Bernhard Mitschang, Norbert Ritter,
Holger Schwarz, Meike Klettke, Andreas Thor,
Oliver Kopp, Matthias Wieland (Hrsg.)

Datenbanksysteme für Business, Technologie und Web (BTW 2017)

Workshopband

06. – 07.03.2017
in Stuttgart, Deutschland

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-266

ISBN 978-3-88579-660-2
ISSN 1617-5468

Volume Editors

Bernhard Mitschang
Universität Stuttgart
Institut für Parallele und Verteilte Systeme, Abteilung Anwendersoftware
70569 Stuttgart, Germany
E-Mail: Bernhard.Mitschang@ipvs.uni-stuttgart.de

Norbert Ritter
Universität Hamburg
Fachbereich Informatik, Verteilte Systeme und Informationssysteme (VSIS)
22527 Hamburg, Germany
E-Mail: ritter@informatik.uni-hamburg.de

Holger Schwarz
Universität Stuttgart
Institut für Parallele und Verteilte Systeme, Abteilung Anwendersoftware
70569 Stuttgart, Germany
E-Mail: holger.schwarz@ipvs.uni-stuttgart.de

Meike Klettke
Universität Rostock
Institut für Informatik
18051 Rostock, Germany
E-Mail: meike.klettke@uni-rostock.de

Andreas Thor
Universität Leipzig
Institut für Informatik, Abteilung Datenbanken
04109 Leipzig, Germany
E-Mail: thor@hft-leipzig.de

Oliver Kopp
Universität Stuttgart
Institut für Parallele und Verteilte Systeme, Abteilung Anwendersoftware
70569 Stuttgart, Germany
E-Mail: oliver.kopp@informatik.uni-stuttgart.de

Matthias Wieland
Universität Stuttgart
Institut für Parallele und Verteilte Systeme, Abteilung Anwendersoftware
70569 Stuttgart, Germany
E-Mail: Matthias.Wieland@informatik.uni-stuttgart.de

Series Editorial Board
Heinrich C. Mayr, Alpen-Adria-Universität Klagenfurt, Austria
(Chairman, mayr@ifit.uni-klu.ac.at)
Dieter Fellner, Technische Universität Darmstadt, Germany
Ulrich Flegel, Infineon, Germany
Ulrich Frank, Universität Duisburg-Essen, Germany
Andreas Thor, HFT Leipzig, Germany
Michael Goedicke, Universität Duisburg-Essen, Germany
Ralf Hofestädt, Universität Bielefeld, Germany
Michael Koch, Universität der Bundeswehr München, Germany
Axel Lehmann, Universität der Bundeswehr München, Germany
Thomas Roth-Berghofer, University of West London, Great Britain
Peter Sanders, Karlsruher Institut für Technologie (KIT), Germany
Torsten Brinda, Universität Duisburg-Essen, Germany
Ingo Timm, Universität Trier, Germany
Karin Vosseberg, Hochschule Bremerhaven, Germany
Maria Wimmer, Universität Koblenz-Landau, Germany

Dissertations
Steffen Hölldobler, Technische Universität Dresden, Germany
Thematics
Andreas Oberweis, Karlsruher Institut für Technologie (KIT), Germany

© Gesellschaft für Informatik, Bonn 2017
printed by Köllen Druck+Verlag GmbH, Bonn

This book is licensed under a
Creative Commons Attribution-NonCommercial 3.0 License.

Vorwort

In den letzten Jahren hat es auf dem Gebiet des Datenmanagements große Veränderungen und Herausforderungen gegeben und auch für die Zukunft zeichnen sich interessante Weiterentwicklungen ab. Im Fokus der Datenbankforschung stehen neben Kerndatenbanktechnologien insbesondere auch Herausforderungen von „Big Data“, also die Analyse von riesigen Datenmengen unterschiedlicher Struktur mit kurzen Antwortzeiten. Neue Architekturansätze für skalierbares Datenmanagement auf der Basis von Cloud Computing gewinnen weiter an Bedeutung, sind aber noch nicht umfassend erforscht.

Zunehmend werden Fragen der effizienten Speicherung, Verarbeitung und Nutzung großer Mengen unstrukturierter Daten adressiert, wie sie unter anderem in sozialen Netzwerken und wissenschaftlichen Anwendungen generiert werden. Hierbei gilt es sowohl allgemeine Konzepte für skalierbares Datenmanagement und effiziente Analysen zu erarbeiten als auch das breite Spektrum an Anwendungsgebieten zu berücksichtigen. Große Datenmengen zu beherrschen ist heute in allen Geschäftsbereichen von Unternehmen ein ebenso essentieller Erfolgsfaktor wie in allen Wissenschaftsbereichen, von den Natur- über Ingenieur- bis hin zu Geisteswissenschaften.

Wie auf jeder BTW-Konferenz gruppieren sich um die Haupttagung eine Reihe von Workshops, die solche speziellen Themen in kleinen Gruppen aufgreifen und sowohl Wissenschaftlern als auch Praktikern und Anwendern einen Rahmen bieten für die intensive Diskussion aktueller Arbeiten in diesen Themenbereichen. Im Rahmen der BTW 2017 finden folgende Workshops statt:

- Big Data Management Systems in Business and Industrial Applications (BigBIA17)
- Big (and small) Data in Science and Humanities (BigDS17)
- Workshop on Industrial Applications of Artificial Intelligence (IAAI’17)
- Präferenzen und Personalisierung in der Informatik (PPI17)
- Scalable Cloud Data Management Workshop (SCDM 2017)

Mit diesen Schwerpunkten reflektiert das Workshop-Programm aktuelle Forschungsgebiete von hoher praktischer Relevanz. Zusätzlich präsentieren Studierende im Rahmen des Studierendenprogramms die Ergebnisse 11 ausgewählter aktueller Abschlussarbeiten im Bereich Datenmanagement. Für jeden Geschmack sollten im Rahmen der BTW 2017 somit interessante Beiträge zu finden sein!

Bei der BTW erstmals neu mit dabei ist eine sog. Data Science Challenge, bei der die Teilnehmer bzw. Teilnehmergruppen einen eigenen Ansatz zur Cloud-

basierten Datenanalyse im Rahmen der gegebenen Aufgabenstellung entwickeln werden. Hierzu müssen zur Verfügung gestellte Datenquellen integriert und ausgewertet werden. Die Teilnehmer haben freie Auswahl der verwendeten Cloud-Plattformen und Analysetechniken. Die Data Science Challenge richtet sich speziell an Doktoranden sowie Studierende. Eine kurze Beschreibung zur Data Science Challenge findet sich hier im Workshop-Band.

Die Materialien zur BTW 2017 werden über die Web-Seiten der Tagung unter http://btw2017.informatik.uni-stuttgart.de/ zur Verfügung stehen.

Die Organisation der BTW-Tagung nebst allen angeschlossenen Veranstaltungen ist nicht ohne die Unterstützung vieler Partner möglich. Diese sind auf den folgenden Seiten aufgeführt; zu ihnen zählen insbesondere die Sponsoren, das Informatik-Forum Stuttgart (infos e.V.), die Universität Stuttgart und auch die Geschäftsstelle der Gesellschaft für Informatik. Ihnen allen gilt unser besonderer Dank!

Stuttgart, im Januar 2017

Bernhard Mitschang, Tagungsleitung/Vorsitzender des Organisationskomitees

Norbert Ritter und Holger Schwarz, Leitung Workshopkomitee

Meike Klettke, Andreas Thor, Leitung Studierendenprogramm

Oliver Kopp und Matthias Wieland, Tagungsband und Organisationskomitee

Tagungsleitung

Bernhard Mitschang, Universität Stuttgart

Organisationskomitee

Johanna Barzen, Univ. Stuttgart
Michael Behringer, Univ. Stuttgart
Uwe Breitenbücher, Univ. Stuttgart
Melanie Herschel, Univ. Stuttgart
Oliver Kopp, Univ. Stuttgart
Frank Leymann, Univ. Stuttgart
Martin Mähler, IBM
Michael Matthiesen, Univ. Stuttgart
Berhard Mitschang, Univ. Stuttgart
Holger Schwarz, Univ. Stuttgart
Matthias Wieland, Univ. Stuttgart
Lena Wiese, Univ. Göttingen

Studierendenprogramm

Meike Klettke, Universität Rostock
Andreas Thor, Hochschule für Telekommunikation Leipzig

Koordination Workshops

Norbert Ritter, Universität Hamburg
Holger Schwarz, Universität Stuttgart

Tutorienprogramm

Johann-Christoph Freytag, HU Berlin

Data Science Challenge

Michael Behrendt, IBM
Pascal Hirmer, Univ. Stuttgart
Martin Mähler, IBM
Kai-Uwe Sattler, TU Ilmenau
Tim Waizenegger, Univ. Stuttgart

Workshop on Big Data Management Systems in Business and Industrial Applications (BigBIA17)

Vorsitz: Benjamin Klöpper, ABB Research; Lena Wiese, Georg-August-Universität Göttingen

Stefan Edlich, Beuth Hochschule
Beate Heisterkamp, Materials Consulting
Holger Kache, IBM
Mikro Kämpf, Cloudera
Carsten Lanquillion, HS Heilbronn
Stefan Mandl, EXASOL AG
Carlos Paiz Gatica, Weidmüller Interface GmbH & Co KG
Daniel Ritter, SAP SE
Klaus Schmid, Univ. Hildesheim
Benedikt Schmidt, ABB Forschungszentrum GmbH
Kerstin Schneider, Hochschule Harz
Heiko Thimm, Hochschule Pforzheim
Liesbeth Vanherpe, École polytechnique fédérale de Lausanne

Workshop on Big (and Small) Data in Science and Humanities (BigDS17)

Vorsitz: Anika Groß, Universität Leipzig; Birgitta König-Ries, Friedrich-Schiller-Universität Jena; Peter Reimann, Universität Stuttgart; Bernhard Seeger, Philipps-Universität Marburg

Alsayed Algergawy, Univ. Jena
Peter Baumann, Jacobs Universität
Matthias Bräger, CERN
Thomas Brinkhoff, FH Oldenburg
Michael Diepenbroeck, Alfred-Wegner-Institut, Bremerhaven
Jana Diesner, University of Illinois at Urbana-Champaign
Christoph Freytag, HU Berlin
Michael Gertz, Univ. Heidelberg
Anton Güntsch, Botanischer Garten und Botanisches Museum, Berlin-Dahlem
Thomas Heinis, IC London
Andreas Henrich, Univ. Bamberg
Jens Kattge, MPI für Biogeochemie
Alfons Kemper, TU München
Meike Klettke Univ. Rostock
Frank Leymann, Univ. Stuttgart
Bertram Ludäscher, University of Illinois at Urbana-Champaign
Alex Markowetz, Univ. Bonn
Jens Nieschulze, Univ. Göttingen
Eric Peukert, Univ. Leipzig
Kai-Uwe Sattler, TU Ilmenau
Uta Störl, Hochschule Darmstadt
Andreas Thor, HfT Leipzig

Workshop on Industrial Applications of Artificial Intelligence 2017 (IAAI'17)

Vorsitz: Alexander Paar, TWT GmbH; Tina Bieth, TWT GmbH; Stefanie Speidel, Karlsruher Institut für Technologie; Cristóbal Curio, Hochschule Reutlingen

Workshop - Präferenzen und Personalisierung in der Informatik (PPI17)

Vorsitz:Markus Endres, Universität Augsburg; Andreas Pfandler, TU Wien und Universität Siegen

Gabor Erdelyi, Univ. Siegen
Jan-Christoph Kalo, TU Braunschweig
Ekaterina Lebedeva, Australian National University
Stefan Mandl, EXASOL AG
Timotheus Preisinger, DEVnet GmbH
Peter Vojtas, Charles Univ. Prague
Antonius Weinzierl, TU Wien
Florian Wenzel, Univ. Augsburg

Scalable Cloud Data Management Workshop (SCDM 2017)

Vorsitz: Norbert Ritter, Universität Hamburg; Felix Gessert, Universität Hamburg

Haopeng Chen, Shanghai Jiao Tong University
Keke Chen, Wright State Univ.
Shiping Chen, CSIRO
Stefan Dessloch, Univ. Kaiserslautern
Sameh Elnikety, Microsoft Research
Nils Gruschka, FH Kiel
Shigeru Hosono, NEC Knowledge Discovery Research Lab
Ching Hsien (Robert) Hsu, Chung Hua University
Meike Klettke, Univ. Rostock
Harald Kosch, Univ. Passau
Sébastien Mosser, Université Nice-Sophia Antipolis
Jiannan Ouyang, Univ. of Pittsburgh
Fabian Panse, Univ. Hamburg
Armin Roth, IBM
Sherif Sakr, Univ. of New South Wales
Holger Schwarz, Univ. Stuttgart
Russel Sears, Pure Storage
Uta Störl, HS Darmstadt
Andreas Thor, Univ. Leipzig
Liqiang Wang, Univ. of Wyoming
Eiko Yoneki, Univ. of Cambridge

Inhaltsverzeichnis

Workshopprogramm

Big Data Management Systems in Business and Industrial Applications (BigBIA17)

Benjamin Klöpper, Lena Wiese
Vorwort 23

Kristof Böhmer, Florian Stertz, Tobias Hildebrandt, Stefanie Rinderle-Ma, Günther Eibl, Cornelia Ferner, Sebastian Burkhart, Dominik Engel
Application and Testing of Business Processes in the Energy Domain . . . 25

Matthias Carnein, Leschek Homann, Heike Trautmann, Gottfried Vossen, Karsten Kraume
Customer Service in Social Media: An Empirical Study of the Airline Industry 33

Cornelius A. Ludmann
Einsatz eines Datenstrommanagementsystems zur Empfehlung von beliebten Nachrichtenartikeln in der CLEF NewsREEL Challenge 41

Holger Eichelberger, Cui Qin, Klaus Schmid
Experiences with the Model-based Generation of Big Data Pipelines . . . 49

Sebastian Czora, Marcel Dix, Hansjörg Fromm, Benjamin Klöpper, Björn Schmitz
Mining Industrial Logs for System Level Insights 57

Eduard Bergen, Stefan Edlich
Post-Debugging in Large Scale Big Data Analytic Systems 65

Big (and small) Data in Science and Humanities (BigDS17)

Anika Groß, Birgitta König-Ries, Peter Reimann, Bernhard Seeger
Vorwort 75

Jihen Amara, Bassem Bouaziz, Alsayed Algergawy
A Deep Learning-based Approach for Banana Leaf Diseases Classification 79

Daniel Kaltenthaler, Johannes-Y. Lohrer, Peer Kröger, Henriette Obermaier
A Framework for Supporting the Workflow for Archaeo-related Sciences: Managing, Synchronizing and Analyzing Data 89

Cornelia Kiefer
Die Gratwanderung zwischen qualitativ hochwertigen und einfach zu erstellenden domänenspezifischen Textanalysen 99

Stephan Kemper, André Petermann, Martin Junghanns
Distributed FoodBroker: Skalierbare Generierung graphbasierter Geschäftsprozessdaten 105

Pascal Hirmer
Effizienz-Optimierung daten-intensiver Data Mashups am Beispiel von Map-Reduce 111

Christian Beilschmidt, Johannes Drönner, Michael Mattig, Marco Schmidt, Christian Authmann, Aidin Niamir, Thomas Hickler, Bernhard Seeger
Interactive Data Exploration for Geoscience 117

Golnaz Elmamooz, Bettina Finzel, Daniela Nicklas
Towards Understanding Mobility in Museums 127

Workshop on Industrial Applications of Artificial Intelligence (IAAI'17)

Alexander Paar, Tina Bieth, Stefanie Speidel, Cristóbal Curio
Vorwort 137

Frederick Birnbaum, Christian Moewes, Daniela Nicklas, Ute Schmid
Data Mining von multidimensionalen Qualitätsdaten aus einer computerintegrierten industriellen Fertigung zur visuellen Analyse von komplexen Wirkzusammenhängen 139

Jasmin Ramadani, Stefan Wagner
Mining Java Packages for Developer Profiles: An Exploratory Study . . . 143

Präferenzen und Personalisierung in der Informatik (PPI17)

Markus Endres, Andreas Pfandler
Vorwort 155

Javier Romero
asprin: Answer Set Programming with Preferences 159

Theresa Csar, Martin Lackner, Reinhard Pichler, Emanuel Sallinger
Computational Social Choice in the Clouds 163

Martin Diller, Anthony Hunter
Encoding monotonic multiset preferences using CI-nets 169

Lena Rudenko, Markus Endres
Personalized Stream Analysis with PreferenceSQL 181

Gábor Erdélyi, Christian Reger
Possible Voter Control in k-Approval and k-Veto Under Partial Information 185

Jan Maly, Stefan Woltran
Ranking Specific Sets of Objects 193

Ladislav Peska, Peter Vojtas
Towards Complex User Feedback and Presentation Context in Recommender Systems 203

Scalable Cloud Data Management Workshop (SCDM 2017)

Felix Gessert, Norbert Ritter
Vorwort 211

Steffen Friedrich, Wolfram Wingerath, Norbert Ritter
Coordinated Omission in NoSQL Database Benchmarking 215

Matthias Kricke, Martin Grimmer, Michael Schmeißer
Preserving Recomputability of Results from Big Data Transformation Workflows 227

Daniel Janusz, Jochen Taeschner
Privatsphäre-schützende Bereichsanfragen in unsicheren Cloud-Datenbanken 237

Andreas Bader, Oliver Kopp, Michael Falkenthal
Survey and Comparison of Open Source Time Series Databases 249

Wolfram Wingerath, Felix Gessert, Steffen Friedrich, Erik Witt, Norbert Ritter
The Case For Change Notifications in Pull-Based Databases 269

Studierendenprogramm

Felix Dreissig, Niko Pollner
A Data Center Infrastructure Monitoring Platform Based on Storm and Trident 281

Araek Tashkandi, Lena Wiese, Marcus Baum
Comparative Evaluation for Recommender Systems for Book Recommendations 291

Julia Romberg
Comparing Relevance Feedback Techniques on German News Articles . . 301

Corinna Giebler, Christoph Stach
Datenschutzmechanismen für Gesundheitsspiele am Beispiel von Secure Candy Castle 311

Florian Pretzsch
Duplikaterkennung in der Graph-Processing-Platform GRADOOP 321

Stefan Noll
Energy Efficiency in Main-Memory Databases 335

Alexander Askinadze
Fake war crime image detection by reverse image search 345

Björn Salgert, Thomas C. Rakow
Modellierung von relationalen Datenbanken mit UML im Round-Trip-Engineering 355

Oliver Swoboda
Serverseitige Aggregation von Zeitreihendaten in verteilten NoSQL-Datenbanken 365

Roland Kahlert, Matthias Liebeck, Joseph Cornelius
Understanding Trending Topics in Twitter 375

Wolfgang Amann
Vergleich und Evaluation von RDF-on-Hadoop-Lösungen 385

Tutorienprogramm

Thomas Seidl
Multimedia Similarity Search 397

Felix Gessert, Wolfram Wingerath, Norbert Ritter
Scalable Data Management: An In-Depth Tutorial on NoSQL Data Stores 399

Data Science Challenge

Tim Waizenegger
BTW 2017 Data Science Challenge (SDSC17) 405

Autorenverzeichnis

Workshopprogramm

Big Data Management Systems in Business and Industrial Applications (BigBIA17)

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 23

Big Data Management Systems in Business and Industrial Applications (BigBIA17)

Benjamin Klöpper[1], Lena Wiese[2]

Big Data stands for the intelligent and efficient handling and usage of large, heterogeneous and fast changing amounts of data. The ultimate goal of big data is the generation of valuable insights from ever growing amounts of data. Hence, the application of Big Data in business and industry contexts has proven to be very valuable. However, several challenges related to big data remain; these challanges go beyond the often used catchphrases: volume, velocity, variety, and veracity and also address security, privacy, linked-data technologies in the context of a wide range of AI applications. Big Data is more than just data analysis since the outcome influences the way how digital businesses and a knowledge economy are organized now and in the future.

This workshop is dedicated to the application of Big Data concepts and technologies in real-world systems for a variety of application domains like Manufacturing, Logistics, Media, Healthcare, and Finance.

For this edition of the workshop, we accepted six papers covering diverse topics like data stream management, business processes, analytics and log mining in and for Big Data Management Systems. We thank the program committee members and the BTW organizers for their support and all authors for their contributions.

1 Workshop Organizers

Benjamin Klöpper (ABB Research)
Lena Wiese (Georg-August-Universität Göttingen)

2 Program Committee

Stefan Edlich, Beuth Hochschule für Technik
Beate Heisterkamp, Materials Consulting
Holger Kache, IBM Deutschland RD GmbH
Mikro Kämpf, Cloudera
Carsten Lanquillion, Hochschule Heilbronn
Stefan Mandl, EXASOL AG

[1] ABB Research, benjamin.kloepper@de.abb.com
[2] Georg-August-Universität Göttingen, wiese@cs.uni-goettingen.de

Carlos Paiz Gatica, Weidmüller Interface GmbH & Co KG
Daniel Ritter, SAP SE
Klaus Schmid, Universität Hildesheim
Benedikt Schmidt, ABB Forschungszentrum GmbH
Kerstin Schneider, Hochschule Harz
Heiko Thimm, Hochschule Pforzheim
Liesbeth Vanherpe, Ecole polytechnique federale de Lausanne

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 25

Application and Testing of Business Processes in the Energy Domain

Kristof Böhmer[1] Florian Stertz[1] Tobias Hildebrandt[1] Stefanie Rinderle-Ma[1]
Günther Eibl[2] Cornelia Ferner[2] Sebastian Burkhart[2] Dominik Engel[2]

Abstract:

The energy domain currently struggles with radical legal and technological changes, such as, smart meters. This results in new use cases which can be implemented based on business process technology. Understanding and automating business processes requires to model and test them. However, existing process testing approaches frequently struggle with the testing of process resources, such as ERP systems, and negative testing. Hence, this work presents a toolchain which tackles that limitations. The approach uses an open source process engine to generate event logs and applies process mining techniques in a novel way.

Keywords: Business Process, Process Testing, Energy, Process Analysis

1 Introduction

The protection of today's energy production and transmission organizations against fraud, misuse, and faults is crucial in order to ensure the stable, easy, and cheap access to electricity [Co09]. Until now the energy domain achieved the required level of protection based on an *isolation driven strategy*, cf. [We10]. However, driven by legislation changes, increased complexity, and new technologies the need for open and standardized approaches becomes obvious, cf. [AB07]. For example, *smart meters* have emerged in recent years. Smart meters provide fine-grained energy consumption measurement and bidirectional communication with various stakeholders, such as energy providers, cf. [DWD11]. Such novel technologies and developments provide a plethora of advantages – but also pose *novel challenges*, cf. [We10]. For example, smart meters do not only foster the stabilization of large complex energy transmission networks but also enable to remotely cut off end customers from their access to electricity, cf. [DWD11]. Hence, it is important to test smart meters and related technologies.

At the same time, the complexity increased in the energy domain. Hence, standardized use cases were defined and agreed upon by major Austrian energy producers and distributors, cf. [OE15]. These use cases describe the core business processes of energy producers and distributors (denoted as *energy processes* in the following) in textual form. For energy

[1] Universität Wien, Workflow Systems and Technology, Währingerstrasse 29, A-1090 Wien, firstname.lastname@univie.ac.at
[2] FH Salzburg, Josef-Ressel-Zentrum, Campus Urstein Süd 1, A-5412 Puch/Salzburg, firstname.lastname@fh-salzburg.ac.at

providers it is a crucial task to implement the energy processes in order to address upcoming energy related challenges.

The process model for the `prepayment` use case is depicted in Fig. 1. This and the following examples are defined using Business Process Model and Notation (BPMN) as the standard process modeling notation. Fig. 1 contains multiple related critical smart metering use cases (e.g, payment handling) that can, if a fault occurs, lead to an unexpected energy turnoff. For the sake of brevity this work will focus on the depicted use case and its connected process model. However other use cases and requirements were also successfully modeled.

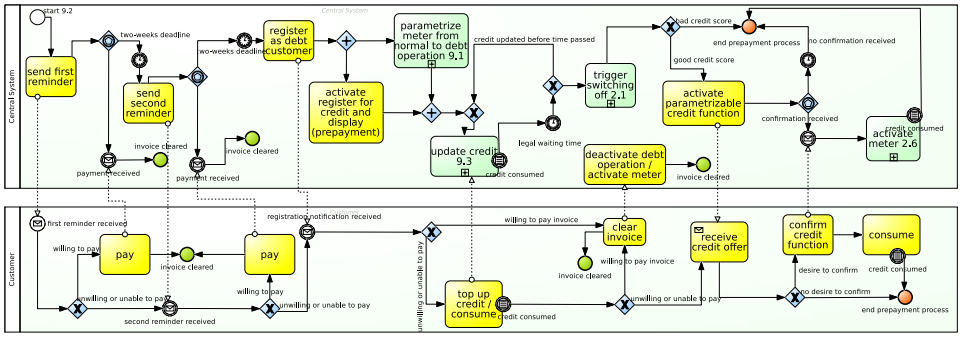


Fig. 1: BPMN model of prepayment process and use cases (modeled using Signavio)

In addition to implementing specifications/use cases it is also necessary to *test* the created processes and their integration of internal and external services. Hereby related faults can be identified early during the process design phase. Business process testing focuses on defining and executing test cases that specify expected behavior. Unfortunately current process testing approaches are somewhat limited. For example, existing approaches struggle with a flexible integration of real world and mocked services into the business processes. Hence existing work abstracts from incorporating external services into the tests, cf. [BR16].

Moreover, existing approaches focus on testing that specified behavior is possible (i.e., positive testing) neglecting negative testing. This is testing that a specific kind of behavior is not possible, e.g., to ensure that an anticipated fault is not present.

Hence, existing work isn't suitable to answer the following research questions:

RQ1 How can business process testing be applied in the energy domain?

RQ2 How to flexibly transition from simulation/test environments into production?

RQ3 How to conduct negative testing in the proposed process testing toolchain?

This work presents a toolchain for modeling and testing real world use cases and processes from the energy domain. Hereby this work especially focuses on automatically testing if all resources (e.g., applications or web-services), which are integrated in the processes, behave "correctly" based on their specification. Moreover it will be discussed how fine granular negative and positive test cases can be defined using conformance rules.

The toolchain is discussed in Section 2. Evaluation results are discussed in Section 3. Section 4 discusses related work. Conclusions and future work is given in Section 5.

2 Toolchain for Flexible Process Testing

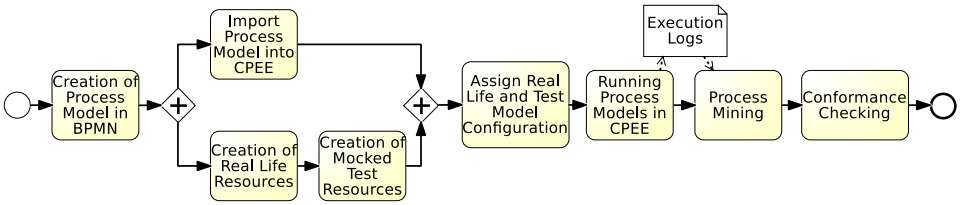


Fig. 2: Toolchain

Existing process testing approaches typically struggle with the integration and testing of resources, cf. [BR15]. Hence, we propose a resource focused testing approach, cf. Fig. 2. The proposed approach assumes that following information is given: a process model that should be tested, a set of resources that are utilized by the model, and test cases. Each test case (short test) holds resource specifications and the data flow variable values required to instantiate/execute the model. Moreover, the presented approach utilizes the Cloud Process Execution Engine (CPEE). It is applied to execute processes under real world conditions but also to simulate executions for test purposes.

The testing starts with importing the model into the CPEE two times with different configurations. The first configuration wires all resources – referenced by the model – to their real life resource implementations (e.g., real web services which would enable to check credit card data). Secondly, a testing configuration is created that utilizes abstracted mocked versions of the real life resources. We refer to the first configuration as the "real life model" and to the second one as the "testing model".

The mocked resources are generated individually, based on resource specifications which are hold by each test. Hence, each test case likely will bring a slightly varying specification. These specifications must fit the use cases and scenarios to be tested. For example, specifications and use cases can be extracted from public documentations, e.g., the use cases published by the Austrian energy domain [OE15]. Note, that the CPEE enables to flexibly configure if a mocked or real life resource should be utilized. Moreover, it enables to flexibly implement the mocked resource behavior programmability with the tool of choice. For example, as a process using a chosen modeling notation or as a service using a programming language such as PHP or Java.

Finally, all the preparatory test artifacts (e.g., mocked resources) are available. Hence, the testing can continue based on a two pronged approach. First, the real life model is executed based on the instantiation/test data defined in the test case. Secondly, the testing model is executed based on the same test data than the real life model and the mocked resources. During the execution of both model configurations the CPEE logging component, cf. [St16], stores all the executed behavior (i.e., execution events) in an execution log. Events and execution logs hold, for example, which steps (e.g., activities) were executed during the

execution of the models. Moreover, they enable to deduce the order of the steps and the data used/exchanged during the execution.

This enables to conduct a wide range of an analysis based on existing conformance checking and process mining techniques, cf. [Aa11]. Conformance checking enables to determine if recorded execution logs conform to expected behavior. For example, conformance checking enables us to determine if each recorded step fits to behavior which is defined in a process model. In comparison, process discovery approaches enable to automatically "identify" the structure/behavior of a process model based on recorded execution logs. In short, recorded execution logs can be converted into a model that is capable of producing the analyzed log – based on existing process mining approaches.

This approach proposes to exploit conformance checking and process mining to ensure the correct implementation of resources which are integrated into process models. Tool support for conformance checking and process mining is provided by, for example, ProM (http://www.promtools.org/). It is proposed to apply process mining on the execution log generated by the model under test. The hereby generated process model is later compared to the execution log generated by the real life model using existing conformance checking techniques, cf. [Aa11]. It is assumed that identified deviations likely indicate either faults at the test/mocked resources or, more likely, at the real life resource implementations.

For example, assume, that a resource has to decide if Eve is allowed to order a product based on her previously determined creditworthiness during a process model execution. Hence, the models' execution differentiates if Eve is creditworthy or not (e.g., an order product step is executed/logged during the models' execution or not). Imagine that the test case defines Eve as *not* creditworthy and also instruments the mocked service accordingly. However, if in the logged real life model events the order product step can be found then the conformance checking will fail and point out a potential fault.

The generated results can be utilized to improve the tested processes in an iterative manner. Hence, the proposed testing approach can be applied multiple times in a row to determine if all identified deviations from the test specification were found/fixed. During each iteration identified deviations can be addressed, either by fixing the real live resource implementation or by adapting a faulty test. Note, we assume that, most likely, multiple test cases will be available. In such a case the previous steps are executed once for each test case.

Finally, we want to point out two additional advantages of the proposed approach. Models can be tested and executed with a mixture of real life and mocked resources. This allows to test the implementation of the resources as soon as they become available. Moreover, this shortens the time to market. This is because the real life process model and execution engine can be used during all tests. Hence, the tested real life model configuration can, after the testing has finished, simply be pushed to the production environment. In addition, negative tests can also be conducted by altering the mocked resources so that they behave "incorrectly". This enables to test/ensure that the specified incorrect behavior is not occurring in the real life process model and its resources.

3 Evaluation

For evaluating the toolchain, we focused on the prepayment process, discussed in Section 1 and 2. We focused on this use case, because the end state of this use case is severe, since a customer would be cut off of his access to power (i.e., electricity), cf. [OE15]. As the first step, we created the BPMN model for this use case. Fig. 1 shows the result of the first step. This process starts with sending a reminder to the customer for paying his bill. A second reminder will be sent, if no payment has been received for two weeks. If the customer does not pay after two weeks the central system of the energy utility will register the customer as a debt customer. The smart meter of the customer will be put in a debt mode and a credit option will be activated for the customer, as a prepayment option. The customer can then top up his credit and consume it afterwards. After his credit is depleted, his power will be cut off. If he is financially powerful, the energy utility, can offer him a credit function for his power supply. If the customer accepts, his power supply will be turned on again and he can consume as much power as his credit covers. Again his power will be cut off, if his credit is depleted. A customer does always have the option of paying his bills to be cleared and registered as a normal customer again.

The next step involves two parts. The first part is building a process model in the CPEE. One approach would be creating an XML file directly for the CPEE, which is tedious and prone to errors. So the CPEE front end (http://cpee.org/~demo/cpee-cockpit) also allows to create process models easily in two ways. It is possible to either create a process model through a graphical editor or to directly importing a model from a BPMN file. The process model can be seen in Fig. 3. Every step of this process sends data to a resource.

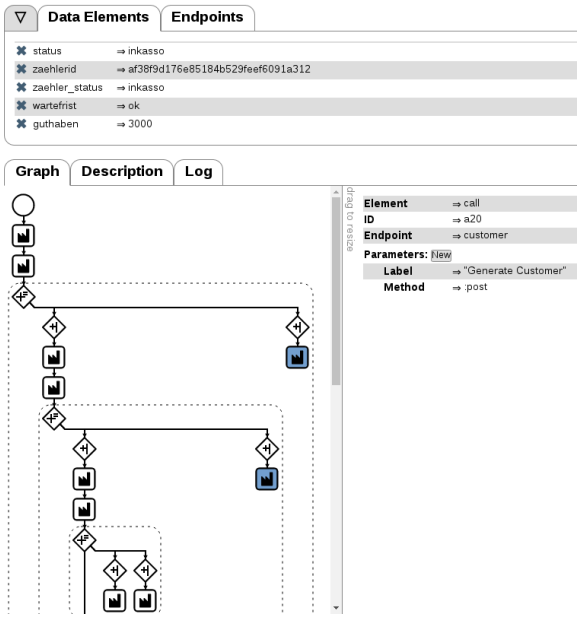


Fig. 3: Prepayment Process created using the CPEE front end

In this scenario, every resource is a web resource and the messages are sent via HTTP. It is important to note, that these resources can be created easily and do not require a big amount of code. For example, PHP based web-services are perfectly suited for this case. After the testing phase, each of these resources can be swapped for a real resource. List. 1 shows a very small entry of an event log. The event “Send first reminder” has been observed on the 5th of October. For this simple query, we wanted to know if a customer with the id of 163 has paid his bill, so we sent his id to the resource. The resource reported back to us and we can see, that the

customer with the id 163 paid his first reminder. The process should end here, since the customer should not be a debt customer now.

List. 1: Example of Log File

```
<event>
  <string key="concept:name" value="Send␣first␣reminder"/>
  <list key="data_send">
    <string key="knr" value="163"/>
  </list>
  <date key="time:timestamp" value="2016-10-05T19:38:26+02:00"/>
  <list key="data_received">
    <string key="sent" value="true"/>
  </list>
</event>
```

How the actions of a resource on the CPEE are logged is explained in more detail in [St16]. The next step describes executing this process model. The CPEE creates a process instance of a process model and executes it. It is possible to create many instances at once, so we can generate multiple event logs which cover the whole behavior of the model. Every process instance creates one event log. These event logs can also be put together to generate one large event log.

The next step of the toolchain is process mining. The basics of process mining are described in Section 2. We created an event log with 1001 processes which consists of 20349 events. With ProM we can mine a process model out of this log and see if it suits our designed process model and our conformance rules through conformance checking. It is important to note, that the implemented conformance checking at the moment only takes the control flow into account for the fault detection. Faults related to data, such as, an incorrect process data variable state, are not detectable at this time. An in depth description of conformance checking can be found in [RA08].

After the conformance checking, the test resources can be altered and new event logs can be created. These steps can be repeated until the conformance checking results fit the real process model and all the test resources were swapped out for the real resources.

4 Related Work

Related work can mainly be found in the *business process testing* domain. There a plethora of approaches, techniques, and concepts are available, cf. [BR15]. Surprisingly it was found that a *flexible integration* of external or internal services and applications (i.e., resources) into test executions is currently hardly provided. Hence, the majority of the existing process testing work abstracts from this challenge, and for example, ignores it, cf. [BR15; ML06].

Alternatively simple approaches are applied that typically simulate resources based on test cases that only hold predefined resource return values, cf. [Br08; BR15; LSD08]. Overall, the most flexible existing integration of resources were found in [LS06] and [LSD08]. In [LS06] the authors propose to transform the process into JAVA code (e.g., activities would become classes), which enables to flexibly mock external resources. For example, basic simulated resource behavior can be implemented in JAVA source code.

This results in a high manual programming effort and still does not enable to easily switch between mocked resources and their real world implementation. So in [LSD08] the authors propose to utilize the Business Process Execution Language (BPEL) to defined the behavior of mocked external resources. Unfortunately their proposed framework only focuses on testing BPEL processes. Overall we came to the conclusion that the integration of resources into test executions is currently limited and tackle this limitation with the proposed testing framework. A similar situation was found when the related work was checked for their *negative testing* capabilities. A majority of the identified existing work focuses on positive testing, cf. [BR15]. Hence, the expected behavior is defined at the test cases, for example, based on the expected variable states or control flow paths. During test execution the observed behavior is compared with the expected one.

However, while existing model checking based testing approaches are also mainly applied for positive testing they can also be utilized for negative testing, cf. [BR15; FBS04; Na06]. Unfortunately, such model testing based approaches frequently require an in depth knowledge of formal rule modeling techniques or custom rule definition languages. We assume that such knowledge can be lacking at typical IT professionals, cf. [BR15]. Overall, we concluded that existing process testing work does not put a strong focus on negative testing – a limitation addressed by this work. In addition we were not able to identify any existing work that focuses or reports on process testing in the energy domain.

5 Conclusion

This paper proposes a process testing toolchain that takes a process model and resources as input and generates test log data as output. In addition existing process conformance and mining approaches are applied in a novel way to test the processes. Hereby, existing resource specifications are exploited to ensure that resources which are integrated into business processes are correctly implemented. In addition the presented toolchain enables, based on the process execution engine CPEE, a flexible integration and switches between real life implementations of resources and customized mocked resources (→ **RQ2**).

It was concluded that this not only enables to start early with resource/process testing (i.e., when the first real life resources become available) but also improves the time to marked for the process models under test. The presented approach also discusses the provision of negative testing capabilities (→ **RQ3**). In order to demonstrate the applicability and feasibility of the toolchain, it has been applied to a real-world uses case/specification from the energy domain [OE15], i.e., the *prepayment* process (→ **RQ1**).

The toolchain can be also applied independently from the energy domain. In future work, the negative testing capabilities could be used in the security domain to ensure that predicted attack vectors cannot be exploited.

Acknowledgement This research was partly funded by the Austrian Research Promotion Agency, project 849914.

References

[Aa11] van der Aalst, W. M.: Process Mining: Discovery, Conformance and Enhancement of Business Processes. Springer, 2011.

[AB07] Armaroli, N.; Balzani, V.: The future of energy supply: challenges and opportunities. Angewandte Chemie International Edition 46/1-2, pp. 52–66, 2007.

[Br08] Breu, R.; Lechner, A.; Willburger, M.; Katt, B.: Workflow Testing. In: Leveraging Applications of Formal Methods, Verification and Validation. Springer, pp. 709–723, 2008.

[BR15] Böhmer, K.; Rinderle-Ma, S.: A systematic literature review on process model testing: Approaches, challenges, and research directions, Technical Report, 2015.

[BR16] Böhmer, K.; Rinderle-Ma, S.: A Testing Approach for Hidden Concurrencies based on Process Execution Logs. In: Service Oriented Computing. Oct. 2016.

[Co09] Cottrell, F.: Energy & society: the relation between energy, social change, and economic development. AuthorHouse, 2009.

[DWD11] Depuru, S. S. S. R.; Wang, L.; Devabhaktuni, V.: Smart meters for power grid: Challenges, issues, advantages and status. Renewable and sustainable energy reviews 15/6, pp. 2736–2742, 2011.

[FBS04] Fu, X.; Bultan, T.; Su, J.: Analysis of interacting BPEL web services. In: World Wide Web. ACM, pp. 621–630, 2004.

[LS06] Li, Z. J.; Sun, W.: Bpel-unit: Junit for bpel processes. In: Service Oriented Computing. Springer, pp. 415–426, 2006.

[LSD08] Li, Z. J.; Sun, W.; Du, B.: BPEL4WS unit testing: framework and implementation. Business Process Integration and Management 3/2, pp. 131–143, 2008.

[ML06] Mayer, P.; Lübke, D.: Towards a BPEL unit testing framework. In: Testing, analysis, and verification of web services and applications. ACM, pp. 33–42, 2006.

[Na06] Nakajima, S.: Model-checking behavioral specification of BPEL applications. Theoretical Computer Science 151/2, pp. 89–105, 2006.

[OE15] OE: Smart Metering Use-Cases für das Advanced Meter Communication System (AMCS), Version 1.0, tech. rep. 1/88, Oesterreichs Energie, 2015.

[RA08] Rozinat, A.; van der Aalst, W. M.: Conformance checking of processes based on monitoring real behavior. Information Systems 33/1, pp. 64–95, 2008.

[St16] Stertz, F.; Rinderle-Ma, S.; Hildebrandt, T.; Mangler, J.: Testing Processes with Service Invocation: Advanced Logging in CPEE. In: Service Oriented Computing. Oct. 2016.

[We10] Wei, D.; Lu, Y.; Jafari, M.; Skare, P.; Rohde, K.: An integrated security system of protecting smart grid against cyber attacks. In: Innovative Smart Grid Technologies. IEEE, pp. 1–7, 2010.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 33

Customer Service in Social Media: An Empirical Study of the Airline Industry

Matthias Carnein[1] Leschek Homann[2] Heike Trautmann[3] Gottfried Vossen[4] Karsten Kraume[5]

Abstract: Until recently, customer service was exclusively provided over traditional channels. Customers could write an email or call a service center if they had questions or problems with a product or service. In recent times, this has changed dramatically as companies explore new channels to offer customer service. With the increasing popularity of social media, more companies thrive to provide customer service also over Facebook and Twitter. Companies aim to provide a better customer experience by offering more convenient channels to contact a company. In addition, this unburdens traditional channels which are costly to maintain. This paper empirically evaluates the performance of customer service in social media by analysing a multitude of companies in the airline industry. We have collected several million customer service requests from Twitter and Facebook and automatically analyzed how efficient the service strategies of the respective companies are in terms of response rate and time.

Keywords: Customer Service, Social Media, Twitter, Facebook, Airline Industry, CRM, Analytics

1 Introduction

Contacting a company to receive help with a service or to make a complaint is often a tedious task. Calling a service centre can lead to long waiting times and waiting for a response to an email can take days or even weeks. Due to these drawbacks of traditional channels, many customers turn to social media and post questions or make complaints on the social media presence of a company. Ignoring these customer inquiries can lead to negative feedback and even bad press for the company. For this reason, many companies offer customer service over social media. This paper reports on an empirical study of social media-based customer service in the airline industry.

Analyzing social media is a challenge due to the volume and structure of the data as well as its velocity. Thousands of tweets and posts arrive as a permanent stream of data that needs to be processed accordingly. Analysing such volumes by hand is impossible and the unstructured nature of the data is demanding for the underlying architecture. This paper evaluates millions of customer inquiries from Twitter and Facebook and analyzes how companies respond to them. The results give a market overview of the current state of customer service in social media and serves as a benchmark for the airline industry. To the

[1] University of Münster, Leonardo-Campus 3, 48149 Münster, Germany, matthias.carnein@uni-muenster.de
[2] University of Münster, Leonardo-Campus 3, 48149 Münster, Germany, homannl@uni-muenster.de
[3] University of Münster, Leonardo-Campus 3, 48149 Münster, Germany, trautmann@wi.uni-muenster.de
[4] University of Münster, Leonardo-Campus 3, 48149 Münster, Germany, vossen@uni-muenster.de
[5] University of Münster, Leonardo-Campus 3, 48149 Münster, Germany, karsten.kraume@uni-muenster.de

best of our knowledge, this work is the first benchmark of customer service in social media of its size and presents interesting findings for the strategies in different market segments.

This paper is organized as follows: Section 2 introduces related work and reviews the current state of customer service. Section 3 describes the technical approach, identifies different airline segments and defines relevant Key Performance Indicators (KPIs) to evaluate customer service performance. Section 4 presents the results and highlights companies that perform well or poorly. Finally, Section 5 concludes with a summary of results and an outlook on future research.

2 Background

In the traditional customer service business, the service level is typically measured using the *Average Handling Time (AHT)* or *Average Queue Time (AQT)*. The AHT describes the average time that an agent requires to handle a customer inquiry. The lower it is, the faster the request or question could be served, leading to lower cost for the service provider. Similarly, the AQT describes the average time that a customer has to wait in a queue until a service agent becomes available and able to respond to the inquiry. In other words, the AQT describes the time that passes until the first response from the company. A lower AQT delivers a much better service experience for the customer and leads to higher satisfaction.

In social media, such KPIs are much harder to define as there is no line of waiting customers. Additionally, the handling time can be difficult to determine due to the unstructured nature of conversations in social media. In the following section we derive suitable performance measures for service in social media and present ways to automatically monitor such KPIs for industries and companies. Our analysis results are demonstrated using the airline industry. In the past, the analysis of customer service in social media for the airline industry has attracted considerable attention from research and practice.

As an example, Talkwalker [Ta16] published a number of descriptive statistics of the last month for 18 large airlines. Their focus lies on number of mentions, sentiment of the posts as well as language and country of origin. Their key findings report, for example, that airlines such as KLM, Qatar, jetBlue, and Flying Emirates attract mostly positive sentiment in social media. In contrast, Air France, American Airlines or Iberia attract mostly criticism and negative feedback. Furthermore, large US airlines appear to receive the most mentions in social media, followed by large European airlines.

Similarly, customer service provider Conversocial [Co16] has recently published a report evaluating service performance in social media. The authors have analyzed up to $2,000$ @-mentions from Twitter in July 2016 for 20 different airlines and have computed the number of tweets and response times. Their results show that Southwest Airlines, Alaska Airlines, KLM and Lufthansa provided the best performance with respect to response time and response rate. In contrast, spirit Airlines, Turkish Airlines and easyJet provide the worst. However, the analysis conducted in this paper has a much wider scope and is thus much more comprehensive.

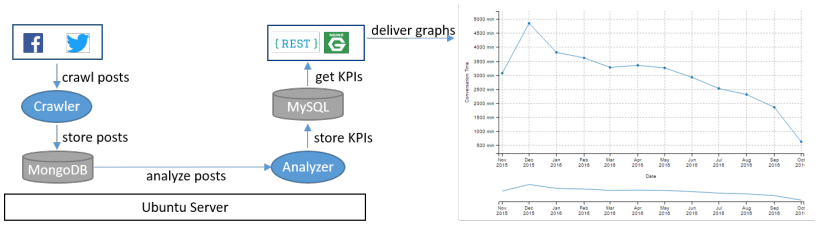


Fig. 1: Visualization of the architecture.

3 Set up

3.1 Architecture

To analyze the posts we set up a dedicated server which provides the necessary databases and also hosts the front end in form of a website. All social media posts are collected via the Facebook Graph API [Fa16] and the Twitter REST API [Tw16]. The data is crawled by using Python scripts that periodically query the APIs and store new posts in a database. The advantage of using Python scripts in our implementation is driven by the intention to create a modular solution without dependencies on other frameworks. The APIs provide each post as a collection of key-values in form of a JSON file. This data is inherently unstructured and may differ between Facebook and Twitter as well as different API versions. For this reason, we utilize a document-based database to store the social media posts. The document-based database allows to store arbitrary key-value data as single objects. Due to its popularity, size of community and maturity status, MongoDB [Mo16] is our choice of database.

Besides the document database, we employ a MySQL database [Or16] that is used to store our defined KPIs for each post together with its time stamp. In contrast to the unstructed JSON objects, the KPIs have a well-defined structure and are best stored in a relational database. Similar to the crawler, the KPIs are periodically calculated by Python scripts that analyze each post and store the KPIs in the MySQL database. Note that we are not dependent on MySQL and could use another relational database, e.g., supporting more OLAP functionality. Finally, the KPIs are delivered to a web service using a REST API hosted by an nginx web server and are visualized using JavaScript. Note that since most calculations are performed asynchronously, we expect our architecture to be able to hold even longer time series without performance issues. The overall structure of the architecture is shown in Figure 1.

3.2 Data Sources

We have identified the airlines with the largest passenger volume, fleet size and revenue targeting various market segments. Our search has resulted in 48 different airlines, ranging from low-cost airlines such as RyanAir to full-service airlines such as Lufthansa. For all

Tab. 1: Number of social media posts per airline from January 2016 – November 2016 and number of passengers as reported on the companies' homepage.

Airline	Number of posts	Passengers per year
KLM	404,964	27,740,000
Iberia	229,677	14,000,000
Etihad Airways	175,609	17,400,000
Air France	166,225	79,016,000
WestJet	139,560	18,500,000
Lufthansa	155,643	107,000,000
easyjet	117,301	70,000,000
Air Berlin	83,696	27,274,777
Thomas Cook Airlines	75,288	6,700,000
Air Canada	73,731	41,000,000

companies, we collected the Facebook and Twitter accounts targeted at the English and German speaking market. This has lead to a total of 66 Twitter and 58 Facebook accounts. We focus our analysis on Facebook and Twitter since they are the most widely used platforms to offer customer service.

For all Twitter accounts, we have collected the *@-mentions* directed at the airlines by querying their Twitter REST API [Tw16]. In other words, we have searched the stream of tweets for all tweets in which one of the companies is addressed. To collect the responses from the airlines, we query the timeline of the airline's Twitter account. This approach allows us to build a conversation structure where posts and their responses are linked.

For all Facebook accounts, we utilize the Facebook Graph API [Fa16] to collect all posts from the airline's Facebook page. This includes so-called *visitor posts*, i.e., posts that users can leave on the page, as well as comments on posts from the company.

Unfortunately, API access is often limited. As an example, the Twitter API restricts search access to approximately the past 9 days. To lift these restrictions we continuously monitored the different accounts over time to gather a total of 6.187.835 posts directed at one of the airlines and 1.777.234 responses from the respective airlines. This allowed us to construct a total of 1.388.922 conversations, i.e., chains of posts and responses which can be used to automatically analyze the service performance. In our analysis we restrict the samples to the current year, i.e., January 2016 – November 2016. For brevity we are not able to present the results for all 48 airlines. Instead, we focus on the ten airlines with the largest volume on social media as shown in Table 1. To put this into perspective, we also listed the number of passengers per year.

3.3 Airline Market Segments

In general, the airline industry can be divided into four main market segments: Low-cost carriers, regional carriers, leisure carriers, and full-service network carriers [Bö14]. Each segment is characterized by service level as well as number of flight routes. While low-

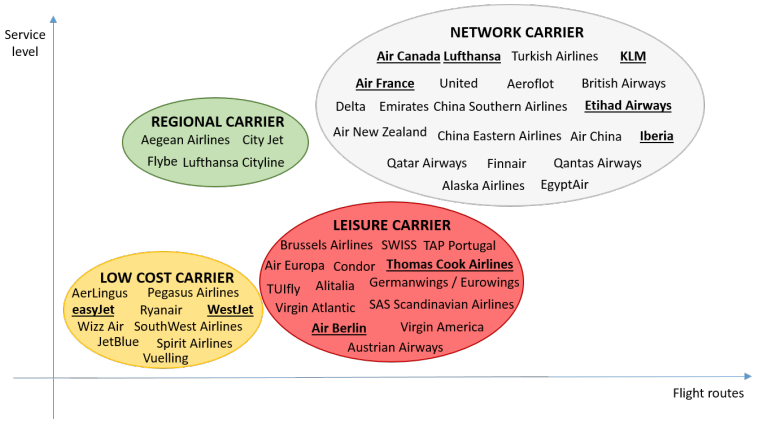


Fig. 2: Monitored airlines and their respective market segments.

cost carriers provide little service and few routes to achieve lower cost, full-service network carriers provide a dense network of flight routes with a high level of service. Leisure carriers focus mostly on popular holiday locations. Their service is often slightly better compared to low-cost carriers. Finally, regional carriers commonly operate in a geographically bounded area with few flights.

Following this classification, each of our monitored airlines can be assigned to one of the four segments. This classification allows us to compare the different strategies of each segment. An overview of all airlines and their respective segment is shown in Figure 2. The ten airlines that we focus on in this paper are marked. Six of them are full-service network carriers and two are low-cost carriers and leisure carriers, respectively.

3.4 Performance Measures

To automatically evaluate service quality, we define several measures based on the time stamps of posts. The goal is to define measures that provide a valuable overview of how fast a solution was found or offered.

As an intuitive measure, we first define **response time**. The response time in minutes can be seen as the equivalent to the AQT, which indicates how long a customer had to wait for the first response from a company. Therefore, it is defined as the delay between the publishing of a new inquiry and the response from the company:

$$\Delta(i) = t_r - t_i, \tag{1}$$

where t_i is the publishing date of the inquiry and t_r the publishing date of the first company response. The **average response time** defines the time the company needs to answer the inquires:

$$\mathrm{ar(C)} = \frac{\sum_{i \in I_C} \Delta(i)}{|I_C|} \tag{2}$$

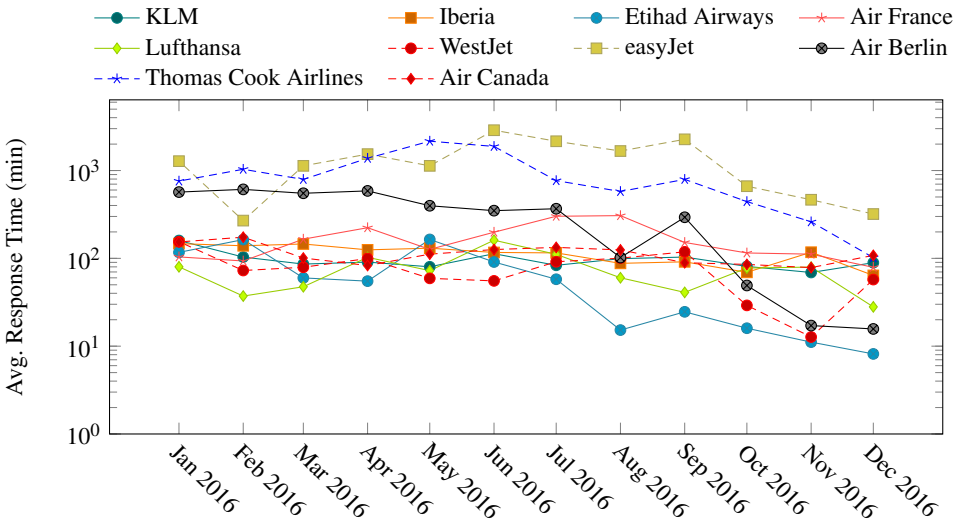


Fig. 3: Average response time of the monitored airlines.

where C is a company and I_C is the set of inquires posted to that company.

In contrast to more traditional channels, a problem with service in social media is that many inquiries are left unanswered. To evaluate this, we utilize the **response rate** as a second KPI. It indicates how many inquiries were left unanswered and is defined as the proportion of service inquiries that receive a response from the company, i.e.:

$$\text{rr(C)} = \frac{\sum_{i \in I_C} r(i)}{|I_C|} \text{ with } r(i) = \begin{cases} 1, \text{ if } \Delta(i) \text{ exists} \\ 0, \text{ else} \end{cases} \tag{3}$$

4 Customer Service Performance

We utilize the above performance indicators to calculate the response time and rate for all posts in our database in the time interval from January 2016 – November 2016. We aggregate the performance of each post to create a monthly report of the customer service performance.

Figure 3 shows the average response time for the analyzed airlines on a logarithmic scale. It becomes obvious that Etihad Airways and WestJet provide the fastest customer service and often answer within the first 15 minutes. In addition, Air Berlin, Lufthansa and KLM answer reasonably fast within less than two hours. The performance of KLM is especially noteworthy, since they receive by far the most posts on social media (cf. Table 1). We observed the worst response time for easyJet and Thomas Cook Airlines who required almost twenty hours to respond. This is surprising, since they are among the companies that received the fewest requests in our study. In addition, Air Berlin and Etihad Airways

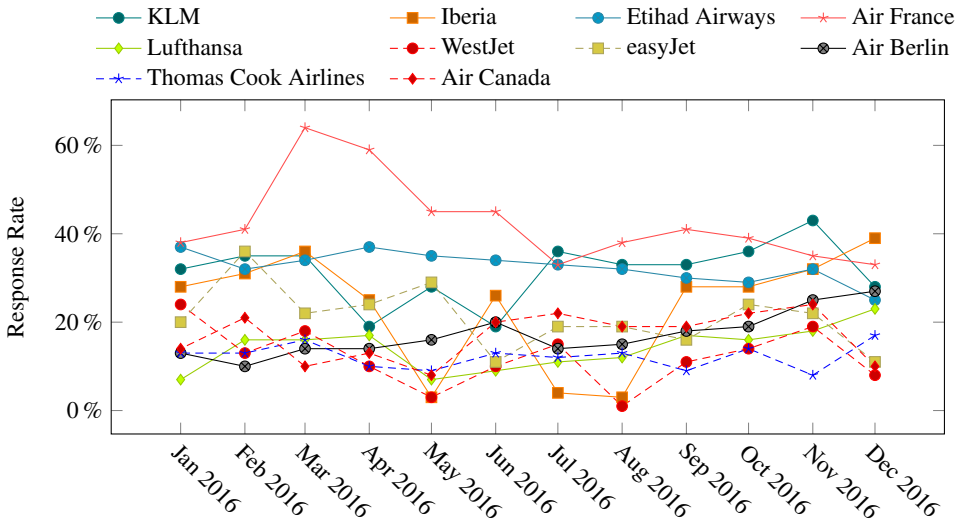


Fig. 4: Response rate of the monitored airlines.

have improved their service within our defined time interval, providing considerably faster response times in the recent months.

Similarly, the response rates of the companies are shown in Figure 4. The figure shows that Air France provides the most responses, replying to roughly 40% of all inquiries on average. In addition, Etihad Airways and KLM reply to more than 30% of inquiries. Both companies are also among the fastest to respond. The worst performance was surprisingly observed for WestJet and Lufthansa. Despite having a low response time, both companies replied to only few posts with a response rate of roughly 15%. Note that the response rate is generally lower than expected due to the fact that many posts on social media might not require a response from the company. As an example, we often observed airplane enthusiasts sharing pictures of an airplane with the company. While KLM often replied to such comments, many other airlines did not deem this necessary.

5 Conclusion

This paper has analyzed millions of customer service requests in social media and has defined several performance measures to automatically evaluate the provided customer service performance. The approach is demonstrated on the ten airlines that have received the most posts in social media since the beginning of the year. We have queried the public APIs of Twitter and Facebook to receive all interactions between the customer and the company. All posts were stored in a document-based database. The posts were then analyzed to derive key performance indicators such as the response time and response rate which were stored in a relational database and served to a web service.

Our results show that the customer service in social media is dominated by the full-service network carriers. Most notably, Etihad Airways and KLM provide fast response times and high response rates despite receiving a larger number of requests each year. This is most likely due to the fact that network carriers have the capacity and financial means to offer social media as yet another channel among many. In contrast, leisure and regional carriers were found to provide the least customer service in social media. For low-cost carriers the result is mixed. Even though WestJet provides fast responses, their overall performance is low. Similarly, easyJet is not able to provide the same service as its competitors.

Our study serves as a starting point for future analysis and can be used to compare the customer service of the airline industry. Furthermore, the same approach is equally applicable to other industries and companies. However, our results are only quantitative and do not consider the content of posts. In order to better evaluate service inquiries, text classification and topic modelling could be applied to get hold of the content of a message. This would allow a more accurate view of performance which also includes whether an issue was resolved or not. Another interesting question is to evaluate whether users with many followers, so called influencers, receive a faster response than others. Additionally, the impact of holidays, strikes, accidents or terrorism could be analyzed. Also, we consider the interplay of social media and classical channels an interesting field of research, as channel preference varies across customer segments and use cases along the customer lifecycle.

References

[Bö14] Bölke, S.: Strategic Marketing Approaches Within Airline Management: How the Passenger Market Causes the Business Concepts of Full Service Network Carriers, Low Cost Carriers, Regional Carriers and Leisure Carriers to Overlap. Anchor Academic Publishing, 2014.

[Co16] Conversocial: Conversocial Airline Benchmark Report — Going the Extra Mile on the Customer Journey, Who's winning, and who's lagging on social?, 2016, URL: http://www.conversocial.com/hubfs/Airline_Report_US.pdf, visited on: 12/18/2016.

[Fa16] Facebook Inc.: Facebook Graph API, 2016, URL: https://developers.facebook.com/docs/graph-api, visited on: 12/18/2016.

[Mo16] MongoDB Inc.: MongoDB, 2016, URL: https://www.mongodb.com/, visited on: 12/18/2016.

[Or16] Oracle Corporation: MySQL, 2016, URL: https://www.mysql.de/, visited on: 12/18/2016.

[Ta16] Talkwalker: Social Media Performance for Airlines, 2016, URL: https://www.talkwalker.com/airlines/social-media-performance-for-airlines, visited on: 12/18/2016.

[Tw16] Twitter Inc.: Twitter REST API, 2016, URL: https://dev.twitter.com/rest/public, visited on: 12/18/2016.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 41

Einsatz eines Datenstrommanagementsystems zur Empfehlung von beliebten Nachrichtenartikeln in der CLEF NewsREEL Challenge

Cornelius A. Ludmann[1]

Abstract: Im Rahmen der CLEF NewsREEL Challenge haben Teilnehmer die Möglichkeit, Recommender-Systeme im Live-Betrieb für die Empfehlung von Nachrichtenartikeln zu evaluieren und sich mit anderen Teilnehmern zu messen. Dazu werden sie durch Events über Impressions informiert und bekommen Requests, auf die sie mit Empfehlungen antworten müssen. Diese werden anschließend den Benutzern angezeigt. Die Veranstalter messen, wie viele Empfehlungen tatsächlich von den Benutzern angeklickt werden.

Eine Herausforderung ist die zeitnahe Verarbeitung der Events, um in einem festgelegten Zeitraum mit Empfehlungen antworten zu können. In diesem Beitrag stellen wir unseren Ansatz auf Basis des Datenstrommanagementsystems „Odysseus“ vor, mit dem wir durch kontinuierlich laufende Queries beliebte Nachrichtenartikel empfehlen. Mit diesem Ansatz konnten wir uns im Rahmen der CLEF NewsREEL Challenge 2016 gegenüber den anderen Teilnehmern behaupten und die meisten Klicks auf unsere Empfehlungen erzielen.

Keywords: Recommender-System, Datenstromverarbeitung, Datenstrommanagementsystem

1 Einführung und Motivation

Das Empfehlen von weiteren Nachrichtenartikeln unter Artikeln auf Nachrichtenportalen ist weit verbreitet. Dabei stellt die Domäne der Nachrichtenartikel besondere Anforderungen an die Berechnung der Empfehlungen:

- Jeden Tag kommt eine große Menge an neuen Nachrichtenartikeln hinzu.
- Viele Nachrichtenartikel sind nur für eine kurze Zeit von Interesse. Insbesondere die Artikel, die sich auf ein aktuelles Geschehen beziehen.
- Die Interaktionen vieler Benutzer können nicht (über einen längeren Zeitraum) einem bestimmten Benutzer zugeordnet werden (z. B. weil sie Cookies löschen oder gar nicht zulassen). Zu diesen Benutzer lässt sich kein Benutzerprofil bilden.

Das stellt die Entwickler von Recommender-Systemen für Nachrichtenartikel vor das Problem, dass das übliche Vorgehen, ein Modell zur Berechnung von Empfehlungen anzulernen und in regelmäßigen Abständen zu aktualisieren, nicht ausreichend ist. Neue Nachrichtenartikel müssen zeitnah bei der Empfehlungsberechnung berücksichtigt werden.

[1] Universität Oldenburg, Abteilung Informationssysteme, Escherweg 2, 26121 Oldenburg, cornelius.ludmann@uni-oldenburg.de

Die Interaktionen von Benutzern bieten wertvolle Informationen, welche Nachrichtenartikel derzeit beliebt sind und müssen somit zeitnah verarbeitet werden.

Im Rahmen der CLEF NewsREEL Challenge [Ho14] sind Forscher an Recommender-Systemen dazu aufgefordert, sich diesen Herausforderungen zu stellen und ein Recommender-System unter realen Bedingungen zu evaluieren. Dazu stellen die Veranstalter der Challenge einen Datenstrom von echten Benutzerinteraktionen (Impressions) zur Verfügung. Die Teilnehmer der Challenge werden durch HTTP POST Requests über Impressions informiert. Außerdem werden sie durch HTTP POST Requests dazu aufgefordert, für einen Benutzer Empfehlungen zu berechnen, die anschließend unter dem Artikel angezeigt werden, die der Benutzer gerade angefordert hat. Vom Veranstalter wird daraufhin die Click Through Rate (CTR; Relation der Anzahl der Klicks auf die Empfehlungen zur Anzahl der Requests) berechnet.

Die Verarbeitung der Datenströme liegt dabei in der Hand der Teilnehmer. Um den Einstieg für Teilnehmer zu erleichtern, wird die Datenrate für jeden Teilnehmer individuell erhöht, solange dieser in der Lage ist, die Daten, insbesondere die Requests, entgegenzunehmen und zu verarbeiten. Während der CLEF NewsREEL Challenge 2016 erhielten wir mit unserer Lösung eine Datenrate für den Events-Datenstrom von ungefähr 10.000 Events pro Minute und ungefähr 10.000 – 15.000 Requests am Tag. Die maximal zu erhaltene Datenrate ist durch die Veranstalter nicht veröffentlicht. Da kein anderer Teilnehmer höhere Datenraten hatte und nach unseren Aufzeichnungen unser System nicht voll ausgelastet war, ist davon auszugehen, dass wir damit die maximale Datenrate erreicht haben.

Da die Teilnehmer unterschiedliche Datenraten und somit auch eine unterschiedliche Anzahl an Requests bekamen, wurde zur Berücksichtigung bei der Challenge von den Veranstaltern gefordert, dass die Anzahl der entgegengenommenen Requests eines Recommender-Systems mindestens 75 % der entgegengenommenen Requests eines Baseline-Recommender-Systems der Veranstalter entspricht. Dies gelang für den Evaluationszeitraum (28. April bis 20. Mai 2016) 21 von 55 teilnehmenden Recommender-Systemen.

In diesem Beitrag stellen wir unseren Ansatz vor, wie wir mit dem Datenstrommanagementsystem „Odysseus“ unter dem Teamnamen *is@uniol* an dieser Challenge teilgenommen haben. Mit unserem Ansatz waren wir in der Lage, durch datengetriebene, inkrementelle Verarbeitung der Events das aktuelle Interesse der Benutzer zu erfassen und somit unter allen Teilnehmern die meisten Klicks auf Empfehlungen zu bekommen.

2 Datenstrommanagementsystem „Odysseus“

Das DSMS Odysseus [Ap12] ist ein Framework zur kontinuierlichen Verarbeitung von (potenziell) unendlichen Datenströmen durch Operatoren. Ähnlich zu Datenbankmanagementsystemen (DBMS) formulieren Benutzer Queries aus denen Anfrageausführungspläne (Query Execution Plans, QEP) erzeugt werden. Diese bestehen aus Operatoren (z. B. die Operatoren der relationalen Algebra) und Verbindungen, die den Datenfluss zwischen Operatoren festlegen. Im Gegensatz zu DBMS sind Anfragen in Odysseus keine einmaligen

und terminierenden Anfragen, sondern werden kontinuierlich ausgeführt. Die Verarbeitung erfolgt datengetrieben durch neue Datenstromelemente.

Odysseus stellt ein Framework zur Verfügung, mit dem das System mit weiteren Operatoren, Map- und Aggregationsfunktionen erweitert werden kann. Queries können durch den Benutzer mithilfe von Anfragesprachen modelliert werden (z. B. CQL [ABW06], PQL [Ap12]). Eine RCP-Anwendung erlaubt das Hinzufügen, Entfernen, Starten und Stoppen von Queries sowie die Anzeige und das Plotten von Datenstromwerten etc.

Zur Realisierung von Fenstern auf den Daten stellt Odysseus einen WINDOW-Operator zur Verfügung, der die Datenstromelemente mit rechtsoffenen Gültigkeitsintervallen (vgl. [KS09]) annotiert. Nachfolgende Operatoren berücksichtigen die Gültigkeitsintervalle. So werden beispielsweise im Aggregationsoperator die Elemente miteinander aggregiert, die überlappende Gültigkeitsintervalle haben. Durch die Annotation von Gültigkeitsintervallen ermöglicht Odysseus die Verarbeitung mit Berücksichtigung der Eventzeit anstatt der Verarbeitungszeit.

3 Berechnung von beliebten Nachrichtenartikeln durch kontinuierliche Queries

Im Rahmen der CLEF NewsREEL Challenge 2016 haben wir zunächst einen einfachen Ansatz umgesetzt, der sowohl als Baseline als auch als Grundlage für Erweiterungen und Anpassungen dient. Dieser Ansatz berechnet kontinuierlich die beliebtesten Artikel je Nachrichtenportal. Dazu aggregieren wir die Anzahl der Impressions je Artikel in einem festgelegten Zeitraum (Window) je Nachrichtenportal und antworten auf einen Request mit der Top-6-Menge der derzeit meistaufgerufenen Artikel. List. 1 zeigt die Definition der Queries in CQL.

List. 1 besteht aus zwei Teilen: der Definition der Ein- und Ausgabedatenströme (entspricht der Data Definition/DDL der SQL; Zeilen 1-5) und der Verarbeitung der Daten (entspricht der Data Manipulation/DML der SQL; Zeilen 7-22).

In den Zeilen 2 und 3 resp. 4 werden die Eingabedatenströme für die Events resp. die Requests definiert. Zeile 5 enthält die Definition des Ausgabedatenstroms, in dem für jedes Request die Empfehlungen ausgegeben werden. Mit `CREATE STREAM` bzw. `CREATE SINK` werden analog zu `CREATE TABLE` in SQL der Datenstromname sowie das Schema definiert. Außerdem wird angegeben, wie auf den Datenstrom zugegriffen wird (Datenverbindung zur Quelle etc.) und wie die Daten zur Überführung in das relationale Schema geparst werden sollen (Parsen von CSV, JSON etc.); in List. 1 aus Gründen der Übersichtlichkeit nicht angegeben. Jedes Datenelement führt zu einem Tupel mit festem Schema.

Anschließend werden die eigentlichen Queries zur Verarbeitung der Daten definiert. Die Verarbeitung haben wir in drei Queries aufgeteilt: die Vorverarbeitung der Daten (Zeilen 8-10), das Aggregieren der Anzahl der Seitenaufrufe je Artikel für jede Nachrichtenplattform (Zeilen 11-13) sowie die Aggregation einer Liste der sechs meistaufgerufenen Artikel je

```
1  CREATE STREAM events (type STRING, articleid INT, publisherid INT, userid INT,
2    userloc INT, ...) ...;
3  CREATE STREAM requests (publisherid INT, userid INT, userloc INT, ...) ...;
4  CREATE SINK recommendations (recs LIST_INT) ...;
5
6  /* Continuous Query Definition */
7  CREATE VIEW impressions FROM (
8    SELECT articleid, publisherid FROM events [SIZE 30 Minutes TIME]
9    WHERE type = "impression" AND articleid > 0);
10 CREATE VIEW counted_impressions FROM (
11   SELECT publisherid, articleid, count(*) AS counts
12   FROM impressions GROUP BY publisherid, articleid);
13 CREATE VIEW topk_sets FROM (
14   SELECT publisherid, nest(articleid) AS most_popular_articles
15   FROM counted_impressions GROUP BY publisherid ORDER BY counts DESC GROUP LIMIT 6);
16
17 /* Join of Requests and TopK Sets */
18 STREAM TO recommendations FROM (
19   SELECT topk_sets.most_popular_articles AS recs
20   FROM topk_sets, requests [SIZE 1 TIME] AS req
21   WHERE topk_sets.publisherid = req.publisherid);
```

List. 1: Queries zur Empfehlung der meistgelesenen Artikel

Nachrichtenplattform. Jede dieser Queries ist als View definiert, sodass die nachfolgenden Queries auf die Ergebnisse zugreifen können.

Die `impressions`-View filtert die Attribute Artikel-ID und Publisher-ID von allen Datenstromelementen vom Typ „impression“ heraus, die eine Artikel-ID angegeben haben. Außerdem werden die Events zur Realisierung eines 30-minütigen, gleitenden Zeitfensters mit (rechtsoffenen) Gültigkeitsintervallen annotiert. Die `counted_impressions`-View greift auf die resultierenden Events zu, um die Anzahl der Seitenaufrufe, die im selben Fenster liegen, zu aggregieren. Die Angabe von `GROUP BY` sorgt für die Partitionierung der Events für die Aggregationsfunktion `count(*)`. Jedes mal, wenn ein neues Datenstromelement eintrifft oder eines ungültig wird, ändert sich das Aggregationsergebnis und die Aggregation gibt ein neues Tupel mit dem aktualisierten Ergebnis aus. Die `topk_sets`-View nutzt anschließend die `nest`-Funktion, um für jedes Nachrichtenportal die sechs Artikel zu einer geordneten Menge zu aggregieren, die die meisten Seitenaufrufe in dem Fenster haben. Dadurch werden die populärsten Nachrichtenartikel je Nachrichtenportal kontinuierlich und inkrementell aggregiert und stehen zur Empfehlung zur Verfügung.

Im letzten Teil (Zeilen 19-22 in List. 1) werden Requests und die Aggregationen (Empfehlungsmengen) aus der `topk_sets`-View per Join verbunden. Die Join-Operation verbindet nur die Datenstromelemente, die überlappende Gültigkeitsintervalle haben (und das Join-Prädikat erfüllen) und somit zum gleichen Zeitpunkt gültig sind. Die Requests sind in dieser Query genau eine Zeiteinheit gültig. Die Aggregations-Operation aus der `topk_sets`-View gibt Tupel mit Gültigkeitsintervallen aus, sodass zu jedem Zeitpunkt genau ein Ergebnistupel (je Gruppe) gültig ist. Das bedeutet, dass sich die Gültigkeitsintervalle nicht überlappen und es keine Lücken zwischen zwei aufeinanderfolgenden Intervallen gibt. Ein Ergebnistupel einer Aggregation entspricht somit der Aggregation aller Eingabetupel die im Gültigkeitsintervall des Ergebnistupel gültig sind (und somit im selben Fenster liegen). Die Join-Operation ordnet somit zu jedem Request genau eine

Empfehlungsmenge (Ergebnistupel der Aggregation aus der `topk_sets`-View) zu. Das Join-Prädikat sorgt dafür, dass zu einem Request die Empfehlungsmenge des richtigen Nachrichtenportals zugeordnet wird.

Die Verarbeitung der Queries ist aus folgenden Operatoren zusammengesetzt:

- Ein WINDOW-Operator $\omega_w(R)$ annotiert Datenstromelemente aus R mit Gültigkeitsintervallen. Um beispielsweise ein Sliding Time Window der Größe w umzusetzen, wird ein Event zum Zeitpunkt t mit einem Intervall $[t_s, t_e)$ mit $t_s = t$ und $t_e = t + w$ annotiert.

  Wir nutzen diesen Operator, um ein Sliding Time Window über die Seitenaufrufe zu realisieren und um die Gültigkeit der Requests auf eine Zeiteinheit zu setzen, damit der Join-Operator diese mit genau einem Aggregationsergebnis verbindet und anschließend verwirft.
- Eine SELECTION $\sigma_\varphi(R)$ entfernt alle Tupel $t \in R$ für die das Prädikat φ nicht gilt.

  Wir nutzen die SELECTION um die Seitenaufrufe aus den Events zu filtern.
- Eine PROJECTION $\pi_{a_1,\ldots,a_n}(R)$ schränkt die Menge der Attribute aller $t \in R$ auf die Attribute $\{a_1, \ldots, a_n\}$ ein. Eine erweiterte Version, MAP genannt, erlaubt die Nutzung von Funktionen (z. B. $\pi_{f(a_1),a_2,f(a_3,a_4),\ldots,a_n}(R)$), die auf eine oder mehrere Attribute angewendet werden.

  Wir nutzen die PROJECTION um die Tupel auf die benötigten Attribute der Seitenaufrufe zu beschränken und die Ergebnistupel mit den Empfehlungen auf das geforderte Format anzupassen.
- Eine AGGREGATION $\gamma_{G,F}(R)$ bekommt eine Menge an Tupeln $t \in R$ und gibt für jede Gruppe, definiert durch die Gruppierungsattribute $g \in G$, ein Tupel aus, das die Ergebnisse der Aggregationsfunktionen $f \in F$ enthält. Typische Aggregationsfunktionen sind SUM, COUNT, AVG, MAX und MIN. Unsere datenstrombasierte Variante aggregiert kontinuierlich und inkrementell Tupel die im selben Fenster liegen (was bedeutet, dass sie überlappende Zeitintervalle haben).

  Wir nutzen die AGGREGATION um die Anzahl der Seitenaufrufe je Nachrichtenartikel zu aggregieren und um die sechs am häufigsten aufgerufenen Artikel zu einer geordneten Menge zusammenzufügen.
- Der JOIN $R \bowtie_\theta S$ kombiniert Tupel von R und S, für die das Prädikat θ gilt. Unsere datenstrombasierte Variante verlangt außerdem, dass Tupel überlappende Gültigkeitsintervalle haben. Das Gültigkeitsintervall des Ausgabetupels ist die Schnittmenge der beiden Intervalle der Eingabetupel.

  Der JOIN-Operator ist verantwortlich für die Zusammenführung von Request und Empfehlungsmenge.

Im folgenden Abschnitt gehen wir auf die wichtigsten Teile der Query, die Berechnung des Ranking Scores (hier die Anzahl der Seitenaufrufe mit der AGGREGATION), das Erstellen der Empfehlungsmengen (AGGREGATION), sowie das Zusammenfügen von Request und Empfehlungsmenge (JOIN), näher ein. Da diese Operatoren, und somit die Qualität der Empfehlungen, stark von der Fenstergröße abhängig sind, evaluieren wir in Abschnitt 3.2 verschiedene Fenstergrößen.

3.1 Berechnung der Empfehlungsmengen

Im Gegensatz zu anderen Lösungen berechnen wir die Empfehlungsmenge kontinuierlich (datengetrieben) und inkrementell. Dazu nutzen wir in Odysseus vorhandene Aggregationsoperatoren. Gegenüber Ansätzen, die in regelmäßigen Abständen die Empfehlungsmengen aufgrund von Daten in Datenbanken neu berechnen, hat unser Ansatz den Vorteil, dass aktuelle Trends (z. B. auch Concept Drifts) sofort berücksichtigt werden können. Gegenüber Ansätzen, die bei Eintreffen eines Requests die Empfehlungsmenge berechnen, hat unser Ansatz den Vorteil, dass ein Request allein durch einen Join mit der bereits berechneten Empfehlungsmenge zeitnah beantwortet werden kann (und damit die von der Challenge geforderten max. 100 ms Latenz eingehalten werden kann).

Unser Aggregationsoperator hält im Hauptspeicher einen Zustand s für jede Gruppe. Der Zustand wird durch eine Aggregationsfunktion $\text{add}(s,e) \mapsto s$ für jedes eingehende Element e aktualisiert (analog zu einer Left-Fold-Funktion). Für die COUNT-Funktion in List. 1 ist die Aggregationsfunktion wie folgt definiert: $\text{add}(s,e) = s+1$.

Da die Aggregation inkrementell über ein Fenster ausgeführt werden soll, müssen ebenso Datenstromelemente, die ungültig werden, entfernt werden. Aus diesem Grund werden beim Eintreffen eines neuen Elements zunächst alle Elemente von Zustand s entfernt, deren t_e kleiner oder gleich dem t_s des eintreffenden Elements sind (und somit nicht im gleichen Fenster wie das eintreffende Element liegen). Dieses geschieht durch eine Funktion $\text{remove}(s,e) \mapsto s$, z. B. für COUNT: $\text{remove}(s,e) = s-1$.

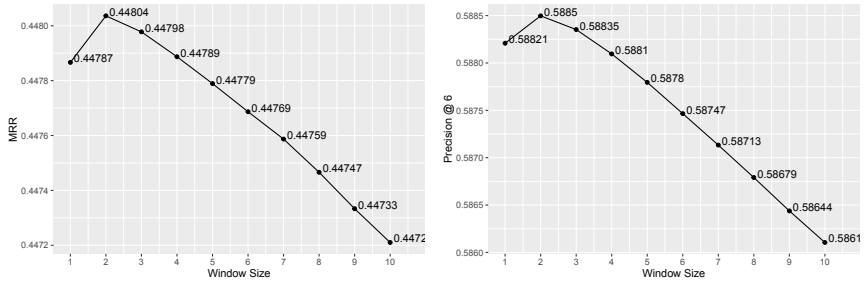
Nach jeder Änderung des Zustands ruft der Aggregationsoperator eine Funktion $\text{eval}(s) \mapsto r$ aus und gibt das Ergebnis r aus. Für COUNT ist $\text{eval}(s) = s$. Der Operator gibt somit einfach den Zustand aus. Andere Funktionen, wie z. B. AVG mit dem Zustand bestehend aus Summe und Anzahl, führen die Funktion $\text{eval}(s) = \frac{s.sum}{s.count}$ aus.

Nachdem mit der COUNT-Funktion die Anzahl der Seitenaufrufe je Artikel berechnet wurde, nutzt ein weitere Aggregationsoperator die NEST-Funktion mit $\text{add}(s,e) = s.\text{insertSorted}(e)$ (absteigend sortiert nach `count`), $\text{remove}(s,e) = s.\text{remove}(e)$ und $\text{eval}(s) = s.\text{sublist}(0,6)$. Die Ausgabe ist eine nach `count` absteigend sortierte Menge an Artikel-IDs der 6 populärsten Nachrichtenartikel.

3.2 Evaluation der Fenstergröße

Ein entscheidender Parameter in der Query ist die Fenstergröße. Da wir die *aktuelle* Anzahl der Seitenaufrufe je Nachrichtenartikel bestimmen möchten, stellt sich die Frage, wie groß das Fenster über dem Datenstrom sein muss, um die *aktuelle* Situation widerzuspiegeln. Ist das Fenster zu groß, so ist es nicht sensibel genug für aktuelle Trends. Ist es zu klein, so sind nicht genug Daten vorhanden, um populäre von unpopuläre Artikel zu unterscheiden.

Um eine angemessene Fenstergröße zu bestimmen, haben wir 21 Queries mit verschiedenen Fenstergrößen (1 bis 10 min, 20 min, 30 min, 40 min, 50 min, 60 min, 90 min, 2 hrs, 3 hrs,



(a) Mean Reciprocal Rank (b) Precision @ 6

Abb. 1: Mean Reciprocal Rank (a) und Precision@6 (b) für die Fenstergrößen 1 – 10 min und dem Nachrichtenportal mit der ID 35774.

6 hrs, 12 hrs, 24 hrs) parallel über den selben Datenstrom ausgeführt. Ähnlich zur *Interleaved Test-Then-Train* (ITTT; vgl. [Ga13]) Evaluationsmethode, haben wir den Reciprocal Rank von jedem Artikel eines Impression-Events in der Menge der 100 populärsten Artikel (entspricht die Empfehlungsmenge von 100 Artikel statt 6 Artikel bzgl. Lst. 1) berechnet, bevor dieses Event die Menge beeinflusst hat. Dies ergibt für jeden Seitenaufruf eines Artikels einen Reciprocal Rank $rr = \frac{1}{r}$ für den Rang r den der Artikel in den populärsten 100 Artikeln bzgl. der in der Query verwendeten Fenstergröße einnimmt. Je höher ein aufgerufener Artikel in der aggregierten (auf 100 Artikel erweiterten) Empfehlungsmenge steht, desto größer ist der Reciprocal Rank (max. $rr = 1$). Diesen Wert haben wir mit einem Aggregationsoperator zu einem Mean Reciprocal Rank (MRR) über mehrere Wochen aggregiert, um die beste Fenstergröße zu ermitteln. Analog dazu haben wir auch die Precision@6 berechnet. Diese gibt an, wie häufig ein aufgerufener Artikel in den Top-6 vorhanden ist.

Da die Nachrichtenportale in der Challenge unterschiedliche Datenraten haben und somit in einem Fenster unterschiedlich viele Daten zur Verfügung stehen, haben wir mithilfe der Gruppierung den MRR für jedes Nachrichtenportal getrennt berechnet.

Abbildung 1 zeigt die Ergebnisse für das Nachrichtenportal mit der ID 35774 (sport1.de). Wie man in Abbildung 1a sehen kann, wird der höchste MRR von 0,44804 für ein Fenster von 2 min. erreicht. Analog verhält sich die Precision@6 in Abbildung 1b, welcher besagt, dass knapp 59 % der aufgerufenen Artikeln ein Artikel aus den sechs am meisten aufgerufenen Artikeln der letzten 2 min. war. Sowohl größere als auch kleinere Fenster haben diese Werte verschlechtert. Dieses Muster zeigt sich auch für andere Nachrichtenportale, wenn auch für weniger frequentierte Portale ein größeres Zeitfenster von bis zu 2 Stunden zum optimalen MRR bzw. Precision@6 führte.

4 Zusammenfassung und Ausblick

In diesem Beitrag haben wir unsere Lösung für die Empfehlung von Nachrichtenartikeln im Rahmen der CLEF NewsREEL Challenge 2016 vorgestellt. Mit unserem Ansatz haben

wir die obersten Plätze in der Online-Evaluation auf echten Nachrichtenportalen belegt. Dazu haben wir das Datenstrommanagementsystem *Odysseus* eingesetzt, welches aus einer Query einen Anfrageausführungsplan aus datenstrombasierten Operatoren erstellt. Wir haben eine Basis-Query vorgestellt, welche zu jedem Zeitpunkt die Artikel empfiehlt die innerhalb der letzten w min. am meisten aufgerufen wurden. Um eine optimale Fenstergröße w zu bestimmen, haben wir 21 verschiedene Fenstergrößen (von 1 min. bis 24 Stunden) im Live-Stream evaluiert. Für ein großes Nachrichtenportal (sport1.de) ergab sich eine optimale Fenstergröße von $w = 2$ min.

In der Challenge hat sich gezeigt, dass eine effiziente Datenstromverarbeitung für Recommender-Systeme ein wichtiges Thema darstellt. Von 55 teilnehmenden Recommender-Systemen haben lediglich 21 (38 %) die Anforderung der Beantwortung der Requests innerhalb von 100 ms zufriedenstellend erfüllt. Durch eine inkrementelle Aggregation der Daten haben wir in unserer Lösung die Empfehlungsmengen kontinuierlich vorberechnet. Dadurch konnten wir ein Request durch einen einfachen Equi-Join, in dem zu jedem Request genau eine Empfehlungsmenge zugeordnet wird, unter Berücksichtigung der neusten Impression-Events, beantworten.

Die Modellierung von Queries mithilfe einer Anfragesprache hat sich zur schnellen Anpassung und Erweiterung des Recommender-Systems als nützlich herausgestellt. Dadurch war es uns ohne großen Aufwand möglich mit mehreren Recommender-Systemen an der Challenge teilzunehmen. Das Starten, Stoppen und Pausieren von Queries innerhalb von Odysseus hat zudem das Deployen neuer Queries ohne lange Ausfallzeiten ermöglicht.

In Zukunft wollen wir verstärkt verschiedene Variationen (z. B. die Auswirkungen verschiedener Gruppierungsattribute) untersuchen. Ein weiterer Aspekt für zukünftige Evaluationen sind content-basierte Methoden, die den Inhalt analysieren (z. B. um Ähnlichkeiten zwischen Artikeln einzubeziehen) sowie modellbasierte Methoden, den den Datenstrom nutzen, um ein Modell kontinuierlich und inkrementell zu aktualisieren.

Literatur

[ABW06] Arasu, Arvind; Babu, Shivnath; Widom, Jennifer: The CQL continuous query language: semantic foundations and query execution. VLDB Journal, 15(2):121–142, 2006.

[Ap12] Appelrath, H.-Jürgen; Geesen, Dennis; Grawunder, Marco; Michelsen, Timo; Nicklas, Daniela: Odysseus: A Highly Customizable Framework for Creating Efficient Event Stream Management Systems. In: DEBS'12. ACM, S. 367–368, 2012.

[Ga13] Gama, Joao; Zliobaite, Indre; Biefet, Albert; Pechenizkiy, Mykola; Bouchachia, Abdelhamid: A Survey on Concept Drift Adaptation. ACM Comp. Surveys, 1(1), 2013.

[Ho14] Hopfgartner, Frank; Kille, Benjamin; Lommatzsch, Andreas; Plumbaum, Till; Brodt, Torben; Heintz, Tobias: Benchmarking News Recommendations in a Living Lab. In: CLEF'14: Proceedings of the 5th International Conference of the CLEF Initiative. LNCS. Springer Verlag, S. 250–267, 09 2014.

[KS09] Krämer, Jürgen; Seeger, Bernhard: Semantics and implementation of continuous sliding window queries over data streams. ACM TODS'09, 34(1):4, 2009.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 49

Experiences with the Model-based Generation of Big Data Pipelines

Holger Eichelberger[1], Cui Qin[1], Klaus Schmid[1]

Abstract: Developing Big Data applications implies a lot of schematic or complex structural tasks, which can easily lead to implementation errors and incorrect analysis results. In this paper, we present a model-based approach that supports the automatic generation of code to handle these repetitive tasks, enabling data engineers to focus on the functional aspects without being distracted by technical issues. In order to identify a solution, we analyzed different Big Data stream-processing frameworks, extracted a common graph-based model for Big Data streaming applications and developed a tool to graphically design and generate such applications in a model-based fashion (in this work for Apache Storm). Here, we discuss the concepts of the approach, the tooling and, in particular, experiences with the approach based on feedback of our partners.

Keywords: Big Data, stream-processing, model-based development, code generation, Apache Storm.

1 Introduction

Developing Big Data applications is complex. Besides the inherent complexity of the various data processing and analysis algorithms comes the need to deal with the technical complexity of distributed high-performance software architectures. While modern Big Data frameworks like Apache Storm[3], Spark[4], Flink[5], or Hadoop[6], reduce this complexity, they also introduce their own level of complexity. Ideally, data scientists would be able to focus only on the functional complexity while ignoring technical complexity, e.g., how to program the links between different data processors or how to transport complex data types among them. We faced this problem directly in our collaborative projects with researchers from this area. Thus, we decided to pursue the possibility of hiding this complexity and easing the development in the area of Big Data stream-processing. This is a particularly challenging area that gets significant attention due to the ability to provide data processing capabilities at (nearly) real-time. The solution for which we opted — and which we will describe here — is to create a model-based approach that allows data engineers to model a solution on an abstract level that fits their understanding of the data processing pipeline and to relate the various steps in the pipeline to adequate implementations.

[1] University of Hildesheim, Software Systems Engineering, Marienburger Platz 1, 31141 Hildesheim, {eichelberger, qin, schmid}@sse.uni-hildesheim.de

[3] http://storm.apache.org

[4] http://spark.apache.org

[5] https://flink.apache.org/

[6] http://hadoop.apache.org

In our research, which was conducted in the FP7-project QualiMaster[7], we actually addressed an even more complex problem as we aimed at the creation of self-adaptive data processing pipelines. Thus, in this project there exist various algorithmic variants , e.g., different algorithms for calculating financial correlations in real-time [Ny16] to be executed on a heterogeneous resource pool consisting of server machines and hardware co-processors. At runtime the layout of the data processing pipeline is constantly optimized to ensure maximum performance. However, here we will focus only on the model-based development of Big Data pipelines (independent of any adaptation).

In the following section, we will discuss the foundations and the meta-model of our model-based approach. We generate the necessary code for the technical parts of the data processing architecture from the models. This will be briefly described in Section 3. Section 4 will then discuss the experiences we made with the approach and, in particular, how this approach contributed to the cooperation with the various data scientists. Finally, in Section 5 we will summarize and provide an outlook on future work.

2 Model-based Design of Big Data Applications

In this section, we detail our approach to model-based development of Big Data streaming applications. We start with an overview on concepts of stream-processing frameworks. Then, we detail the modeling concepts and, finally, the design tool that we realized to ease the creation of the models.

As a foundation for creating an application meta model for the QualiMaster project, we analyzed the data processing concepts used in more than 12 open source and research stream-processing frameworks that we considered as candidates for our infrastructure [Qu14]. During this analysis, we identified as relevant concepts for the meta model two types of stream processors and one common concept. We found approaches:

- Supporting only a **fixed set of pre-defined data stream analysis operators**. Examples are Borealis [Ab05] or PIPES [KL09].
- Focusing on **user-defined algorithms**, such as Storm, Heron [Ku15] or IBM System S [Ge08]. In particular, the algorithms themselves can be distributed over a cluster.

Some recent open source approaches such as Spark and Flink tend to combine both types. As a common concept, all stream processors we examined rely on a data flow graph (sometimes also called network of operators or query plan). A data flow graph consists of operator nodes (data sources, data processors, sometimes also data sinks) connected by data flow edges. In some approaches, like Storm, the data flow graph can be cyclic, i.e., data processors can feed back their results to their predecessors. Technically, the data flow graph is often realized in terms of a program (Spark, Storm, Heron, Flink), through a DSL, e.g., in [Ge08], or as a model [Ab05]. However, none of the approaches supports heterogeneous processing involving hardware co-processors out of the box.

[7] http://qualimaster.eu

Several model-based approaches to stream-processing applications have been developed, which are typically also based on the concept of data flow graphs. For example, Borealis [Ab05] ships with a a graphical editor and interprets the designed query model at runtime. jStorm[8] realizes a Domain-Specific Language (DSL) for Storm, which is interpreted at startup time of the application. Apache Nifi [9] enables the graphical composition of static Big Data streaming applications. Moreover, Guerriero et al. [Gu16] model and generate Big Data applications for different data processing frameworks. However, our approach differs in its specific capabilities for flexibly exchanging alternative data processing algorithms and utilizing a heterogeneous resource pool including hardware co-processors.

The requirements for designing an **application meta model** for the QualiMaster project include the support for user-defined algorithms, flexible (runtime) algorithm exchange as well as a heterogeneous resource pool. The resource pool may include commodity servers and specialized hardware co-processors such as Field Programmable Gate Arrays (FPGAs) [Ny16]. At the heart of our model is a **processing pipeline**, i.e., a data flow graph consisting of data sources, processors and data sinks. For testing, experiments and runtime adaptation, we allow the flexible exchange of the algorithms before and at runtime. Our modeling concept for enabling such exchanges is the **algorithm family**, which represents a group of algorithms performing a similar data analysis task at potentially different runtime characteristics. Thus, in our model, data processors specify the implementing family, which, in turn, defines its constituting data processing algorithms. Algorithms can be simple Java classes, co-processor algorithms, which were designed and compiled for a certain FPGA architecture or distributed algorithms represented as sub-pipelines. Similarly, data sources and data sinks specify their implementation, i.e., components that obtain data items from a data provider or pass data items to an end-user application, respectively. Moreover, the meta model allows capturing available and required processing resources. Available resources can be commodity servers and hardware co-processors. Required resources are declared by the pipeline nodes or by an algorithm if it runs only on a certain co-processor. More details on the underlying meta-modeling approach are discussed in [Ei16].

Not all possible models can be successfully executed by a stream-processing framework. To avoid illegal model instances as well as data processing problems at runtime, we equipped our meta model with a number of constraints. The constraints ensure that:

- A pipeline represents a **valid data flow graph**, i.e., there are no incoming flows to data sources and no outgoing flows from data sinks. Depending on the underlying processing frameworks, cycles in data flows can be forbidden by a further constraint.
- A **valid data flow** from node n_1 to n_2 ensures that n_2 can consume the type of data produced by n_1. Requiring this form of type safety avoids serious runtime errors. While some processing frameworks support such restrictions, e.g., Flink, others like Storm focus on processing untyped objects only.

[8] https://github.com/alibaba/jstorm
[9] https://nifi.apache.org

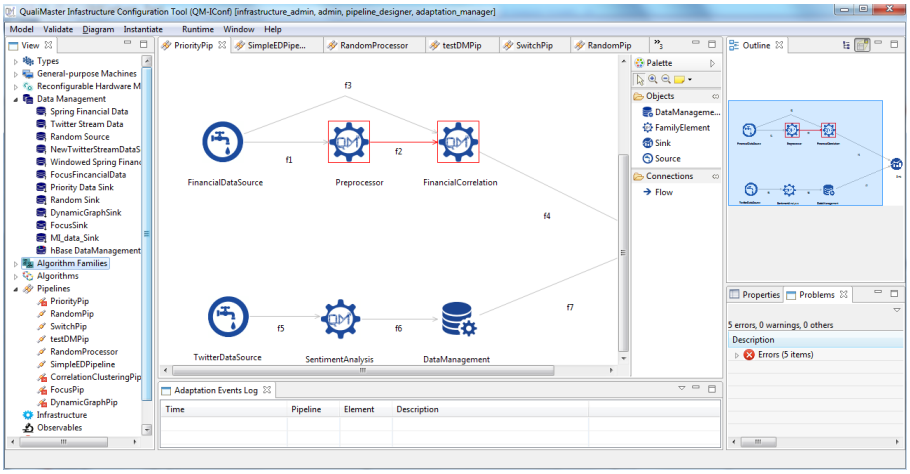


Fig. 1: Interactive pipeline design and generation tool QM-IConf indicating validation errors.

- **Interface compatibility** of exchangeable algorithms hold, i.e., algorithm families and contained members must be interface compatible regarding the consumed and produced types of data.

To ease the creation of application models, we developed a design tool, the QualiMaster Infrastructure Configuration (QM-IConf) tool. QM-IConf is an Eclipse RCP application implemented on top of an own model management infrastructure [Ei16]. Figure 1 depicts a screenshot of QM-IConf. On the left, the tool displays all model concepts ranging from the data types for pipelines over resources, algorithms, algorithm families to pipelines in the suggested sequence of defining them. For pipelines, we integrated a graphical drag-and-drop editor, which we generate using the Eugenia/Epsilon framework[10]. Figure 1 depicts a fragment of a financial risk analysis pipeline. QM-IConf allows validating whether a model fulfills all constraints using the forward-chaining reasoner implemented by our model management infrastructure. For illustration, validating a model with 9 pipelines each with 5 nodes on average takes around 150 ms on a Dell Latitude 6430 laptop. Figure 1 shows a validation error for two pipeline nodes marked by red rectangles. More detailed information on the error is available in the error view in the lower right corner.

3 Model-based Generation of Big Data Applications

In this section, we discuss the code generation for pipelines including the technical parts of the data processing architecture based on the meta-model introduced in Section 2.

Figure 2 illustrates the architecture overview of the model-based generation of data processing pipelines. We consider the execution environments including the stream-processing

[10] http://www.eclipse.org/epsilon/doc/eugenia/

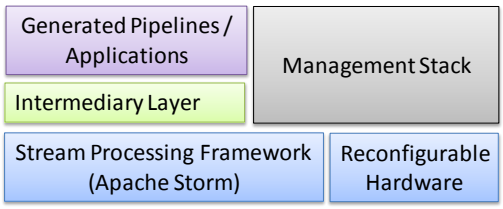


Fig. 2: Architecture overview.

Framework (Apache Storm) and reconfigurable hardware as a heterogeneous resource pool. Based on the model-based design discussed in Section 2, we generate pipelines as executable code for the underlying stream-processing framework. To structure the generated implementation as well as to simplify the generation and the generated code, we rely on an Intermediary Layer, i.e., a library with extended Storm concepts, such as a processing node which reacts on external signals. On top of the generated pipelines, we provide a Management Stack allowing us to control and monitor runtime execution through QM-IConf (see Figure 1). More specifically, we mainly perform the following generation steps:

- **Interfaces**. For processing nodes in a pipeline, we generate data source, family and data sink interfaces based on their input and output types to enforce type-safe implementation of pluggable components, in particular data processing algorithms. This allows the data engineers to design pipelines first and to implement algorithms later.
- **Data serialization support**. In a heterogeneous processing environment, data including complex nested types must be transferred among various types of processing frameworks: Storm supports default Java serialization and the faster kryo[11] framework, while co-processors are realized in different programming languages. As manually implementing serialization mechanisms is a tedious and error-prone task, we generate the required implementation of type-specific serializers based on the modeled data types, in our case kryo and protobuf[12] for the co-processors.
- **Integration of hardware co-processors**. Hardware co-processors must be fitted into the communication patterns of the processing framework. This requires network connections as well as sender/receiver functionality in terms of pipeline nodes. While the user just models the algorithm, the generation produces transparently the required communication nodes based on the aforementioned serializers.
- **Pipelines**. We traverse the modeled data flow graph of a pipeline starting at its data sources following the data flows to all subsequent pipeline nodes. For each node type, we apply a specific code generation template creating an implementation based on the the respective family interface, which can handle runtime parameter changes and algorithm exchanges. For a hardware co-processor algorithm, we integrate the code generated in the previous step. For the generation of complex algorithms given

[11] https://github.com/EsotericSoftware/kryo
[12] https://developers.google.com/protocol-buffers/

as reusable sub-pipelines, we perform a recursive pipeline instantiation. Finally, we create and execute a Maven build specification, which compiles the code, integrates all required dependencies and derives deployable pipeline artifacts.

The data engineers can execute the generation process through QM-IConf. For example, for the example model described in Section 2 the generation including compilation and download of the most recent versions of algorithm components takes around 4 minutes.

4 Experiences and Discussion

We report now on the experiences we made in applying our approach and the feedback we received from our project partners. 7 data engineers from 3 groups participated, all familiar with Storm, Hadoop, and Java. They modeled 5 different data analysis pipelines. We structure the discussion in chronological fashion, starting with the project initial phase, the introduction of the model-based approach, its application and its evolution over time.

At the **beginning of the project**, the data engineers were rather skeptical about a model-based approach as this was unfamiliar to most of them. As the model management framework already existed before the project, we were able to provide an initial version of the meta model and QM-IConf already after some months. To support our partners applying the approach, we gave some hands-on workshops **introducing the approach**, QM-IConf and how to implement the component interfaces. Here, the data engineers were puzzled about the enforced type safety as this is not required by Storm as well as the (generated) component interfaces as exchanging algorithms was not in their focus. After their first modeled pipeline was successfully generated and executed, the data engineers started to use the approach and to focus more and more on the implementation of the data analysis algorithms. In this phase we received a lot of feedback on desired features, but also on problems in QM-IConf and the generation, which helped us stabilizing the approach.

In **later phases**, the data engineers acknowledged the abstraction level provided by the models and the structuring effect of the interfaces on their work and the usefulness of typed data items. Moreover, as some data engineers never looked into the generated pipeline code and inspected how the algorithm interfaces are used, they were surprised about the ”magic” of transparent data serialization or runtime algorithm changes. This is in particular true for complex pipeline structures, e.g., co-processor integration or efficient algorithm switching [QE16], which require additional pipeline nodes and specific communication patterns. For a specific financial algorithm [Ny16], we compared two variants aiming at the same computation, a generated sub-pipeline and a manual software implementation. We found that both variants produce the same results and operate on the same cluster / distribution settings at the same performance. Thus, the model-based generation successfully shields data engineers from complex or error-prone development tasks and allows working on a higher level of abstraction, e.g., families or sub-pipelines. However, we also noticed negative effects during **integration phases**: Modeling and code generation became the center of the integration work and integrating changes in models by different partners at the same time and fixing issues in the generated code significantly increased our workload.

The realization work in the project follows agile ideas, i.e., changing and restructuring code to introduce new or advanced capabilities is not unusual. In addition to the algorithms, also the meta model, the pipeline models and the code generation undergo an **evolution**. According to our experience, model-based generation helps to quickly and consistently deploy changes and enhancements in a consistent manner across a set of pipelines. This releases the data engineers from maintaining pipeline code over time. Of course, they still have to maintain and optimize the functionality of their algorithms. However, also bugs can easily be deployed into multiple pipelines as an accident. Here, an important support mechanism is **continuous integration**. We apply continuous integration to both, the code base for the algorithm components as well as the validity of the model and the code generation using a headless version of our tool. This ensures the deployment of up-to-date pipelines and acts as an early indicator of integration problems, such as accidental changes in the models. However, intensive unit tests for some components increase the overall build time, so for identifying problems it is faster to change and modify (even the generated) code locally, to perform intensive testing and to integrate the changes back into their algorithms, the model using QM-IConf or the generation templates.

We conclude that model-based development has several positive effects on the development of Big Data streaming applications. Technical benefits include avoiding issues of manually implementing schematic code, easing and speeding up the realization of complex pipeline structures, e.g., algorithm switching or consistently changing and evolving code over time. Our approach allows data engineers to concentrate on their core competences, i.e., developing new data analysis algorithms and composing them to Big Data applications. In our case, the more the data engineers worked with the approach, the more their confidence in stability and correctness increased. However, this also requires an increasing level of quality assurance to avoid that (avoidable) issues affect the achieved trust.

5 Conclusions and Future Work

Developing Big Data applications involves complex tasks, in particular as actual processing frameworks focus on the programming of the applications. Model-based development can release the data engineers from programming tasks and allow them to focus on their specific expertise. In this paper, we presented a model-based approach to the development of Big Data streaming applications. The approach enables flexible exchange of data analysis algorithms at runtime and allows utilizing a heterogeneous resource pool. We discussed an analysis of stream-processing frameworks, our meta model for Big Data streaming applications, tool support for designing streaming applications and the generation process to obtain implementations for Apache Storm. We discussed our practical experiences with the approach and conclude that model-based design and code generation of Big Data stream applications pays off, in particular to realize schematic code or complex capabilities, which are not supported by actual stream-processing frameworks. Moreover, data engineers in our project appreciate the approach and its effects on their work. However, this does not come for free. Effort is needed for creating the models, realizing model transformations and code generation as well as maintaining and evolving the approach over time.

In the future, we aim at the generation of more optimized self-adaptive code, e.g., to increase the efficiency of switching among alternative algorithms. We also consider extending our approach to support alternative stream processors as target platform.

Acknowledgment

We thank Sascha El-Sharkawy, Roman Sizonenko, Aike Sass, Niko Nowatzki and Patrik Pastuschek for their work on QM-IConf. This work was partially supported by the QualiMaster (grant 619525) funded by the European Commission in the 7th framework programme. Any opinions expressed herein are solely by the authors and not of the EU.

References

[Ab05] Abadi, D. J.; Ahmad, Y.; Balazinska, M.; Cetintemel, U.; Cherniack, M.; Hwang, J.-H.; Lindner, W.; Maskey, A.; Rasin, A.; Ryvkina, E.; Tatbul, N.; Xing, Y.; Zdonik, S. B.: The Design of the Borealis Stream Processing Engine. In: Conference on Innovative Data Systems Research (CIDR '05). Pp. 277–289, 2005.

[Ei16] Eichelberger, H.; Qin, C.; Sizonenko, R.; Schmid, K.: Using IVML to Model the Topology of Big Data Processing Pipelines. In: International Software Product Lines Conference (SPLC '16). Pp. 204–208, 2016.

[Ge08] Gedik, B.; Andrade, H.; Wu, K.-L.; Yu, P. S.; Doo, M.: SPADE: The System S Declarative Stream Processing Engine. In: Intl. Conf. on Management of Data (SIGMOD '08). Pp. 1123–1134, 2008.

[Gu16] Guerriero, M.; Tajfar, S.; Tamburri, D. A.; Di Nitto, E.: Towards a Model-driven Design Tool for Big Data Architectures. In: Intl. Workshop on BIG Data Software Engineering (BIGDSE '16). Pp. 37–43, 2016.

[KL09] Klein, A.; Lehner, W.: Representing Data Quality in Sensor Data Streaming Environments. J. Data and Information Quality 1/2, 10:1–10:28, Sept. 2009.

[Ku15] Kulkarni, S.; Bhagat, N.; Fu, M.; Kedigehalli, V.; Kellogg, C.; Mittal, S.; Patel, J. M.; Ramasamy, K.; Taneja, S.: Twitter Heron: Stream Processing at Scale. In: Intl. Conf. on Management of Data (SIGMOD '15). Pp. 239–250, 2015.

[Ny16] Nydriotis, A.; Malakonakis, P.; Pavlakis, N.; Chrysos, G.; Ioannou, E.; Sotiriades, E.; Garofalakis, M.; Dollas, A.: Leveraging Reconfigurable Computing in Distributed Real-time Computation Systems. In: Workshop on Big Data Processing - Reloaded. http://ceur-ws.org/Vol-1558/, 2016.

[QE16] Qin, C.; Eichelberger, H.: Impact-minimizing Runtime Switching of Distributed Stream Processing Algorithms. In: EDBT/ICDS Workshop on Big Data Processing - Reloaded. http://ceur-ws.org/Vol-1558/, 2016.

[Qu14] QualiMaster consortium: Quality-aware Processing Pipeline Modeling, Deliverable D4.1, http://qualimaster.eu, 2014.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 57

Mining Industrial Logs for System Level Insights

Sebastian Czora[1], Marcel Dix[2], Hansjörg Fromm[1], Benjamin Klöpper[2], Björn Schmitz[1]

Abstract: Industrial systems are becoming more and more complex and expensive to operate. Companies are making considerable efforts to increase operational efficiency and eliminate unplanned downtime of their equipment. Condition monitoring has been applied to improve equipment availability and reliability. Most of the condition monitoring applications, however, focus on single components, not on entire systems. The objective of this research was to demonstrate that a combination of visual analytics and association rule mining can be successfully used in a condition monitoring context on system level.

Keywords: condition monitoring; predictive maintenance; log file analysis; data mining

1 Introduction

Most of the condition monitoring applications known today are focusing on single units of equipment (e.g. pumps, motors, transformers, turbines, etc.), not on entire systems (e.g. factories, sub-stations, power plants). This leaves possible interactions and cause-and-effect relationships among different units of equipment unconsidered. The objective of this research was to demonstrate that a combination of visual analytics and association rule mining can be successfully used in a condition monitoring context on system level. More precisely, the goal was to identify patterns of events (recorded in log files) that occur closely together in time or are correlated with certain types of alarms or failures.

The study was conducted for the downstream section of a petrochemical plant with data that was captured by the installed SCADA (supervisory control and data acquisition) system. The SCADA system generates a log file that reports events (alarms and warnings) originating from condition monitoring systems attached to various types of equipment (pumps, motors, handling equipment). In their monitoring function, SCADA systems tend to be event-driven rather than process-driven [GH13]. The step from continuous process data to discrete event data (called logs) reduces the amount of data communicated significantly. On the other hand, some detailed information is lost. Still, a SCADA of even a small installation can generate thousands of potentially alarming events per day [HBH12] and the data volume generated by a factory-wide SCADA system can reach an order of magnitude that easily can be considered as big data

[1] Karlsruhe Institute of Technology (KIT), Karlsruhe Service Research Institute (KSRI), Kaiserstr. 89, 76133 Karlsruhe, Germany
[2] ABB AG, Corporate Research Center, Wallstadter Str. 59, 68526 Ladenburg, Germany

[FB14,ZE11]. Monitoring these vast amounts of data exceeds human capabilities by far and calls for technologies that are nowadays called big data analytics [ZE11].

Two important requirements were formulated for this analytics solution: Firstly, there should be no extensive software development. Algorithms should be used that are available in commercial data mining tools or open source solutions with a strong development community. Secondly, the algorithms should be comprehensible ('white box' instead of 'black box') and generate results that are explainable and understandable by domain and plant experts. It will be shown in the course of this paper that these requirements could be fulfilled.

2 Related Work

Most existing condition monitoring solutions concentrate on errors of one particular equipment [Ei15]. However, there can be complex interdependencies between equipment-level faults and system-level faults [Lu09]. To improve the availability and reliability of the entire system, a system-level condition monitoring is required [Ei15]. The processing of multiple sensor data, however, quickly reaches a complexity that exceeds computational tractability. To still maintain a system-wide view, the data from multiple sources must be reduced in volume by appropriate preprocessing or filtering techniques. This is exactly done by SCADA systems which transform the raw process data into events and report these events in a logging system.

Log message contain information about the conditions under which certain events occurred. Typical attributes are: the time of occurrence, a message sequence number, the event origin (e.g. a unit of equipment), a severity code (in case of alarms), a message type and a standardized message text. A collection of log messages is called a log file. Equipment event logs have been investigated by Devaney et al. [HGH15] using natural language processing and case-based reasoning. Sipos et al. [Si14] have applied multi-instance learning to mine event logs.

Log files are not only used in industrial systems, but also in computer systems and telecommunications networks. Similar are *web logs*, which record user activities when users browse a web site. Different data mining algorithms have been used for log file analysis in these areas. Among them are clustering, association/frequent pattern mining, sequential pattern mining, multi-instance learning, naïve Bayes classifiers, hidden Markov and semi-Markov models, support vector machines, text mining, and visual analytics. The extensive literature on these methods cannot be listed in this short paper.

3 Industrial Case Study

The study was conducted for the aforementioned downstream section of a chemical plant with data that was captured by the installed SCADA system. The data consisted of log

messages (alarms and events) originating from the condition monitoring systems of the individual units of equipment. The main phases of the project that was conducted according to the CRISP-DM[3] process were *data understanding and preparation, followed by the actual data analysis*.

3.1 Data Understanding and Preparation

As already mentioned, data for this study originates from a SCADA system installed at the downstream section of a petrochemical plant. A SCADA system is a distributed hierarchical IT system connecting local control units attached to physical devices with a SCADA control center. Log entries are generated by additional servers in the network communicating over standard OPC4 protocols, the OPC Alarm and Event Server (OPC AE), the OPC Data Access Server (OPC DA), and a Data Concentrator.

Log entries contain the time of occurrence (Timestamp), a message sequence number (MessageId), the event origin (DeviceId), a severity code (Severity), a message type (StatementId) and a corresponding message text (Statement). Every time an equipment condition changes, the OPC server records an event and sends it to the central controller of the SCADA system. Warning and failure alarms are events of a higher severity code (i.e. Severity = 500, Severity = 1000) than simple condition changes.

We used an iterative approach in order to develop an understanding of this data. Several workshops were conducted with industrial domain experts, who ultimately work with the analysis results. Besides talking to the industrial domain experts, a visual, interactive data exploration tool that was developed based on R's Shiny web application framework5 was extremely helpful. Its primary objective was to visualize event patterns and to develop a deeper understanding of relationships that might exist in the data set. Data quality was assessed by checking missing values, plausibility of data, and inconsistencies between the fields in the log file. Identified quality problems resulted in adjustments in the real system to improve data collection in the future. Since the analysis was based on data that had been collected over a long timeframe in the past, the existing data had to be corrected and improved as best as possible for further processing.

3.2 Data Analysis

As already mentioned, the objective of this research project was to find patterns of events that occur closely together. This co-occurrence of events is a problem that was originally investigated in *market basket analysis* and is known as *pattern mining*, *frequent itemset mining* or *association analysis*. A number of algorithms have been developed over time to address the problem of association rule mining. Among them are

[3] CRISP-DM = Cross Industry Standard Process for Data Mining
[4] OLE (Object Linking and Embedding) for Process Control (OPC), https://opcfoundation.org/about/what-is-opc/
[5] https://cran.r-project.org/package=shiny

Aprior, *Eclat* and *FP-growth* [HKP11].

In our context, transactions correspond to buckets of 100 events that precede a warning alarm, and groups of items correspond to groups of events. Let A and B be nonempty, disjoint subsets of an overall set of possible events I. Then, an association rule is an implication of the form

$$A \Longrightarrow B$$

where A is called the *antecedent*, and B is called the *consequent* of the rule. This means: if a group of events A is occurring in a certain time window (e.g. a number of alarms), then a group of events B (e.g. a failure) is likely to occur in the same time window. *Support* and *confidence* are measures to describe the significance of the association rule:

$$A \Rightarrow B\ [support = 2\%, confidence = 60\%]$$

A support of 2% means that 2% of all transactions (observed time intervals) show that the events A and events B are occurring together. A confidence of 60% means that 60% of all intervals which contain events A also contain events B. Formally, let $P(A)$ be the probability that a transaction contains the set of events A, and, accordingly, $P(A \cup B)$ be the probability that a transaction contains the *union* of the sets A and B, or *both* A and B. Then $P(A \cup B)$ is called the $support$ of the rule $(A \Rightarrow B)$

$$support(A \Rightarrow B) = P(A \cup B)$$

and $P(B \mid A)$ is called the $confidence$ of the rule $(A \Rightarrow B)$

$$confidence(A \Rightarrow B) = P(B \mid A)$$

If *support (A)* is greater than a predefined support threshold, then A is called a *frequent itemset*. In market basket analysis, the focus is on identifying co-occurring *frequent* itemsets. In our context, the focus lies on co-occurring *rare* patterns - patterns that occur infrequently (i.e. with low support) but are of critical importance. Such events are warnings, failures or other deviations from normal operating conditions which are expected to occur infrequently. *Rare pattern mining* has also become important in other areas such as fraud and intrusion detection. A review of the field is given by Koh and Ravana [KR16].

Classical association mining algorithms generate rules that exceed a minimum support threshold (*minsup*), i.e. they are designed for discovering frequent patterns. For rare patterns, the minsup threshold has to be set very low (in our case, minsup < 1%). As a consequence, the algorithms generate a very high number of rules, and not all of these rules are interesting [KR16]. To identify relevant rules and filter out irrelevant ones, additional measures of interestingness are required that reveal more intrinsic pattern characteristics than only support and confidence. Such measures are *lift*, the *Kulczynski measure*, and the *imbalance ratio*. Wu et al. [WCH10] compare seven measures of interestingness and come to the conclusion that the *Kulczynski* measure [Ku27] in conjunction with the *imbalance ratio* is a very promising combination. Both measures

are null-invariant. *Kulczynski* proposed a measure based on the average of the two conditional probabilities

$$Kulc(A,B) = \frac{1}{2}\Big(P\big(A \,\big|\, B\big) + P\big(B \,\big|\, A\big)\Big)$$
$$= \frac{1}{2}\big(confidence(B \Rightarrow A) + confidence(A \Rightarrow B)\big)$$

A *Kulczynski* value near 0 or 1 indicates that the patterns A and B are either negatively or positively correlated and we have an interesting rule. If the value is 0.5, there is no such indication. For this case, the *imbalance ratio* [WCH10]

$$IR(A,B) = \frac{\big|\, support(A) - support(B) \,\big|}{support(A) + support(B) + support(A \cup B)}$$

provides more insight. It assesses the imbalance of two patterns A and B where 0 is perfectly balanced and 1 is very skewed. If *Kulc(A,B) = 0.5* and *IR(A,B) = 0*, the rule is completely uninteresting. If *Kulc(A,B) = 0.5* and *IR(A,B) = 1*, the rule might be worth looking at.

By applying measures like the Kulczynsi measure or the imbalance ratio, additional concepts such as *multilevel mining* and *constraint-based mining* can be used [HKP11]. Since we were interested in using easy-to-understand algorithms, we focused on the well-known, standard pattern mining algorithms available in commercial or well-supported open source packages and selected an algorithm that has proven to perform well with low support thresholds. Based on an evaluation by Saabith et al. [SSB15], FP-growth was chosen.

3.3 Big Data Impementation

FP-growth typically performs well on large datasets if the minsup parameter is set high enough to prune the combinatorial search space. The smaller minsup is chosen - and this is necessary for discovering rare patterns such as failures in industrial plants -, the larger the search space and the longer the algorithm's runtime and memory requirements. Experiments demonstrated that the large search space did not fit entirely into the memory of a single computer. This calls for new technical solutions that cope with the complexity of the problem. These solutions are based on massively parallel architectures and are known as *big data technologies* [FB14] [Ma11].

We chose IBM's InfoSphere BigInsights [ZE11], which is based on the distributed big data processing environment Apache Hadoop. IBM's InfoSphere BigInsights allows the integration of open source analytics packages and we decided to use Mahout's parallel FP-growth algorithm[6] to take full advantage of the distributed hardware architecture. As an extension of Mahout's FP-growth implementation, we implemented more

[6] https://github.com/apache/mahout/tree/mahout-0.8/core/src/main/java/org/apache/mahout/fpm/pfpgrowth

sophisticated pattern evaluation measures. The implementation and the experiments were conducted at the Smart Data Innovation Lab (SDIL).[7] The hardware architecture used in this project consists of 6 IBM Power S822 full rack nodes, each with 20 cores, 512 GB RAM, 260 hard disk drives with over 300 TB storage capacity and a 40 GB/s network connection. For the implementation and the experiments, Hadoop version 2.7, version 3 of IBM's InfoSphere BigInsights, and version 0.8 of the Mahout library were available.

The automated log data transformation and the parallel FP-growth algorithm as well as the pattern analysis were embedded into an overall workflow with MapReduce jobs for data transformation, the parallel FP growth, and the calculation of interestingness measures and the validation and interpretation of results. We implemented configurable MapReduce jobs that automatically create transactions (baskets) for the FP-growth algorithm. These MapReduce jobs implement the logic how the baskets should be constructed from the logs that should be mined. Here, we decided to split the log file every time a warning alarm occurs. Furthermore, each transaction consists of 100 events that occurred before the warning alarm was registered in the log.

3.4 Results

The result of the analysis phase is a collection of association rules together with their corresponding measures of rule interestingness. A short excerpt of this rule collection is shown in Table 1. Every row in this table represents an association rule. Thus, a row can be interpreted as a set of events that occur together in the analyzed log file. The first column contains the *RuleId*, the second column the consequent of the association rule and the further columns the association rule's antecedent set. Additionally, the last columns represent the measures of rule interestingness *support*, *confidence*, *Kulczynski* and *imbalance ratio*.

The association rules shown in Table 1 describe co-occurrence of failures across different units of equipment like pumps, fans and lifts. The *Kulczynski* value for the respective association rules varies in a range between 0.75 and 1. Therefore, the *imbalance ratio* does not need to be considered for further interpretation and the association rules are interesting. The rules with *RuleIds* 69, 85, and 195 describe situations where the data concentrator lost its connection to the central controller. Consequently, each of the equipment units connected to the respective data concentrator also lost their connection to the central controller. This is a situation that was already observed in the preprocessing phase with the visual data exploration tool.

Additionally, further interesting association rules were found. For example, a rule representing a situation in which the communication between a motor starter power module and the control unit of the motor starter is disrupted. Further analysis of the alarms in the antecedent set disclosed that the co-occurrence of the respective alarms reflect the process of intentionally disconnecting equipment from the condition

[7] Smart Data Innovation Lab: http://www.sdil.de/en

monitoring system. Another interesting association rule describes a situation in which equipment exceeded the predefined failure level of one of its operational parameters due to a specific handling of the equipment.

The discussion with industrial domain experts confirmed that the above mentioned association rules are understandable and explainable and denote either undesired communication problems, problems in setting up a manufacturing facility, or actual operational issues in the manufacturing systems. Thus, they are valid and interesting from an engineering point of view and can be used to improve plant operations. Limitations of association rule mining have been demonstrated by comparing patterns that are identified by the visual data exploration tool with the patterns identified by the parallel FP-growth algorithm. To sum up, the study has shown that visual analytics in combination with association rule mining are suitable methods for condition monitoring on the system level.

Rule ID	Rule	S	C	K	IR
69	Fan ID A67 is damaged or missing & Material Life AB 69 and Oil pump is damaged or missing => Concentrator ID 1 is missing	0.31	1	1	0
85	Immersion Pump P1015 is damaged or missing & Fan ID A67 & Hydrauliuc Oil pump is missing => Concentrator ID 2 is missing	0.17	0.75	0.75	0
195	Fan ID A89 is damaged & Fan ID A89 is damaged & Lift Workshop is damaged or missing => Concentrator ID 21 is missing	0.31	1	1	0

Tab. 1: Excerpt of the association rules together with their measures of interestingness S=support, C=confidence, K=Kulczyinski measure, IR=imbalance ratio

4 Conclusion

In the real-world industrial environment described in the beginning, we were able to demonstrate that a system-level analysis of the log files generated by a SCADA system is achievable with appropriate algorithms and technology. For the detection of rare events with low support and confidence, association rule mining requires new technical approaches. A pattern mining algorithm could successfully be implemented on a scalable big data architecture. Not surprisingly, good data quality is one of the key requirements in achieving meaningful results.

References

[Ei15] Eickmeyer, J.; Li, P.; Givehchi, O.; Pethig, F.; Niggemann, O.: Data driven modeling for system-level condition monitoring on wind power plants. In: 26th International

Workshop on Principles of Diagnosis (DX 2015), 2015.

[FB14] Fromm, H.; Bloehdorn, S.: Big Data – Technologies and Potential, In: Enterprise-Integration, ed. by Schuh, G.; Stich, V., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, chap. 9, pp. 107–124.

[GH13] Galloway, B.; Hancke, G. P.: Introduction to industrial control networks, Communications Surveys & Tutorials, IEEE 15/2, pp. 860–880, 2013.

[HBH12] Hadziosmanović, D.; Bolzoni, D.; Hartel, P. H.: A log mining approach for process monitoring in SCADA, International Journal of Information Security 11/4, pp. 231–251, 2012.

[HKP11] Han, J.; Kamber, M.; Pei, J.: Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

[HN15] Herzig, K.; Nagappan, N.: Empirically detecting false test alarms using association rules. In: Proceedings of the 37th International Conference on Software Engineering-Volume 2, IEEE Press, pp. 39–48, 2015.

[KR16] Koh, Y. S.; Ravana, S. D.: Unsupervised Rare Pattern Mining: A Survey, ACM Transactions on Knowledge Discovery from Data (TKDD) 10/4, 45:1–45:29, 2016.

[Ku27] Kulczynski, S.: Die Pflanzenassoziationen der Pieninen, Bulletin de l'Academie polonaise des sciences. Serie des sciences biologiques, pp. 57–203, 1927.

[Lu09] Lu, B.; Li, Y.; Wu, X.; Yang, Z.: A review of recent advances in wind turbine condition monitoring and fault diagnosis. In: 2009 IEEE Power Electronics and Machines in Wind Applications, IEEE, Lincoln, NE, pp. 1–7, 2009.

[Ma11] Manyika, J.; Chui, M.; Brown, B.; Bughin, J.; Dobbs, R.; Roxburgh, C.; Byers, A. H.: Big data: The next frontier for innovation, competition, and productivity, 2011.

[Si14] Sipos, R.; Fradkin, D.; Moerchen, F.; Wang, Z.: Log-based predictive maintenance. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining – KDD '14, ACM Press, New York, NY, USA, pp. 1867–1876, 2014.

[SSB15] Saabith, A. L. S.; Sundararajan, E.; Bakar, A. A.: Comparative Analysis of Different Versions of Association Rule Mining Algorithm on AWS-EC2. In: International Visual Informatics Conference. 2015, Springer International Publishing, 2015.

[WCH10] Wu, T.; Chen, Y.; Han, J.: Re-examination of interestingness measures in pattern mining: a unified framework, Data Mining and Knowledge Discovery 21/3, pp. 371–397, 2010.

[ZE11] Zikopoulos, P.; Eaton, C.: Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media, 2011.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 65

Post-Debugging in Large Scale Big Data Analytic Systems

Eduard Bergen[1] Stefan Edlich[2]

Abstract: Data scientists often need to fine tune and resubmit their jobs when processing a large quantity of data in big clusters because of a failed behavior of currently executed jobs. Consequently, data scientists also need to filter, combine, and correlate large data sets. Hence, debugging a job locally helps data scientists to figure out the root cause and increases efficiency while simplifying the working process. Discovering the root cause of failures in distributed systems involve a different kind of information such as the operating system type, executed system applications, the execution state, and environment variables. In general, log files contain this type of information in a cryptic and large structure. Data scientists need to analyze all related log files to get more insights about the failure and this is cumbersome and slow. Another possibility is to use our reference architecture. We extract remote data and replay the extraction on the developer's local debugging environment.

Keywords: Software debugging, Bug detection, localization and diagnosis, Java Virtual Machine, JVMTI, Bytecode instrumentation, Apache Flink, Application-level failures

1 Motivation

Fast and responsive data processing in the era of Big Data separated by the four dimensions, volume, variety, velocity and veracity lead to a high complexity of data management. Complex data management systems offer interfaces to get information about the current system state, but there is a gap of subset observation in an executed large scale big data analytic system (LSA).

When starting an investigation process to find the root cause of a failure, data scientists not only need to investigate a big amount of diverse log-data. Data scientists also need detailed knowledge about the executed processing engine such as Apache Flink to find out where to start the search.

In our application, we are interested in the identification of failure patterns in the data that user defined functions (UDFs) process. Within the identified failure pattern we observed, the subset of the execution graph is of interest. The observation methodology allows doing an extraction of the required information and application values to do a replay. We assume most failure patterns at the runtime layer in the analysis process. Failures, which bring the whole system down such as kernel crashes from the operating system, are not in the scope of this work. Since a kernel crash is not safe and a crashed application cannot detect its own crash without additional and complex tooling.

[1] Department of Computer Science and Media, Beuth University of Applied Science Berlin, Germany, Luxemburger Str. 10, 13353 Berlin, eduard.bergen@beuth-hochschule.de
[2] Department of Computer Science and Media, Beuth University of Applied Science Berlin, Germany, Luxemburger Str. 10, 13353 Berlin, stefan.edlich@beuth-hochschule.de

Currently, the system works as a prototype in a holistic system. The observation, extraction, and replay need multiple steps. In this paper, we pursue the observation of a subset in an LSA such as Apache Flink and emphasize obliviate techniques for a full automated observation process. Although we integrated our prototype in Apache Flink, not all components are optimal and efficient. Hence, response time and performance are not in the focus of this work. The result of our work is a system that replays certain input records on the developer's local machine. This allows a much faster investigation of the big data analytic job and brings us a step forward to find possible execution errors i.e. exceptions in minimal time.

This paper has following structure. First, we briefly present some background regarding observation in Section 2, classify the runtime environment and failure types, address related technologies, and describe their deficiencies. Section 3 is the main part and introduces our approach by filling the gap through isolated local debugging. Within Section 4 we look into our reference architecture, analyze behavior and show a use case. Finally, we conclude our work.

2 Background and Related Work

In recent years more and more LSAs such as Apache Flink [AF16] and Apache Spark [AS16] address fast parallel data processing using distributed system concepts. In distributed systems capabilities to digest and interpret the root cause of an error fail because they are too complex and large to handle failure analysis in production. Additionally, complex error investigation techniques like traces suffer from getting in touch with the faulty environment. Common approaches to discover a failure is through monitoring, remote debugging, data provenance [CAA11] or tracing the lineage [CWW00]. LSAs with limited control over the execution of a process and manipulation of data make it difficult to discover runtime errors. Runtime errors result from code that is syntactically correct but violates the semantics of a programming language.

```
public class SubText {
public static void main(String[] args) {
String printText = "SubText";
System.out.println(printText.substring(7,4));
}
}
```

List. 1: Source code sample *printText* with exceptional part inside the substring method

Typically, LSAs are complex unit interaction environments where messages need to exchange between the units correct, deterministic and fault-tolerant. Inside units, the environmental messages interact through interfaces. The model of an implemented interface contains consistent entities. Every message interacts with a fixed protocol and connected ports also known as channels. Independent failures do occur in units or channels. The challenge is to detect these failures. At the application level, a failure category consists of applicational failure models (AFM) and functional failure models (FFM).

Listing 1 represents a Java source code sample and shows a common scenario for a usual runtime error. For simplicity reasons, we choose this example. In the fourth line of Listing 1 the value of variable *printText* will produce a Java exception of type *StringIndexOutOf-BoundsException* because the value length of variable *printText* is seven and not eleven. The method *substring* tries to access a range of chars between char seven and eleven. The result is an abnormal program execution.

A program break inside LSAs is more than a classical abnormal program execution. Mostly it is time-consuming to finish a broken job if a program breaks in a big cluster setup, because of added additional communication overhead to a job. The common scenario is to identify parts of the cluster as vulnerable manually or semi-automated and wait until these parts restarted to a clean state. Other strategies are defining a model for the provenance of a data item or querying [Vi10].

After introducing some preliminaries, we will briefly recall known methodologies in debugging of distributed systems, specialized on record and replay (RnR). While there are many different RnR systems, proposes because of varied applications such as program debugging, online program analysis, fault tolerance, performance prediction and intrusion analysis probably the most important application is program debugging. The most common operation of debugging is bug reproduction, diagnosis and fixing. Program debugging consists of several imperative steps such as executing the job multiple times, pausing and investigating the state of variables and tracing [HT14].

It is also challenging to debug distributed software which often has complex structures [Wi12]. The reproduction of identical executed jobs requires the same runtime. LSAs like Apache Flink and Apache Spark have multiple signal controls, flows, application state, intermediate results and shared resources. For reproduction purpose the replication of an erroneous behavior of the used LSA [Fe15] becomes very hard because of first a different ordering of data tuples, second the execution resources and third the runtime environment.

There are plenty of tools for debugging distributed systems such as Inspector gadget [OR11] and Arthur [An13], Spark-BDD [Co15], Daphne [JYB11], and NEWT [LDY13]. These tools are specific and run inside the same process as the application itself. Normally the application uses the maximum level of the Java Virtual Machine (JVM) heap. Thus, it is not possible to attach remote debuggers. Furthermore the debugger is using an expensive heap space during the runtime, which often conflicts with the application heap space.

The more critical issue is the garbage collector itself. If the debugger runs in the same JVM and owing to the fact that a garbage collector is working during a Java program termination, in that case the debugger will also terminate. No further control is possible as to introduce a new active polling process. This leads to a bad design where further issues occur as a high CPU usage. Apart from internal observation, an external observation as descriptive debugging information used in DTrace [CSL04] needs detailed knowledge about the application runtime.

DiSL [Ma13] introduces the investigation process on the external and internal level. Basically DiSL uses a Java Virtual Machine Tool Interface (JVMTI) [OC16] implementation

of bytecode instrumentation. Furthermore the DiSL Java framework does bytecode transformation. In DiSL each compilation step sends and receives bytecode data between the native agent and the DiSL framework through a Transmission Control Protocol (TCP) socket connection. DiSL uses the combination of internal and external observation and decouples the bytecode transformation to a separate process. The communication is mainly socket based. If the compiler does not return the correct bytecode, the instrumentation of bytecode is a bottleneck. In general the result of the integrated faulty bytecode leads to an unpredictable JVM process termination.

If there is a need to get the fastest instrumentation of code transformation, implemented prototypes should be able to manage and solve any locking techniques during the instrumentation phase. Thus, the post debugging approach needs to process deterministic in the external observation way.

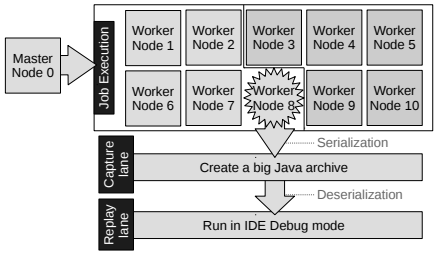


Fig. 1: An overview of our environment extraction and replay methodology

3 Reference Architecture

In this Section, we present the reference architecture according to the requirements described in Section 2. Figure 1 shows the architecture of the prototype. Our implementation of the reference architecture uses open source technologies. There is a plan to open source the reference architecture in several steps.

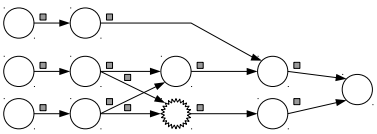


Fig. 2: Example of a job graph in Apache Flink

As depicted in Figure 1 the job fails in worker node number eight. Furthermore Figure 1 shows our reference architecture which consists of two parts: the *capture lane* and the *replay lane*. The *capture lane* allows users to get a subset of a defined workflow of UDFs using not only second order functions of operators such as map and reduce. During the *replay lane* a process in the reference architecture translates the recorded subset.

In Apache Flink there is a master and multiple worker nodes [Ba10]. The Apache Flink *JobManager* is the master node and the *TaskManagers* are the worker. In order to accomplish a record and replay workflow, the replay process requires complete information about the

executed job. First a data scientist ensures that a native agent attaches to a specific worker node. Every time a failure occurs in the attached mode within the cluster, a native agent creates a big Java archive on the filesystem.

This paper primarily focuses on our novel process for subset extraction and replay. Although other systems introspect extractions in the form of a whole job graph, our approach goes a step further by determining low-level characteristics to generate a small environment for later analysis. Additionally, systems exploring and tracing code via remote connections either suffer from getting the right place of a tuple value or need to generate source code to connect with precompiled operators.

While the investigated application handles specific application errors by itself, with small pieces of code we extract and serialize also the environment, additional runtime values, such as current input data of used operator. A native agent appends the recorded information about the environment to the Java archive file during the exception handling in the Java process.

Figure 3 shows the workflow of extraction and replay. Inside the web interface, archived jobs also contain the corresponding subset of the failed job. Hence, users are able to download the created archived jar-File for further local inspection.

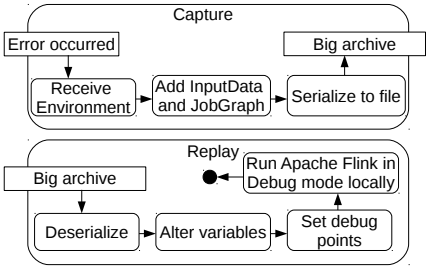


Fig. 3: An overview of extraction and replay activity in Apache Flink

The replay component takes in Figure 3 the archive, extracts the subset and starts a translation process in a local development environment. The translation process consists of three steps. First, the native agent serializes needed variables. Then updates of the configuration follow and at the end there is a need to use specifically overridden classes to inflate current input data for the used operator.

Before the replay process starts, data scientists execute run configurations on a homogeneous and small local machine. After the setup of workflow, users set debug points in specific lines in the source code. Since the current job executes, the attached debugger stops at user defined debug points. Data scientists usually use in debug session an Integrated Development Environment (IDE) to step through the call graph and observe the values of variables. Now that we have introduced our reference architecture, the following Section gives an impact about the usage and behavior of our reference architecture.

4 Experiences

In a RnR system such as our reference architecture that offers comprehensive code coverage, the challenge for data scientists quickly becomes figuring out what not to track. The main technique for filtering noisy data objects is to define transformation classes. In the record component as depicted in Figure 3, an implementation of a JVMTI native agent offers such a possibility.

During the attachment phase at the JVM, the native agent activates common capabilities to start receiving events such as class calls in a bytecode structure. Each time the JVM invokes a callback with the name *ClassFileLoadHook* the native agent executes a registered method. Accordingly, the callback method initiates a class file transformation and basically exchanges current bytecode with the desired bytecode of the transformation class.

In order to enable the class transformation, the native agent requires the desired transformation classes. Thus, the deployment of the native agent bundles compiled Java classes. We have used our reference architecture extensively to understand system behavior mainly in development environments. One issue [AFI14] in which our reference architecture could be useful is a long running Apache Flink job fail because of an *IndexOutOfBoundsException*. We have recreated this exception in a simple case as shown in Listing 1. Instead of using a constant value as in Listing 1, we used to input data from the production environment.

Figure 4 shows the replayed Java stack trace and expresses the proof of a working implementation of our reference architecture. Thus, data scientists begin root-causing as depicted in Figure 4 at line 94 of the WordCount-class, step-through the call graph until DataSourceTask-class and try to analyze why the program breaks in the invoke-method locally.

```
/usr/lib/jvm/java-7-oracle/bin/java ...
java.lang.StringIndexOutOfBoundsException: String index out of range: -4
    at java.lang.String.substring(String.java:1911)
    at org.apache.flink.client.testjar.WordCount$Tokenizer.flatMap(WordCount.java:94)
    at org.apache.flink.client.testjar.WordCount$Tokenizer.flatMap(WordCount.java:82)
    at org.apache.flink.runtime.operators.chaining.ChainedFlatMapDriver.collect(ChainedFlatMapDriver.java:79)
    at org.apache.flink.runtime.operators.DataSourceTask.invoke(DataSourceTask.java:180)
```

Fig. 4: A Java stack trace after replay with our reference architecture in Apache Flink

With the subset record and replay reference architecture, we are able to debug in a local environment and find the root cause. Compared to the time it takes to reproduce the failure in issue [AFI14], with our reference architecture, data scientists and developers are able to focus more on finding the root cause and solving the problem.

5 Conclusion

We have described our reference architecture, a new facility for debugging of LSAs through dynamic bytecode instrumentation. We have described and shown the primary features of our reference architecture, including details of record process and entry parts for further development. Also, we have demonstrated the use in root-causing a given problem. Although it is hard to design a single general approach of value extraction that replays all runtime

failure types immediately, it is still feasible to design application-oriented schemes for specific application scenarios.

6 Future Work

Our reference architecture provides a reliable and extensible foundation for further work to enhance our possibility to observe and post debug. We actively extend and update our reference architecture. Further development of components focuses first on automation in deployment and managing of agents, second in a plug-in system to address different observed program versions, and third in the visualization of subset reduced call graphs.

7 Acknowledgement

This work is generously funded by the Federal Ministry of Education and Research under the reference number 01IS14013D and was created as a part of the BBDC.berlin project.

References

[AF16] Apache Software Foundation: Flink Fast and reliable large-scale data processing engine, 2016, URL: http://flink.apache.org, visited on: 03/15/2016.

[AFI14] Apache Software Foundation: ASF JIRA: [FLINK-1000] Job fails because an IndexOutOfBoundsException, 2014, URL: https://issues.apache.org/jira/browse/FLINK-1000, visited on: 03/15/2016.

[An13] Ankur, D.; Zaharia, M.; Shenker, S.; Stoica, I.: Arthur: Rich Post-Facto Debugging for Production Analytics Applications, 2013, URL: http://ankurdave.com/dl/arthur-atc13.pdf, visited on: 03/15/2016.

[AS16] Apache Software Foundation: Spark Lightning-fast cluster computing, 2016, URL: http://spark.apache.org, visited on: 03/15/2016.

[Ba10] Battré, D.; Ewen, S.; Hueske, F.; Kao, O.; Markl, V.; Warneke, D.: Nephele/-PACTs: A Programming Model and Execution Framework for Web-scale Analytical Processing. In: Proceedings of the 1st ACM Symposium on Cloud Computing. SoCC '10, ACM, Indianapolis, Indiana, USA, pp. 119–130, 2010, ISBN: 978-1-4503-0036-0.

[CAA11] Cheney, J.; Ahmed, A.; Acar, U. a.: Provenance As Dependency Analysis. Mathematical. Structures in Comp. Sci. 21/6, pp. 1301–1337, 2011, ISSN: 0960-1295.

[Co15] Condie, T.; Gulzar, M. A.; Interlandi, M.; Kim, M.; Millstein, T.; Tetali, S.; Yoo, S.: Spark-BDD: Debugging Big Data Applications. In: the 16th International Workshop on High Performance Transaction Systems (HPTS). 2015.

[CSL04] Cantrill, B. M.; Shapiro, M. W.; Leventhal, A. H.: Dynamic Instrumentation of Production Systems. In: Proceedings of the Annual Conference on USENIX Annual Technical Conference. ATEC '04, USENIX Association, Boston, MA, pp. 2–2, 2004.

[CWW00] Cui, Y.; Widom, J.; Wiener, J. L.: Tracing the Lineage of View Data in a Warehousing Environment. ACM Transactions on Database Systems 25/2, pp. 179–227, 2000, ISSN: 0362-5915.

[Fe15] Ferber, M.: FerbJmon Tools - Visualizing Thread Access on Java Objects using Lightweight Runtime Monitoring. In: Euro-Par 2015: Parallel Processing Workshops: Euro-Par 2015 International Workshops, Vienna, Austria, August 24-25, 2015, Revised Selected Papers. Springer International Publishing, Cham, pp. 147–159, 2015, ISBN: 978-3-319-27308-2.

[HT14] Honarmand, N.; Torrellas, J.: Replay Debugging: Leveraging Record and Replay for Program Debugging. In: Proceeding of the 41st Annual International Symposium on Computer Architecuture. ISCA '14, IEEE Press, Minneapolis, Minnesota, USA, pp. 445–456, 2014, ISBN: 978-1-4799-4394-4.

[JYB11] Jagannath, V.; Yin, Z.; Budiu, M.: Monitoring and Debugging DryadLINQ Applications with Daphne. In: IPDPS Workshops. IEEE, pp. 1266–1273, 2011, ISBN: 978-1-61284-425-1.

[LDY13] Logothetis, D.; De, S.; Yocum, K.: Scalable Lineage Capture for Debugging DISC Analytics. In: Proceedings of the 4th Annual Symposium on Cloud Computing. SoCC '13, ACM, Santa Clara, California, 17:1–17:15, 2013, ISBN: 978-1-4503-2428-1.

[Ma13] Marek, L.; Zheng, Y.; Ansaloni, D.; Bulej, L.; Sarimbekov, A.; Binder, W.; Qi, Z.: Introduction to Dynamic Program Analysis with DiSL. In: Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering. ICPE '13, ACM, Prague, Czech Republic, pp. 429–430, 2013, ISBN: 978-1-4503-1636-1, visited on: 03/15/2016.

[OC16] Oracle Corporation: JVM Tool Interface Version 1.2.3, 2016, URL: https://docs.oracle.com/javase/8/docs/platform/jvmti/jvmti.html, visited on: 03/15/2016.

[OR11] Olston, C.; Reed, B.: Inspector Gadget: A Framework for Custom Monitoring and Debugging of Distributed Dataflows. PVLDB 4/12, pp. 1237–1248, 2011, visited on: 03/15/2016.

[Vi10] Vicknair, C.: Research Issues in Data Provenance. In: Proceedings of the 48th Annual Southeast Regional Conference. ACM SE '10, ACM, Oxford, Mississippi, 20:1–20:4, 2010, ISBN: 978-1-4503-0064-3.

[Wi12] Wieder, A.; Bhatotia, P.; Post, A.; Rodrigues, R.: Orchestrating the Deployment of Computations in the Cloud with Conductor. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation. NSDI'12, USENIX Association, San Jose, CA, pp. 27–27, 2012.

Big (and small) Data in Science and Humanities
(BigDS17)

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 75

Big (and Small) Data in Science and Humanities

Anika Groß[1] Birgitta König-Ries[2] Peter Reimann[3] Bernhard Seeger[4]

The importance of data has dramatically increased in almost all scientific disciplines over the last decade, e. g., in meteorology, genomics, complex physics simulations, biological and environmental research, and recently also in humanities. This development is due to great advances in data acquisition and data accessibility, such as improvements in remote sensing, powerful mobile devices, popularity of social networks and the ability to handle unstructured data (including texts). On the one hand, the availability of such data masses leads to a rethinking in scientific disciplines on how to extract useful information and on how to foster research. On the other hand, researchers feel lost in the data masses because appropriate data management, integration, analysis and visualization tools have not been available so far. However, this is starting to change with the recent development of big data technologies and with progress in natural language processing, semantic technologies and others that seem to be not only useful in business, but also offer great opportunities in science and humanities.

A typical scientific data analytics pipeline covers several crucial phases to allow for complex data processing, integration, analysis and visualization. First, relevant data need to be acquired and integrated from different, often heterogeneous data sources. This includes various forms of data such as measurements, textual documents, images, graph data or spatio-temporal data. Depending on the size and structure of data, suitable data storage solutions have to be selected, e. g., from the wide spectrum of NoSQL data stores. Before pushing data to the actual analytics phase, it is crucial to detect and eliminate data quality problems by applying a careful data cleaning and enrichment process. Data analysis typically comprises data aggregation and filtering as well as sophisticated analysis methods. By now, we have access to a huge set of big data frameworks (e. g., Apache Spark, Apache Flink, Apache Storm) that support batch and/or stream processing and already provide many algorithms for data mining, machine learning or graph analytics. However, generic systems do not always offer good solutions for specialized problems. Hence, individual domain-specific analysis methods might be adopted in the pipeline as well. Finally, experts need to interpret results and incorporate novel insights into their real-world applications. Since it is infeasible to review very large result data sets, a clear representation and visualization is indispensable. Moreover, interactive user interfaces are necessary, e. g., to give extensive user support or to incorporate user's feedback and know-how during the data analysis process.

[1] Universität Leipzig
[2] Friedrich-Schiller-Universität Jena
[3] Universität Stuttgart
[4] Philipps-Universität Marburg

The analysis of scientific data faces several challenges. One main challenge is the generation of *high quality* results. Big data analytics need to deal with large heterogeneous amounts of data impeding quality assurance during data processing and analysis. Usually, it is very challenging to assess the achieved quality since gold standard data sets and benchmarks are rare and often not fully representative for a considered real-world problem. Another important aspect is the *scalability* of the applied methods to very large and possibly growing or changing data sets. Data is often not static, but underlies *evolution* or covers inherent temporal aspects, e. g., when data is acquired periodically to enable temporal analysis. Further challenges arise when privacy requirements must be respected as it is often the case for personal data, e. g., in the medical domain. Another important challenge and aim is the realization of scientific data analysis workflows as flexible *end-to-end solutions*. In many domains, data analysis still faces the problem of interrupted, bit-by-bit data processing, although many manual steps could (easily) be automated. A manual execution of data analysis tasks is time-consuming and prone to human error. It is therefore crucial to build automated solutions that involve human interactions if and only if this is useful. It is further important to incorporate data *provenance* aspects and traceability of the adopted processes within scientific workflows to enable the *reproducibility* of analysis results. Finally, a key challenge is the *visualization* of intermediate and final results. Visual analytics can offer valuable new insights when humans reveal patterns and trends based on meaningful data visualization.

This workshop intends to bring together scientists from various disciplines with database researchers to discuss real-world problems in data science as well as recent big data technology. We selected seven contributions that address several challenges of data-driven analysis. The contributions cover scientific data from various domains, such as geographical, environmental and agricultural science, biodiversity, archeology, humanities and business. The proposed approaches address topics that are related to the analysis of temporal and spatial data, decision support, interactive data exploration, deep-learning, data quality, text analysis, scientific workflows as well as the scalability of the adopted methods.

Three papers present data analysis approaches to support users in their decisions within different domains. *Amara et al.* use a deep-learning-based approach to classify banana leaf diseases based on image data sets. The authors show the effectiveness of their approach under varying conditions, e. g., regarding illumination and resolution size. The proposed method mainly enables early disease recognition and decision support for farmers that need to identify banana plant diseases. *Elmamoozet et al.* follow a graph-based approach to model continuous mobility in museum exhibitions. The authors discuss challenges to cope with a huge amount of sensor data, graph dynamics, as well as heterogeneous user groups within the environment of a museum. The system aims to support curators in their guiding task as well as museum visitors. *Beilschmidt et al.* present an intuitive web-based user interface for interactive exploration in geosciences. The system deals with spatio-temporal data from biodiversity research. The approach addresses scientific users by supporting decisions based on visual analytics for effective data-driven science on large and heterogeneous data.

The paper of *Kaltenthaler et al.* presents a framework for supporting the scientific workflow for managing and analysing archeo-related species. In this domain, working online is usually

not possible since there is no reliable internet connection, e. g., in the field of an excavation. This makes synchronization of local and global data essential during the data gathering, sharing and analysis process. The provided tools are applied within German zooarchaeology projects.

Cornelia Kiefer analyzes and discusses important challenges of domain-specific text analysis. Domain-specific texts can show very different and varying characteristics compared to standard texts like newspaper articles. In her work, she hence reveals serious data quality problems when applying standard text analytics tools to domain-specific texts during all phases of the analytics workflow.

Two further papers address scalability and efficiency aspects in the context of data analysis and data generation. *Kemper et al.* present a scalable solution of a data generator in the context of graph-based business analytics. Data generation and simulation are necessary when no or little representative data is available within a domain. In order to scale for the generation of huge business process graphs, an existing system has been extended to allow for distributed execution based on Apache Flink and Gradoop. Within their experiments, the authors show the scalability of the system to large data sets. Furthermore, *Pascal Hirmer* presents a concept to optimize the efficiency of data-intensive Mashups. The optimized execution is currently based on Map Reduce. The approach includes policy annotations to decide which services need to be executed in particular steps of a data mashup pipeline. The accompanying Mashup tool supports domain experts in their explorative analysis tasks.

We are deeply grateful to everyone who contributed to this workshop, in particular, the authors, the reviewers, the BTW team, and all participants.

1 Workshop Organizers

Anika Groß (Universität Leipzig, DE)
Birgitta König-Ries (Friedrich-Schiller-Universität Jena, DE)
Peter Reimann (Universität Stuttgart, DE)
Bernhard Seeger (Philipps-Universität Marburg, DE)

2 Program Committee

Alsayed Algergawy (Uni Jena, DE)
Peter Baumann (Jacobs Universität, DE)
Matthias Bräger (CERN, CH)
Thomas Brinkhoff (FH Oldenburg, DE)
Michael Diepenbroeck (Alfred-Wegner-Institut, Bremerhaven, DE)
Jana Diesner (University of Illinois at Urbana-Champaign, US)
Christoph Freytag (Humboldt Universität Berlin, DE)
Michael Gertz (Uni Heidelberg, DE)
Anton Güntsch (Botanischer Garten und Botanisches Museum, Berlin-Dahlem, DE)

Thomas Heinis (Imperial College, London, UK)
Andreas Henrich (Universität Bamberg, DE)
Jens Kattge (Max-Planck-Institut für Biogeochemie, DE)
Alfons Kemper (TU München, DE)
Meike Klettke (Universität Rostock, DE)
Frank Leymann (Universität Stuttgart, DE)
Bertram Ludäscher (University of Illinois at Urbana-Champaign, US)
Alex Markowetz (Uni Bonn, DE)
Jens Nieschulze (Uni Göttingen, DE)
Eric Peukert (Universität Leipzig, DE)
Kai-Uwe Sattler (TU Ilmenau, DE)
Uta Störl (Hochschule Darmstadt, DE)
Andreas Thor (Hochschule für Telekommunikation Leipzig, DE)

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 79

A Deep Learning-based Approach for Banana Leaf Diseases Classification

Jihen Amara,[1] Bassem Bouaziz,[2] and Alsayed Algergawy [3]

Abstract:

Plant diseases are important factors as they result in serious reduction in quality and quantity of agriculture products. Therefore, early detection and diagnosis of these diseases are important. To this end, we propose a deep learning-based approach that automates the process of classifying banana leaves diseases. In particular, we make use of the *LeNet* architecture as a convolutional neural network to classify image data sets. The preliminary results demonstrate the effectiveness of the proposed approach even under challenging conditions such as illumination, complex background, different resolution, size, pose, and orientation of real scene images.

Keywords: Banana plant diseases, Deep learning, Classification

1 Introduction

Crop diseases are major sources of famine and food insecurity on our planet. In fact, it is estimated that plant pathogens may account for annual crop yield losses of up to 16% globally [Oe06]. Furthermore, the current solutions to fight different diseases demand the massive use of crop protection products, which are dangerous for the environment and the user. Microscope and DNA sequencing-based methods are effective to identify and discover different kinds of diseases. Even though many of the farmers around the world do not have access to these diagnostics tools, the vast majority of them possesses a cell phone. In fact, the Ericsson company forecasts that mobile subscriptions will reach 9.3 billion in 2019 and 5.6 billion of these will be smartphone subscriptions [Mo15]. Hence, a phone based tool that helps in diagnosing crop diseases based on capturing and analyzing automatically a picture of a plant leaf is a promising solution.

In this paper, we take a first step towards such a tool. However, we limit our study to classify banana leaves diseases. Banana is threatened by different types of diseases, such as *banana sigatoka* and *banana speckle*. The black sigatoka is caused by the fungus *Mycosphaerella fijiensis*. Its symptoms start by minuscule, chlorotic spots and it then develops into thin brown streaks that are bounded by leaf veins (see Fig.1). Leaf speckle is a fungal disease. Its symptoms start as light brown little spots and with time they increase in size

[1] Digital Research Center of Sfax (CRNS)- MIRACL Laboratory, Tunisia, jihene.amara@uni-jena.de, amarajihen@gmail.com
[2] Higher Institute of Computer Science and Multimedia, University of Sfax, MIRACL Laboratory, Tunisia, bassem.bouaziz@uni-jena.de, bassem.bouaziz@isims.usf.tn
[3] Heinz-Nixdorf-Chair for distributed information systems, Institute for Computer Science, Friedrich-Schiller University of Jena, Germany, alsayed.algergawy@uni-jena.de

and become black. Left untreated, these diseases will kill the plant. If, however, they are diagnosed early, they can be treated and the plant can be saved.

(a) Black Sigatoka (b) Banana Speckle

Figure 1: Example of two banana diseases

Given the limited availability of resources and expertise on banana pathology worldwide, the need for a system to classify and identify banana diseases automatically is urgent. Also, the proliferation of using phones and the internet all over the world make it easily accessible to all kind of people. Computer vision and machine learning techniques have been applied to different disease detection such as tomatoes, grapes, potatoes and cotton. Motivated by advances in computer vision, especially convolutional neural networks (CNNs), which managed to produce remarkable results in the field of image classification, we propose a new system based on CNNs for banana plant disease recognition. Our method, based on LeNet architecture [Le89] requires minimal image preprocessing. The model can learn visual features directly from images. The developed model is able to recognize two different types of diseased leaves out of healthy ones.

The rest of the paper is organized as follows: in Section 2, we present the related work. Section 3 explains technical details of the proposed approach and the architecture of the used convolutional neural network. Furthermore, experimental evaluation and results are reported in Section 4. Finally, Section 5 concludes the paper and provides an outlook to future work.

2 Related work

Plant diseases cause major production and economic losses in agriculture and forestry. Recent developments in agricultural technology have led to a demand for a new era of automated non-destructive methods of plant disease detection. Hence, a number of approaches have turned to computer vision and machine learning techniques to create a fast method for plant diseases detection at the early onset of the symptoms. Most of the studies presented in the literature of plant disease identification follow the steps shown in Fig. 2 [Al11, Cu10, PC13].

As shown in Fig. 2, the identification process starts by an image acquisition step where different digital devices are used to capture healthy and infected plant images. Then, further analysis is needed to edit the image and prepare it for later treatment, such as image enhancement, segmentation, color space conversion and filtering. In particular, image segmentation methods, like thresholding, are frequently used to detect boundaries in images.

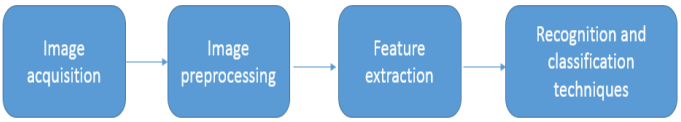


Figure 2: General steps applied to plant disease identification

Within the feature extraction step, features such as color, shape and texture are calculated from the image. Finally, the classification step is performed. Different classification algorithms are used in the literature such as neural network [Sa13], support vector machine [BKS16] and rule based classification [Cu10, Sc16]. In the following, we present a set of the state-of-the-art approaches following the general architecture in Fig. 2.

An automated tool is presented in [Cl15] to measure the foliar bleaching of leaf caused by the insect known as Corythucha ciliata (SAY). In this study, a CCD camera was used for collecting infected plant images. The authors used the digital color image analysis discrimination method in which the original RGB leafs images were converted into HIS and L*a*b color spaces. The leaves containing the disease are determined based on the color difference between them and the healthy leaves. The method is based on the use of the *Otsu method* to isolate the leaf from its background and the chlorophyll histogram to detect discolorations caused by the lace bug. In [Al11], a color based approach is introduced to identify five types of leaf diseases which are *early scorch*, *cottony mold*, *ashen mold*, *late scorch* and *tiny whiteness*. The approach starts by identifying green pixels based on the calculation of a global threshold according to the *Otsu method*. After removing of non green pixels, the infected areas are clustered using the K-means technique. Furthermore, texture features are extracted using the co-occurrence matrix as an input to a neural network to identify the disease. In [Cu10], the authors develop an automatic threshold method to separate the infected areas from the healthy ones. They use a multispectral CCD camera to capture the images of the leaves then each leaf image is converted from the RGB format to the HIS format. After the infected and healthy pixels are segmented, the RIA (Ratio of Infected Area) and RCI (Lesion Colour Index) values were calculated. Based on these values, the approach is able to determine if a leaf is healthy or not.

Another approach, represented in [BKS16], starts by a segmentation step of the leaf in order to remove the background using the guided active contour. The approach then computes the deviation of each pixel from the green color in order to segment the symptom. The greener the pixel the healthier the part of the leaf is. The segmented symptoms and lesions are then transformed from the RGB space to the HSV, L*a*b* and CMYK color spaces. The classification step is then performed using color histograms. A new system is introduced in [MBP16] to classify different diseases that infect the paddy plants such as brown spot disease, leaf blast disease and bacterial blight disease. In this system, the authors extract the scale invariant feature transform (SIFT) feature and then use KNN and SVM for classification. [Ob14] introduce a prototype for the detection of mycotic infections on tomato crops. The authors have created a manually annotated dataset of 147 im-

ages including 31 and 28 images of healthy and infected plant leaves, respectively. Then they use different color descriptors (Color Structure Descriptor (CSD), Scalable Color Descriptor (SCD) and Color Layout Descriptor (CLD)) to characterize the leaflets. The authors reported that the CSD showed better results as compared to SCD and CLD. The author in [Sa13] presented a disease identification approach of a medical herb called Phyllanthus Elegans Wall (Asin-Asin Gajah) which is used to cure breast cancer. The proposed method starts by enhancing contrast as a preprocessing step before segmentation and features extraction from leaves images. Then, two feed forward neural networks which are multi-layer perceptrons and radial basis function RBF were applied to classify the images into healthy or unhealthy.

Even though different methods have achieved good classification results in identifying and recognizing some of the diseases, they suffer from some limitations. For example, segmentation is used in most methods as the first step in the leaf disease analysis. If the leaf image is captured with a black background, the segmentation is straightforward and no obstacles should be faced. However, when the background contains other leaves or plants, the segmentation may be questionable. Most of the methods will fail to effectively extract the leaf from its background which will lead to unreliable results. Also, some disease symptoms do not have well represented edges and they could gradually fade into healthy tissue [Ba16]. This may disturb solutions like color based methods and thresholding. Furthermore, a number of the methods rely on hand-crafted features such as color histograms, texture features, shape features and SIFT that requires expensive work and depends on expert knowledge. However, these methods do not generalize well and they are not effective when dealing with a large amount of data that could contain significant varieties. For example, the black leaf streak disease in banana produces heterogeneous symptoms and hence requires more powerful methods to detect them effectively. Hence, in our work we will investigate the application of convolution neural network (CNN) which is a method that avoids segmentation and hand-crafted features. Also, the presence of a large amount of banana leaves data and powerful computing infrastructure made CNN a suitable candidate for the current application.

In the next section, we detail the proposed method for banana disease identification.

3 Proposed method

To deal with the mentioned challenges, we introduce a deep learning-based approach to classify and identify banana leaves diseases. The general architecture of the proposed framework is illustrated in Fig. 3. The figure shows that the framework consists of two main components: *image preprocessing* and *deep learning-based classification*. In the following, we present details about each component.

3.1 Image preprocessing

The dataset stored in either local or global repositories contains a large number of images of healthy and infected leaves. The images are taken with a standard digital camera. Each

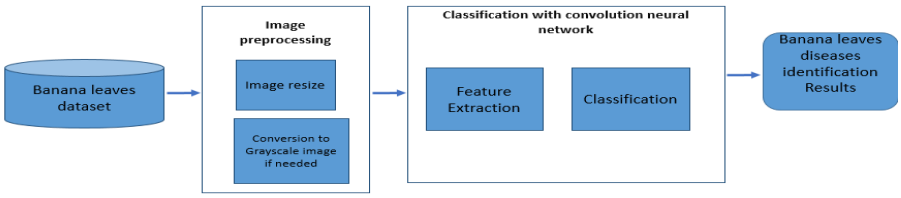


Figure 3: Proposed framework architecture.

image has three channels which are red (R), green (G), and blue (B). In our experiments we will test the applicability of our approach to both the RGB images and the grayscale images. To this end, we perform a preprocessing step where each image in our dataset is resized to $60 \star 60$ pixels and converted to grayscale.

3.2 Deep learning based classification

Neural networks contains multiple neurons arranged in layers. The neurons in the adjacent layers are connected to each other. These neurons learn how to convert inputs (pre-extracted and pre-processed features) into corresponding output (labels). In particular, convolutional neural networks (CNNs) are a family of multilayered neural networks and are considered as the first successful trial for deep learning approachs where many layers of a hierarchy are successfully trained in a robust manner. CNNs are known for their robustness toward low variation in inputs, they require low preprocessing for their execution. They are also able to extract appropriate features while simultaneously performing discrimination [De14, ARK10]. More specifically, in the current implementation we make use of the LeNet [Le89] architecture for the convolution neural network.

As shown in in Fig. 4, a CNN is composed of three main parts which are convolution, pooling and fully connected layers. The convolution and pooling layers act as feature extractors from the input images while the fully connected layer acts as a classifier. The essential purpose of convolution is to extract features automatically from each input image. The dimensionality of these features is then reduced by the pooling layer. At the end of the model, the fully connected layer with a softmax activation function makes use of the learned high-level features to classify the input images into predefined classes. In summary, the LeNet model is composed of two main parts: the first part is the self-taught feature extraction model and the second part is the classification model. In the rest of this section, we will detail these two components.

3.2.1 Feature extraction model

The feature extraction model is the part where the network learns to detect different high-level features from the input images. It consists of a sequence of convolution and pooling layers.

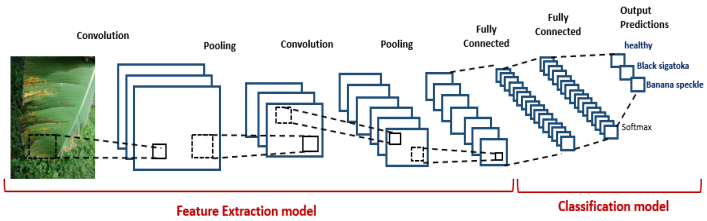


Figure 4: A graphical depiction of a LeNet model

- **Convolution map.** The convolution layer is an elementary unit in the CNN architecture. The goal of convolution is to extract features from the input image. It consists of a set of learnable filters. Each filter is applied to the raw pixel values of the image taking into account the red, green and blue color channels in a sliding window fashion, computing the dot product between the filter pixel and the input pixel. This will result in a 2-dimensional activation map of the filter called the *feature map*. Hence, the network learns filters (ie. edges, curves) that will activate when they find known features in the input.The CNN learns the values of these filters on its own during the training process. The Convolution operation is presented in Equation 1. A convolution layer is configured by the number of convolution maps it contains M_i, the size of the filters which are often squared $k_x \star k_y$. The feature map M_i is computed as follows:

$$M_i = b_i + \sum_k W_{ik} \star X_k \tag{1}$$

where $\star$ is the convolution operator, X_k is the k^{th} input channel, W_{ik} is the sub kernel of that channel and b_i is a bias term. In other words, the convolution operation being performed for each feature map is the sum of the application of k different 2D squared convolution features plus a bias term. Hence, In comparison with traditional image feature extraction that relies on crafted general feature extractors (SIFT, Gabor filter, etc), the power of CNN is noted in its ability to learn the weights and biases of different feature maps which lead to task specific powerful feature extractors. Moreover, the rectified nonlinear activation function *(ReLU)* is performed after every convolution to introduce nonlinearity to the CNN. The ReLU is a very popular activation function which is defined as $f(x) = max(0,x)$ where x is the input to a neuron.

- **Max-pooling map:** In the architecture of convolutional neural network, convolution layers are followed by sub-sampling layers. Each sub-sampling layer reduces the size of the convolution maps, and introduces invariance to (low) rotations and translations that can appear in the input. A layer of max-pooling is a variant of such layer that has shown different benefits in its use. The output of max-pooling layer is given by the maximum activation value in the input layer over sub windows within each feature map. The max-pooling operation reduce the size of the feature map.

3.3 Classification model

Within the classification step we use fully connected layers where each neuron provides a full connection to all learned feature maps issued from the previous layer in the convolution neural network. These connected layers are based on the softmax activation function in order to compute the classes scores. The input of the softmax classifier is a vector of features resulting from the learning process and the output is a probability that an image belongs to a given class. The softmax function ς takes as input a C-dimensional vector z and outputs a C-dimensional vector y of real values between 0 and 1. This function is calculated as below:

$$y_c = \varsigma(\mathbf{z})_c = \frac{e^{z_c}}{\sum_{d=1}^{C} e^{z_d}} \quad for\, c = 1 \cdots C \tag{2}$$

In the next section, we will present the conducted experiments and results.

4 Experimental Evaluation

To validate the performance of the proposed approach, we conducted a set of experiments using a real dataset of banana diseases obtained from the PlantVillage project [HS15, MHS16]. The plant village project [3] contains thousands of images of healthy and diseased crop plants that are open and available on the web. The images in our dataset are annotated as belonging to three different categories which are healthy (1643 images), black sigatoka (240 images) and black speckle (1817 images). Fig. 1 illustrates some samples of the dataset which contains in general 3700 images. These images are captured with different sizes, orientation, poses, backgrounds and illumination. In our implementation we used the deeplearning4j [4] as an open source deep learning library which supports the use of GPUs to make the execution of the deep learning algorithms faster. Our goal is to evaluate the predictive performance of our model on unseen data of banana diseases. Hence, in our experiments, we decided to test all the different range of train and test set splits to evaluate the robustness of our proposed algorithm and its ability to avoid overfitting. This means that a percentage of the whole dataset is used for training and the rest is used for testing. The training dataset varies from 80%, 60%, 50%, 40 % to 20 % with the use of the same hyper parameters described in Table 1. These parameters are determined empirically according to a series of experiments carried on the whole dataset that give the best results of classification.

Table 1: hyper parameters choices

Parameter	optimization algorithm	learning rate	momentum	weight decay	batch size	activation function	iterations
Value	SGD	0.001	0.9	0.005	10	Sigmoid	30

As shown in Table 1, the stochastic gradient descent (SGD) algorithm is used in our model to learn the best set of weights and biases of the neural network that minimize the loss

[3] https://www.plantvillage.org/en/
[4] https://deeplearning4j.org/

function. While learning, the SGD algorithm works by randomly choosing a small number of training inputs. We refer to this as the batch size which is set to 10. The learning rate is set to 0.001. It is the rate at which a function move through the search space. A small learning rate leads to more precise results but it requires more training time. The momentum is an additional factor to determine how fast the SGD algorithm converges on the optimum point. It is set to 0.9.

To evaluate the effectiveness of the proposed system, we make use of a combination of *accuracy*, *precision*, *recall*, and *F1-score*. Results are reported in Table 2 across all our experimental configurations.

Table 2: Precision, recall, accuracy and F1 score for the corresponding experimental configuration.

		Color				Gray Scale			
Train	Test	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score
20%	80%	0.9861	0.9867	0.986	0.9864	0.9444	0.9479	0.9444	0.9462
40%	60%	0.9861	0.9865	0.9859	0.9863	0.9757	0.9764	0.975	0.976
50%	50%	0.9972	0.9970	0.9972	0.9971	0.8528	0.889	0.8527	0.8705
60%	40%	0.9676	0.969	0.9677	0.9683	0.9282	0.9314	0.9283	0.9298
80%	20%	0.9288	0.9299	0.9288	0.9294	0.8594	0.8678	0.8594	0.8636

these results were obtained using our deep learning model for identifying three banana images classes (healthy, black sigatoka and black speckle). As shown in Table 2, our model was able to find good results when applied to classify the diseases of banana from its leaves. The obtained results confirm the the importance of the color information in plant disease identification. Hence, a green color always refer to a healthy leaf while a leaf with black or brown spots may be considered unhealthy.

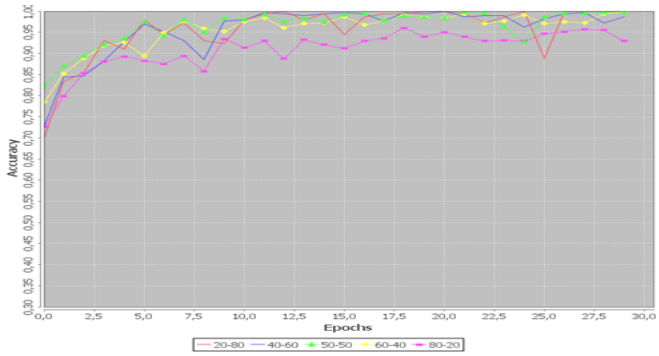


Figure 5: Comparison of progression of overall accuracy grouped by train-test set splits

Fig. 5 shows the accuracy of the different train and tests splits choices while the number of iterations is varied. As we can see, in some splits the model take more time to converge. However in most of the test splits, the model starts to stabilize from iteration 25 and achieve good accuracy at the final iteration.

5 Conclusion

Agriculture suffers from a severe problem, plant diseases, which reduces the production and quality of yield. Besides, the shortage of diagnostics tools in underdeveloped countries has a devastating impact on their development and quality of life. Hence, there is an urgent need to detect the plant diseases at the early stage with affordable and easy to use solutions. To this end, in this paper, we presented an approach based on convolution neural networks to identify and classify banana diseases. The proposed model can serve as a decision support tool to help farmers to identify the disease in the banana plant. Hence, the farmer can take a picture of the leaf with the symptoms and then the system will identify the type of the disease. Our main contribution is to apply deep neural networks to detect two famous banana diseases which are banana sigatoka and banana speckle in real scene and under challenging conditions such as illumination, complex background, different images resolution, size, pose and orientation. After several experimentations our system was able to find good classification results. This has proven that the proposed method can significantly support an accurate detection of leaf diseases with little computational effort. Encouraged by the optained results, we intend in our future work to test more banana and plants diseases with our model. Besides, we will target the automatic severity estimation of the detected disease since it is an important problem that can help the farmers in deciding how to intervene to stop the disease.

6 Acknowledgments

We are grateful to Birgitta König-Ries for their discussions and their technical supports. A part of this research was supported by The DAAD funding through the BioDialog project. A. Algergawy'work is partly funded by DFG in the INFRA1 project of CRC AquaDiva.

References

[Al11] Al-Hiary, H.; Bani-Ahmad, S.; Reyalat, M.; Braik, M.; ALRahamneh, Z.: Fast and accurate detection and classification of plant diseases. International Journal of Computer Applications, 17(1):0975–8887, 2011.

[ARK10] Arel, Itamar; Rose, Derek C.; Karnowski, Thomas P.: Deep Machine Learning - A New Frontier in Artificial Intelligence Research [Research Frontier]. IEEE Comp. Int. Mag., 5(4):13–18, 2010.

[Ba16] Barbedo, A.: A review on the main challenges in automatic plant disease identification based on visible range images. Biosystems Engineering, 144:52–60, 2016.

[BKS16] Barbedo, Jayme Garcia Arnal; Koenigkan, Luciano Vieira; Santos, Thiago Teixeira: Identifying multiple plant diseases using digital image processing. Biosystems Engineering, 147:104–116, 2016.

[Cl15] Clement, Alain; Verfaille, T; Lormel, C; Jaloux, B: A new colour vision system to quantify automatically foliar discolouration caused by insect pests feeding on leaf cells. Biosystems Engineering, 133:128–140, 2015.

[Cu10] Cui, Di; Zhang, Qin; Li, Minzan; Hartman, Glen L.; Zhao, Youfu: Image processing methods for quantitatively detecting soybean rust from multispectral images. Biosystems Engineering, 107(3):186–193, 2010.

[De14] Deng, Li: A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing, 3, 2014.

[HS15] Hughes, David P.; Salathé, Marcel: An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. CoRR, abs/1511.08060, 2015.

[Le89] LeCun, Yann; Boser, Bernhard; Denker, John S; Henderson, Donnie; Howard, Richard E; Hubbard, Wayne; Jackel, Lawrence D: Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

[MBP16] Mohan, K. Jagan; Balasubramanian, M.; Palanivel, S.: Detection and Recognition of Diseases from Paddy Plant Leaf Images. International Journal of Computer Applications, 144(12):34–41, 2016.

[MHS16] Mohanty, Sharada Prasanna; Hughes, David; Salathe, Marcel: Using Deep Learning for Image-Based Plant Disease Detection. 2016.

[Mo15] Mobility, Ericsson: , Interim Update: Ericsson Mobility Report, 2015.

[Ob14] Oberti, Roberto; Marchi, Massimo; Tirelli, Paolo; Calcante, Aldo; Iriti, Marcello; Borghese, Alberto N.: Automatic detection of powdery mildew on grapevine leaves by image analysis: Optimal view-angle range to increase the sensitivity. Computers and Electronics in Agriculture, 104:1–8, 2014.

[Oe06] Oerke, E-C: Crop losses to pests. The Journal of Agricultural Science, 144(01):31–43, 2006.

[PC13] Patil, Sagar; Chandavale, Anjali: A Survey on Methods of Plant Disease Detection. International Journal of Science and Research (IJSR), 6(14), 2013.

[Sa13] Sannakki, Sanjeev S; Rajpurohit, Vijay S; Nargund, V B; Kulkarni, Pallavi: Diagnosis and Classification of Grape Leaf Diseases using Neural Networks. In: Fourth International Conference on Communications and Networking Technologies. 2013.

[Sc16] Schor, Noa; Bechar, Avital; Ignat, Timea; Dombrovsky, Aviv; Elad, Yigal; Berman, Sigal: Robotic Disease Detection in Greenhouses: Combined Detection of Powdery Mildew and Tomato Spotted Wilt Virus. IEEE ROBOTICS AND AUTOMATION LETTERS, 1(1):354–360, 2016.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 89

A Framework for Supporting the Workflow for Archaeo-related Sciences: Managing, Synchronizing and Analyzing Data

Daniel Kaltenthaler[1] Johannes-Y. Lohrer[1] Peer Kröger[1] Henriette Obermaier[2]

Abstract: In this paper we analyze the challenges of the workflow in archaeo-related disciplines. We extract the main parts of the workflow and consider them in our solution for an eScience service in this domain. First, we describe the dynamic framework *xBook* to be used for different archaeo-related databases. Afterwards, we show the *Synchronization* supporting the collaboration with colleagues and sharing data. Finally, we describe the *Analysis Tool*, that allows analyses to be executed directly inside the framework. Concluding, we show a sample analysis in *OssoBook*, a zooarchaeological database.

Keywords: Analysis, Framework, Synchronization, OssoBook, Workflow, xBook

1 Introduction

The recovery and analysis of material culture is the main focus of archaeo-related work. The corpus of finds like artifacts, faunal remains, or human burial remains are excavated, described, categorized, and analyzed in projects all over the world. The results of analyzing collected data make us learn about the past. Having access to big data volumes may be essential to run significant analyses. Therefore, sharing data is important. A long time data was only recorded analog, the digital recording, if available, was normally done in Excel sheets. This made the exchange of primary data, which are often not published, complicated and difficult, thereby complex analysis of archaeo-related data were infrequent.

The usage of digital databases for the recording of data has increased in the last decades. There are some databases that offer complex and versatile options to enter data, like the zooarchaeological database *The York System* [Ha03] or the archaeobotanical database *ArboDat* [PDK11]. However, these databases lack other convenient features – they provide methods to export data, but sharing data comfortable via the Internet is not supported. For the basic and frequent problem statements the data is retrieved with predefined queries, but there are no complex analyses available that are directly built into the application. Data has normally to be analyzed manually or in extern tools using the exported data. Additionally, there are a wide range of databases that were customly created by freelanced archaeologists and bioarchaeologists, which functionalities are limited to their requirements, and are not published.

[1] Ludwig-Maximilians-Universität München, Institute for Informatics, Database System Group, Oettingenstraße 67, 80538 München, {kaltenthaler,lohrer,kroeger}@dbs.ifi.lmu.de
[2] Bavarian State Collection for Anthropology and Palaeoanatomy Munich, Kaulbachstraße 37, 80539 München, henriette.obermaier@palaeo.vetmed.uni-muenchen.de

While the workflow of all disciplines is similar, there still are some differences in the nature of the data. Therefore our ambition is to provide a flexible framework, that offers the creation of databases which supports the composition of all input options, where data can be entered, shared, and analyzed within the same application. This would support archaeologists and bioarchaeologists in their workflow.

In summary, the main contribution in this paper are as follows: We classify three main parts of the workflow of archaeo-related work in Chapter 2 and consider them in our solution for the digital realization. Our framework is described in Chapter 2.1. We explain the functionality to collaborate and share data with others, and to support working offline without an Internet connection, the *Synchronization*, in Chapter 2.2. The data collection and sharing forms the basis for documentations and for analyses inside the excavation project, that is provided by the built-in *Analysis Tool*, which we describe in Chapter 2.3. Finally, we show the composition of a frequently used analysis in the zooarchaeological database *OssoBook* as an example with the analysis framework in Chapter 3.

2 Creation of the xBook Framework

Concerning the archaeological and bioarchaeological work, we extracted three main challenges that have to be fulfilled to support the work of archaeologists and bioarchaeologists:

1. **Data Gathering:** Saving of archaeo-related data digitally in a database.
2. **Data Sharing:** Collaborating with colleagues and sharing data with other users.
3. **Data Analyses:** Executing analyses on the available data.

The solution of all three challenges must be as dynamic and flexible as possible to be usable in several archaeo-related databases. Below, we describe our solution for these challenges for the archaeo-related disciplines, which can also be transferred to other disciplines.

2.1 Development and Implementation of the xBook Framework

We set the challenge to provide a generic database solution for all disciplines, that is as customizable as possible to allow all required information about the specific data to be gathered. We created *xBook*, a generic open-source framework including the common and basic features for a database for archaeo-related disciplines. Below, we use the term “Book” to describe an incarnation of *xBook*. Each Book provides these common features:

In *xBook* the gathered data is separated to single data groupings called **Projects**. All findings of an excavation or a partial area of the excavation are combined to a single Project. This enables the collaboration and the sharing of data of specific excavations, or parts of it.

Most of the values that are entered to the database have a basic and simple type, like text, drop-down box, numbers, time, date, boolean values, etc. So the framework offers a

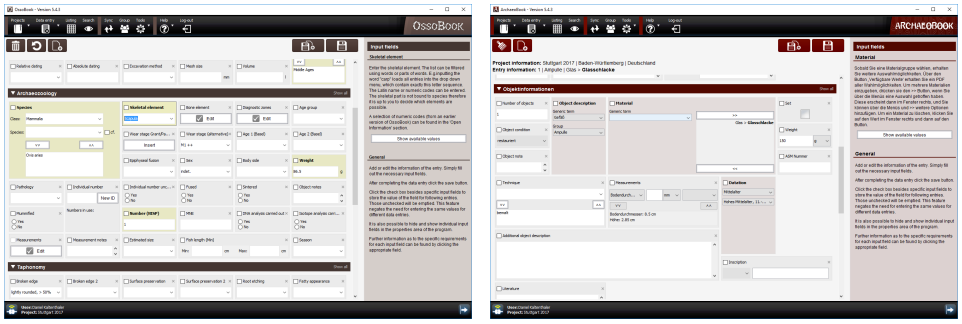


Fig. 1: The input mask of *OssoBook* (left) and *ArchaeoBook* (right), based on the *xBook* framework, that provides a basic GUI and functions, but allows customization like e.g. individual input fields.

dynamic and flexible **Data Entry Mask** that reuses the provided, most common types of input fields for each Book. There are already available several different input fields that can be used without having to implement a new type of input field. However, if a Book needs a specific, maybe much more complex type of input field, it can easily be added and be used for a specific Book. So, the Data Entry Mask of each Book is composed of the necessary, specific input fields and offers the required possibilities to enter the data, dependent on the archaeo-related discipline. As an example, a comparison of the Data Entry Mask of *OssoBook* and *ArchaeoBook*, two incarnations of *xBook*, is shown in Figure 1.

An overview about all data of a specific project can be displayed in the **Listing**. All entries of the Data Entry Mask are displayed as a table form. If there are available more than one Data Entry Mask, a selection of the mask is displayed. The table can be sorted and filtered.

To search the saved data for one or more specific data sets, *xBook* provides two types of **Data Search**. The first one is the Filter. It allows to filter the Listing for entries with specific terms or values. All data sets that do not match the Filter settings are hidden in the Listing. The Filter is made to regain specific entries. The second one is a Project Search where the user can search for local Projects that are matching with specific terms or values.

To be able to collaborate and share data, *xBook* provides an **Project Right Management**. Each Project can be shared to specific users, or to a specific user group. Each user or user group can be granted rights to read or write the data, and permission to edit the Project information. These rights can be extended and set individually.

Further features, like **User Management**, an **Exporter**, and an automatic **Update Function** is also included to the *xBook* framework.

All features are wrapped into a common **Graphical User Interface** which structures can be reused. So every Book provides the same basic frame, like navigation, footer, notifications, etc. The similarity of the graphical representation can be viewed in Figure 1. Still, customizations are possible, like new elements in the navigation, or the adjustment of the application logo. The common features are shared with all Books, that all benefit from new features and special implementations for a single Book. The features match the

requests of archaeologists and bioarchaeologists to gather and save data entry for their excavations.

2.2 Synchronization of Data

To collaborate and share the data with others as well as store the data, a synchronization process is required. Since working online is not possible in environments in which a reliable Internet connection cannot be guaranteed, e.g. directly in the field of an excavation, we must provide a way to enter the data locally. These data must then be synchronized to the global server and all updated or added entries in the global database must be synchronized to the local database again. While there are already a wide range of different database systems available that offer possibilities for synchronization (like Microsoft SQL Server, Oracle, and MySQL), none of them fulfills all the requirements needed for our framework. Most of them do not "provide the ability to work offline and none of them allows synchronization of single data sets" [Lo16].

Concept: The *Synchronization* uses timestamps to keep track of the last time the entry was updated in the global database. Additionally the status within the local database is saved in another column. The status can be `synchronized`, if the entry was not updated locally since the last time it was synchronized with the global database; `updated`, if the entry was updated locally since the last synchronization; and `conflicted`, if a conflict was detected. A conflict occurs if the same entry is updated on two different local databases, without synchronizing the entry in between. Because incarnations of *xBook* can have multiple independent input units, the *Synchronization* synchronizes one complete entry – with all information that are saved in connected tables – at a time, one input unit after another.

Implementation: Each time an entry is added or updated in the local database, the status of the entry is set to `updated`. This defines which entry has been changed since the last time the *Synchronization* was executed. If the synchronization process is started, each input unit is checked which data sets are not synchronized. One after the other the unsynchronized data sets are uploaded to the server. If no conflicts are recognized, the status of the uploaded entries is set to `synchronized`. On the server, a trigger sets the timestamp to the current time every time an entry is uploaded or updated. If there is no unsynchronized entry left, the entries are synchronized back to the local database in the order of the timestamp on the server, starting with the entry that has the lowest timestamp on the server, but is higher than the highest timestamp on the local database. If the currently synchronized entry was updated locally or is marked as `conflicted`, this entry is ignored. This procedure is repeated for all available input units.

Conflicts: If a conflict was detected during the *Synchronization* process, the conflict has to be solved before the entry can be synchronized again. This has to be done manually by the users inside an own panel. When solving the conflict, the users gets the differences between the local and the global version displayed in a diff-screen. There, the users can select which value they want to save. After the users have decided the version they want to use for all differences, the entry is saved locally with the updated values as `synchronized`,

but without updating the timestamp. Then the entry is additionally saved globally (without the *Synchronization*) with the timestamp of the global entry. Therefore, the timestamp of the local entry is updated during the next synchronization process. This can not be done directly. Otherwise entries would not be synchronized to the local database, if they were updated globally since the last synchronization process, because they would have a lower timestamp than the conflicted entry.

Integration: The integration in the application must consider that most of the users of *xBook* are not familiar to work almost exclusively with a computer. That is the reason why it is absolutely necessary to hide the complexity of the *Synchronization* behind an intuitive input mask that is easy to use. In this Synchronization Panel, all global Projects for which a user has read and/or write permission must be downloadable from the server. Therefore the corresponding Projects can be selected in the Project selection on the right side of the Synchronization Panel. The local Projects that have not been synchronized with the server before must be able to be uploaded to the server. These Projects can be selected in the Project selection on the left side. Existing Projects, that are either saved in the local or global database, must be updatable. For this purpose the corresponding Projects must be selected, as explained above. The synchronization process can be started by pressing the "Synchronize" button.

2.3 Embeddable Analyses Tool

Having provided the framework for creating databases for archaeo-related work (cf. Chapter 2.1) and enabled a feature for sharing the collected data and collaborate with colleagues (cf. Chapter 2.2), we now want to concentrate on possibilities to analyse this data. So far, data is usually exported from a database in the form of spreadsheets — e.g. Microsoft Excel, LibreOffice Calc, etc. -– or comma-separated values files to meet the goals of specific researches in the archaeo-related domain. Although this is enabled by using the Export function in *xBook*, this method can be aggravating, time consuming, and error-prone, but it is still common practice. Our ambition is to provide a dynamic, flexible, and powerful tool that allows specific queries from archaeological databases without having any prior knowledge of programming. We want to support archaeologists and bioarchaeologists in their research and to provide the ability to select specific data from the database as is needed, with as few limitations as possible. The target is to provide a flexible tool that scientists can use for visually querying the data from their databases and create almost every composition for their data analysis. Furthermore, we want to offer a possibility to generate graphical results that scientists can use for their analyses and publications, without having to export any data and use external spreadsheet or analysis tools. This tool must work independent of possible changes of the database scheme.

Concept: The *Analysis Tool* was developed for *xBook*, but must be flexible enough to be able to be embedded to other Java applications as well. To achieve this, the tool must provide a well defined set of functions that has to be implemented by the application, the tool is being embedded to. At the same time the tool needs an API that allows new components to be registered that might be specific for an application. This also requires

the communication of the components to be generic, because the type of the connected components is not known beforehand. The tool must provide a predefined graphical user interface that can be directly embedded into these applications. However, it must also be possible to implement a custom graphical representation, in case another design of the elements is desired, or if the GUI library is not applicable. So, an abstract connection between the logic and the view layer of the components is required. The *Analysis Tool* is designed to be embeddable to other applications to support the integration of specific solutions in any database application. With these specifications, the tool is flexible and extensible, and supports individual implementations dependent on the requirements of the scientific domain.

Implementation: The different analyses must be composed with several single components ("Workers"), that each represent another task in the composition, like the retrieval of data, the filtering or sorting of tables, or for the joining of different data sets. Each Worker is composed of several Properties. These describe the structure of the Workers, that means they define which inputs and outputs are available and which settings can be set by the user. The graphical user interface checks for each Worker and displays a composition dependent on the available Properties. The communication with Properties is done in different ways. The communication to the Property is realized by the methods that are defined in the class `Property` (cf. Figure 2). The communication from the Property has to be done with an event handling system. For this, the object that wants to be notified about changes of Property has to be registered directly to the Property, in form of a Property Listener.

Workers can have Properties for the input and output of data which can be used to connect a Worker to other Workers. To carry out their logic, Workers can query the previous Workers for the output. Furthermore, they provide their output for other Workers. To reduce calculation time and resources, the Workers only calculate the output data if specifically queried. But they always provide the current output scheme which allows the connected Workers to use the data structure to display the selectable data to the users. To retrieve data from the application, the Workers can use the interface `IWorker` (cf. Figure 2) that has to be implemented by the base application.

The API of the *Analysis Tool* provides a registry class, which allows Workers to be registered for inclusion in the analysis. The indirect registration allows externally created Workers to be included to the analysis by scanning the .jar files in the folder "./analyses/workers". All classes that implement the interface `IWorker` are added to the list of available Workers in the *Analysis Tool*.

Integration: To be able to reuse the *Analysis Tool* in different base applications, it is important that the integration requires as few changes to the base application as possible. But since the *Analysis Tool* cannot know how the data is stored, or how the connection to the data is realized, the base application must implement some wrapper methods. The *Analysis Tool* itself is composed in a `JPanel` or `JFXScene`. For this, the base application can either create a new `AnalysisSwing` or `AnalysisFX` instance, which both require an implementation of the `IController` (cf. Figure 2). The `IController` is the interface that serves as the connection of the base application to the analysis. The base application therefore has not to know any internals of the analyses or other way around. Since the

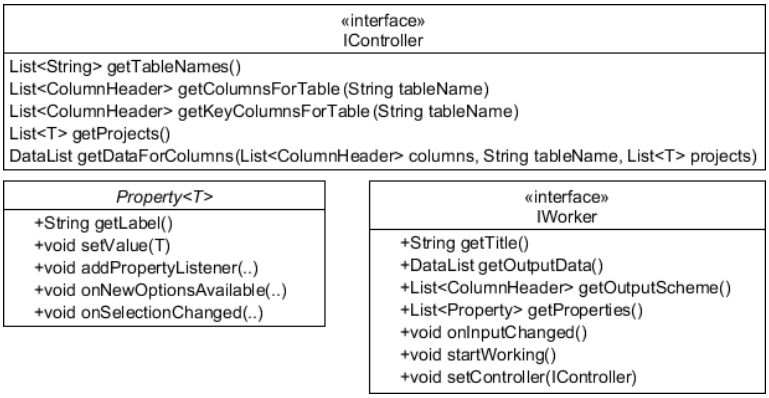


Fig. 2: Schematic representation of the interfaces *IController* and *IWorker*, and the class *Property*.

analysis is done inside one panel, it can easily be included into the base application, which can decide where and when the analysis shall be displayed.

3 Sample Analysis: Age distribution of Animals in OssoBook

The database we focus on in this paper is *OssoBook* [Ka16]. Since its conversion to Java and MySQL in 2001, it has been continuously developed, and became part of the *xBook* framework with release of version 5.0 in 2013 [Ka15]. The application is currently used by approximately 200 users including scientists, Ph.D. students, and students in institutes, universities, museums, and scientific collections with zooarchaeology as field of research, as well as freelance zooarchaeologists in Europe. In the context of the JANUS[3] project of the *German Archaeological Institute*, Berlin, Germany, *OssoBook* will serve as a standard for the zooarchaeology domain in Germany. It contains internationally agreed standards for this branch of science. There are annually workshops in European countries which serve as an introduction to the application, as well as discussion forums on further development.

OssoBook already provided analyses for different scientific research questions, which were not able to be predefined in a framework like *xBook*. Therefore, a plug-in interface allowed the integration of analyses that were dynamically added to the application. Based on this development, it was already possible to implement different tools for data analyses that were integrated to *OssoBook* as plug-ins. [Lo12] [Ka12] The disadvantage of the plug-in interface was that it was only compatible to a specific database scheme of the *OssoBook* application. Each time the database scheme was updated, the plug-ins had to be updated as well. Most of the plug-ins became incompatible and were not further developed, so they could not be used any longer. Still, these analyses are very important.

We demonstrate the functionality of the *Analysis Tool* described in Chapter 2.3 on the basis of an analysis that was already implemented for the out-dated plug-ins interface of *OssoBook*, the age distribution of animals on base of the age determination of teeth [Ka12].

[3] http://www.ianus-fdz.de

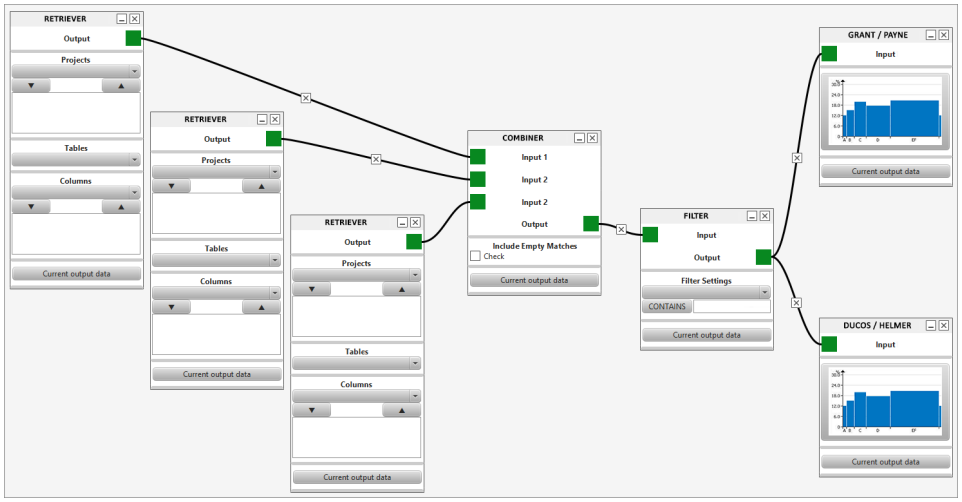


Fig. 3: The simplified composition of the age distribution of animals.

Age distribution of Animals: In zooarchaeology, the age distribution of animals at their slaughtering age is a frequent research question. The age distribution allows conclusions on the main usage purpose of these animals in the past, e.g. for milk, meat, fleece, or lamb production. There are different methods to achieve a result for this analysis. One is described by Grant [Gr82] and Payne [Pa73] and uses the description of the wear stage of the teeth. Another one is defined by Ducos revised by Helmer [He95] and uses the measurements of the crown of the teeth. Both of them distribute the animals dependent on the age of the teeth to groups. Using these improvements in zooarchaeological techniques, actual interpretation of slaughtering age profiles were developed which reflect animal management strategies. [HV07]

Composition in the Analysis Tool: The realization of the two methods consists of several, single Workers. Some of the default Workers can be reused for composing the necessary data. However it is necessary to implement an individual Worker for the calculation of the specific distribution of the animals into the groups. The compositions of the data, that are required to execute the individual calculations, are similar, but differ in one necessary information: While the approach of Grand/Payne uses the wear stage values, the one of Ducos/Helmer needs the measurements to calculate the distribution. So, most of the composition in the *Analysis Tool* can be reused for the two approaches. Due to the nature of the structure of the *Analysis Tool*, there are several possible options to generate the required data for the analyses. We describe a plain way, that allows to easily change the parameters that are used to filter the source data before running one of the two different analysis methods. First all information about the bone of the animal is gathered with three different Retrievers. The first Retriever fetches the species and skeletal element information for all entries, the second one gathers all wear stage information, and the third one gets all measurement information. In the second step all information is combined in a Combiner, that uses the primary key to find matching entries. The combined data can then be filtered in

a Filter. Of course it is possible to use multiple filters, if required. Finally, the filtered data is forwarded to both of the two individual Workers for the calculation of the age determination. These Workers were created inside the *OssoBook* source code and are registered to the framework over the API. The Workers require all necessary columns to be available to be able to carry out the analysis. Since the columns are all defined inside *OssoBook* and therefore are known beforehand, no mapping of the columns is needed and the Workers can easily check if all required columns are available. The final composition of the analysis in the *Analysis Tool* can be viewed in Figure 3.

Results: Once the composition is completed and all Workers are connected, the Worker for two approaches (after Ducos/Helmer and Grant/Payne) calculate the necessary data for the analysis. The Workers hold the result of the calculation which can be displayed in table form by clicking the button "Current output data". Furthermore, the Worker is extended with a functionality to display the data in a histogram, a typical representation of the data of the age distribution.

4 Discussion

In this paper, we explained the workflow of archaeo-related disciplines and described three main challenges of this area: The gathering, sharing and analysis of data. We first explained our solution for the digital data gathering in *xBook*, a dynamic and flexible framework for data gathering. We then explained the realization of the *Synchronization*, a function to work together with colleagues and share data with other scientists. Afterwards we described the *Analysis Tool* that we developed to be able to compose flexible and complex analysis with data from a database. Finally, we showed the functionality of the *Analysis Tool* by composing the age distribution of animals in the zooarchaeological database *OssoBook*.

While the *xBook* framework and the *Synchronization* are already used for scientific and inventory databases, there are also some possible improvements for the performance of the Synchronization process. A possible speed increase could be achieved by using compressing methods for the transmitted data. The usage of a special data type for the transmitted data would decrease the size of the data sent from server to client, or the other way around. Furthermore, the entries are currently only marked as "deleted" on the server, if a data set is deleted by the user. A solution to be able to delete the file sets in the database as well would decrease the necessary amount of data for the database.

The *Analysis Tool* is still at an early stage of development and there are still issues that could be addressed in the future, to make working with it go more smoothly. With the provided basic Workers, the *Analysis Tool* already allows a wide range of possible analyses. But the creation of further Workers would extend the possibilities of the *Analysis Tool*, especially Workers for predefined calculations, including statistical, mathematical or clustering algorithms. Besides, frequent used combinations of single Workers may be integrated as own Workers to increase the usability. Also the number of provided diagrams for the graphical representation of the data could be increased.

For a lot of analysis types, the provided Workers are sufficient to create analyses completely without programming skills. But still there are specific calculations that are not directly possible by using the existing Workers, like the calculation of the age distribution in the example in this paper (cf. Chapter 3). For creating a new Worker like these some programming skills are still required. In addition, access to the source code is also required to add a new Worker for specific calculations.

References

[Gr82] Grant, Annie: The use of tooth wear as a guide to the age of domestic ungulates. In (Wilson, Bob; Grigson, Caroline; Payne, Sebastian, eds): British Archaeological Reports British Series, volume 109, pp. 91–108. Book Publishing Inc., Oxford, 1982.

[Ha03] Harland, Jennifer F.; Barrett, Hames H.; Carrott, John; Dobney, Keith; Jaques, Deborah: The York System: An integrated zooarchaeological database for research and teaching. Internet Archaeology, 13, 2003.

[He95] Helmer, Daniel: Biometria i arqueozoologia a partir d'alguns exemples del Proxim Orient. Cota Zero, 11:51–60, 1995.

[HV07] Helmer, Daniel; Vigne, Jean-Denis: Was milk a "secondary product" in the Old World Neolithisation process? Its role in the domestication of cattle, sheep and goats. Anthropozoologica, 42(2):9–40, 2007.

[Ka12] Kaltenthaler, Daniel: Visual Cluster Analysis of the Archaeological Database OssoBook in Consideration of Aspects of Data Integrity and Consistency. Diplomarbeit, Ludwig-Maximilians-Universität Munich, 2012.

[Ka15] Kaltenthaler, Daniel; Lohrer, Johannes-Y.; Kröger, Peer; van der Meijden, Christiaan; Obermaier, Henriette: Synchronized Data Management and its Integration into a Graphical User Interface for Archaeological Related Disciplines. In: Design, User Experience, and Usability: Users and Interactions - 4th International Conference, DUXU 2015, Held as Part of HCI International 2015, Los Angeles, CA, USA, August 2-7, 2015, Proceedings, Part II. pp. 317–329, 2015.

[Ka16] Kaltenthaler, Daniel; Lohrer, Johannes-Y.; Kröger, Peer; van der Meijden, Christiaan; Granado, Eduard; Lamprecht, Jana; Nücke, Florian; Obermaier, Henriette; Stopp, Barbara; Baly, Isabelle; Callou, Cécile; Gourichon, Lionel; Pöllath, Nadja; Peters, Joris; Schiebler, Jörg: OssoBook v.5.4.3. München, Basel, 2016.

[Lo12] Lohrer, Johannes-Y.: Density Based Cluster Analysis of the Archaeological Database OssoBook in Consideration of Aspects of Data Quality. Diplomarbeit, Ludwig-Maximilians-Universität Munich, 2012.

[Lo16] Lohrer, Johannes-Y.; Kaltenthaler, Daniel; Kröger, Peer; van der Meijden, Christiaan; Obermaier, Henriette: A Generic Framework for Synchronized Distributed Data Management in Archaeological Related Disciplines. Future Generation Computer Systems, 56:558–570, 2016.

[Pa73] Payne, Sebastian: Kill-off patterns in sheep and goats: the mandibles from Aşvan Kale. Anatolian studies, 23:281–303, 1973.

[PDK11] Pokorná, Adéla; Dreslerová, Dagmar; Křivánková, Dana: Archaeobotanical Database of the Czech Republic, an Interim Report. Interdisciplinaria Archaeologica, Natural Sciences in Archaeology, 2:49–53, 2011.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 99

Die Gratwanderung zwischen qualitativ hochwertigen und einfach zu erstellenden domänenspezifischen Textanalysen

Cornelia Kiefer[1]

Abstract: Die Textanalyse ist zu einem entscheidenden Werkzeug in verschiedenen Domänen wie den Geisteswissenschaften, Naturwissenschaften sowie auch in der Industrie geworden. Eine der größten Herausforderungen bei domänenspezifischen Textanalyseprojekten besteht darin, das Wissen aus den Bereichen IT und Text Mining mit dem Wissen aus der Domäne zusammenzubringen. Viele Textanalysetoolkits werden deshalb speziell für den Gebrauch durch Domänenexperten ohne oder mit wenig IT und Textanalysewissen vereinfacht. In diesem Beitrag diskutieren wir, inwiefern diese Vereinfachungen zu Qualitätsproblemen bei der Analyse von unsauberen Daten führen können.

Keywords: Textanalyse, Datenqualität, Analysequalität, überwachte maschinelle Lernverfahren, Textanalyse in den Geisteswissenschaften.

1 Einleitung

Viele Fragen in den Geisteswissenschaften, Naturwissenschaften und in der Industrie können beantwortet werden, indem Informationen aus großen Textkorpora extrahiert werden [FS07]. So werden zum Beispiel in den Sozialwissenschaften, in der Biologie, Linguistik und in der Automobilindustrie Textanalysen verwendet, um Fragen aus den jeweiligen Bereichen zu beantworten. All diese domänenspezifischen Projekte müssen sich derselben Herausforderung stellen: das Expertenwissen aus IT, Textanalyse und der Domäne muss zusammengebracht werden, um die Fragestellungen adäquat beantworten zu können. Eine Möglichkeit, die Verschmelzung der Kompetenzen zu ermöglichen, besteht darin, Menschen in allen dreien oder zumindest zwei der Kompetenzen auszubilden [Co]. Ein weiterer Ansatz ist es, das notwendige IT und Textanalysewissen stark zu reduzieren und vereinfachte Analyseumgebungen zur Verfügung zu stellen, die es dem Domänenexperten erlauben IT und Textanalyse für die Beantwortung seiner Forschungsfrage zu nutzen. Zum Beispiel wurde der Leipzig Corpus Miner (LCM) für Sozialwissenschaftler ohne IT-Hintergrund entwickelt [LW16]. Eine einfache Verwendbarkeit von Analysetools sollte jedoch nicht auf Kosten der Qualität gehen. In diesem Artikel stellen wir zwei Datenqualitätsindikatoren mit Blick auf die automatische Analyse von Textdaten in Textanalysepipelines vor.

[1] Universität Stuttgart, Graduate School of Excellence advanced Manufacturing Engineering (GSaME), Nobelstr. 12, 70569 Stuttgart, cornelia.kiefer@gsame.uni-stuttgart.de

2 Motivation und Anwendungsszenario

Um Geisteswissenschaftlern zu ermöglichen, selbst Textanalysen durchzuführen, werden Textanalysen vereinfacht. In diesen vereinfachten Analysetools werden zum Teil voreingestellte Trainingskorpora verwendet. Diese Vereinfachung kann jedoch zu qualitativ schlechten Textanalysen und zu falschen Forschungsergebnissen führen, wenn unsaubere Daten für Fragestellungen in der Domäne analysiert werden sollen. Auch wenn unterschiedliche Texttypen wie etwa Bewertungen, Forumsnachrichten und Tweets aus den sozialen Medien in einem Stream in Echtzeit analysiert werden sollen, kann die Verwendung von Analysetools, die nur für einen Datentyp trainiert wurden, zu schlechten Ergebnissen führen. Diese Probleme können durch entsprechende Datenqualitätsindikatoren angezeigt werden.

Im Folgenden beschreiben wir ein beispielhaftes **Anwendungsszenario**, das diese Probleme aufzeigen soll: Ein Linguist möchte die Jugendsprache in sozialen Netzwerken analysieren. Hierzu werden unsaubere Chat,- und Twitterdaten betrachtet. Im ersten Schritt werden alle englischen Beiträge herausgefiltert. Auf den herausgefilterten Beiträgen werden nachfolgend einige Vorverarbeitungsschritte durchgeführt, wie etwa, Sätze zu segmentieren (=Satzsegmentierung), Sätze in seine kleinsten bedeutungstragenden Elemente zu zerlegen (=Tokenisierung), und jedem Token die korrekte Wortart zuzuordnen (=Wortartentagging). Danach kann nach Adjektiven oder Nomen gefiltert werden um schließlich eine Häufigkeitsverteilung zu berechnen und die Ergebnisse in den letzten Schritten zu visualisieren und manuell auszuwerten.

3 Verwandte Arbeiten

In [Ki16] legen wir den Fokus auf die Messung der Qualität von unstrukturierten Daten und schlagen automatische Metriken für Textdaten vor. In dieser Arbeit illustrieren wir zwei dieser Datenqualitätsindikatoren anhand eines Anwendungsszenarios aus den Geisteswissenschaften.

Während es zur Qualität von strukturierten Daten sehr viel Forschung gibt, etwa zu Dimensionen [WS96], sowie Methoden zum Messen und Verbessern der Qualität von strukturierten Daten [Se13], gibt es bisher noch kaum Forschung zur Qualität von unstrukturierten Daten. Die Notwendigkeit, Methoden für die Messung und Verbesserung der Qualität von unstrukturierten Daten zu finden, wurde jedoch erkannt [Sc12]. In [So04] werden vier Kategorien für Datenqualitätsdimensionen für unstrukturierte Textdaten und konkretere Indikatoren, wie z.B. lexikalische Ambiguität und der Anteil an Rechtschreibfehlern aufgelistet. Wir illustrieren hingegen zwei Datenqualitätsprobleme in einem Anwendungsszenario aus den Geisteswissenschaften und zeigen Probleme bei der Analyse von unsauberen Daten an Beispielen mit echten Textkorpora auf.

4 Messen der Interpretierbarkeit von Textdaten

In dieser Arbeit fokussieren wir die Datenqualitätsdimension Interpretierbarkeit. Diese messen wir über die Ähnlichkeit zwischen den vom Datenkonsumenten erwarteten Daten zu den Eingabedaten (vgl. [Ki16]). Die Eingabedaten sind dabei die zu analysierenden Daten und die erwarteten Daten können im Falle eines überwachten Klassifikators konkret über die verwendeten Trainingsdaten dargestellt werden. Die *Ähnlichkeit von Trainingsdaten und Eingabedaten* kann in diesem Fall mit Textähnlichkeitsmetriken berechnet werden, die die semantische, strukturelle oder stilistische Ähnlichkeit von Texten messen (für Textähnlichkeitsmetriken siehe etwa [DTI13]). Weiterhin gibt der *Anteil an noisy data* Aufschluss darüber, wie gut ein Standardtool für das Natural Language Processing, das zumeist saubere Daten erwartet, diese Daten interpretieren kann. Dieser Anteil kann z.B. über den Anteil an Rechtschreibfehlern, unbekannten Wörtern und Abkürzungen bestimmt werden.

5 Identifikation von Problemen der Interpretierbarkeit im Anwendungsszenario

In diesem Abschnitt zeigen wir Probleme auf, die mit Hinblick auf die oben beschriebenen Indikatoren bei der Erstellung der Analysepipeline zu dem Anwendungsszenario aus Kapitel 2 auftreten können. Die im Anwendungsszenario skizzierte Analysepipeline fassen wir in Abbildung 1 zusammen. Für jeden Schritt in der Analysepipeline muss die Ähnlichkeit zwischen den erwarteten Daten und den Eingabedaten gemessen werden, um Datenqualitätsprobleme mit Blick auf die Interpretierbarkeit der Textdaten feststellen zu können. Wir diskutieren im Folgenden beispielhaft die automatische Erkennung der Sprache („Sprache“ in Abbildung 1) und die automatische Annotation von Wortarten („Wortarten“ in Abbildung 1). Die Tools zu beiden Vorverarbeitungsschritten basieren auf überwachten maschinellen Klassifikatoren, die auf großen Mengen manuell annotierter Daten trainiert wurden.

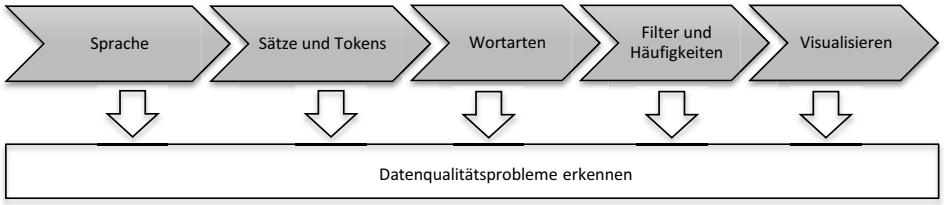


Abb. 1: Datenqualitätsprobleme können in jedem Analyseschritt in Textanalysepipelines auftreten

Im ersten betrachteten Vorverarbeitungsschritt, **„Sprache“**, soll für jeden Datensatz automatisch die korrekte Sprache erkannt werden. Da viele domänenspezifische Textanalysen unsaubere Daten verarbeiten müssen und die Domänenexperten die standardmäßig verwendeten Trainingsdaten zumeist nicht verändern, kann es zu einer

schlechten Erkennungsrate kommen. In Tabelle 1 stellen wir zur Illustration der Problematik die erreichte Genauigkeit (als prozentualen Anteil korrekt erkannter Sätze) für drei Spracherkenner, Apache *Tika*[2], den *language-detector*[3] und den *LanguageIdentifier*[4] für unterschiedliche Datentypen dar.

	Genauigkeit		
Datensatz	Tika	language-detector	Language Identifier
Zeitungstexte etc. (Penn Treebank[5], siehe [MMS93])	86	96	96
Prosa (Brown, siehe [FK79])	84	89	91
Tweets (Twitter Korpus, siehe [De13])	47	72	77
Chatnachrichten (NPS Chat, siehe [FLM])	20	22	33

Tab. 1: Genauigkeit der automatischen Filterung nach englischsprachigen Sätzen

Während Apache Tika ein Standardtool ist, das überwiegend auf sauberen Daten trainiert wurde, wurde der language-detector ebenfalls auf annotierten Tweets trainiert. Der LanguageIdentifier basiert auf [CT94], hier wurden Newsgroups als Trainingsdaten verwendet. Die Korpora sollten für dieses Beispiel jeweils Satzweise annotiert werden, wobei wir annehmen dass korrekterweise jeder Satz als englisch annotiert werden sollte[6]. Für saubere Daten, wie Zeitungstexte und Prosa liegt der Anteil an korrekt als Englisch erkannter Sätze bei allen drei Tools über 80%. Bei Tweets sinkt diese Rate erheblich für den Tika Spracherkenner, der saubere Daten erwartet (auf 47%) und weniger stark (auf 72% und 77%) für den language-detector und den LanguageIdentifier, die beide auch auf Tweets bzw. Newsgroups trainiert wurden. Für Chatnachrichten sinkt die Rate bei allen drei Erkennern erheblich, auf 20-33%. Bei den Chatnachrichten und Tweets sind insbesondere kurze Sätze mit einem hohen Grad an *noisy data* falsch klassifiziert worden. Zu den problematischen Sätzen zählen bspw. *„ah well“*, *„where did everyone gooo ?”, “RT @xochinadoll: I fel so blah today.”*. Die Messung des *Anteils an noisy data* sowie der *Ähnlichkeit zwischen Trainingsdaten und Eingabedaten* könnte diese Probleme indizieren.

Als nächsten beispielhaften Vorverarbeitungsschritt betrachten wir die automatische Annotation von **„Wortarten“**. In den Standardtools für die Wortartenannotation von Textdaten wird zumeist ein ausgewählter Trainingskorpus als Standard verwendet, der für die Analyse von sauberen Daten besonders geeignet ist. Wie schon bei der Komponente zur Spracherkennung kann eine Verwendung von Standardtools je nach Qualität der Daten zu Problemen führen. Zur Illustration der Problematik nennen wir in Tabelle 2 die erreichte Genauigkeit von verschiedenen Standardmodulen zur automatischen Bestimmung von Wortarten (als prozentualen Anteil der korrekt

[2] https://tika.apache.org/
[3] https://github.com/optimaize/language-detector
[4] Aus der Bibliothek DKPro Core: https://dkpro.github.io/dkpro-core/
[5] Verwendet wurde der in NLTK zur Verfügung gestellte Ausschnitt aus der Penn Treebank.
[6] Die Satzgrenzen wurden jeweils manuell bestimmt (bzw. es wurden entsprechende Goldannotationen für die Trennung der Korpora in Sätze verwendet).

zugewiesenen Wortarten von allen zugewiesenen Wortarten[7]). Wir haben drei Standardtools auf den unterschiedlichen Korpora getestet: Tool 1 ist der *NLTK* Maxent Treebank Tagger, Tool 2 ist der *Stanford* Tagger, und Tool 3, der *OpenNLP* Tagger. Im *Leipzig Corpus Miner* wird der OpenNLP Tagger für das Wortartentagging verwendet.

	Genauigkeit		
Datensatz	NLTK	Stanford	OpenNLP
Zeitungstexte etc. (Penn Treebank)	100	91	90
Prosa (Brown)	60	63	63
Tweets (Twitter Korpus)	65	67	70
Chatnachrichten (NPS Chat)	64	62	62

Tab. 2: Genauigkeit bei der automatischen Annotation von Wortarten

Sowohl die Goldannotationen als auch die vorhergesagten Wortarten basieren auf dem Penn Treebank Tagset [MMS93]. Tool 1 (NLTK) wurde auf der Penn Treebank trainiert, Tool 2 (Stanford) auf den Artikeln aus dem Wall Street Journal, die Teil der Penn Treebank sind und Tool 3 (OpenNLP) wurde auf eigenen Trainingsdaten von OpenNLP trainiert[8]. Bei sauberen Daten, wie etwa Zeitungstexten funktionieren die Standardtools einwandfrei, wohingegen bei unsauberen Daten wie Chatposts und Twitterdaten alle getesteten Tools versagen.

Wie in diesem Abschnitt anhand von zwei beispielhaften Vorverarbeitungsschritten aufgezeigt wurde, können bei jedem Analyseschritt in Textanalysepipelines Probleme auftauchen, die sich mit Vereinfachungen bzw. fehlenden Anpassungen durch Anwender ohne Textanalyseexpertise begründen lassen. Diese Problematik wird noch zusätzlich dadurch erschwert, dass die genannten Probleme in einer Analysepipeline propagieren und sich aufsummieren und letztlich die durch Textanalyse gewonnenen Antworten zur Forschungsfrage des Domänenexperten verfälschen können. Die illustrierten Probleme könnten durch eine Messung der Qualität der Textdaten, wie in Kapitel 4 vorgeschlagen, angezeigt werden.

6 Fazit und Ausblick

Um die Verschmelzung von Kompetenzen zu IT, Textanalyse und zu einer bestimmten Domäne zu ermöglichen, können vereinfachte Analyseumgebungen und Standardtools des Natural Language Processing eingesetzt werden. Die Vereinfachung von Textanalysetools sollte jedoch nicht auf Kosten der Qualität der Analysen gehen. In dieser Arbeit wurden hierzu zwei Datenqualitätsprobleme in einem konkreten Anwendungsszenario aus den Geisteswissenschaften illustriert. In der anschließenden Forschungsarbeit werden wir die entsprechenden Datenqualitätsindikatoren (*Ähnlichkeit*

[7] Es wurden jeweils die manuell annotierten Satz,- und Tokengrenzen verwendet.
[8] Zur Zusammenstellung letzterer Trainingsdaten gibt es keine näheren Angaben: https://opennlp.apache.org/documentation/manual/opennlp.html

von Trainingsdaten und Eingabedaten und *Anteil an noisy data*) und daraus abgeleitete Problemlösungen implementieren und den Ansatz in Experimenten validieren.

Literaturverzeichnis

[Co] Universität Jena: Computational and Data Science: Ein neuer Studiengang für eine neue Wissenschaftsdisziplin. http://www.cds.uni-jena.de/, 04.11.2016.

[CT94] Cavnar, W. B.; Trenkle, J. M.: N-gram-based text categorization. In Ann Arbor MI, 1994; S. 161–175.

[De13] Derczynski, L. et al.: Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data: Proceedings of the International Conference on Recent Advances in Natural Language Processing. Association for Computational Linguistics, 2013.

[DTI13] Daniel Bär; Torsten Zesch; Iryna Gurevych: DKPro Similarity: An Open Source Framework for Text Similarity: Proceedings of the Association for Computational Linguistics, Stroudsburg, USA, 2013.

[FK79] Francis, W. N.; Kučera, H.: Manual of Information to Accompany A Standard Corpus of Present-day Edited American English, for Use with Digital Computers. Brown University, Department of Lingustics, 1979.

[FLM] Forsyth, E.; Lin, J.; Martell, C.: The NPS Chat Corpus. http://faculty.nps.edu/cmartell/NPSChat.htm, 03.11.2016.

[FS07] Feldman, R.; Sanger, J.: The text mining handbook. Cambridge University Press, New York, 2007.

[Ki16] Kiefer, C.: Assessing the Quality of Unstructured Data: An Initial Overview. In (Ralf Krestel; Davide Mottin; Emmanuel Müller Hrsg.): CEUR Workshop Proceedings. Proceedings of the LWDA, Aachen, 2016; S. 62–73.

[LW16] Lemke, M.; Wiedemann, G.: Text Mining in den Sozialwissenschaften. Springer Fachmedien, Wiesbaden, 2016.

[MMS93] Marcus, M. P.; Marcinkiewicz, M. A.; Santorini, B.: Building a Large Annotated Corpus of English: The Penn Treebank. In Comput. Linguist., 1993, 19; S. 313–330.

[Sc12] Schmidt, A.; Ireland, C.; Gonzales, E.; Del Pilar Angeles, M.; Burdescu, D. D.: On the Quality of Non-structured Data. http://www.iaria.org/conferences2012/filesDBKDA12/DBKDA_2012_PANEL.pdf, 03.11.2016.

[Se13] Sebastian-Coleman, L.: Measuring data quality for ongoing improvement. A data quality assessment framework. Elsevier Science, Burlington, 2013.

[So04] Sonntag, D.: Assessing the Quality of Natural Language Text Data: GI Jahrestagung, 2004; S. 259–263.

[WS96] Wang, R. Y.; Strong, D. M.: Beyond accuracy: what data quality means to data consumers. In J. Manage. Inf. Syst., 1996; S. 5–33.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 105

Distributed FoodBroker: Skalierbare Generierung graphbasierter Geschäftsprozessdaten

Stephan Kemper[1], André Petermann[2] und Martin Junghanns[2]

Abstract:

Graphen eignen sich zur Modellierung und Analyse komplexer Zusammenhänge zwischen beliebigen Objekten. Eine mögliche Anwendung ist die graphbasierte Analyse von Geschäftsprozessen. Für die Entwicklung und Evaluierung entsprechener Analysetools werden Datensätze benötigt. FoodBroker ist ein Datengenerator, welcher vordefinierte Geschäftsprozesse simuliert und die Daten in Form von Graphen lokal auf einem Rechner erzeugt. Um Graphen beliebiger Größer erstellen zu können, zeigen wir in diesem Beitrag wie FoodBroker mit Hilfe der Open-Source-Frameworks GRADOOP und Apache Flink auf verteilten Systemen implementiert werden kann.

Keywords: Datengenerierung, Geschäftsprozesse, Verteilte Systeme, Gradoop, Apache Flink

1 Einleitung

Für die Entwicklung und Evaluierung von Tools zur graphbasierten Analyse von Prozessdaten werden Datensätze benötigt. Selbst für rein wissenschaftliche Zwecke ist es jedoch schwer an reale Geschäftsdaten zu gelangen. Dies kann an Datenschutzgründen liegen, aber auch daran, dass Firmen selbst gewinnbringende Erkenntnisse aus den Daten ziehen wollen. Eine Alternative ist die Verwendung eines Generators, der Daten mit realäquivalenten Eigenschaften erzeugt. FoodBroker [Pe14] ist ein solcher Generator, der auf der Simulation von konkreten Geschäftsprozessen basiert. Die aktuelle Implementierung unterstützt jedoch nur die Parallelisierung über mehrere Threads eines einzelnen Rechners und ist daher ungeeignet um Datenvolumen zu erzeugen, wie sie in sehr großen Unternehmen anfallen und für Skalierbarkeitsuntersuchungen neuer analytischer Verfahren benötigt werden. Hierdurch motiviert entstand der in diesem Beitrag beschriebene Ansatz, FoodBroker unter Verwendung von BigData-Technologien neu zu implementieren. Dies geschieht in der Absicht die horizontal skalierbare Ausführung auf einem Rechencluster zu ermöglichen.

Die vorliegende Arbeit beschreibt Distributed FoodBroker. In Kapitel 2 wird die FoodBroker-Simulation zunächst kurz eingeführt. Die für die verteilte Ausführung verwendeten Frameworks, sowie die Implementierung des Generators werden in Kapitel 3 beschrieben. Kapitel 4 fasst die Ergebnisse verschiedener Experimente zur Skalierbarkeit zusammen. In Kapitel 5 werden verwandte Arbeiten diskutiert und in Kapitel 6 wird der Artikel zusammengefasst.

[1] Universität Leipzig, Abteilung Datenbanken, kemper@studserv.uni-leipzig.de
[2] Universität Leipzig & ScaDS Dresden/Leipzig, [petermann,junghanns]@informatik.uni-leipzig.de

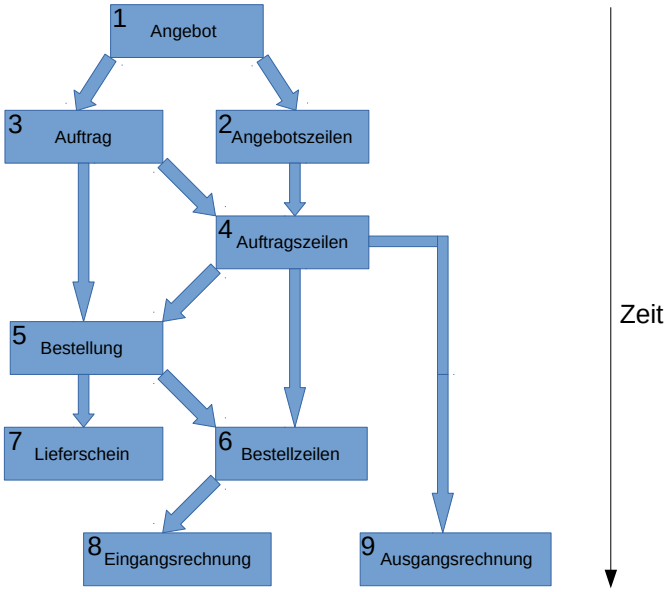


Abb. 1: Zeitlicher Ablauf des Brokerage-Prozesses. Die Reihenfolge der Erstellung wird durch die Zahlen wiedergegeben.

2 FoodBroker

FoodBroker simuliert vordefinierte Geschäftsprozesse eines Unternehmens, welches Nahrungsmittel handelt. Dabei werden Daten über alle wesentlichen Schritte, vom Angebot bis hin zur Lieferung und Tickets, z.B. aufgrund von Qualitätsmängeln oder verspäteter Lieferung, erstellt. Die Simulation erzeugt zwei Arten von Daten. Zunächst gibt es Stammdaten, die während der gesamten Simulation vorhanden sind und für jeden Kaufvorgang gleichwertig zur Verfügung stehen. Beispiele für Stammdaten sind Zulieferer, Kunden oder Produkte. Jedes dieser Objekte erhält einen der Qualitätswerte *gut*, *normal* oder *schlecht*, welche zur Steuerung der Prozesssimulation benötigt werden. Des Weiteren gibt es transaktionale Daten, die erst während der Ausführung eines Geschäftsprozesses erstellt werden. Hierzu gehören unter anderem das Verkaufsangebot, die Rechnung oder ein mögliches Ticket. Beide Arten von Daten können als Knoten und deren Beziehungen als Kanten eines Graphen modelliert werden. Ein vollständiges Schema aller Datenobjekte und deren Beziehungen ist in [Pe14] dargestellt.

Der Geschäftsprozess ist in zwei wesentliche Teilprozesse unterteilt: die Vermittlung (*Brokerage*) als Hauptvorgang und die optional nachgelagerte Problembehandlung (*Complaint Handling*). Der zeitliche Ablauf der Generierung transaktionaler Daten während des Brokerage-Prozesses wird in Abbildung 1 dargestellt. Des Weiteren zeigt die Abbildung die Abhängigkeiten der einzelnen Objekte voneinander. Betrachtet man die Ausgangsrechnung, so wird deutlich, dass diese vom Auftrag abhängt. Zeitlich gesehen wird sie jedoch erst zum Schluss erstellt.

3 Verteilte Ausführung

Nachfolgend wird zunächst eine kurze Einführung zu den BigData-Frameworks Apache Flink [Ca15] und GRADOOP [Ju15, Ju16] gegeben. Diese werden verwendet um die Ausführung FoodBrokers auf mehrere Maschinen zu verteilen.

Apache Flink ist ein Open-Source-Framework, welches für Big Data-Analysen eine Laufzeitumgebung zur Verfügung stellt. Diese ermöglicht es, analytische Programme durch Transformationen verteilter Objektmengen (sog. *DataSets*) als Datenfluss zu beschreiben. Die Ausführung eines solchen Programms wird von Flink koordiniert und automatisch auf die Maschinen eines Rechnerclusters verteilt. Mögliche Datenquellen und -senken für Flink-Programme sind bspw. das verteilte Dateisystem HDFS [Sh10] oder relationale bzw. NoSQL Datenbanken wie Apache HBase.

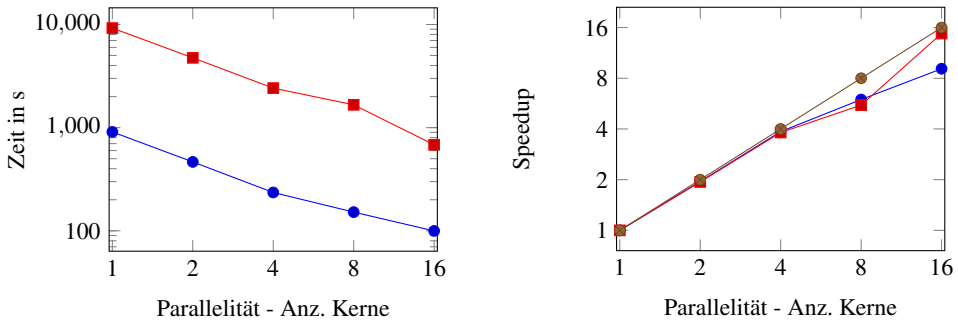
GRADOOP ist ein Open-Source-Framework, welches an der Universität Leipzig entwickelt und für die verteilte Analyse von Graphen eingesetzt wird. GRADOOP implementiert das schema-freie ***E**xtended **P**roperty **G**raph **M**odel* (EPGM) [Ju16] und ermöglicht hierdurch die Darstellung mehrerer, möglicherweise überlappender Graphen innerhalb einer Datenbank. So können die während einer Geschäftsprozess-Simulation erzeugten Daten zusammengefasst und als Graph, nachfolgend als *Transaktion* bezeichnet, gespeichert werden. Hierbei können Überlappungen enstehen, wenn ein Stammdatum in mehreren Transaktionen referenziert wird. Neben der Repräsentation bietet das EPGM analytische Graph-Operatoren, welche auf einzelnen Graphen und Graphmengen ausgeführt werden können [Ju16].

Um mittels FoodBroker eine verteilte Datengenerierung zu ermöglichen, müssen die zeitlichen Abhängigkeiten aller Objekte berücksichtigt werden. Hierdurch ergibt sich eine Aufteilung in drei Phasen. Zunächst müssen die Stammdaten erstellt und in DataSets gespeichert werden. Diese stehen in der nachfolgenden Brokerage-Phase zur Verfügung in welcher der Großteil der transaktionalen Daten erzeugt wird. Abhängig von den Ergebnissen der Transaktionen wird die dritte Phase, das Complaint Handling, aufgerufen. Anschließend müssen alle relevanten Daten aus den einzelnen Transaktionen mit den Stammdaten zusammengeführt werden.

Der **Vermittlungsprozess** (s. Abb. 1) bietet wenig Potenzial zur Parallelisierung. So könnte bspw. die Erstellung der Eingangs- und Ausgangsrechnung gleichzeitig stattfinden, da beide auf der gleichen zeitlichen Ebene liegen. In der bisherigen Implementierung[3] wird der Prozess linearisiert und mehrere Simulationen parallel (auf Prozessorebene) ausgeführt. Dieser Ansatz hat sich auch als beste Lösung für die Verteilung auf mehrere Rechenknoten herausgestellt. Hierzu wird die Flink-Transformation *MapPartition* verwendet.

Zu Beginn wird ein N-elementiges DataSet erzeugt, wobei jedes Element Startpunkt einer Distributed FoodBroker-Simulation ist. Durch den Aufruf der *MapPartition*-Transformation lässt sich die Ausführung des Vermittlungsprozesses gleichmäßig auf R Recheneinheiten verteilen. Jede Recheneinheit bearbeitet eine Partition des initialen DataSet und führt nun $N \div R$ Simulationen unabhängig aus. In jeder Simulation muss auf die

[3] `https://github.com/dbs-leipzig/foodbroker`, commit: 548e51d



(a) Ausführungszeiten bei einem Skalierungsfaktor von *100* (blau) und *1000* (rot).

(b) Speedup beider Skalierungsfaktoren im Vergleich zu linearem Speedup (braun).

Abb. 2: Ergebnisse

Stammdaten zugegriffen werden, was erfordert, dass diese an alle Rechner gesendet werden müssten. Um die zu verteilende Datenmenge zu minimieren, werden diese vorher mittels *Map*-Transformation auf ihre Id und den zugehörigen Qualitätswert reduziert und als Parameter an die *MapPartition*-Transformation übergeben. Innerhalb dieser Operation ist der Ablauf analog zur bisherigen Implementierung.

4 Auswertung

Im Folgenden wird die Performanz der verteilten Datengenerierung betrachtet. Hierbei wird sowohl die Skalierbarkeit hinsichtlich wachsender Datenmengen als auch unter Hinzunahme von Ressourcen evaluiert. Um die produzierten Datenmengen zu erhöhen wird in FoodBroker ein Skalierungsfaktor verwendet. Die Anzahl der simulierten Geschäftsprozesse hängt linear und die Anzahl der Stammdaten linear mit Dämpfungsfaktor von diesem Skalierungsfaktor ab. Die Parallelität wird abhängig von der Anzahl verwendeter Prozessorkerne angegeben. Für die Messung wurden jeweils 10 unabhängige Distributed FoodBroker Abläufe mit der Standardkonfiguration gestartet und die durchschnittliche Ausführungszeit ermittelt.

Das **Testsystem** ist ein Cluster von fünf PC-Systemen, jeder Rechner besteht aus einem Intel XEON W3520 (4 x 2.6 Ghz) und besitzt 6 GB RAM. Das Betriebssystem ist Ubuntu Server 14.04 (64-Bit) und es ist eine 500 GB Samsung Seagate HDD für die Datenspeicherung verbaut. Für die Ausführung verwenden wir Apache Flink 1.1.2 (1 Jobmanager, 4 Taskmanager), Hadoop 2.5.2 und GRADOOP Version 0.3.0-SNAPSHOT. Die Implementierung ist online verfügbar[4].

Die **Ergebnisse** sind in Abbildung 2 dargestellt, wobei Abbildung 2(a) die Ausführungszeit unter verschiedenen Parallelitätsgraden und Skalierungsfaktoren zeigt. Die Tests starten bei sequenzieller Ausführung, was einem Grad von *1* entspricht. Für die Generierung

[4] `https://github.com/dbs-leipzig/gradoop`
package : org.gradoop.flink.datagen.transactions.foodbroker

bei Skalierungsfaktor *100* werden ca. 909 Sekunden, bei Skalierungsfaktor *1000* ca. 9226 Sekunden benötigt. Wird die Parallelität auf *2* erhöht, so dauert die Ausführung im ersten Fall im Durchschnitt ca. 465 Sekunden und ist damit nur bedingt langsamer, als für eine lineare Skalierbarkeit zu erwarten ist. Bei einem Skalierungsfaktor von *1000* ist es ähnlich, hier werden im Schnitt ca. 4753 Sekunden im Vergleich zu 4613 Sekunden benötigt. Wird die Parallelität auf *4* erhöht, so erreichen beide Skalierungsvarianten eine um fast 50% kürzere Ausführungszeit im Vergleich zur vorherigen Messung. Ab einer Parallelität von *8* verringern sich die Ausführungszeiten im ersten Fall jedoch nicht mehr so stark. Bei einem Skalierungsfaktor von *1000* werden jedoch bis zu einem Parallelitätsfaktor von einschließlich *16* signifikante Erhöhungen in der Ausführungsgeschwindigkeit erreicht. Dies ist darin begründet, dass der Einfluss des konstanten Anteils in der Berechnung der Stammdatenanzahl abnimmt.

Betrachtet man den in Abbildung 2(b) dargestellten Speedup beider Ausführungsvarianten (blau, rot) im Vergleich zur linearen Steigerung (braun), so ist zu erkennen, dass sich die resultierenden Kurven bis zu einer Parallelität von *4* überlappen. Bei den anschließenden Parallelitätsgraden nimmt der Speedup bei einem Skalierungsfaktor von *100* immmer weiter ab. Neben dem genannten Verhältnis zu den Stammdaten kann dies an der geringen Anzahl erzeugter Daten je Kern liegen. Denn bei einem Skalierungsfaktor von *1000*, bei dem deutlich mehr Daten erzeugt werden, wird bei einem Parallelitätsgrad von *16* ein Speedup von ca. 13,5 erreicht. Es gibt also für jeden Skalierungsfaktor eine bestimmte Anzahl an Kernen, die verwendet werden sollte, um eine möglichst effiziente Generierung auszuführen.

5 Verwandte Arbeiten

Bisher existieren nur wenige Datengeneratoren für graphbasierte oder prozessorientierte Daten. Netzwerkgeneratoren im Allgemeinen sind bspw. im Bereich des Semantic Web vorhanden. Hier gibt es unter anderem den *Berlin SPARQL Benchmark (BSBM)* [Bi09] für die Simulation eines Internethandels, bei dem Händler Produkte anbieten und Kunden diese bewerten können. Des Weiteren gibt es beispielsweise *SP²Bench* [Sc08], welcher Zitiernetzwerke generiert, sowie *LUBM* [Yu05] für die Generierung einer Universitätsorganisation. Hierbei simuliert lediglich *BSBM* einen vollständigen Geschäftsprozess. Eine weitere wichtige Domäne für graphbasierte Generatoren sind soziale Netzwerke. Hier gibt es u.a. den *Social Network Benchmark (SNB)* der LDBC-Organisation [Er15] sowie *Graph500*[5]. Diese befassen sich jedoch nicht mit einer graphbasierten Geschäftsprozess-Modellierung, zur Datengenerierung für Analysetools, wie es bei FoodBroker der Fall ist. Durch die Einbindung von Distributed FoodBroker in GRADOOP ist es zudem möglich, Graphen verteilt zu generieren und diese anschließend direkt für die Evaluierung von Graph-Operatoren zu verwenden.

[5] `http://www.graph500.org`

6 Zusammenfassung

In diesem Beitrag wurde zunächst ein Überblick zur aktuellen Implementierung des Datengenerators FoodBroker gegeben. Der Fokus lag hierbei auf der Simulation des Brokerage-Prozesses. Da der Datengenerator die Ausführung nur auf einer Maschine unterstützt, ist die maximal mögliche Graphgröße limitert. Der Schwerpunkt dieses Papiers ist die Verteilung der Datengenerierung mittels FoodBroker unter Verwendung von Apache Flink und GRADOOP.

Durch unsere Experimente konnte gezeigt werden, dass sich Distributed FoodBroker annähernd linear skaliert. Bei erhöhter Parallelität mit geringem Skalierungsfaktor bricht der Speedup jedoch ein. Grund hierfür ist das Verhältnis zwischen einer hohen Anzahl an Stammdaten und wenig ausgeführten Simulationen. Das Verhältnis nimmt jedoch bei steigendem Skalierungsfaktor ab. Daher muss für jeden Skalierungsfaktor der Parallelisierungsgrad manuell festgelegt werden. Distributed FoodBroker ist somit auch für große Datenmengen und erhöhter Parallelisierung einsetzbar.

Literaturverzeichnis

[Bi09] Bizer, Christian; Schultz, Andreas: The Berlin SPARQL Benchmark. International Journal on Semantic Web & Information Systems, Vol. 5, Issue 2, S. 1–24, 2009.

[Ca15] Carbone, Paris et al.: Apache Flink™: Stream and Batch Processing in a Single Engine. IEEE Data Eng. Bull., 38(4), 2015.

[Er15] Erling, Orri et al.: The LDBC Social Network Benchmark: Interactive Workload. In: Proc. ACM SIGMOD. SIGMOD '15, ACM, New York, NY, USA, S. 619–630, 2015.

[Ju15] Junghanns, Martin et al.: Gradoop: Scalable Graph Data Management and Analytics with Hadoop. Bericht, University of Leipzig, 2015.

[Ju16] Junghanns, Martin et al.: Analyzing Extended Property Graphs with Apache Flink. Proc. SIGMOD, 2016.

[Pe14] Petermann, André et al.: FoodBroker - Generating Synthetic Datasets for Graph-Based Business Analytics. 5th Works. on Big Data Benchmarking (WBDB), LNCS 8991, 2014.

[Sc08] Schmidt, Michael et al.: SP2Bench: A SPARQL Performance Benchmark. CoRR, abs/0806.4627, 2008.

[Sh10] Shvachko, Konstantin et al.: The Hadoop Distributed File System. In: 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). S. 1–10, May 2010.

[Yu05] Yuanbo, Guo et al.: LUBM: A Benchmark for OWL Knowledge Base Systems. Web Semantics: Science, Services and Agents on the World Wide Web, 3(2-3), 2005.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 111

Effizienz-Optimierung daten-intensiver Data Mashups am Beispiel von Map-Reduce

Pascal Hirmer[1]

Abstract: Data Mashup-Ansätze und -Tools bieten einen einfachen und schnellen Weg, um Daten zu verarbeiten und zu analysieren. Über eine grafische Oberfläche können dabei – in der Regel grafisch – Datenquellen und Datenoperationen sowie der Datenfluss einfach modelliert werden. Hierdurch ergeben sich vor allem Vorteile durch einfache Bedienbarkeit durch Domänennutzer sowie einer explorativen Vorgehensweise. Jedoch legen vorhandene Data Mashup-Ansätze und -Tools wenig Wert auf die Effizienz der Ausführung, was dadurch begründet wird, dass durch Data Mashups in der Regel kleine Datenmengen verarbeitet werden. Zu Zeiten von Big Data gilt dies jedoch nicht mehr; schon scheinbar kleine Szenarien enthalten oftmals eine Vielzahl an Daten. Um mit diesem Problem zukünftig umzugehen, stellen wir in diesem Paper eine Konzeptidee am Beispiel von Map-Reduce vor, mit der die Ausführung von Data Mashups bzgl. Effizienz optimiert werden kann.

Keywords: Data Mashups, Map-Reduce, Big Data, Effizienzoptimierung

1 Einführung

In der heutigen Zeit steigen die Datenmengen immer weiter an, wodurch viele Vorteile durch die Generierung von Wissen aus diesen Daten gewonnen werden können [Mc12]. Zum Beispiel führen große Konzerne wie Facebook und Google komplexe Analysen auf einer riesigen Datenmenge durch, um den Nutzern personalisierte Werbung vorzuschlagen. Die Hauptherausforderungen sind hierbei die sich ständig ändernden Daten, deren Verteiltheit sowie deren Heterogenität. Jedoch kann wichtiges Wissen nicht nur von großen Konzernen sondern auch im Kleinen gewonnen werden. Insbesondere kleine Firmen sowie Privatnutzer sehnen sich nach einer einfachen Möglichkeit Daten zu verarbeiten sowie Analysen auszuführen. Um dies zu ermöglichen, wurden in der Vergangenheit viele Data Mashup Tools geschaffen, oftmals auch als ETL-Tools oder SPSS-Systeme bezeichnet [DM14]. Diese bieten in der Regel eine grafische Oberfläche basierend auf dem Pipes and Filters Pattern [Me95], um einen Datenflussgraphen bestehend aus Datenquellen sowie Datenoperationen (Filter, Aggregation, Analysen) zu modellieren. Dies ermöglicht nicht nur die Nutzung durch Domänenexperten, wie z.B. Wirtschaftsanalysten, sondern auch eine explorative Vorgehensweise bei der Ergebnisfindung. Das Datenflussmodell kann dabei leicht angepasst und erneut ausgeführt werden. Dies wird so lange wiederholt, bis das gewünschte Ergebnis eingetreten ist. Als Konsequenz kann dieses Ergebnis anschließend robust, z.B. als ETL-Prozess mit angeschlossenem Data Warehouse, umgesetzt werden.

Bestehende Data Mashup-Lösungen fokussieren sich nicht auf eine effiziente Ausführung sondern legen Wert auf die grafische Oberfläche und deren Bedienbarkeit. Des Weiteren

[1] Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Universitätsstraße 38, 70569 Stuttgart, Pascal.Hirmer@ipvs.uni-stuttgart.de

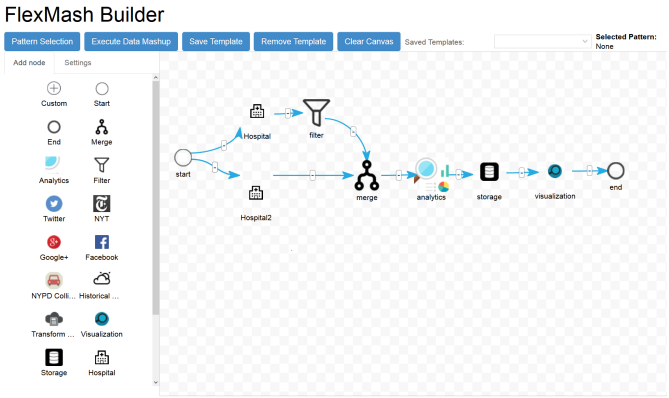


Abb. 1: Grafische Oberfläche des Data Mashup Tools FlexMash

ist die Datenmenge aufgrund von proprietären, nicht-skalierbaren Implementierungen oftmals stark begrenzt. Eine Verknüpfung der einfachen, benutzerfreundlichen Modellierung mit einer skalierbaren Verarbeitung sehr großer Datenmengen kann die Anwendbarkeit dieser Ansätze stark erhöhen. So können beispielsweise große Datenbanken angeschlossen werden. In dieser Arbeit stellen wir eine Konzeptidee am Beispiel von Map-Reduce vor, die den an der Universität Stuttgart entwickelten Data Mashup-Ansatz *FlexMash* sowie dessen prototypische Implementierung um eine optimierte, skalierbare Ausführung von Datenoperationen und -analysen erweitert. Dabei sollen für auszuführende Datenoperationen Map-Reduce Jobs erstellt werden, die anschließend verteilt auf einem großen Rechencluster ausgeführt werden können. Hierdurch kann die Effizienz der Ausführung stark optimiert werden. Eine ähnliche Vorgehensweise wurde bereits in verwandten Tools und Ansätzen verfolgt, z.B. durch eine Transformation von Datenfluss-Sprachen wie Pig oder JaQL auf Map-Reduce. Für die Domäne Data Mashups und deren grafische Modellierung von Datenflüssen wurde dies jedoch noch nicht umgesetzt.

Dieses Paper ist wie folgt aufgebaut: In Abschnitt 2 wird kurz der Data Mashup-Ansatz FlexMash beschrieben, der als Grundlagen für dieses Paper dient. Anschließend wird der Hauptbeitrag in Abschnitt 3 vorgestellt. Abschnitt 4 beschreibt verwandte Arbeiten und Abschnitt 5 fasst den Beitrag dieses Papers zusammen und gibt einen Ausblick auf zukünftige Arbeiten.

2 FlexMash

FlexMash [HM16] ist ein Data Mashup-Ansatz und eine Toolimplementierung entwickelt an der Universität Stuttgart, welches eine ad-hoc Modellierung von Data Mashups sowie eine explorative Vorgehensweise der Ausführung ermöglicht. Die Modellierung der Datenflussgraphen basiert auf dem Pipes and Filters-Pattern. Im Vergleich zu anderen Arbeiten bietet FlexMash einige Vorteile, darunter die Abstraktion von technischen Details bei der Modellierung durch sogenannte *Modellierungs-Patterns* [Hi15; HM16]. Diese ermöglichen

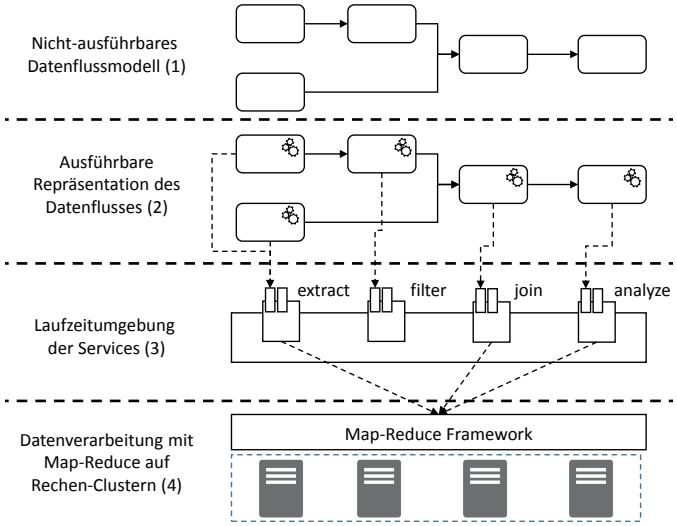


Abb. 2: Gesamtansatz zur Ausführung mit Map-Reduce

eine weitestmögliche Abstraktion, sodass Domänenexperten nur mit Konzepten arbeiten müssen, die sie auch kennen. Beispielsweise können Wirtschaftsanalysten Systeme und Programme modellieren (z.B. SAP-Systeme), mit denen sie täglich arbeiten und mit deren Daten sie vertraut sind. Ein weiteres Alleinstellungsmerkmal von FlexMash ist die flexible Ausführung der Datenverarbeitung basierend auf nicht-funktionalen Nutzeranforderungen. Nach der Modellierung kann der Anwender aus einem Katalog seine spezifischen Anforderungen an die Ausführung auswählen, beispielsweise das diese besonders robust gegen Datenverluste oder besonders sicher (verschlüsselt) durchgeführt werden soll. Die Art der Ausführung hängt dann von diesen Anforderungen ab, d.h., die Technologien und Softwarekomponenten werden dynamisch, baukasten-artig zusammengesetzt, um eine personalisierte Ausführungsumgebung zu schaffen [HM16]. Die grafische Oberfläche des Tools zur Modellierung der Datenflussgraphen ist in Abb. 1 zu sehen. In dem dargestellten Beispiel sollen Daten von zwei Krankenhäusern zusammengeführt und analysiert werden. Da eines der Krankenhäuser für die Analyse unwichtige Daten erfasst, werden diese vor der Zusammenführung gefiltert. Eine mögliche Analyse wäre bspw. zu untersuchen, welche Nebenwirkungen durch die Einnahme eines bestimmten Medikaments entstehen.

3 Effizienz-Optimierung daten-intensiver Data Mashups

In diesem Abschnitt wird der Hauptbeitrag dieses Papers beschrieben: eine Konzeptidee für die optimierte Ausführung von Data Mashups am Beispiel von Map-Reduce. Im bisherigen FlexMash-Konzept wird das nicht ausführbare grafische Datenflussmodell zur Beschreibung der Datenverarbeitung und -analyse in eine ausführbare Repräsentation überführt (bspw. in Workflowsprachen wie BPEL). Diese ausführbare Repräsentation ruft dann für jeden Datenoperator im Modell einen dazugehörigen Service auf, der die Daten verarbeitet. Dies

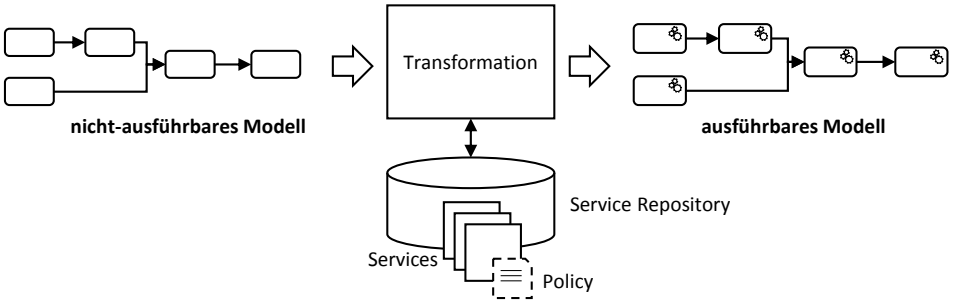


Abb. 3: Policy-basierte Auswahl der Services

skaliert jedoch nicht, da die Services die Daten immer als Ganzes verarbeiten. Um die erforderliche Skalierbarkeit zu erreichen wird daher eine neue Ebene der Datenverarbeitung eingeführt, die von den Services abstrahiert wird. Der Vorteil hiervon ist es, dass das bestehende Konzept nicht angepasst werden muss.

Dies ist in Abb. 2 dargestellt. Dabei ist zu sehen, dass mehrere Abstraktionsebenen existieren: auf oberster Ebene (1) liegt der nicht-ausführbare Datenflussgraph, der durch Transformation auf eine ausführbare Repräsentation abgebildet wird, die eine Ebene darunter liegt (2). Diese Repräsentation kann beispielsweise auf etablierten Workflow-Sprachen wie BPEL basieren. Bei der Ausführung dieser werden für jeden Datenoperator im Modell entsprechende Services aufgerufen (3), die die Daten verarbeiten. In bisherigen Ansätzen wird die Datenverarbeitung von jedem Service direkt selbst ausgeführt (z.B. in Java), wodurch die Skalierbarkeit jedoch stark eingeschränkt wird. In diesem Paper führen wir eine zusätzliche Ausführungsschicht auf unterster Ebene ein (4), die die Datenverarbeitung verteilt mittels Map-Reduce ausführen kann. Wie in Abb. 2 zu sehen ist, sollte jedoch nicht pauschal jede Datenoperation mit Map-Reduce ausgeführt werden. Für Datenoperationen mit geringer Laufzeitkomplexität (Filterung, Sortierung, etc.) macht es Sinn die Datenverarbeitung direkt in dem Service auszuführen, um den durch das Map-Reduce Framework entstehenden Overhead zu vermeiden. Jedoch muss dies dynamisch entscheidbar sein, da sich der Overhead ab einer bestimmten Datenmenge wiederum lohnen kann. Wichtig ist, dass die Daten für die Verarbeitung mittels Map-Reduce oftmals in einem verteilten Dateisystem abgelegt werden müssen (z.B. HDFS in Hadoop). Dies erfordert jedoch, abhängig von der verwendeten Verarbeitungsplattform, eine Verschiebung von Daten in das entsprechende Dateisystem. Der hierdurch resultierende Overhead und wie hiermit umgegangen werden kann wird in zukünftigen Arbeiten betrachtet.

Um zu entscheiden welche Services ausgeführt werden annotieren wir diese mit Policies (z.B. Servicebeschreibungen wie WS-Policys), die definieren für welche Art von Daten und für welche Datenmenge sie geeignet sind. Bei geringen Datenmengen sowie einer geringen Komplexität kommen Services in Frage die Daten ohne die Verwendung von Map-Reduce verarbeiten, im umgekehrten Fall sollten diese Techniken wiederum angewendet werden. Die Entscheidung welche Services letztendlich für die Datenverarbeitung genutzt werden wird bei der Transformation des nicht-ausführbaren Modells in die ausführbare Repräsentation getroffen, dies ist in Abb. 3 dargestellt. Dabei werden die Art der Daten sowie

die Datenmenge mit den Policies geeigneter Services abgeglichen. Erfüllt eine Service-Policy alle Anforderungen, die durch die Daten gegeben sind wird ein entsprechender Service-Aufruf in die ausführbare Repräsentation eingefügt. Erfüllen mehrere Services eine Policy sollte die Auswahl entweder zufällig geschehen oder es müssen weitere Faktoren wie z.B. Kosten, Laufzeitkomplexität, benötigte Rechenressourcen (z.B. Größe der Cluster) etc. der Implementierung der Services hinzu genommen werden. Dies erfordert komplexe Kostenmodelle, die in Zukunft entwickelt werden. Erfüllt keiner der Services die Policy in vollem Umfang wird der bestmögliche Service ausgewählt. Wir fokussieren uns in diesem Paper nicht auf die genauen Charakteristiken der Map-Reduce Jobs, da die Datenverarbeitung mittels Map-Reduce bereits durch eine Vielzahl von Arbeiten abgedeckt wurde. Wir gehen davon aus, dass die Datenoperationen sowohl mittels Map-Reduce, als auch ohne umgesetzt werden können. Eine erste Untersuchung hat ergeben, dass dies auf die gängigen Datenoperationen (z.B. Filter) zutrifft.

4 Verwandte Arbeiten

Hadoop[3] ist eine bekannte Plattform für die Ausführung von Map-Reduce, welches auf dem Hadoop File System (HDFS), einem verteilten Dateisystem, arbeitet. Basierend auf dem HDFS ermöglicht Hadoop eine auf mehreren Rechenclustern verteilte Verarbeitung großer Datenmengen mittels Map-Reduce. In den letzten Jahren wurde Hadoop zum de-facto Standard für die Ausführung von Map-Reduce. Für eine erste Implementierung des in diesem Paper vorgestellten Ansatzes wurde Hadoop verwendet. Seit einiger Zeit sind jedoch moderne, effizientere Ansätze auf dem Vormarsch. Zu diesen gehören Apache Spark und Apache Flink[4]. Apache Spark ermöglicht eine deutliche Effizienzsteigerung im Vergleich zu Hadoop [GA15]. Des Weiteren bietet Apache Spark eine Vielzahl weiterer Features, die vor allem auch Grundfunktionen der Datenverarbeitung (Filterung, Aggregation) beinhalten. Außerdem ermöglicht Spark eine Unterstützung von strombasierter Datenverarbeitung, was eine höhere Abdeckung verschiedenartiger Szenarien ermöglicht. Eine weitere Plattform für effiziente Datenverarbeitung ist Apache Flink. Apache Flink ist auf strombasierte Datenverarbeitung fokussiert und bietet eine ähnliche Funktionalität wie Spark. Flink hat somit wie Spark einen Effizienzvorteil gegenüber Hadoop. Ein umfassender Vergleich von Spark und Flink ist in [Ma16] beschrieben.

Zusammengefasst ergibt diese Diskussion, dass es sinnvoll ist, abhängig vom Anwendungsfall, verschiedene Frameworks und Datenverarbeitungstechniken, z.B. strombasiert, zu verwenden. Durch den in diesem Paper beschriebenen Ansatz ist dies einfach möglich. Die Abstraktion der eigentlichen Datenverarbeitung durch Services erlaubt es, verschiedene Service-Implementierungen für dieselben Datenoperationen anzubieten. Diese können dann beispielsweise mit Hadoop, Spark, Flink, oder anderen Plattformen ausgeführt werden. Wichtig ist, dass die Policys für jede zusätzlich unterstützte Plattform erweitert werden müssen. Dies kommt daher, dass diese Plattformen zusätzliche Funktionalitäten bieten könnten, die in den Policys repräsentiert sein sollten.

[3] http://hadoop.apache.org/
[4] https://flink.apache.org/

5 Zusammenfassung und Ausblick

In diesem Paper wird eine Konzeptidee für eine effizienz-optimierte Ausführung von Data Mashups am Beispiel von Map-Reduce beschrieben. Dabei steht insbesondere die Transformation des nicht-ausführbaren Modells in die ausführbare Repräsentation im Vordergrund. Durch das Anheften von Policies an die Services, die definieren für welche Art Daten und für welche Datenmenge sie am besten geeignet sind, können wir die Transformation der nicht-ausführbaren Datenflussmodelle auf eine ausführbare Transformation dahingehen anpassen, dass immer die zeiteffizienteste Ausführung ausgewählt wird. Die Ausführung selbst kann dabei beispielsweise verteilt mittels Map-Reduce durchgeführt werden, wodurch die Effizienz vor allem bei großen Datenmengen optimiert werden kann.

In Zukunft werden wir die vorgestellte Konzeptidee konkretisieren sowie eine zugehörige Implementierung anfertigen. Dabei muss auch das Format der Policy genauer definiert werden sowie mehrere in Frage kommende Policy-Frameworks untersucht werden.

Literatur

[DM14] Daniel, F.; Matera, M.: Mashups - Concepts, Models and Architectures. Springer, 2014.

[GA15] Gopalani, S.; Arora, R.: Comparing Apache Spark and Map Reduce with Performance Analysis using K-Means. International Journal of Computer Applications 113/1, S. 8–11, März 2015.

[Hi15] Hirmer, P.; Reimann, P.; Wieland, M.; Mitschang, B.: Extended Techniques for Flexible Modeling and Execution of Data Mashups. In: Proceedings of the 4th International Conference on Data Management Technologies and Applications (DATA). 2015.

[HM16] Hirmer, P.; Mitschang, B.: Rapid Mashup Development Tools: First International Rapid Mashup Challenge, Revised Selected Papers. In. Springer International Publishing, Kap. FlexMash – Flexible Data Mashups Based on Pattern-Based Model Transformation, 2016.

[Ma16] Marcu, O. C.; Costan, A.; Antoniu, G.; Pérez-Hernández, M. S.: Spark Versus Flink: Understanding Performance in Big Data Analytics Frameworks. In: 2016 IEEE International Conference on Cluster Computing (CLUSTER). S. 433–442, 2016.

[Mc12] McAfee, A.; Brynjolfsson, E.; Davenport, T. H.; Patil, D.; Barton, D.: Big data. The management revolution. Harvard Bus Rev/, 2012.

[Me95] Meunier, R.: The pipes and filters architecture. In: Pattern languages of program design. 1995.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 117

Interactive Data Exploration for Geoscience

Christian Beilschmidt[1], Johannes Drönner[1], Michael Mattig[1], Marco Schmidt[2], Christian Authmann[1], Aidin Niamir[2], Thomas Hickler[2,3], Bernhard Seeger[1]

Abstract: Data-driven research requires interactive systems supporting fast and intuitive data exploration. An important component is the user interface that facilitates this process. In biodiversity research, data is commonly of spatio-temporal nature. This poses unique opportunities for visual analytics approaches. In this paper we present the core concepts of the web-based front end of our VAT (Visualization, Analysis and Transformation) system, a distributed geo-processing application. We present the results of a user study and highlight unique features for the management of time and the generalization of data.

Keywords: Visualization, Biodiversity, Scientific Workflows

1 Introduction

Recently, research has become increasingly data-driven. Researchers often form new ideas by exploring large databases and identifying interesting patterns, instead of collecting data with a concrete hypothesis already in mind. Visual analytics plays an important role in this approach. It provides the necessary tools for researchers in order to facilitate effective data exploration. In biodiversity research a large fraction of the data is inherently spatio-temporal, i.e. the position of objects can be represented in a coordinate system at a certain point in time. This makes it especially appealing for a visual analytics approach, as spatial data can naturally be visualized on a map.

While data-driven research offers many new scientific opportunities, it also poses challenges for users regarding data integration, cleansing, filtering and lineage. First, there is a multitude of publicly available heterogeneous data which users want to combine, possibly also with their own data. There are different types of data, e.g. vector and raster data, and different reference systems for space and time. Second, data often appear as time series and computations need to take this into account in order to produce valid results. Third, the size of individual data sets poses challenges as high resolution raster images easily exceed hundreds of gigabytes. It is not feasible to store and process such data on regular desktop hardware using standard software. Fourth, specific subsets of data have often quality issues. An appropriate visualization has to support identifying errors and relevant subsets. Fifth, the flexible composition of workflows via a user-friendly interface makes it difficult

[1] University of Marburg, Dept. of Mathematics and Computer Science, Hans-Meerwein-Str., 35032 Marburg, {authmanc, beilschmidt, droenner, mattig, seeger}@mathematik.uni-marburg.de
[2] Senckenberg Biodiversity and Climate Research Centre (BiK-F), Senckenberganlage 25, 60325 Frankfurt am Main, {firstname.lastname}@senckenberg.de
[3] Department of Physical Geography, Goehte University, Altenhöferallee 1, 60438 Frankfurt am Main, Germany

for users to keep track of the data lineage. In particular, this poses challenges in correctly citing the source data. To the best of our knowledge, we are not aware of a single system addressing all these challenges for biodiversity science.

Our Visualization, Analysis and Transformation system (VAT) [Au15a; Au15b] aims to support such an interactive data exploration for biodiversity data. It consists of a back end for low-latency geo processing called MAPPING (Marburg's Analysis, Processing and Provenance of Information for Networked Geographics) and a web front end called WAVE (Workflow, Analysis and Visualization Editor). The main purpose of this system is to pre-process data and to export results for further analysis in custom tools. The user interface is of utmost importance in order to effectively enable data-driven science on large and heterogeneous data for scientific users with little background in information technology.

Hidden behind the user interface are so-called exploratory workflows representing a composition of atomic scientific tasks. WAVE creates these workflows on the basis of an intuitive user interface. Moreover, workflows can process data with provided building blocks for investigating different approaches. When users obtain meaningful results, all steps that led to their computation are available as workflows. These workflows can be stored, shared and adjusted for similar use cases on different data sets.

The VAT system is already in use in two ongoing projects. GFBio[4] [D+14] is a national data infrastructure for German biodiversity research projects. It offers an archive for long-term access in order to facilitate data sharing and re-usage. VAT provides added value services for exploring and processing the data sets of the GFBio archives. Idessa[5] deals with sustainable range-land management in South African savannas. Here, the VAT system is used as a toolbox for implementing a web-based decision support system for farmers to avoid land degradation.

The main contributions of this paper are: we present an interface for exploratory workflow creation, effective data generalization and previews, linked time series computations, and automatic provenance and citation tracking.

The rest of the paper is structured as follows. First, we discuss briefly in Section 2 the motivation of building a new system by shortly presenting other work in this area. Then, we introduce the design of the user interface of VAT and describe some aspects in more detail, like our mechanisms for data generalization. Section 4 presents an evaluation of our interface by discussing a user study. Finally, Section 5 concludes the paper.

2 Related Work

There is an ongoing scientific interest in interactive map applications [AAG03; Ro13]. This stresses the importance of immediate responses of operations. Current visual analytic approaches [S+13] follow this approach but lack the ability to track workflow provenance and modify configurations.

[4] www.gfbio.org
[5] www.idessa.org

Typically, data processing in geo sciences is either done using scripting languages like R or Geographic Information Systems (GIS) like QGIS[6]. Writing R programs requires knowledge of the language and the required packages. The development takes time and the processing speed is limited. GIS offer a graphical user interface that requires less programming skills. However, desktop GIS also suffer from slow processing as they are limited to local resources and do not exploit modern hardware sufficiently well [Au15b]. Workflow builders for GIS do not support an exploratory usage. To the best of our knowledge there exists no GIS that support time series as a core concept.

Web-based applications allow for ubiquitous access and are able to provide more processing power than desktop applications. There are specialized applications like Map Of Life [JMG12] that aim to solve specific use cases very well. More general functionality is provided by cloud-based GIS like CartoDB[7] and GIS Cloud[8]. However, the processing capabilities of these systems is still limited as they mainly focus on map creation rather than scientific processing tasks.

Workflow systems like Taverna [W+13], Kepler [A+04] and Pegasus [M+13] are building workflows upfront, with the goal of executing them on multiple data sets. This is contrary to our approach of exploratory workflows that are transparently built in the background during data exploration. Additionally, they do not allow web-based workflow creation, offer little geo functionality and limited processing throughput [Au15b].

3 WAVE: A Workflow, Analysis and Visualization Editor

This section describes our system's user interface. It starts with an overview of the fundamentals. Then, it discusses different concepts for solving the challenges introduced in the Introduction. This covers methods for data generalization and temporal support as well as project and citation management.

3.1 Basic Principles

The fundamental idea for WAVE is to offer an intuitive web-based user interface for interactive exploration of biodiversity data. Users can select data from a data repository and upload custom files. They perform operations for filtering or enriching data by combining them with other sources of information. Finally, they export data for further analysis in their preferred custom tools on a different system. The complete workflow of computations is always accessible and allows a reproduction of results anytime later.

Biodiversity data is mostly spatio-temporal. The system thus supports temporal collections of data from one of the following types: points, lines, polygons and rasters. Examples for such data are occurrences of species as points, roads as lines, country borders as polygons

[6] www.qgis.com
[7] www.carto.com
[8] www.giscloud.com

and environmental variables like elevation as rasters. The system treats every collection as a time series. Points, lines and polygons are all homogeneous object collections. Every object has an associated time interval expressing its validity. In a raster every cell shares the same temporal validity.

Computations on the available data are specified as workflows. They describe a dataflow from source collections over processing operators to a final result. A JSON representation makes the workflows storable and shareable. WAVE builds a workflow on-the-fly in the background when a user performs actions on selected data. It thus keeps track of all the applied processing steps. A query consists of a workflow, a spatial window and a reference time that selects the temporal valid objects. Queries are processed on MAPPING, the back end of the VAT system.

The central part of the data visualization is a panel with a map that consists of multiple layers of geographic data and shows them in their spatial and temporal context. The map supports different projections and allows for panning and zooming. It is linked to a data table that contains further information about non-spatial attributes. All data, either from sources or results of computations, are represented as separate layers. Users can also specify the order in which layers are drawn on the map. Layers refer to a workflow that is part of the query processed in MAPPING. They also serve as inputs for operators in order to create a new workflow (c.f. the operator list above the map panel in Figure 1 for examples). Beside layers, a workflow can also output plots, e.g. histograms or scatter-plots.

Layers and plots are linked to a component that allows the selection of the temporal reference. Here, users specify the point in time that is transferred to MAPPING as part of the spatio-temporal context. A change of the temporal reference triggers a re-computation of all layers and plots. WAVE supports a video mode for which the user specifies a time interval. Then, the computation slides over the interval, continuously producing the outputs of layers and plots. This effectively visualizes the changes in the data over time.

Adding a new layer consists of either selecting data from a source or applying an operator on one or more existing layers. In order to allow users to easily combine their own data with important environmental information, WAVE offers an interface to access a repository of raster and vector data hosted by MAPPING. In addition, users may upload their own data represented in the popular CSV data format. Operators allow the selection of appropriate data sets as inputs. One of the benefits of WAVE is that there is a check whether inputs fit to operators. If not, it tries to transform the data, e.g. adapt to the coordinate system, to make it compatible. By automatically applying such transformations, users can easily integrate heterogeneous data sets in their scientific workflows.

Figure 1 shows a map of an application containing three layers and two plots. Here, import dialogues were used to gather repository and custom data of African and Forest Elephants. The user applied filters for cleansing the vector occurrence points with a polygon and removed outliers by applying numeric range filters. Additionally, combination operators added environmental data from rasters to the vector occurrence points. There is support to create statistics by either using built-in operators or calling user-defined R scripts. It is

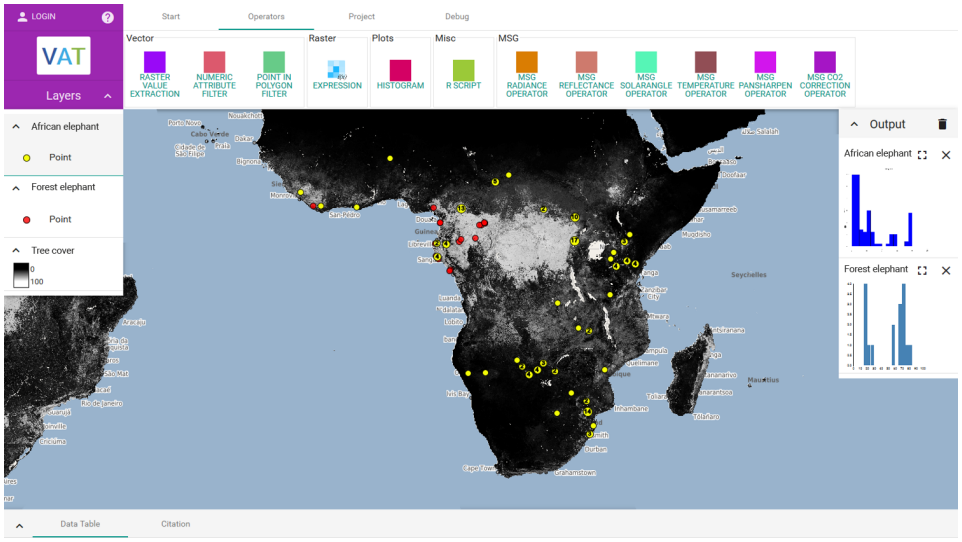


Fig. 1: Screenshot of the use case result

possible to export the output of our workflows as files. A more convenient approach is our R package that supports calling the workflows directly from a user program.

Users can work in multiple projects, each of them offers its specific data sets, workflows, layers and plots. The auto-save feature of WAVE ensures that projects are always up-to-date. A flexible rights management allows sharing projects within a team. In addition, projects can be published to other users with the option to restrict the right of changing the workflows and other aspects.

3.2 Data Generalization

There are two major objectives when interactively visualizing data for the user: One is to compute results in a near-realtime fashion for the user's exploration experience. The other is to provide an abstraction of the data that facilitates detecting interesting patterns. Data generalization can address both concerns.

The generalization of raster data is possible by aggregating multiple adjacent cells and representing them in a lower resolution. This requires less storage but comes at the expense of losing information. However, the amount of visible cells is naturally limited by the amount of pixels on the user's screen. Thus, it is sufficient to output raster images in this resolution for previews. Moreover, it is also sufficient to use source rasters and intermediate results of queries in this resolution instead of restricting the aggregation only to the results. This allows us to compute preview results with low latency. Users can afterwards trigger the computation in full resolution to produce scientifically valid results. In addition, the user can increase the accuracy of the data processing and the data visualization incrementally by simply zooming into interesting areas.

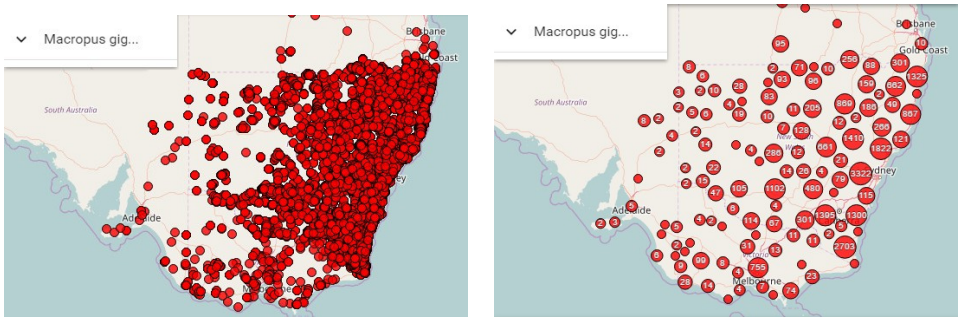


Fig. 2: Clustering a set of kangaroo observations in Australia to improve density information

For the generalization of vector data, a first popular approach is to transform vector data into a raster, but this goes along with a loss of attribute information. A second approach is to apply simplification techniques. Beside of being expensive, this also causes semantic changes of queries. We will examine these techniques in our future work.

WAVE offers an approach to generalizing big point sets to speed up their visualization and to identify cluster patterns. Displaying each point with its associated attributes exceeds the capabilities of current browsers on modern hardware even for sizes of less than one million points. Additionally, the size of transferring the data in the GeoJSON standard format stresses the internet connection of mobile devices. An example is 23,039 kangaroo (Macropus giganteus) occurrence points from GBIF[9], c.f. the left hand side of Figure 2. The uncompressed size with 20 common attributes is ~15 MB even for this relatively small data set. Furthermore, it is hard to recognize in the original plot that there is a very dense population of kangaroos in the south of Canberra. WAVE uses an adapted tree implementation of the hierarchical method developed by Jänicke, et al. [Jä12] to cluster data for the purpose of visualization. This allows combining nearby data points dependent on the zoom level and map resolution. By zooming in, the user gets a smaller excerpt of the map in more detail such that clusters break up and more details reveal. We represent the clusters as circles with logarithmically scaled area based on the number of included points. Additionally, the circles contain the number of points as labels. As circles are non-overlapping it is easy to identify clusters, c.f. the right hand side of Figure 2.

The tabular view is linked to the map and reacts to changes that occur in the layer as well. The view only has the possibility to present the clustered point data. Otherwise we would have to transfer all data which conflicts with data compression. Therefore, the table shows exactly the same data points as the map in the same resolution. The basic idea is to keep the attributes of the original data and to report an aggregate derived from the data in the cluster. For numeric attributes, we use the mean and standard deviation that can be computed in linear time. By zooming in, the amount of points in a cluster decreases and so does the standard deviation. This means the information gets more exact by diving into the data. For textual attributes, WAVE keeps a small number of representative points (typically three to five) for each cluster. Among the many options for selecting representative points,

[9] Global Biodiversity Information Facility: www.gbif.org

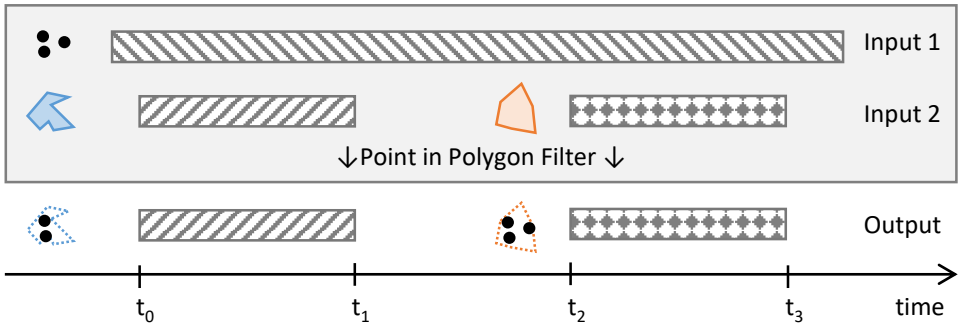


Fig. 3: Temporal point-in-polygon filter of three points and two polygons

we decided to use the points closest to the cluster center. The reason for our choice is that this information improves in accuracy when zooming in.

3.3 Support for Temporal Operations

Support for operations on time-series is a unique feature of the VAT system in comparison to other GIS-like systems. Our system supports the definition of a date and time as a global temporal reference. Recall that each query consists of a workflow with a spatio-temporal context, in particular a time interval that expresses the validity. The temporal reference slices a time series result such that it only contains elements that are valid at the given point in time.

For example, a user may add the WorldClim[10] mean annual temperature data set as a raster layer to a project. This data set is a time series that contains monthly climate variables. By means of the temporal reference the system is now able to choose the valid raster for the current month from the time series and add it to the map. Consequently, operations that include this data set also incorporate the correct raster with respect to the temporal reference. In comparison, traditional GIS oblige a user to manually add the correct raster from the data set beforehand. This is obviously a cumbersome and error-prone task.

The temporal validity of a data object is defined as an interval from a start time to an end time in which the object is incorporated into computations. When data with different validities are combined, data objects may have to be split into multiple items with different validities. Figure 3 shows an example of a point-in-polygon filter, where the output is a time series with two separate objects due to different validities of the polygons.

A problem arises when users want to combine data with non-overlapping temporal validity. One example is to compare measurements from today with measurements from the same day of the last year. In order to support this important type of temporal operations, WAVE offers a temporal shift operator to change the temporal validity of objects. A shift can either be absolute or relative. It is applied to a query first before any other processing

[10] www.worldclim.org

takes place. The result has to adapt its temporal validity to the temporal operation. In the previous example, after retrieving the measurements from last year, we have to set their validity to the current year in order to be able to compare them with each other.

When a user changes the temporal reference, WAVE triggers a re-computation of all views. Thus, there will be an update of the map, the layers, the connected table and the associated plots. The incorporation of temporal functionality in the user interface is still object of our future research. We currently only allow the specification of the temporal reference. Harmonizing the validity of different data sets is not yet possible. We will perform a user study in order to find an appropriate and intuitive way to extend our user interface for such kind of operations.

3.4 Provenance Information

Citations are very important for scientific work. They allow researchers to classify, comprehend and reproduce published results. They are also important for the publishers of those results, as they facilitate assessing the impact of their work. Aside from scientific results, also raw data has to be properly cited for the above reasons. Today, papers on topics related to data are encouraged by organizations and journals to facilitate data sharing. A recent article [BDF16] stressed the importance of citations in scientific data management.

Our system automatically keeps track of the citations of the involved data sets. We call the combination of (1) citation, (2) license and (3) a URI (e.g. link to a landing page) provenance information. All source operators are responsible for collecting the provenance information for the outputs, given a specific input. Processing operators combine the provenance information of their inputs via a duplicate eliminating union operation. This behavior can however be altered for specific operators.

The provenance information for a layer is always accessible by the user in WAVE. When the user exports a layer, a zip-archive is created. In addition to the actual data it contains two files. One file contains the workflow representing the computation of the layer, including all applied transformations. The other contains all the provenance information of the data sets involved in the computation.

4 Evaluation

We conducted a user study to gain insights about an appropriate user interface design beforehand to the product development. This included creating several use cases together with domain experts in biodiversity research to (1) create realistic scenarios and (2) cover as many concepts of WAVE as possible. We created a paper prototype to try out different interface variations. This allowed us stepping back from any implementation details and focusing on concepts on a sketch board. The advantage was that it is very inexpensive to discard doubtful concepts. And in conclusion this led to rapid concept development with domain experts.

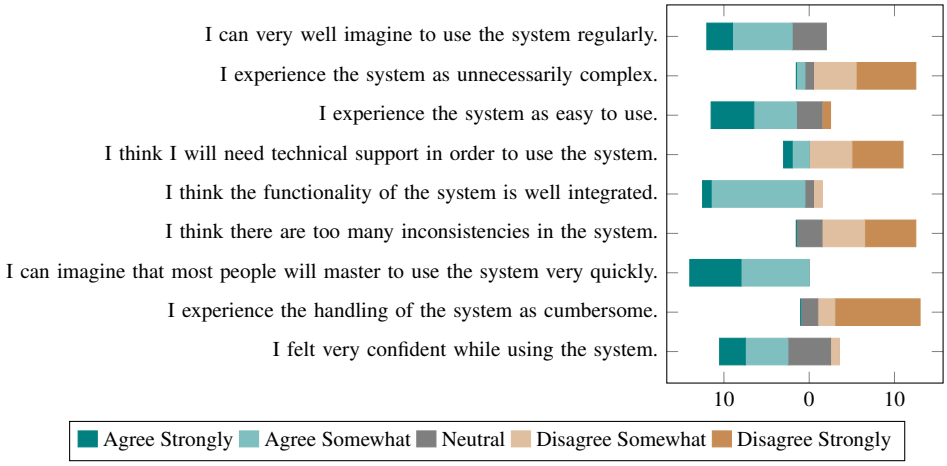


Fig. 4: Results of the use case of the user study regarding the paper prototype

The user study consisted of two parts. The first was an introduction of a use case and a 20 minute time span to solve a specific task. The users should work on the task independently without any system introduction or explanation. We observed the behavior and timed steps of certain sub tasks. The second part included a questionnaire of nine fixed questions and an additional field for free text comments. The participants had ten minutes time for this feedback. The questions aimed at different impressions about the system usage. As answers we used a symmetric typical five-level Likert scale with a neutral element.

We asked 15 users from the biodiversity domain at the Senckenberg Biodiversity and Climate Research Institute (BiK-F) in Frankfurt am Main to process our use case. Figure 4 shows the result of the study. Together with the additional comments (which we excluded here for space reasons) we did not find any reason for major changes in our design proposal. Nevertheless, we identified minor weaknesses and were able to get a better understanding of how users work with our system. One interesting fact to mention was the expectation of the users to interact with the application like in desktop GIS. This included right-clicking on elements to perform actions. This was a strong contrast to our previous experience in web application development.

5 Conclusion

We presented the design process, the fundamental concept and selected features of WAVE as the front end of the VAT system, a distributed system for supporting scientific geo applications. We conducted a user study to validate the design of the user interface. Several unique features including the support for temporal data, approaches for generalizing data, the integration of R scripts and the automatic management of workflows and citations distinguish WAVE and the VAT system from other solutions.

In our future work we will explore the possibilities of extending the system into an application builder. Here, users will be able to create custom applications using built-in functionality. For example, they create a workflow to solve a parametrized problem. Other users can then make use of this workflow in a custom user interface where they specify the required parameters without having the need to know the internals of the processing chain.

References

[A+04] Altintas, I.; Berkley, C.; Jaeger, E., et al.: Kepler: An Extensible System for Design and Execution of Scientific Workflows, In: 16th Int. Conf. on Scientific and Statistical Database Management, 2004. Proceedings. 2004, pp. 423–424.

[AAG03] Andrienko, N.; Andrienko, G.; Gatalsky, P.: Exploratory spatio-temporal visualization: an analytical review, Journal of Visual Languages & Computing 14/6, pp. 503–541, 2003.

[Au15a] Authmann, C.; Beilschmidt, C.; Drönner, J.; Mattig, M.; Seeger, B.: Rethinking Spatial Processing in Data-Intensive Science, In: Datenbanksysteme für Business, Technologie und Web (BTW) - Workshopband, 2015, pp. 161–170.

[Au15b] Authmann, C.; Beilschmidt, C.; Drönner, J.; Mattig, M.; Seeger, B.: VAT: A System for Visualizing, Analyzing and Transforming Spatial Data in Science, Datenbank-Spektrum 15/3, pp. 175–184, 2015.

[BDF16] Buneman, P.; Davidson, S.; Frew, J.: Why Data Citation is a Computational Problem, Commun. ACM 59/9, pp. 50–57, Aug. 2016, ISSN: 0001-0782.

[D+14] Diepenbroek, M.; Glöckner, F.; Grobe, P., et al.: Towards an Integrated Biodiversity and Ecological Research Data Management and Archiving Platform: The German Federation for the Curation of Biological Data (GFBio), In: GI: Informatik 2014 – Big Data Komplexität meistern, 2014.

[Jä12] Jänicke, S.; Heine, C.; Stockmann, R.; Scheuermann, G.: Comparative Visualization of Geospatial-Temporal Data. In: GRAPP/IVAPP, 2012, pp. 613–625.

[JMG12] Jetz, W.; McPherson, J. M.; Guralnick, R. P.: Integrating biodiversity distribution knowledge: toward a global map of life, Trends in Ecology & Evolution 27/3, pp. 151–159, 2012.

[M+13] McLennan, M.; Clark, S.; Deelman, E., et al.: Bringing Scientific Workflow to the Masses via Pegasus and HUBzero, In: Proceedings of the 5th International Workshop on Science Gateways, 2013, p. 14.

[Ro13] Roth, R. E.: Interactive maps: What we know and what we need to know, Journal of Spatial Information Science 2013/6, pp. 59–115, 2013.

[S+13] Steed, C. A.; Ricciuto, D. M.; Shipman, G., et al.: Big Data Visual Analytics for Exploratory Earth System Simulation Analysis, In: Computers & Geosciences, vol. 61, Elsevier, 2013, pp. 71–82.

[W+13] Wolstencroft, K.; Haines, R.; Fellows, D., et al.: The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud, In: Oxford Univ Press, 2013.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 127

Towards Understanding Mobility in Museums

Golnaz Elmamooz[1] Bettina Finzel[2] Daniela Nicklas[3]

Abstract: Data mining techniques can provide valuable insight to understand mobility in museums. However, the results of such techniques might not be easily understood by the museum staff. In this paper, we propose a graph-based approach to model museum exhibitions, sensor locations, and guiding tasks. We further discuss how route-based trajectory mining can be adapted to work with this graph model and which challenges need to be addressed to cope with the graph dynamics and the continuous flow of sensor data. Based on the demands of two target groups, curators and visitors, three applications are proposed: a *museum graph editor*, a *mobile museum guide*, and a *curator decision support*. We propose an architecture for a platform that provides context information and data mining results to such applications. We claim that our proposed platform can cover many aspects and demands that arise in the museum environment today.

Keywords: mobility in museum, property graph model, museum knowledge management, incremental trajectory data mining

1 Introduction

Museum exhibitions are areas where mobility plays an important role. The museum staff designs the layout of exhibits so that an optimal visiting experience can be gained, often by assuming certain movement patterns or routes. In addition, they may offer audio guides, info boards, sign posts, and even mobile gaming apps [EM11, Va10] that should further help visitors to follow proposed routes or topics. Visitors may or may not be aware of such offers; they might only follow their interests and show a more or less predictable behavior. A case study analysis of an ambient intelligent museum guide [WE05] describes a museum as ecology: as an information ecology, it is a system of people, practices, values, and technologies in a local environment; and to understand that ecology, understanding visitor mobility is a key factor. Nowadays, a multitude of sensors could be used to automatically observe visitor mobility in a museum: light barriers at doors, cameras [Ru13], the usage of electronic guides [BF16], or dedicated sensors like in the work of Martella [Ma16]. Here, proximity sensors attached to visitors and exhibits were used to sense the fact that someone is close to an exhibit. All these data sources provide raw data of varying data quality. To gain information about the movement (represented as trajectories) and to derive useful knowledge out of such data, various trajectory mining techniques can be exploited. However, as has been demonstrated in [Ma16], the results of such techniques might not be easily understood by the museum staff. In addition, such techniques often work on maps

[1] Otto-Friedrich-Universitt Bamberg, Lehrstuhl fr Informatik, An der Weberei 5, 96047 Bamberg, golnaz.elmamooz@uni-bamberg.de
[2] bettina.finzel@stud.uni-bamberg.de
[3] daniela.nicklas@uni-bamberg.de

and coordinates. However, it might be tedious to model exact floor plans, exhibit locations and sensor locations for a whole museum, in particular since many aspects of such digital maps will change with new exhibitions. Hence, we propose a graph-based approach to model museum exhibitions, sensor locations, and guiding tasks (either for navigation or for location-based games). We further discuss how route-based trajectory mining can be adapted to work with this graph model and which challenges need to be addressed to cope with the graph dynamics. Based on the demands of two target groups in the museum, curators and visitors, two types of application are needed. First, a *mobile museum guide* for the visitor to find their way easily and, secondary, a *curator decision support* and a *museum graph editor* that helps the curators organize the museums in an efficient way. The paper is organized as follows. In Section 2 we propose the main architecture of our platform. In Section 3, we describe the features of the museum graph model designed according to two main challenges: the dynamic environment and the public environment in a museum. We use museum graph model to provide context information for mobility model management which we present in Section 4. We show that different preprocessing and mining techniques take use of that context information. We conclude with Section 5 and give an outlook on future work.

2 Architecture Overview

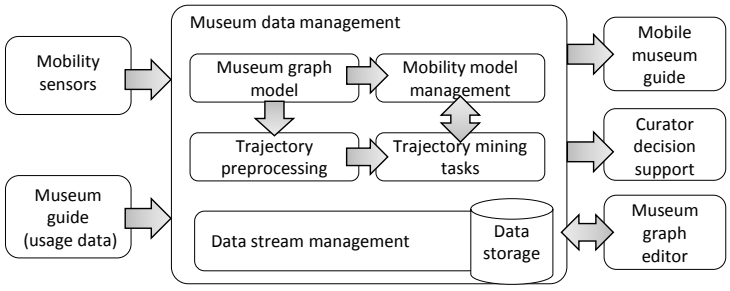


Fig. 1: Architecture overview

To manage mobility data we propose an architecture as depicted in Figure 1 to support three applications (right side): *Mobile museum guides* are mobile multi-media applications that inform visitors about exhibits, propose tours, and could contain scavenger-hunt-like games. In addition, by recognizing the mobility pattern and interests of the current user, such guide can individually recommend certain exhibits, topics, or tours ("'visitors who liked exhibits A and B also liked room C"'). The *Curator decision support* helps the museum staff to understand mobility. It is a domain-specific data analysis application that visualizes the results of various data analysis like popularity of exhibits, most popular paths, or classes of visitor behavior. Finally, we offer the *Museum graph editor* that provides an easy way for the museum staff to model exhibitions, game tasks, and room layouts. The *Museum data management* continuously processes incoming data from *Mobility Sensors* like cameras or WIFI trackers and from usage of the *Mobile museum guide*. Locations of visitors are *preprocessed* and mapped onto the *Museum graph model* (as provided by the

corresponding editor) that represents the location of rooms, exhibits, and sensors. The results are semantic trajectories that are fed into various *Trajectory mining tasks* to produce mobility models like frequent mobility patterns or trajectory clusters. Since the museum graph model changes whenever new exhibits or game tasks occur, we need to manage these models so that they are compliant with the exhibition. For this, changes in the museum graph model are published to the *Mobility model management* which controls the mining tasks. In the following sections, we give more details on the Museum graph model and the Mobility model management.

3 Museum Graph Model

Based on previous evaluations on geometric and symbolic location models [BD05] we propose an attributed graph-based location model of the museum environment to represent the context information which is used in analysis of mobility inside the museum. We need to model exhibits, visitors, tasks, areas, sensor locations, routes and the spatial and non-spatial relationships between such assets and objects. Assets and objects are modelled as nodes. Relationships are represented as edges. Both, nodes and edges, can be further attributed. Relationships that express containment and connectedness are sufficient and necessary to answer common spatial queries [BD05] but to enable analysis that goes beyond spatial relationships, further properties like topics or temporal availability need to be included. We therefor chose the property graph approach [Su15, Da14] to design the model. For the implementation, Neo4j[4] will be used. Figure 2 shows an example graph that could be implemented as part of a bigger graph representing the whole museum. The graph consists of seven nodes: three are of type *POI* (point of interest), two of type *location*, one of type *passage* and one of type *activity*. Passage refers to objects connecting locations, like doors, stairs or elevators. Activity refers to different actions a visitor can take in the museum, like tasks related to exhibits. The nodes are attributed with subtypes, names and descriptions to ease the retrieval of individual nodes or sets of nodes. Each node has an unique ID and some human-readable values for each attribute. The nodes are connected through different kinds of relationships.

Figure 2 shows relationships that will be mainly used in our graph: *connected*, *inside* and *assigned*. In the example graph, two rooms – the entrance hall and the room named Vogelsaal – are *connected* by a door. One POI *inside* the entrance hall could be the cash desk, where visitors pay for their visit. In the Vogelsaal, two exhibits (vitrines) which show some birds and animals, are located. The *mobile museum guide* could offer some tasks to the visitors to increase their museum experience. Here, we *assigned* a task to the second vitrine, containing padded birds. For example, the visitor could be asked by the mobile museum guide to estimate the number of feathers as soon as she passes the vitrine.

The example shows that the graph model can be easily understood by humans. Curators and museum education officers should be able to understand the model as they might want to create and adapt it via the museum graph editor. A graph-based model provides a

[4] https://neo4j.com/

well suited representation and has a medium modelling effort compared to other location model approaches [BD05]. Furthermore the model can help to understand the reasons behind certain mobility patterns of visitors in the museum. A visitor might spent some time infront of the second vitrine to count the feathers.

Regarding museum data management, we face two main challenges: (i) the environment is dynamic, and (ii) the environment is public. Challenge (i) results in a variable frequency of updates, updates on different parts of the model (e.g. a property "'accessible"' is added for nodes of the type "'room"') and changing availability (e.g. a relationship exists only within a certain time interval). Because of challenge (ii), we need to deal with heterogeneous user groups and we need to integrate privacy-preserving methods into the data management platform [St16]. The graph model is scalable and adaptable to changes in the museum. Properties, nodes and relationships in the instantiated graph can be easily added or removed. However, such changes over time and within space must be considered in further processing steps. To control the knowledge that is stored in the graph, restricted relationships and unique nodes can be defined. Thus, the dynamics of the graph itself can be limited. In the future we will evaluate features of other graph variants and compare them to our property graph model. For improving the scalability of the graph and for enabling the organization of entities in subgraphs, we intend to use multiple relation-based layers [Bo12]. For investigating how fine-grained semantical information can be included into our model (e.g. exhibit B is between exhibit A and exhibit C) we will consider hypergraphs [ERV06].

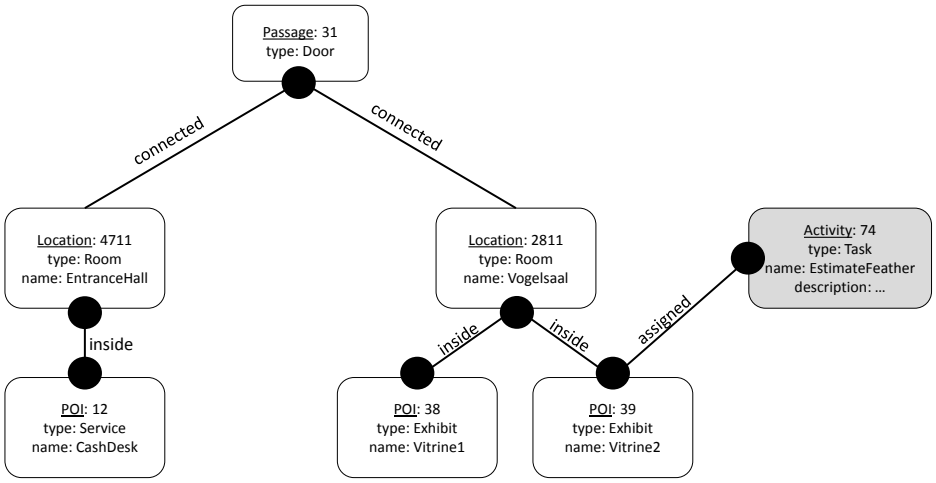


Fig. 2: Museum graph example

4 Mobility Model Management

In contrast to the museum graph model which represents the context information, mobility models represent the knowledge on movement of visitors. Examples for mobility model

are frequent mobility patterns or trajectory clusters. The two main applications use such mobility models to provide their service. Curators can arrange their museum in a way to have a constant distribution of visitor population instead of some crowded area and some empty rooms. In addition, recommendations that are based on mobility models can help visitors find a specific exhibit in a shorter time and choose the best path based on their interests.

We use the information gained from Museum graph model as context information to preprocess the observations from mobility sensors and the usage data from the mobile museum guide. Therefor we can extract relations and points that build primary routes. For example, considering the attribute "'accessible"', we know that a potential route through a room is possible or not. Combining primary routes together with preprocessing techniques like stay point detection and map-matching lead to more accurate results [Kh08]. Additionally, we can use trajectory segmentation technique when we need to consider trajectories in some particular time or particular exhibition in the museum. We develop trajectory mining tasks such trajectory clustering and trajectory classification based on preprocessed semantic trajectories to produce several mobility models. Finally, we need to manage all the knowledge obtained from different mobility models to serve the applications.

Trajectory clustering in the museums is the process of finding frequent trajectory patterns shared by visitors and of grouping similar trajectories into clusters. Trajectory clustering has been presented for clustering the trajectories both in free space and in a route-based setting [Zh15]. As we mapped the information of the museum environment into a graph model, our clustering task is based on the trajectories in route-based setting. According to the fact that a museum is a changing environment and also the interests of visitors could change during time intervals, we need to implement incremental trajectory mining algorithms [Li10]. In such approaches, first a trajectory segmentation technique is applied on trajectories and then the last segment of trajectory in a time window can be considered. One of the benefits of this method is that if the visitors had different paths before the start of the time window, it does not affect the results of clustering within the time-window, and we can focus on analyzing recent movement patterns [SZM16]. For example, we can study the direct effects of some events or changes to the Museum graph model on the distribution of visitor population.

Based on the example mentioned in the previous section, we can apply stay point detection as preprocessing step and after extracting stay points from single trajectories we can apply clustering techniques to define different clusters of stay points based on duration or number of detected stay points. In addition, using points of interest defined in graph model can help us to find stay points easier. However, in stay point detection step we may find some other stay points which are not defined in the graph model as POI. For clustering the stay points incrementally, we define a window size and consider the stay points just within a time window. As described before, in the entrance hall there is a cash desk where all of the visitors should stop, even for few seconds, to buy a ticket for the museum. In this scenario, by applying stay point detection on single trajectories and clustering these points based on number on detected stay points we can extract three main stay points: a stay point in front of the cash in the entrance hall and two stay points in front of the vitrines in the Vogelsaal.

In incremental clustering, events can change the results of clustering during different time intervals. For example, a curator adds the activity of feather estimation at time t_2. Later, at time t_3, she is interested in the effects of this change in the mobility models.

As illustrated in Figure 3, we can define two time intervals based on the graph changes: one before t_2 and one from t_2 to now (which is t_3). After adding the activity, visitors become more interested in the birds in vitrine 2. Therefore, the number of detected stay points in the second time window increases. On the other hand, since there are no changes to the entrance hall, we do not define a new time window for this room. In the example, 20 visitors visit the museum within the time interval t_1 to t_3. We know all of the visitors stop at cash desk, so we can use this stay point to count the number of all visitors. From t_1 to t_2, 8 visitors are interested at vitrine 1 and 3 visitors are interested in vitrine 2. At time t_2 by changing the graph model of Vogelsaal the number of detected stay points for vitrine 2 changes. The number of detected stay points from t_1 to t_2 and from t_2 to t_3 is shown in table 1.

Based on the information obtained from preprocessed semantic trajectories we can classify different kinds of visitors based on their visiting style and time schedule. For the same reason mentioned for clustering techniques, like changes in museum environment and the interests of visitors, we use incremental classification [Le08]. The obtained knowledge helps curators to arrange the exhibits in a special order that makes visitors spend a shorter or longer time in the museum, and helps visitors by providing more accurate suggestions in order to have a more desirable path through the museum.

Location name	SPs $t_1 - t_2$	SPs $t_2 - t_3$
Cash desk	20	
Vitrine 1	8	8
Vitrine 2	3	12

Tab. 1: Detected number of stay points in different time windows

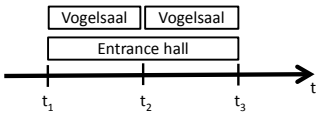


Fig. 3: Defined time windows

5 Conclusion and Outlook

We presented a platform that addresses many aspects and challenges regarding the museum environment. This platform can be deployed in any museum, regardless the focus of the exhibitions. The goal of this platform is to understand the mobility in the museums in order to support three applications that match different requirements. The platform contains a *museum graph editor* that can be used by museum staff to represent and update the graph model of the museum environment. The graph model further serves several data mining tasks which provide mobility models to a *mobile museum guide* and a *curator decision support*. Using different preprocessing and mining tasks on semantic trajectories can provide accurate mobility models that result in a deeper understanding of mobility in museums. However, for this we have to solve the challenge of a dynamic mobility model management that controls the time windows and data scopes of incremental trajectory mining algorithms so that it stays consistent with the real world, as it is represented by the

Museum graph model. Future works will cover investigations on requirements and therefore cooperation with museums. Throughout the development process we will evaluate the property graph model and decide on how it can be extended by further features to cover different application and mining use cases. The important aspect in developing a mobile museum guide application and a curator decision support application is that the applications should provide consistent information to the curators and visitors at the same time. For example, a new event like offering snack in buffet should provide new suggestion for visitors and help the curators to organize the event efficiently.

References

[BD05] Becker, Christian; Duerr, Frank: On Location Models for Ubiquitous Computing. Personal Ubiquitous Comput., 9(1):20–31, January 2005.

[BF16] Bay, Herbert; Fasel, Beat: , Interactive Museum Guide - Semantic Scholar, 2016.

[Bo12] Boden, Brigitte; Guennemann, Stephan; Hoffmann, Holger; Seidl, Thomas: Mining Coherent Subgraphs in Multi-layer Graphs with Edge Labels. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '12, ACM, New York, NY, USA, pp. 1258–1266, 2012.

[Da14] Das, Souripriya; Srinivasan, Jagannathan; Perry, Matthew; Chong, Eugene; Banerjee, Jayanta: , A Tale of Two Graphs: Property Graphs as RDF in Oracle, 2014.

[EM11] Economou, Maria; Meintani, Elpiniki: Promising Beginning? Evaluating Museum Mobile Phone Apps. Limerick, 2011.

[ERV06] Estrada, Ernesto; Rodriguez-Velazquez, Juan A.: Complex Networks as Hypergraphs. Physica A: Statistical Mechanics and its Applications, 364:581–594, May 2006. arXiv: physics/0505137.

[Kh08] Kharrat, Ahmed; Popa, Iulian Sandu; Zeitouni, Karine; Faiz, Sami: Clustering Algorithm for Network Constraint Trajectories. In (Ruas, Anne; Gold, Christopher, eds): Headway in Spatial Data Handling, Lecture Notes in Geoinformation and Cartography, pp. 631–647. Springer Berlin Heidelberg, 2008. DOI: 10.1007/978-3-540-68566-1_36.

[Le08] Lee, Jae-Gil; Han, Jiawei; Li, Xiaolei; Gonzalez, Hector: TraClass: Trajectory Classification Using Hierarchical Region-based and Trajectory-based Clustering. Proc. VLDB Endow., 1(1):1081–1094, August 2008.

[Li10] Li, Zhenhui; Lee, Jae-Gil; Li, Xiaolei; Han, Jiawei: Incremental Clustering for Trajectories. In (Kitagawa, Hiroyuki; Ishikawa, Yoshiharu; Li, Qing; Watanabe, Chiemi, eds): Database Systems for Advanced Applications, Lecture Notes in Computer Science 5982, pp. 32–46. Springer Berlin Heidelberg, April 2010. DOI: 10.1007/978-3-642-12098-5_3.

[Ma16] Martella, Claudio; Miraglia, Armando; Cattani, Marco; Steen, Maarten van: Leveraging proximity sensing to mine the behavior of museum visitors. In: 2016 IEEE International Conference on Pervasive Computing and Communications, PerCom 2016, Sydney, Australia, March 14-19, 2016. pp. 1–9, 2016.

[Ru13] Rubino, Irene; Xhembulla, Jetmir; Martina, Andrea; Bottino, Andrea; Malnati, Giovanni: MusA: Using Indoor Positioning and Navigation to Enhance Cultural Experiences in a Museum. Sensors, 13(12):17445–17471, December 2013.

[St16] Steuer, Simon; Benabbas, Aboubakr; Kasrin, Nasr; Nicklas, Daniela: Challenges and Design Goals for an Architecture of a Privacy-preserving Smart City Lab. Datenbank-Spektrum, 16(2):147–156, July 2016.

[Su15] Sun, Wen; Fokoue, Achille; Srinivas, Kavitha; Kementsietsidis, Anastasios; Hu, Gang; Xie, Guotong: SQLGraph: An Efficient Relational-Based Property Graph Store. ACM Press, pp. 1887–1901, 2015.

[SZM16] Silva, Ticiana L. Coelho da; Zeitouni, Karine; Macedo, Jose A.F. de: Online Clustering of Trajectory Data Stream. IEEE, pp. 112–121, June 2016.

[Va10] Van Hage, Willem Robert; Stash, Natalia; Wang, Yiwen; Aroyo, Lora: Finding your way through the Rijksmuseum with an adaptive mobile museum guide. In: The Semantic Web: Research and Applications, pp. 46–59. Springer, 2010.

[WE05] Wakkary, Ron; Evernden, Dale: Museum as ecology: A case study analysis of an ambient intelligent museum guide. In: Museums and the Web. volume 2005, pp. 151–164, 2005.

[Zh15] Zheng, Yu: Trajectory Data Mining: An Overview. ACM Transaction on Intelligent Systems and Technology, September 2015.

Workshop on Industrial Applications of Artificial Intelligence (IAAI'17)

Workshop on Industrial Applications of Artificial Intelligence (IAAI'17)

Alexander Paar[1] Tina Bieth[1] Stefanie Speidel[2] Cristóbal Curio[3]

Ob Softwareentwicklung, Ingenieurwesen oder Medizin – die vierte industrielle Revolution verlangt nach neuen Methoden der Assistenz in den unterschiedlichsten Branchen. Die industrielle Arbeitswelt ist dabei in vielen Bereichen wie nie zuvor von höchsten Anforderungen an Prozesseffizienz und prozessuale Sicherheit geprägt. Fundamentale Einflussfaktoren wie der demographische Wandel, Offshoring und zusätzliche Anwendungsfälle in der Industrie 4.0 oder mit Big Data verlangen nach neuartigen Formen der Unterstützung von Wissensarbeitern.

Bereits heute integrieren produktiv eingesetzte Informationssysteme Intelligenz und Lernfähigkeit auf der Basis moderner Datenbanktechnologie. Die Informationsbasis die für eine automatisierte Entscheidungsfindung zur Verfügung steht wächst dabei stetig und ermöglicht kontinuierliche Verbesserungen bei Präzision und Ausbeute der Vorhersagen.

So wurden Recommender Systeme im Bereich der Softwareentwicklung von der syntaktischen Autovervollständigung zu intelligenten Empfehlungsdiensten basierend auf natürlichsprachlichen Anforderungsspezifikationen und Software Repository Mining weiterentwickelt.

Im Ingenieursbereich ermöglichen lernfähige Klassifikationssysteme die Kapselung von langjähriger Ingenieurserfahrung und gleichen so effektiv Wissensverluste durch Mitarbeiterfluktuation aus, ermöglichen effizient die Erschließung neuer Anwenderkreise für Offshoring und erhalten unmittelbar wertschöpfende Tätigkeiten in Hochlohnländern. Die Integration automatischer Bewertungssysteme erhöht nachweislich die prozessuale Sicherheit und vermeidet kostspielige Defekte in späten Entwicklungsphasen.

Im Bereich der Medizin profitieren Ärzte und Patienten von der Auswertung umfangreicher Behandlungsdaten für algorithmisch fundierte Therapieempfehlungen weit über dem möglichen Erfahrungshorizont eines Einzelnen.

Der heutige Reifegrad von Ansätzen der künstlichen Intelligenz sowie deren unmittelbare Verfügbarkeit in Form von Softwarebibliotheken, Tools und Entwicklungsumgebungen ermöglichen neuartige und einfach anzuwendende Verfahren für automatische Entscheidungsunterstützungs- und Bewertungssysteme.

[1] TWT GmbH Science & Innovation
[2] Karlsruher Institut für Technologie
[3] Hochschule Reutlingen

Dieser Workshop bietet Wissenschaftlern und Praktikern ein moderiertes Forum für den Abgleich von Verfahren der künstlichen Intelligenz mit industriellen Anwendungsfällen für die Identifikation beiderseitiger Optimierungspotentiale sowie neuen Forschungsrichtungen.

Den Einstieg in die beiden Sessions bilden zwei eingeladene Impulsvorträge aus den Bereichen der assistenzgestützten Softwareentwicklung und der computerintegrierten Fertigung. Der Vortrag „Mining Java Packages for Developer Profiles: An Exploratory Study“ erläutert die automatische Definition von Entwicklerprofilen basierend auf einer Analyse der Mitarbeit in Softwareentwicklungsprojekten. Die Entwicklerprofile ermöglichen schließlich eine effiziente und fundierte Zuweisung von Aufgaben.

Der Vortrag „Data Mining von multidimensionalen Qualitätsdaten aus einer computerintegrierten industriellen Fertigung zur visuellen Analyse von komplexen Wirkzusammenhängen“ betrachtet die komplexen Wirkungszusammenhänge in modernen Fertigungsprozessen und skizziert, basierend auf einem Visual Analytics-Ansatz, wie Domänenexperten besonders effizient ein Zugang zu diesem Wissensschatz ermöglicht wird.

1 Workshop Organizers

Dr. Alexander Paar leitet bei der TWT GmbH Science & Innovation die Fachgebiete Data Analytics & Learning und Enterprise Data Solutions.

Dr. Tina Bieth leitet bei der TWT GmbH Science & Innovation im Bereich Consulting das Fachgebiet Communication.

Dr. Stefanie Speidel leitet am Karlsruher Institut für Technologie am Institut für Anthropomatik und Robotik die Gruppe Medizintechnik.

Dr. Cristóbal Curio ist an der Hochschule Reutlingen Professor für das Fachgebiet Cognitive Systems.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 139

Data Mining von multidimensionalen Qualitätsdaten aus einer computerintegrierten industriellen Fertigung zur visuellen Analyse von komplexen Wirkzusammenhängen

Frederick Birnbaum[1] Christian Moewes[2] Daniela Nicklas[3] Ute Schmid[4]

Keywords: Data Mining, industrielle Fertigung, visuelle Analyse

1 Motivation

Die industrielle Fertigung ist ein komplexer Prozess. Heutzutage sind viele Faktoren am Fertigungsprozess von Teilen beteiligt, darunter Maschinen, Menschen, Zulieferer sowie auch zahlreiche Umweltfaktoren. Der Produktionsprozess ist dabei durch Spezifikationen für jedes zu produzierende Teil vorgegeben und kann unter anderem mit Hilfe von Modellierungssprachen zur besseren Übersicht erfasst werden [QKZ11]. Diese Spezifikationen basieren jedoch auf Erfahrungen, Annahmen und Expertenwissen [H+06] was eine zentrale Anpassung der Maschinen deutlich erschwert. Ein komplettes Verständnis des Produktionsprozesses ist heutzutage ohne Hilfsmittel nicht mehr möglich [AK05]. Es ist nicht klar wie sich die Veränderung einer Maschineneinstellung auf andere Funktionen in der Produktionslinie auswirkt. Dazu kommt, dass bisher noch unbekannte Zusammenhänge, z.B. zwischen Mensch und Maschinen nicht richtig erfasst werden können. Im Zuge des Wandels zur Industrie 4.0, welcher eine völlig autonom gesteuerte Fabrik ermöglicht [Lu13], fallen riesige Datenmengen an. Diese können nun auch für Qualitätsverbesserungen im Produktionsprozess verwendet werden. Solche Qualitätsverbesserungen in Bezug auf Ausschussfrüherkennung sollen in dieser Arbeit besonders betrachtet werden.

2 Multidimensionale Qualitätsdaten

Für die Fertigung relevante Daten entstehen aus mehreren Informationssystemen. So erfassen an den Maschinen angebrachte Sensoren die jeweiligen Umweltfaktoren (wie Temperatur, Feuchtigkeit) in Echtzeit [PKP16]. Sogenannte Manufacturing Execution Systems (MES) bilden die digitale Fertigungsplanung auf der produktiven Ebene ab. Die dabei verwendeten Stammdaten, also nicht veränderbare Werte, sind währenddessen in Enterprise Ressource Plannung-Systemen (ERP) abgespeichert, um einen Bezug zum Prozess

[1] Otto-Friedrich-Universität Bamberg, 96045 Bamberg, frederick-leon.birnbaum@stud.uni-bamberg.de
[2] Robert Bosch GmbH, 96050 Bamberg, Christian.Moewes@de.bosch.com
[3] Otto-Friedrich-Universität Bamberg, 96045 Bamberg, daniela.nicklas@uni-bamberg.de
[4] Otto-Friedrich-Universität Bamberg, 96045 Bamberg, ute.schmid@uni-bamberg.de

herstellen zu können [WMK05]. Es besteht weiter die Möglichkeit, die Daten in einem Data Warehouse [IGG03] zu erfassen, zu integrieren und zu Analysezwecken für das Management oder Produktionsmitarbeiter aufzubereiten.

3 Geplante Data Mining-Methoden

Auf ein Data Warehouse aufbauend entstehen nun neue Möglichkeiten des Data Minings. Das Sammeln und der Einsatz der Maschinendaten mit passenden Verfahren bringt Wirkungszusammenhänge der einzelnen Produktionsfaktoren hervor. Diese wirken sich entweder direkt oder indirekt auf die Qualität des Fertigungsprozesses aus. Vorgegebene Toleranzwerte aus ERP und MES-Systemen helfen dabei, passende Qualitätsregeln aufzustellen. Darunter fällt z.B. das heuristische Einstellen der Parameter einer Maschine in einer Linienfertigung. Eine lokale Veränderung der Einstellgrößen einer Maschine kann auf nachfolgenden Maschinen zum Ausschuss führen, auch wenn diese nicht angepasst werden. Dies liegt vornehmlich daran, dass der Fertigungsprozess heutzutage so komplex ist, dass linearen Abhängigkeiten zwischen Prozessparametern auf unterschiedlichen Maschinen faktisch nicht existieren. Jedoch kann der produzierte Ausschuss als Erfahrung dienen, um gerichtete Suchverfahren für bessere Einstellungen heranzuziehen.

In der Industrie hat sich hierzu der “Cross Industry Standard Process for Data Mining” (CRISP-DM) über die Jahre als Standard durchgesetzt. Dieser gliedert den Data Mining Prozess in die Phasen Business Understanding, Data Understanding, Data Preperation, Modelling, Evaluation und Deployment [Ch00].

Basierend auf dem CRISP-DM Ansatz befasst sich diese Arbeit mit der Anwendung passender Data Mining Verfahren auf diskrete und numerische Daten, um möglichst früh Ausschuss zu erkennen. Die Ergebnisse sind Wirkungszusammenhänge, die als Assoziationsregeln aufgestellt und in geeigneter Reporting Software visualisiert werden können. Da diese Regeln lediglich auf diskrete Daten ausgelegt sind, muss hierbei auch der Vergleich zu herkömmlichen Data Mining-Verfahren gezogen werden. Dies geschieht mit Hilfe der damit verbundenen Gütemaße.

4 Vorgehen

In einem ersten Schritt befasst sich diese Arbeit mit der Analyse von Diskretisierungsverfahren, um eine passende Data Preparation nach CRISP-DM durchzuführen. Direkt im Anschluss soll die Menge und die Art der zur Verfügung stehenden Daten analysiert werden, um eine geeignete Auswahl an Data Mining Verfahren zu ermitteln.

Im zweiten Schritt werden die ausgewählten Verfahren (Assoziationsregeln, Naive Bayes-Klassifikator, ...) auf die Maschinendaten angewandt. Hierbei dienen die Toleranzwerte als Mittel, um den Ausschuss zu klassifizieren. Im Prozess sollen dabei mehrere Simulationen der Verfahren auf unterschiedlichen Datenmenge erfolgen.

Die Ergebnisse der Durchläufe werden im dritten Schritt mit der Reporting-Software Tableau[5] anschaulich visualisiert. Die Gütemaße der eingesetzten Verfahren werden ebenfalls erfasst und untereinander verglichen. Der Prozess und die Visualisierung der Assoziationsregeln erfolgt dabei basierend auf der Vorarbeit von [St13], wo bereits auf die Analyse und Visualisierung von Wirkungszusammenhängen mittels Assoziationsregeln eingegangen wird.

Im letzten Schritt soll eine Beurteilung der eingesetzten Verfahren erfolgen. Dazu spielen die bereits erwähnten Gütemaße eine Rolle. Außerdem wird auf die Performance und Analysierbarkeit der Visualisierungen in Tableau eingegangen und ein Fazit gezogen. Dabei ist zu erwarten, dass z.B. der Einsatz von Assoziationsregeln höhere Kosten verursachen wird als herkömmliche Verfahren, da der Einsatz von Diskretisierungsverfahren eine Vorbedingung ist. Auf der positiven Seite stehen dafür eine sehr deutliche Visualisierung der Wirkungszusammenhänge bei Ausschuss und ein leichtes Erkennen von Veränderungen.

5 Zusammenfassung und Ausblick

Insgesamt ergibt sich aus dem skizzierten Vorgehen ein hohes Potential, Fertigungsprozesse und deren Wirkungszusammenhänge durch Data Mining besser verstehen zu können.

Für einen produktiven Einsatz müssen jedoch noch einige zentrale Herausforderung gemeistert werden. Zum einen ist die korrekte Auswahl passender Simulationsdaten aus dem riesigen multivariaten Datenschatz des Data Warehouses zu nennen. Zum anderen stellt sich die Frage nach einem optimalen Satz an Variablen- bzw. Features [GE03].

Bei der Übertragung auf den Echtzeit-Einsatz müssen zum einen Fragen des Concept Drift und der kontinuierlichen Anpassung der Analysemethoden an die aktuelle Produktion betrachtet werden; zum anderen müssen die gelernten Methoden geeignet umgesetzt werden, um auch mit niedriger Latenz und hohem Durchsatz in Echtzeit arbeiten zu können. Hierfür müssen inkrementelle Verfahren und der Einfluss unterschiedlicher Fenstergrößen für die Analysen erforscht werden.

Schließlich muss untersucht werden, welche Visualisierungen am besten geeignet sind, um auch Domänenexperten ohne besonderes Data Science-Vorwissen einen Zugang zu dem reichen Wissensschatz zu ermöglichen, der mit Data Mining-Methoden aus multidimensionalen Qualitätsdaten in der industriellen Fertigung gewonnen werden kann.

Literatur

[AK05] Agard, B.; Kusiak, A.: Data mining for selection of manufacturing processes. In: Data Mining and Knowledge Discovery Handbook. Springer, 2005, S. 1159–1166.

[5] http://www.tableau.com

[Ch00] Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; Wirth, R.: CRISP-DM 1.0 Step-by-step data mining guide./, 2000.

[GE03] Guyon, I.; Elisseeff, A.: An introduction to variable and feature selection. Journal of machine learning research 3/Mar, S. 1157–1182, 2003.

[H+06] Harding, J.; Shahbaz, M.; Kusiak, A. et al.: Data mining in manufacturing: a review. Journal of Manufacturing Science and Engineering 128/4, S. 969–976, 2006.

[IGG03] Imhoff, C.; Galemmo, N.; Geiger, J. G.: Mastering data warehouse design: relational and dimensional techniques. John Wiley & Sons, 2003.

[Lu13] Lucke, D.: Smart Factory. In: Digitale Produktion. Springer, 2013, S. 251–269.

[PKP16] Petersen, S. A.; van der Kooij, R.; Puhar, P.: Connecting Business Processes and Sensor Data in Proactive Manufacturing Enterprises. In: Working Conference on Virtual Enterprises. Springer, S. 101–109, 2016.

[QKZ11] Qiao, L.; Kao, S.; Zhang, Y.: Manufacturing process modelling using process specification language. The International Journal of Advanced Manufacturing Technology 55/5-8, S. 549–563, 2011.

[St13] Steinbrecher, M.: Discovery and visualization of interesting patterns, Diss., Magdeburg, Universität, Diss., 2013, 2013.

[WMK05] Wiendahl, H.-H.; Mußbach-Winter, U.; Kipp, R.: MES-Lösungen: Wann braucht man sie, wozu sind sie nützlich? In: automatisierungs-ATLAS 2005/06. Technik-Dokumentations-Verlag, 2005.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 143

Mining Java Packages for Developer Profiles: An Exploratory Study

Jasmin Ramadani[1] and Stefan Wagner[2]

Abstract: Not all developers have the same degree of knowledge of all parts of a software system. For allocating new task expertise, it would be interesting to have different developer profiles explicit. The state of the practice is to find out manually who might be most experienced in a certain area. A clear understanding how to automate this analysis is missing. Our goal is to explore to what degree the analysis of couplings of packages can be useful for this automation. Our analysis approach uses the idea that packages reflect the organization of the source code into basic functionalities. We use data mining on the version history to identify the sets of the packages that were most frequently changed together in different occasions. We present a case study where we analyze three open-source software systems to define developer expertise profiles based on the aggregation of the packages. Our results identify different developer profiles. They can be especially useful in analyzing projects with a larger number of developers saving time and effort by limiting the data sets to be investigated to find relevant software changes.

Keywords: Version history;Packages;Developers expertise

1 Introduction

Software systems are developed and changed by a number of developers over time. The contribution of developers to a project consists of the combination of their actions performed during software development [GKS08]. This kind of expertise is called *implementation expertise* [SZ08]. Yet, software developers involvement in the source code becomes diverse and awkward to follow. As the development team grows, it becomes more complicated to allocate the developers according to their expertise considering their contribution to the project. One approach defines that the developers who changed a file are considered to have the expertise for that file [SZ08]. This information can be found by mining changes in the software repositories.

In Java projects, packages are important to group source code based on its functionality. By scanning the changes in the versioning system, we can extract the packages of files being changed. For the reason that the package structure is more stable and does not change often, we use the packages instead of classes or methods to define developer expertise profiles.

An approach how to find file changes that happened together frequently using data mining has been described in [KYM06; RW16b; Yi04]. The changes are grouped based on the developer that performed the commits. These changes are called *coupled file changes* and

[1] University of Stuttgart, jasmin.ramadani@informatik.uni-stuttgart.de
[2] University of Stuttgart, stefan.wagner@informatik.uni-stuttgart.de

can be offered to the developers as recommendations when solving some maintenance task. We use this technique to extract the packages which changed most frequently together to define developer profiles. The profiles can be used to identify who worked on similar issues on the project.

1.1 Problem Statement

Developers working on maintenance tasks could use the information who worked on which part of the source code to assign work or ask for help. Having many developers on a project, it is difficult to identify the experts working on topics related to their tasks.

1.2 Research Objective

The aim of our study is to identify the developers' expertise profiles based on the projects' package structures. We use the package organization of the source code because it reflects the grouping of the system functionalities and defines the fundamental features or layers of the system. Using data mining, we investigate the software repositories to extract the sets of packages that were most frequently changed together by a given developer during software development. The fact that these sets of package changes are repeating in different occasions give us better expertise description than simply listing how often the packages have been changed by the developer. Based on the functionalities behind these sets, we differentiate the developer expertise.

1.3 Contribution

We present an exploratory case study where we define developer profiles of expertise based on the aggregated information about the system packages that were most frequently changed together. By differentiating developer profiles, the effort for the analysis drops because the users involved in maintenance do not have to examine the data from all developers on the project. They can choose the data from those developers who implemented changes in the system relevant to their tasks.

2 Background

2.1 Java Packages

Packages in Java represent namespaces to organize set of classes and interfaces which prevent conflicts in the source code. They can be related according to some specific functionality. For example, the *java.io* package groups the classes dealing with input and output. There are two fundamental approaches for creating packages in Java. Package by feature reflects a set of features of the software. It organizes the classes related to a single feature. Package by layer reflects various application levels instead of features.

2.2 Data Mining

Hidden dependencies between source code files are also known as logical dependencies or couplings [GHJ98]. Coupled file changes describe a situation where someone changes a file and afterwards also changes another file. We use the same principle to investigate the version history for frequent change sets on a package instead of file level. One of the most popular data mining techniques is the discovery of frequent item sets.

We use a heuristic where the commits containing the package names are grouped based on the developers who committed the changes. The number of developers identifies the number of groups of commits for which the mining process will be performed. The user-defined support threshold identifies the frequency of the package couplings.

Tab. 1: Developer Profiles

	Package	Profile
Dev.1	astpa/src/astpa/controlstructure/controller/ astpa/src/astpa/controlstructure/create astpa/src/astpa/controlstructure/command/ astpa/src/astpa/controlstructure/controller/editParts/ astpa/src/astpa/controlstructure/policy/	control structure
Dev.2	astpa/src/astpa/ui/interfaces/ astpa/src/astpa/ui/common/grid astpa/src/astpa/ui/sds/	user interface

2.3 Developer Profiles

Developers working on a project can be differentiated based on their contribution to the functionalities of the software. There are developers that commit a significant number of commits to the system whereby others only contribute [ADG08]. Existing approaches investigate developer expertise based on the authorship of these changes [Fr10]. In our study, we define profiles of developers expertise on the level of packages. Using developers' profiles we capture the characteristics of their changes according to their contribution in the source code. Schuler et al. [SZ08] defined a process for aggregating developer expertise profiles. Similarly, we use the changed packages from the version history to aggregate the expertise profile of the developer. The profile contains all coupled packages changed by the developer. We rank the coupled packages to find the most frequent package couplings to define the profile.

Users working on some maintenance task can use the profiles to decide which developer's data given the implementation expertise matches to their work the best. For example, lets suppose that a developer has some maintenance task related to some change in the *control structure* of the application. He or she looks up at the developers profiles to find the developer whose most frequent package couplings are relevant for his feature. In Table 1 we have two developers, their most frequently changed packages and the defined profiles. We see that the most frequently coupled packages by the first developer cover features related to the *control structure*. We aggregate the features covered in these packages in a developer profile: *control structure*. The second developer profile describes package features related to the *user interface* and are not relevant for the task mentioned above.

3 Related Work

Various studies define the developer expertise based on the changes in the code. McDonald [Mc01] presents a recommendation system using the change history information based on the Line 10 rule. Here, the developer whose name shows up on line 10 of the change log is the last person who worked on the file. A developer is considered to be an expert if he or she is the last person having the file in memory [RR13]. Gitba et al. [Gi05] measure the code ownership based on who is the last committer. We consider all the developers working on a particular part of the code, not only the last one. A similar approach is defined by Mockus and Herbsleb [MH02] where the number of times the file is changed by a developer is measured for the expertise. Nakakoji et al. [Na02] define an onion like structure to describe the contributing developers to a project. Our approach differs because we do not investigate methods or files to define developer profiles but Java packages. We investigate frequently changed couplings of packages and not only the number of changes.

Manto and Murphy propose developer expertise [MM07] where the expert is defined using data from versioning system and explore how often a developer changed a particular file. The study by Schuler et al. [SZ08] recommends experts by mining of version archives based on the methods they have changed. Another approach, presented by Anvik and Murphy [AM07] defines a three-way approach using the source repository check-in logs, bug reports and the modules containing changes determined on file or package level. Alonso et al. [ADG08] uses CVS to identify and visualize developer expertise. Similarly, we use the version history of the project, in our case it is the Git repository. However, we investigate not on a method but on a package level. Joblin et al. presents an empirical study about classification of developers in to core and peripheral roles [Jo16].

There are several studies where the package information in the project has been investigated. Robles et al. [Ro06] studied the characteristics of software including the packages, lines of codes or the programming language. In our study, we do not describe the content of the classes in the packages, we identify the way the source code is divided in packages based on the features or layers. The project analysis studies performed in [Ha08; SD07] involve dependency analysis between the packages. We investigate logical couplings between the packages and not architectural dependencies.

Most of the studies dealing with identifying coupled changes from software repositories use some kind of data mining for this purpose [KYM06; RW16a; RW16b; Yi04; Zi04]. In [KYM06; Yi04] the authors investigate the couplings based on the file level. In [FGP05; Zi04] the researchers identify couplings between parts of files like classes, methods or modules. We investigate higher level couplings based on the project packages.

4 Case Study Design

4.1 Research Questions

RQ1: Which package couplings are most frequent per developer? We examine which packages are most frequently changed together by particular developers. We use this

information to investigate the functionalities the developers were mostly involved during the software development.

RQ2: What kind of developer profiles can we define based on the packages? Based on the changed packages and their functionalities, we aggregate them to define their profiles related to their expertise on the system software. Using this information, developers can explicitly identify the software changes related to their task.

4.2 Case Selection

For our study, we use three open source projects: ASTPA, an Eclipse based Java project, RIOT, Java and Android based software and VITA, Java based text analysis software. They were all developed at the University of Stuttgart and were found on the local Gitlab. The projects have been selected based on their structure having a number of packages and developers and their availability for the study.

4.3 Data Collection Procedure

We extract the Git log from the project repositories and format the output to separate the commits according to the developer who has done the changes. We enlist the commits with all changed files represented by the file names containing the relative file paths including the packages and sub-packages of the project. To prepare the data for analysis, we remove empty commits or commits having only a single entry. We group the commits based on the developer heuristics. We create data sets of commits for every developer who contributed to the repository.

4.4 Analysis Procedure

- *Extracting the packages from the commits:* We extract all names of the files changed in a commit. They include the relative file path on the system which includes the names of the package and sub-packages. We scan the file paths from the back and remove the filenames from the path. The resulting string identifies the name of the package or sub package.
- *Mining packages:* We perform a frequent item sets analysis on the packages to extract the most frequent couplings from the the repository. Due to the high number of file couplings, we use a relatively high user-defined support level for the frequent item sets analysis. This way we include only the coupled packages which happened frequently and not by chance.
- *Define developer profiles:* We rank all the coupled packages for a developer starting with the most frequently changed package to identify the most frequent ones. After the ranking of the packages, we join the group of most frequent packages to the developer. This will identify the expertise of the developer marking the functionalities

he or she was most involved into. We look up the features that are involved in the files behind this packages. We aggregate developers expertise profile based on the most frequently changed packages.

5 Results and Discussion[3]

We have extracted the most frequent package couplings for the developers in all three projects. Depending on the outcomes of the mining algorithm, we set an average support level of 80% for the first project, 60% for the second and 40% for the third project. These values define the strength of the couplings.

Our results show that the number of different coupled packages vary from developer to developer and from project to project. These values are mainly influenced by the number of functionalities the developers have been involved into.

Tab. 2: Frequent packages and expertise profiles

	ASTPA	
	packages	profiles
Dev1	astpa.src.astpa.controlstructure.controller.editParts	control structure
Dev2	astpa.src.astpa.controlstructure astpa.src.astpa.controlstructure.controller.editParts astpa.src.astpa.controlstructure.figure	control structure
Dev3	astpa.src.astpa.controlstructure.controller.editParts astpa.src.astpa.model.interfaces astpa.src.astpa.ui.common.grid	control structure+ user interface+model
Dev4	astpa.src.astpa.model.interfaces	user interface+model
Dev5	astpa.astpa.intro.graphics.icons	graphics
Dev6	astpa.src.astpa.ui.sds, astpa.src.astpa.ui.acchaz	user interface
Dev7	astpa.src.astpa.ui.common.grid	user interface
Dev8	astpa.icons.buttons.systemdescription	graphics
	RIOT	
Dev.1	android.riot.res.layout, android.riot.res.drawable-xhdpi android.riot.res.drawable-mdpi, android.riot.res.drawable-hdpi, android.riot	android layout
Dev.2	android.res.layout android.res.layout android.src.main.java.de.uni_stuttgart.riot.android.management	android layout
Dev.3	commons.src.main.java.de.uni_stuttgart.riot.server.commons.db	database
Dev.4	android.riot, android.riot.settings, android.riot.res.drawable-hdpi android.riot.res.drawable-mdpi, android.riot.res.drawable-xhdpi android.riot.res.drawable-xxhdpi, android.riot.res.layout android.riot.res.menu, android.riot.res.values-de	android layout
Dev.5	android.src.main.java.de.uni_stuttgart.riot.android.account	android account
Dev.6	android.src.main.java.de.uni_stuttgart.riot.android.idea.libraries .idea.libraries	android libraries
Dev.7	usermanagement.src.main.java.de.uni_stuttgart.riot. usermanagement.data.sqlQueryDao.impl	database
Dev.8	commons.src.main.java.de.uni_stuttgart.riot.thing	riot things
Dev.9	webapp	webapps
Dev.10	commons.src.main.resources.languages, webapp	webapps
	VITA	
Dev.1	src.main.resources.gate_home.plugins.annie.resources.gazetteer	ANN plugins
Dev.2	src.main.java.de.unistuttgart.vis.vita.services	services
Dev.3	src.main.front-end.app.partials	frontend
Dev.4	src.main.java.de.unistuttgart.vis.vita.analysis	analysis
Dev.5	src.main.java.de.unistuttgart.vis.vita.importer.epub	importer
Dev.6	src.main.java.de.unistuttgart.vis.vita.importer.epub	importer
Dev.7	src.main.front-end.app.partials	frontend

[3] The complete list of all couplings and profiles are available at: http://dx.doi.org/10.5281/zenodo.51302.

5.1 Most frequent package couplings per developer (RQ1)

For every developer, we have extracted a list of package couplings covering various software features. For example, in Table 2, we see that the first developer on the ASTPA project changed mostly files in the *controller.editParts* sub-package, whereby the sixth and the seventh developer changed mostly the *ui* related packages.

The first, the second and the fourth developer who worked on the RIOT project changed very similar packages related to the layout whereby the third and the seventh developer worked on the database packages.

The second developer who worked on the VITA project worked on the services whereby the third and the seventh developer worked on packages related to the front-end. The results show that some of them changed functionalities in files that belong to the same package, whereby others worked on different packages. Some of the developers share the packages meaning that two or more developers worked on the same packages. In other cases the developers split their work and contributed into totally different packages.

5.2 Developer profiles (RQ2)

Based on most frequent package couplings we have aggregated various developer profiles for all three projects as presented in Table 2. For the first project we identify profiles of developers related to the changes on the control structure, user interface and graphics. For the second project we define developer profiles covering expertise on the android layout, database, account, libraries, android functionalities and web apps. For the developers on the third projects we define profiles including involvement in the plugins, services, front end, analysis and import features.

The difference in the profiles is directly influenced by the areas of the system where the developers have worked on. Also the project organization and the number of packages affect the precision of the profiles.

5.3 Discussion

Using the most frequently changed packages, various functionality topics in the source code structure have appeared. For some developers we have limited set of changed system functionalities which gives clear information what they were mainly working on. This way, the profiles related to a small number of functionalities and show a more precise developer expertise. This is useful to define a precise expertise profile and clearly identifies changes related to a particular maintenance task for developers working on a concrete part of the system.

Other package changes show developers working on various topics in the system. They show more general expertise working on different areas of the system. Their profiles do not

clearly represent a set of changes related to a specific functionality. They are based on a set of changes covering broader system contribution. However, they can be still valuable for developers working on various parts of the software or on new functionalities.

6 Threats to Validity

A threat to construct validity could be that the developers by creating the packages influence the organization of the code. The names of the packages could lead to a group of classes which are not related or similar. We look up in the classes and other files in the packages to inspect manually if they are related to the package name.

As an internal threat we can mention that the relatively high support level for the data mining algorithms provides relative small number of package couplings. However, this ensures that these couplings happened frequently and not by chance.

The generalization of our approach represents an external threat because it is limited and we investigated a small number of Java projects. However, it is possible to apply our approach on any project having a clear package or namespace structure and having several developers working on it.

7 Conclusion

We conclude that we can successfully define developer profiles based on the packages that most frequently changed together. There are developers working on similar system functionalities which can lead to more precise developer profiles. This can limit the number of data sets for data mining, which decreases the effort for the analysis and can be very helpful in projects with a large number of developers. For the developers working on various or totally different parts of the code, the profiles show more general coverage of system expertise.

The resulting profiles differentiate a number of functionalities, which can help the users working on their maintenance tasks to choose the most relevant implementation. The next steps will be to formalize the automation of the expertise analysis process of the profile description according to the package structure of the system.

References

[ADG08] Alonso, O.; Devanbu, P. T.; Gertz, M.: Expertise Identification and Visualization from CVS. In: Proceedings of the 2008 International Working Conference on Mining Software Repositories. MSR '08, pp. 125–128, 2008.

[AM07] Anvik, J.; Murphy, G. C.: Determining Implementation Expertise from Bug Reports. In: Proceedings of the Fourth International Workshop on Mining Software Repositories. MSR '07, pp. 2–, 2007.

[FGP05] Fluri, B.; Gall, H.; Pinzger, M.: Fine-Grained Analysis of Change Couplings. In: SCAM. Pp. 66–74, 2005.

[Fr10] Fritz, T.; Ou, J.; Murphy, G. C.; Murphy-Hill, E.: A Degree-of-knowledge Model to Capture Source Code Familiarity. In: Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1. ICSE '10, pp. 385–394, 2010.

[GHJ98] Gall, H.; Hajek, K.; Jazayeri, M.: Detection of Logical Coupling Based on Product Release History. In: Proceedings of the International Conference on Software Maintenance. ICSM '98, pp. 190–, 1998.

[Gi05] Girba, T.; Kuhn, A.; Seeberger, M.; Ducasse, S.: How Developers Drive Software Evolution. In: Proceedings of the Eighth International Workshop on Principles of Software Evolution. IWPSE '05, pp. 113–122, 2005.

[GKS08] Gousios, G.; Kalliamvakou, E.; Spinellis, D.: Measuring Developer Contribution from Software Repository Data. In: Proceedings of the 2008 International Working Conference on Mining Software Repositories. MSR '08, 2008.

[Ha08] Hautus, E.: Improving Java Software Through Package Structure Analysis, 2008, URL: http://ehautus.home.xs4all.nl/papers/PASTA.pdf/.

[Jo16] Joblin, M.: Classifying Developers into Core and Peripheral: An Empirical Study on Count and Network Metrics, 2016, URL: http://arxiv.org/abs/1604.00830.

[KYM06] Kagdi, H.; Yusuf, S.; Maletic, J. I.: Mining Sequences of Changed-files from Version Histories. In: Proceedings of the 2006 International Workshop on Mining Software Repositories. MSR '06, pp. 47–53, 2006.

[Mc01] McDonald, D. W.: Evaluating Expertise Recommendations. In: Proceedings of the 2001 International ACM SIGGROUP Conference on Supporting Group Work. GROUP '01, pp. 214–223, 2001.

[MH02] Mockus, A.; Herbsleb, J. D.: Expertise browser: a quantitative approach to identifying expertise. In: ICSE '02: Proceedings of the 24th International Conference on Software Engineering. Pp. 503–512, 2002.

[MM07] Minto, S.; Murphy, G. C.: Recommending Emergent Teams. In: Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007). Pp. 5–5, 2007.

[Na02] Nakakoji, K.; Yamamoto, Y.; Nishinaka, Y.; Kishida, K.; Ye, Y.: Evolution Patterns of Open-source Software Systems and Communities. In: Proceedings of the International Workshop on Principles of Software Evolution. IWPSE '02, pp. 76–85, 2002.

[Ro06] Robles, G.; Gonzalez-Barahona, J. M.; Michlmayr, M.; Amor, J. J.: Mining Large Software Compilations over Time: Another Perspective of Software Evolution. In: Proceedings of the International Workshop on Mining Software Repositories (MSR 2006). Pp. 3–9, 2006.

[RR13] Robbes, R.; Rothlisberger, D.: Using developer interaction data to compare expertise metrics. In. Pp. 297–300, 2013.

[RW16a] Ramadani, J.; Wagner, S.: Are Suggestions of Coupled File Changes Interesting? In: Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering. Pp. 15–26, 2016.

[RW16b] Ramadani, J.; Wagner, S.: Which Change Sets in Git Repositories Are Related? In: International Conference on Software Quality, Reliability and Security (QRS). 2016.

[SD07] S.Ducasse; D.Pollet; M.Suen; andI. Alloui, H.: Package Surface Blueprints: Visually Supporting the Understanding of Package Relationships. In: 2007 IEEE International Conference on Software Maintenance. Pp. 94–103, 2007.

[SZ08] Schuler, D.; Zimmermann, T.: Mining Usage Expertise from Version Archives. In: Proceedings of the 2008 International Working Conference on Mining Software Repositories. MSR '08, pp. 121–124, 2008.

[Yi04] Ying, A. T. T.; Murphy, G. C.; Ng, R. T.; Chu-Carroll, M.: Predicting Source Code Changes by Mining Change History. IEEE Transactions on Software Engineering 30/9, pp. 574–586, 2004.

[Zi04] Zimmermann, T.; Weisgerber, P.; Diehl, S.; Zeller, A.: Mining Version Histories to Guide Software Changes. In: Proceedings of the 26th International Conference on Software Engineering. ICSE '04, pp. 563–572, 2004.

Präferenzen und Personalisierung in der Informatik

(PPI17)

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 155

Workshop Präferenzen und Personalisierung in der Informatik (PPI)

Markus Endres[1] Andreas Pfandler[2]

Die erste Edition des Workshops *„Präferenzen und Personalisierung in der Informatik“* (PPI17) wurde im Rahmen der BTW 2017[3], der 17. Fachtagung „Datenbanksysteme für Business, Technologie und Web“, ins Leben gerufen. Die Motivation des Workshops ergibt sich aus folgenden Beobachtungen:

Präferenzen spielen im Alltag jedes Menschen eine große Rolle. Einfache Präferenzen wie z.B. „Zum Frühstück mag ich lieber Kaffee als Tee, Schokocroissants sind besser als ein Käsebrot“ oder „Eine Unterkunft in einem 4-Sterne Hotel in der Nähe der Konferenz ist mir wichtiger als eine möglichst günstige Übernachtung“ sind uns allen bekannt. Besonders im Bereich der Informatik gewinnen Präferenzen immer mehr an Bedeutung. So haben Präferenzen seit langem Einzug bei Touristikportalen oder Firmen wie Amazon, Google und Facebook gehalten, um personalisierte Produktempfehlungen vorzunehmen.

Präferenzen und Personalisierung sind dabei nicht mehr nur von akademischem Interesse, sondern haben weitgehende Auswirkungen auf Industrie und Wirtschaft. Angesichts der immer zunehmenden Fülle an Informationen genügt es heute nicht mehr, eine Menge gespeicherter Daten möglichst schnell bei den Nutzern „abzuliefern“. Vielmehr müssen die richtigen Daten in der gegebenen Situation zur rechten Zeit im adäquaten Zusammenhang unter dem relevanten Blickwinkel und in der gewünschten Form zur Verfügung stehen. Die Notwendigkeit und das Potential des präferenzbasierten und personalisierten Datenmanagements wird dadurch deutlich; nicht zuletzt, da dessen Ergebnisse auch unmittelbare Anwendungen in der Industrie finden (werden). Die Entwicklung passender Methoden zum Erlernen und Verarbeiten von Präferenzinformationen bleibt – insbesondere aufgrund der riesigen Datenmengen – eine Herausforderung für Wissenschaft, Wirtschaft und Gesellschaft.

Der Workshop *„Präferenzen und Personalisierung in der Informatik“* (PPI17) widmet sich den oben angesprochenen Themen, um Informationssysteme der Zukunft zu schaffen, die Präferenzverarbeitung als ein zentrales Konzept der Personalisierung ansehen. Theoretische und praktisch relevante Beiträge, Software-Demos sowie sich noch in der Entwicklung befindende Arbeiten zu Präferenzen und Personalisierung waren herzlich willkommen.

[1] Universität Augsburg, Institut für Informatik, Universitätsstr. 6a, D 86159 Augsburg, endres@informatik.uni-augsburg.de
[2] Technische Universität Wien, Favoritenstr. 9, A 1040 Wien und Universität Siegen, Unteres Schloß 3, D 57072 Siegen, pfandler@dbai.tuwien.ac.at
[3] http://www.btw2017.informatik.uni-stuttgart.de

Hier eine Aufzählung einiger relevanter Themen:

- Präferenzen in der Künstlichen Intelligenz und in Datenbanken
- Präferenzen in Multiagenten-Systemen
- Anwendung von Präferenzen, wie z.B. personalisierte Recommender
- Modellierung und Repräsentation von Präferenzen
- Preference Elicitation und Preference Learning
- (Computational) Social Choice im Allgemeinen
- Spieltheoretische Aspekte von Präferenzen
- Software-Demos zu Präferenzen und Personalisierung

Der Workshop dient auch dem gemeinsamen Austausch zu aktuellen Forschungsfragen in diesem Bereich, der Kommunikation zwischen den Wissenschaftlern im deutschsprachigen Raum sowie der Möglichkeit neue Kontakte zu knüpfen. Der Workshop ist nicht nur für renommierte Wissenschaftler/-innen, sondern auch für Jung-Wissenschaftler/-innen gedacht, die ihre aktuellen Arbeiten in einem kleinen Rahmen vorstellen und diskutieren wollen. Willkommen sind auch Beiträge aus der Wirtschaft sowie von Doktoranden oder Studierenden, die ihre Abschlussarbeiten mit Bezug zu Präferenzen und/oder Personalisierung präsentieren möchten. Der Workshop bietet so die Gelegenheit zu einem intensiven Kontakt mit den anderen Teilnehmern und wird durch einen Invited Talk abgerundet.

Javier Romero von der Universtiät Potsdam hat sich in seinem Invited Talk bereit erklärt das „asprin“ Framework zur Formulierung und Auswertung von Präferenzen mittels Answer Set Programming vorzustellen. Für das wissenschaftliche Programm wurden drei Langbeiträge sowie drei Kurzbeiträge angenommen, die sich sowohl mit theoretischen als auch anwendungsorientierten Aspekten zur Weiterentwicklung von präferenzbasierten Methoden befassen. Besonders positiv hervorzuheben ist, dass diese Beiträge ein breites Spektrum an relevanten Themen abdecken.

Wir bedanken uns bei allen Mitgliedern des Programmkomitees und den Reviewern für ihren Einsatz und freuen uns nunmehr gemeinsam mit allen Teilnehmern auf spannende und vielfältige Vorträge, interessante Diskussionen und zahlreiche neue Ideen und Erkenntnisse.

1 Organisatoren

Markus Endres, Universität Augsburg
Andreas Pfandler, TU Wien und Universität Siegen

2 Programmkomitee

Gábor Erdélyi, Universität Siegen
Jan-Christoph Kalo, TU Braunschweig
Ekaterina Lebedeva, Australian National University
Stefan Mandl, EXASOL AG

Timotheus Preisinger, DEVnet GmbH
Antonius Weinzierl, TU Wien
Florian Wenzel, Universität Augsburg

Zusätzliche Gutachter: Christian Reger und Yongjie Yang von der Universität Siegen

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 159

asprin: Answer Set Programming with Preferences

Javier Romero[1]

Abstract: Answer Set Programming (ASP) is a well established approach to *declarative problem solving*, combining a rich yet simple modeling language with high-performance solving capacities. In this talk we present *asprin*, a general, flexible and extensible framework for preferences in ASP. *asprin* is general and captures many of the existing approaches to preferences. It is flexible, because it allows for the combination of different types of preferences. It is also extensible, allowing for an easy implementation of new approaches to preferences. Since it is straightforward to capture propositional theories and constraint satisfaction problems in ASP, the framework is also relevant to optimization in Satisfiability Testing and Constraint Processing.

Keywords: Preference, Optimization, Answer Set Programming

1 Introduction

Answer Set Programming (ASP; [Ba03]) is a well established approach to *declarative problem solving*. Rather than solving a problem by telling a computer *how to solve the problem*, the idea is to simply describe *what the problem is* and leave its solution to the computer. The success of ASP is due to the combination of a rich yet simple modeling language with high-performance solving capacities. Modeling has its roots in the fields of Knowledge Representation and Logic Programming, while solving is based in methods from Deductive Databases and Satisfiability Testing (SAT; [Bi09]). ASP programs resemble Prolog programs, but they are interpreted according to the stable models semantics [GL88], and the underlying solving techniques are closely related to those of modern SAT solvers.

For solving real-world applications it is often necessary to represent and reason about preferences. This was realized quite early in ASP, leading to many approaches to preferences [BNT03; Br04; SP06]. Departing from there, we have developed our approach [Br15a; Br15b] and the resulting *asprin*[2] system providing a general and flexible framework for quantitative and qualitative preferences in ASP. Our framework is *general* and captures many of the existing approaches to preferences. It is *flexible*, providing means for the combination of different types of preferences. And it is also *extensible*, allowing for an easy implementation of new approaches to preferences. Since it is straightforward to capture propositional theories and constraint satisfaction problems in ASP, the approach is also relevant to optimization in Satisfiability Testing and Constraint Processing.

[1] University of Potsdam, javier@cs.uni-ptosdam.de
[2] *asprin* stands for "*AS*P for *pr*eference handl*in*g".

2 The Travelling Salesperson Problem in Answer Set Programming

The basic idea of ASP is to represent a given problem by a logic program whose stable models correspond to solutions, and then to use an ASP solver for finding stable models of the program. As an illustration, let us consider the Travelling Salesperson Problem (TSP). In our example, the salesperson starts in the city `a`, and should visit the cities `b`, `c`, and `d`. We are given the information about the existing roads between the cities, and their corresponding distances. In ASP, this could be represented by the following set of facts:

```
start(a). city(a). city(b). city(c). city(d).
road(a,b,10). road(b,c,20). road(c,d,25). road(d,a,40).
road(b,d,30). road(d,c,25). road(c,a,35).
```

where, for instance, `road(a,b,10)` means that there is a road from `a` to `b` of `10` kilometres. In the most basic formulation of the problem, a solution is a route visiting once every city and returning to the starting city. The following rules capture the problem:

```
{ travel(X,Y) : road(X,Y,D) }.

visited(Y) :- travel(X,Y), start(X).
visited(Y) :- travel(X,Y), visited(X).

:- city(X), not visited(X).
:- city(X), 2 { travel(X,Y) }.
:- city(X), 2 { travel(Y,X) }.
```

Line 1 chooses a set of `travel(X,Y)` atoms, where `X` and `Y` are two cities connected through a `road`. The next rules describe when a city has been `visited`: if we `travel` to it from the `start` city (Line 3), or if we `travel` from an already `visited` city (Line 4). The last lines represent constraints on the solutions. Line 6 says that it cannot be the case that a `city` is not `visited`, line 7 forbids travelling from one `city` to two or more cities and, similarly, line 8 forbids reaching a city from two or more cities. Putting together the previous facts and rules in a logic program, it turns out that the solutions of the original problem correspond to the stable models of the program, which can be computed by an ASP solver. For example, stable model M_1 contains $\{$`travel(a,b)`, `travel(b,c)`, `travel(c,d)`, `travel(d,a)`$\}$ and corresponds to the solution where first `a` is visited, then `b`, `c`, `d` and `a` again. Another stable model M_2 contains $\{$`travel(a,b)`, `travel(b,d)`, `travel(d,c)`, `travel(c,a)`$\}$.

3 *asprin*: Answer Set Programming with Preferences

asprin extends ASP with preference relations among the stable models of a logic program. Formally, a *preference relation* is a strict partial order $\succ$ over the stable models of a logic program P. Given two stable models X and Y of P, $X \succ Y$ means that X is preferred to Y with respect to $\succ$. Then, a stable model X of P is *optimal* with respect to $\succ$ if there is no other stable model Y such that $Y \succ X$.

In *asprin*, preference relations are declared by *preference statements*, composed of an identifier, a type, and a set of preference elements. The identifier names the preference relation, whereas its type and elements define the relation. Coming back to the classical formulation of the TSP, the problem is to find a route of minimum distance. The preference for shorter routes can be represented as follows:

```
#preference(distance,less(weight)){
  D :: travel(X,Y), road(X,Y,D)
}.
```

This statement[3] declares a preference relation named `distance` of type `less(weight)` with a single preference element `D :: travel(X,Y), road(X,Y,D)`. The preference type aims at reducing the sum of the weights, and the preference element says that if we travel from `X` to `Y` and the distance is `D`, we obtain a weight of `D`. Hence, the resulting relation `distance` prefers stable models that induce the minimum sum of distances. In our example, M_1, whose sum of distances is `95`, is optimal with respect to `distance`, and is preferred to M_2, whose sum of distances is `100`. Indeed, M_1 is a solution to the classical TSP.

Extending the problem definition, we could consider a conditional preference of type `aso` (standing for Answer Set Optimization, [BNT03]):

```
#preference(balance,aso){
  travel(Y,Z), road(Y,Z,D'), D'<25 || travel(X,Y),
                                      road(X,Y,D), D>25
}.
```

expressing that, whenever we travel through a long road (`D>25`), we prefer afterwards to travel through a short one (`D'<25`). And going further, both preferences could be combined:

```
#preference(all,pareto){
  **distance; **balance
}.
```

defining a new preference relation `all` which is the Pareto ordering of the two preferences `distance` and `balance`.

One important feature of *asprin* is that the semantics of preference types are not predefined. The idea is that we may define the semantics of a new preference type, and implement those semantics in *asprin* simply by writing a logic program, called a *preference program*. This program takes two (reified) stable models and decides whether one is preferred to the other following the semantics of the preference type. Then we may write a preference statement of the new type, and use it importing the corresponding preference program.

For computing optimal models, *asprin* does repeated calls to an underlying ASP solver: first, an arbitrary stable model is generated; then this stable model is "fed" to the preference

[3] In ASP, meta statements are preceded by '#'.

program to produce a preferred one, etc. Once the program becomes unsatisfiable, the last stable model obtained is an optimal one.

asprin is implemented in Python, and available for download at `https://github.com/potassco/asprin`. *asprin* provides a *library* containing a number of common, preference types along with the necessary preference programs. Users happy with what is available in the library can thus use the available types without having to bother with preference programs at all. However, if the predefined preference types are insufficient, users may define their own ones (and so become preference engineers). In this case, they also have to provide the preference programs *asprin* needs to cope with the new preference relations.

References

[Ba03] Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, 2003.

[Bi09] Biere, A.; Heule, M.; van Maaren, H.; Walsh, T., eds.: Handbook of Satisfiability. IOS Press, 2009.

[BNT03] Brewka, G.; Niemelä, I.; Truszczyński, M.: Answer Set Optimization. In (Gottlob, G.; Walsh, T., eds.): Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03). Morgan Kaufmann Publishers, pp. 867–872, 2003.

[Br04] Brewka, G.: Complex Preferences for Answer Set Optimization. In (Dubois, D.; Welty, C.; Williams, M., eds.): Proceedings of the Ninth International Conference on Principles of Knowledge Representation and Reasoning (KR'04). AAAI Press, pp. 213–223, 2004.

[Br15a] Brewka, G.; Delgrande, J.; Romero, J.; Schaub, T.: asprin: Customizing Answer Set Preferences without a Headache. In (Bonet, B.; Koenig, S., eds.): Proceedings of the Twenty-Ninth National Conference on Artificial Intelligence (AAAI'15). AAAI Press, pp. 1467–1474, 2015.

[Br15b] Brewka, G.; Delgrande, J.; Romero, J.; Schaub, T.: Implementing Preferences with asprin. In (Calimeri, F.; Ianni, G.; Truszczyński, M., eds.): Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'15). Vol. 9345. Lecture Notes in Artificial Intelligence, Springer-Verlag, pp. 158–172, 2015.

[GL88] Gelfond, M.; Lifschitz, V.: The Stable Model Semantics for Logic Programming. In (Kowalski, R.; Bowen, K., eds.): Proceedings of the Fifth International Conference and Symposium of Logic Programming (ICLP'88). MIT Press, pp. 1070–1080, 1988.

[SP06] Son, T.; Pontelli, E.: Planning with Preferences using Logic Programming. Theory and Practice of Logic Programming 6/5, pp. 559–608, 2006.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 163

Computational Social Choice in the Clouds

Theresa Csar[1] Martin Lackner[2] Reinhard Pichler[3] and Emanuel Sallinger[4]

Abstract: In the era of big data we are concerned with solving computational problems on huge datasets. To handle huge datasets in cloud systems dedicated programming frameworks are used, among which MapReduce is the most widely employed. It is an important issue in many application areas to design parallel algorithms which can be executed efficiently on cloud systems and can cope with big data. In computational social choice we are concerned with computational questions of joint decision making based on preference data. The question of how to handle huge preference datasets has not yet received much attention. In this report we summarize our recent work on designing and evaluating algorithms for winner determination in huge elections using the MapReduce framework.

Keywords: Computational Social Choice, Cloud Computing, MapReduce, Parallel Computing, Distributed Computing

1 Introduction

Especially with the rise of e-business platforms, huge preference datasets are becoming more and more common. Some interesting examples are the comparison of results of search engines, preferences in online movie or book stores, surveys among a large number of people, etc. Computing winner sets over such preferences is an important topic in computational social choice. Yet the handling of huge sets of preference data has not received much attention. Often these large datasets are deployed and analysed in cloud systems. In such cloud systems MapReduce [DG08] is the most widely used framework for data intensive parallel computation.

In [Cs17] algorithms for winner determination in huge elections are proposed, analysed and experimentally evaluated. In this report we summarize our recent work in [Cs17] and explain one MapReduce algorithm for winner determination in more detail. We provide a short introduction to MapReduce, methods of social choice and winner determination and at the end give possible directions for future work.

[1] TU Wien, Abteilung für Datenbanken und Artificial Intelligence, 1040 Wien, csar@dbai.tuwien.ac.at
[2] University of Oxford, Department of Computer Science, Oxford, martin.lackner@cs.ox.ac.uk
[3] TU Wien, Abteilung für Datenbanken und Artificial Intelligence, 1040 Wien, pichler@dbai.tuwien.ac.at
[4] University of Oxford, Department of Computer Science, Oxford, emanuel.sallinger@cs.ox.ac.uk

2 Preliminaries

2.1 MapReduce

The challenges of handling huge data sets in the cloud is an issue in many research areas, ranging from core database tasks such as computing joins [AU10, BKS13] to data intensive computations in general [Af13]. MapReduce, and its open source implementation Hadoop, are widely used frameworks in cloud computing. MapReduce was first introduced by Google [DG08]. The MapReduce model is divided into three phases: map, shuffle and reduce. In the map phase the input data is read and key-value pairs are produced. In the shuffle phase all key-value pairs are assigned to a corresponding reduce task with the same key. The reduce tasks then perform simple calculations on the received values. The reduce tasks cannot communicate among each other and write their results back to the distributed file system after the computation is finished. If a computation has to be done in several MapReduce rounds, the data is read again in each map phase and is written to the distributed file system after the reduce phase.

In the theoretical analysis of MapReduce algorithms the following performance characteristics are of interest: total communication cost, wall clock time, replication rate [AU10] and the number of MapReduce rounds. These performance characteristics describe the communication cost and the computation time in more detail:

Communication Cost. We analyse two measures of communication cost: the total communication cost and the replication rate. The total communication cost refers to the total number of values transmitted by the map and reduce tasks, whereas the replication rate is a relative measure of communication cost. It is defined as the ratio of the input size to the reduce phase and the input size to the map phase.

Computation Time. There are two important measures to describe computation time. First one can consider the number of MapReduce rounds needed to complete the total computation. Since MapReduce is typically not optimized for iterative computations, each additional MapReduce round takes time for writing and reading the data from/to disks. The second measure is wall clock time, which refers to the maximum time needed for the parallel execution.

2.2 Computational Social Choice: Winner Determination

A central problem in computational social choice is winner determination. Given a set of candidates C and a list of votes, the goal is to compute the set of all winners according to a given voting rule. The input is a set of votes, and the votes (preferences) can be given as partial or total orders (e.g., $\{a > c > d > b, a > e\}$).

Preferences can be aggregated to a weak and/or strict dominance matrix. The (weak/strict) dominance matrix D is a $(0,1)$-matrix, where $D[a,b] = 1$ means that a (weakly/strictly) dominates b. A candidate a dominates b, if there are more votes preferring a over b.

In Winner Determination we are concerned with selecting a subset of all candidates as winners. There are many different approaches for selecting such a winner set. A seemingly straight forward method is selecting the *Condorcet winner*, which is a single candidate that dominates all other candidates in a pairwise comparison. Unfortunately such a winner does not necessarily exist. Many other methods for selecting subsets of candidates as winners have been developed. The computational complexity of some of them is studied in [BFH09], one of them is the Smith set. Due to space reasons we choose the Smith set as the only method outlined in this report, since the algorithm illustrates the overall idea of MapReduce very well. The Smith set is defined as the smallest possible set of candidates that dominate all outside candidates. If there exists a Condorcet Winner, the Condorcet Winner is the only candidate in the Smith set. The proposed algorithm for computing the Smith set works very well with MapReduce and is explained in the next section.

3 Computational Social Choice in the Clouds

In [Cs17] we develop parallel algorithms for winner determination in huge elections using MapReduce. In particular, we present parallel algorithms for computing the *Schwartz Set*, *Smith Set*, *Copeland Set* and *dominance graphs*. Furthermore, we show that some methods for winner determination are unlikely to be parallelizable, e.g. we show that the single-transferable vote (STV) rule is P-complete and thus a highly parallelizable algorithm may not be hoped for. In this report the computation of the Smith Set is explained in more detail. The algorithm is based on non-trivial observations of [BFH09] and fits very well into the MapReduce framework. It is noteworthy that preference data sets can be large in two dimensions (many votes and/or many candidates) and both scenarios can be found in real world data. A closer inspection of algorithms from [Cs17] show that a large number of voters is far easier to handle than a large number of candidates; hence the focus of the paper is on the more challenging case of data sets with many candidates.

All algorithms presented in [Cs17] use the MapReduce framework for parallelization. Some performance characteristics of the proposed algorithms have been studied theoretically and experiments have been performed. In the theoretical analysis we focused on the following parameters: total communication cost, wall clock time, replication rate and the number of MapReduce rounds (see Section 2.1).

3.1 MapReduce Algorithm for computing the Smith Set

This is a brief sketch of the MapReduce algorithm for computing the Smith set proposed in [Cs17]. A candidate a is in the Smith set if for every candidate b there is a path from a to b in the weak dominance graph [BFH09]. A naive algorithm for computing the Smith set would thus first compute the transitive closure of the weak dominance graph. This would take logarithmically many MapReduce rounds. We can do much better by applying the following property: Let $D^k_{\succeq}(v)$ denote the set of vertices reachable from vertex v by a path of length $\leq k$ in the weak dominance graph. [BFH09] show that in the weak dominance

graph a vertex t is not reachable from a vertex s if and only if there exists a vertex v such that $D^2_{\succeq}(v) = D^3_{\succeq}(v)$, $s \in D^2_{\succeq}(v)$, and $t \notin D^2_{\succeq}(v)$. This characteristic is put to value in our MapReduce algorithm for computing the Smith set.

Each vertex is saved and processed as a list of three sets: **old** contains the set of vertices that are already known to be reached from a, **new** is the set with newly found vertices that can be reached from a and **reachedBy** contains all vertices known to reach a.

Map Phase. The vertices are read from the input file and for each vertex the following key-value pairs are produced: For every vertex r in *reachedBy* as key, the whole set *new* is emitted with mode 'new' as value. The resulting key-value pair is: (key=r,value= (set *new*, mode='new')). Likewise, for every vertex n in *new* the pair (key=n,value= (set *reachedBy*, mode='reachedBy')) is emitted.

Reduce Phase. Each reduce task is responsible for one candidate/vertex and combines the received values. The outputs are the corresponding vertices with the sets *new*, *old* and *reachedBy*. The set *reachedBy* is the union of all input values with mode 'reachedBy'; the same applies to the set *old*. The set *new* must not contain any values that are already listed in *old*, therefore we first create the set *new* as the union of all received values with mode 'new', but then we set $new = new - old$.

The map and reduce phase are done twice and then an additional round for post-processing is needed to identify the candidates contained in the Smith set. The algorithm takes three MapReduce rounds in total.

The proposed MapReduce algorithm for computing the Smith set of a data set with m candidates has a replication rate of $rr \leq 2m+1$ and the number of keys is m, where m is the number of candidates. The wall clock time is $\leq 6m^2 + 8m$ and the total communication cost is $\leq 6m^3 + 8m^2$ [Cs17].

4 Future Work

Real World Data: Identify Social Choice Problems in the Clouds. An important next step will be to evaluate the algorithms on real world data, since the experiments in [Cs17] where performed solely on synthetic datasets. Other problems in computational social choice that can make use of such parallel algorithms have to be identified. It will be important to identify what preference data sets are already available in the cloud.

Alternatives to MapReduce/Hadoop. MapReduce and its Open Source implementation Apache Hadoop are widely used for parallel computation. However MapReduce is typically not ideal for iterative computations. There are already many new adaptations and extensions available. In [Le16] an experimental comparison of iterative MapReduce frameworks has been performed. As expected Hadoop performs worse for iterative computations, since Hadoop/MapReduce is optimized for batch processing. Spark outperforms all other tested frameworks. In comparison to Hadoop, Spark [Za10] loads all data into memory and organizes it as Resilient Distributed Datasets (RDDs), while Hadoop reads the data

in junks from the distributed file system. Spark also allows for more generic data flows and is not limited to the strict MapReduce API [Le16]. An interesting topic is to analyze for which problem settings Spark outperforms Hadoop and the other way around.

5 Conclusion

We have made a first step to take computational social choice to the cloud. Yet, more work has yet to be done. This report provides an overview of recent work and possible future research directions. The challenges of cloud computing start from dataset management in a distributed file system, choosing a fitting parallel framework and finally lead to designing algorithms to efficiently solve computation problems. All these challenges have to be overcome to take computational social choice to the cloud.

References

[Af13] Afrati, Foto N.; Sarma, Anish Das; Salihoglu, Semih; Ullman, Jeffrey D.: Upper and Lower Bounds on the Cost of a Map-Reduce Computation. Proceedings of the VLDB Endowment, 6(4):277–288, 2013.

[AU10] Afrati, Foto N.; Ullman, Jeffrey D.: Optimizing joins in a map-reduce environment. In: EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings. volume 426 of ACM International Conference Proceeding Series. ACM, pp. 99–110, 2010.

[BFH09] Brandt, Felix; Fischer, Felix; Harrenstein, Paul: The computational complexity of choice sets. Mathematical Logic Quarterly, 55(4):444–459, 2009.

[BKS13] Beame, Paul; Koutris, Paraschos; Suciu, Dan: Communication steps for parallel query processing. In: Proc. PODS 2013. ACM, pp. 273–284, 2013.

[Cs17] Csar, Theresa; Lackner, Martin; Pichler, Reinhard; Sallinger, Emanuel: Winner Determination in Huge Elections with MapReduce. In: Proceedings of AAAI-17. AAAI Press, 2017. To appear.

[DG08] Dean, Jeffrey; Ghemawat, Sanjay: MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1):107–113, 2008.

[Le16] Lee, Haejoon; Kang, Minseo; Youn, Sun-Bum; Lee, Jae-Gil; Kwon, YongChul: An Experimental Comparison of Iterative MapReduce Frameworks. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. ACM, pp. 2089–2094, 2016.

[Za10] Zaharia, Matei; Chowdhury, Mosharaf; Franklin, Michael J; Shenker, Scott; Stoica, Ion: Spark: cluster computing with working sets. HotCloud, 10:10–10, 2010.

Acknowledgments: This work was supported by the Austrian Science Fund projects (FWF):P25207-N23, (FWF):P25518-N23 and (FWF):Y698, by the Vienna Science and Technology Fund (WWTF) through project ICT12-015, by the European Research Council (ERC) under grant number 639945 (ACCORD) and by the EPSRC programme grant EP/M025268/1.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 169

Encoding monotonic multiset preferences using CI-nets[1]

Martin Diller[2], Anthony Hunter[3]

Abstract: *CP*-nets and their variants constitute one of the main AI approaches for specifying and reasoning about preferences. *CI*-nets, in particular, are a *CP*-inspired formalism for representing ordinal preferences over sets of goods, which are typically monotonic. Considering also that goods often come in multisets rather than sets, a natural question is whether *CI*-nets can be used more or less directly to encode preferences over multisets. We here provide some initial ideas about this by first presenting a straight-forward generalisation of *CI*-nets to multisets with bounded multiplicities, which we show can be efficiently reduced to *CI*-nets. Second, we sketch a proposal for a further generalisation which allows for encoding preferences over multisets with unbounded multiplicities, yet characterise reasoning in this framework in terms of the first. We finally show a potential use of our generalisation of *CI*-nets for personalization in a recent system for evidence aggregation.

Keywords: Multiset preferences, *CI*-nets, evidence aggregation

1 Introduction

CI-nets [BEL09] are part of several languages for specifying and reasoning about preferences that are inspired by CP-nets [Bo04]. These languages have in common that assertions regarding preferences are interpreted via the "ceteris-paribus" ("all remaining things being equal") semantics. I.e. "A is preferred to B" is interpreted as shorthand for "A is preferred to B, ceteris paribus". This allows the formulation of an "operational semantics" in terms of "worsening flips" for verifying statements regarding preferences computationally. *CI*-nets distinguishing feature is that they are tailored to ordinal preferences over sets of goods. These are also typically monotonic, i.e. more goods are usually preferred to less goods.

Also taking in account the fact that more often than not goods come in multisets rather than sets, a natural question is whether *CI*-nets can be easily generalised to specify and reason about preferences over multisets as well as sets of goods. We here present ideas on how to generalise *CI*-nets to deal with what we identify as the two main differences of preferences over multisets and preferences over sets of goods. The first of the differences is obviously that, while preferences over sets involve comparing different combinations of a fixed number of elements (namely one of each item), when considering multiset preferences also the multiplicity of the items needs to be taken in account. So, for example, while in the set scenario preferring apples over oranges always is interpreted as "irrespective of the number of apples and oranges", in the multiset scenario it is possible to say, for instance, that one prefers having an apple over an orange if one doesn't already have any apples, but

[1] Supported by the Austrian Science Fund FWF project W1255-N23.
[2] TU Wien, Institute of Information Systems , Favoritenstraße 9-11, A-1040 Wien, mdiller@kr.tuwien.ac.at
[3] University College London, Dept. of Computer Science, 66-72 Gower Street, London WC1E 6EA, anthony.hunter@ucl.ac.uk

one prefers having an orange over some number (say, up up to three) apples if one already has some (e.g. two or more) apples.

A slightly more subtle issue is that, while when talking about preferences over sets there is a natural limit to the number of items one is considering (namely, one of each), in the case of preferences over multisets it is often the case that it is artificial to impose any a-priori upper bound on the multiplicity of the items. For example, when one says that one prefers having an apple and an orange over say even up to three pears, this also means that one prefers having two apples and two oranges over three pears, three apples and one orange over three pears, etc. If one is using the preferences as a guide as to what choice to take regarding some outcome, e.g. choosing between different baskets of fruits, then the upper bound of apples, oranges, and pears is given by the "evaluation context" (in this case, the upper bound of the fruits in the baskets that are available), but is not part of the preference relation per se. I.e., the same preference relation should be of use when considering a different "evaluation context", e.g. a different set of fruit baskets.

Concretely, in this work we first present a simple generalisation of CI-nets to multisets with fixed multiplicities (Section 3.1). We call the resulting framework $C^m I$-nets ("m" stands for "multiset"). We show that reasoning on $C^m I$-nets can be efficiently reduced to reasoning on CI-nets (Section 3.2). We then sketch a proposal for a further generalisation, $C^{\aleph_0} I$-nets (Section 4.1), which allows for encoding preferences over multisets with unbounded multiplicities (hence, the $\aleph_O$ in $C^{\aleph_0} I$-nets), yet characterise reasoning in this framework in terms of reasoning about $C^m I$-nets (Section 4.2). The result is that at least a restricted form of reasoning on $C^{\aleph_0} I$-nets, which we call "confined reasoning", can ultimately be efficiently reduced to reasoning on CI-nets. Hence, computational procedures and systems for CI-nets [SBH16] can also be used or easily adapted to the multiset scenario.

To further motivate our generalization of CI-nets we give an example of its use in the context of a recent system for the aggregation of evidence from clinical trials [HW12]. We show how $C^{\aleph_0} I$-nets can be applied to order the evidence, which is then subject to further critical analysis by the system, based on personalized criteria (Section 5)[3].

2 Background: CI-nets.

We begin by introducing CI-nets following [BEL09]. Let O be a finite set of objects, items or goods. A *CI-net* on O consists in a set of *CI-statements*: expressions of the form $S^+, S^- : S_1 \rhd S_2$ with S^+, S^-, S_1, S_2 pairwise disjoint subsets of O, $S_1 \neq \emptyset$, $S_2 \neq \emptyset$. The intended meaning is: "if I have all the items in S^+ and none of those in S^-, I prefer obtaining all items in S_1 to obtaining all those in S_2, ceteris paribus". S^+ and S^- are the positive and negative precondition respectively; if they are both empty we write $S_1 \rhd S_2$. The formal semantics of CI-nets on O are given in terms of monotonic preference relations over 2^O. A (strict) *preference relation* is a strict partial order $>$ over 2^O; it is *monotonic* if $S_a \supset S_b$ entails $S_a > S_b$ (S_a "*dominates*" S_b) for any $S_a, S_b \in 2^O$. The preference relation $>$ *satisfies* $S^+, S^- : S_1 \rhd S_2$ if for every $S' \subseteq (O \setminus (S^+ \cup S^- \cup S_1 \cup S_2))$, $(S' \cup S^+ \cup S_1) > (S' \cup S^+ \cup S_2)$.

[3] A longer version of this work is available on arXiv.org [DH16].

A preference relation over 2^O *satisfies a CI-net* $\mathcal{N}$ if it satisfies each CI-statement in $\mathcal{N}$ and is monotonic. A CI-net $\mathcal{N}$ is *satisfiable* if there is a preference relation satisfying $\mathcal{N}$. Our main interest is in the *induced preference relation*, denoted $>_\mathcal{N}$. If $\mathcal{N}$ is satisfiable, this is the smallest preference relation satisfying $\mathcal{N}$.

An alternative operational semantics of CI-nets is given in terms of sequences of worsening flips. Let $\mathcal{N}$ be a CI-net on O, and $S_a, S_b \subseteq O$. Then $S_a \rightsquigarrow S_b$ is a *worsening flip* w.r.t $\mathcal{N}$ if either (i) $S_a \supset S_b$ ($\supset$ flip) or (ii) there is an $S^+, S^- : S_1 \rhd S_2 \in \mathcal{N}$ and $S' \subseteq (O \setminus (S^+ \cup S^- \cup S_1 \cup S_2))$ s.t. $S_a = (S' \cup S^+ \cup S_1)$ and $S_b = (S' \cup S^+ \cup S_2)$ (CI flip). See [BEL09] for a more operational version of the latter condition. We denote there being a sequence of worsening flips from S_a to S_b w.r.t. $\mathcal{N}$ as $S_a \hookrightarrow_\mathcal{N} S_b$ and say that a CI flip is w.r.t. the CI-statement that "justifies" it ; a sequence of flips is then w.r.t. the set of CI-statements that justify the flips in the sequence. Now, if $\mathcal{N}$ is satisfiable, $S_a >_\mathcal{N} S_b$ iff $S_a \hookrightarrow_\mathcal{N} S_b$. Also, $\mathcal{N}$ is satisfiable iff there is no S_a s.t. $S_a \hookrightarrow_\mathcal{N} S_a$.

CI-nets on O can express all monotonic preference relations on 2^O. The flipside is that satisfiability and dominance of CI-nets is PSPACE-complete. Nevertheless, for instance any CI-net with an "acyclic preference graph" (can be checked in PTIME) is satisfiable.

3 Encoding preferences on multisets with fixed multiplicites

3.1 C^mI-nets

We identify a multiset M on a set of objects O via its multiplicity function m_M; $m_M(o)$ is the number of occurrences of $o \in O$ in M. We will often represent such an M in the form $\{(o, m_M(o)) \mid o \in O, m_M(o) \geq 1\}$. We also use standard notation for sets to be interpreted for multisets. The following is a straightforward generalisation of CI-statements tailored to encoding finite multiset preferences, i.e. the multiplicities of the items are fixed.

Definition 1 (C^mI-statements)**.** Let M be a finite multiset on a set of objects O. A C^mI *statement* on M is an expression of the form $M^+, M^- : M_1 \rhd M_2$ where $M^+ \subseteq M$, $M^- \subseteq (M \setminus M^+)$, $M_1, M_2 \subseteq (M \setminus (M^+ \cup M^-))$, $M_1 \neq \emptyset$, $M_2 \neq \emptyset$, and $(M_1 \cap M_2) = \emptyset$.

C^mI-nets consist in a set of C^mI-statements. The semantics of C^mI-nets on M are defined in terms of preference relations over 2^M, with $>$ over 2^M *satisfying a C^mI-statement* $M^+, M^- : M_1 \rhd M_2$ if for every $M' \subseteq (M \setminus (M^+ \cup M^- \cup M_1 \cup M_2))$, we have $(M' \cup M^+ \cup M_1) > (M' \cup M^+ \cup M_2)$ (the conditions on Definition 1 assure that M' is well defined). The notions of a preference relation *satisfying a C^mI-net*, a *C^mI-net being satisfiable*, as well as the *induced preference relation for a C^mI-net* $\mathcal{N}$ ($>_\mathcal{N}$), are also defined analogously as for CI-nets. It is easy to see that C^mI-nets are indeed a generalisation of CI-nets and that a C^mI-net on M can express all monotonic preference relations on 2^M.

Example 1. Let $\mathcal{N}$ be the C^mI-net on $M = \{(a, 6), (b, 6), (c, 6)\}$ consisting of the following three (numbered, separated by ";") C^mI-statements: **(1)** $\{(a, 1)\} \rhd \{(b, 6), (c, 6), (d, 6)\}$; **(2)** $\{(a, 1)\}, \emptyset : \{(b, 1)\} \rhd \{(c, 3), (d, 3)\}$; **(3)** $\{(a, 3)\}, \{(b, 4)\} : \{(c, 3)\} \rhd \{(d, 3)\}$. In

Example 3 we reduce $\mathcal{N}$ to a CI-net; we can deduce that $\mathcal{N}$ is satisfiable from the fact that the latter has an acyclic dependency graph. The C^mI-statement **3** expresses that if one has three of a but doesn't have four of b (i.e. one has up to two of b), then one prefers having three more of c than three more of d.

Let $M_a, M_b \subseteq M$, then $M_a \rightsquigarrow M_b$ is a *worsening flip* w.r.t. a C^mI-net $\mathcal{N}$ on M if either (i) $M_a \supset M_b$ ($\supset$ flip) or (ii) there is a C^mI-statement $M^+, M^- : M_1 \rhd M_2 \in \mathcal{N}$ and an $M' \subseteq (M \setminus (M^+ \cup M^- \cup M_1 \cup M_2))$ s.t. $M_a = (M' \cup M^+ \cup M_1)$ and $M_b = (M' \cup M^+ \cup M_2)$ (CI flip). The latter condition can be verified as follows: if $\overline{M} = (M \setminus (M^+ \cup M^- \cup M_1 \cup M_2))$, then (i) $(M \setminus (M^- \cup M_2)) \supseteq M_a \supseteq (M_1 \cup M^+)$, (ii) $(M \setminus (M^- \cup M_1)) \supseteq M_b \supseteq (M_2 \cup M^+)$, and (iii) $(\overline{M} \cap M_a) = (\overline{M} \cap M_b)$.We again denote there existing a sequence of worsening flips from M_a to M_b w.r.t. $\mathcal{N}$ as $M_a \hookrightarrow_{\mathcal{N}} M_b$. The following proposition can be proven as Theorems 7 and 8 in [Bo04] (but also follows from the results in Section 3.2).

Proposition 1. Let $\mathcal{N}$ be a satisfiable C^mI-net on M; $M_a, M_b \subseteq M$. Then $M_a >_{\mathcal{N}} M_b$ if and only if $M_a \hookrightarrow_{\mathcal{N}} M_b$. Also, $\mathcal{N}$ is satisfiable iff there is no $M_a \subseteq M$ s.t. $M_a \hookrightarrow_{\mathcal{N}} M_a$.

Example 2. Consider again the C^mI-net $\mathcal{N}$ from Example 1. The following is a sequence of flips from which $\{(a,3),(b,3)\} >_{\mathcal{N}} \{(a,3),(b,2),(d,5)\}$ can be derived: $\{(a,3),(b,3)\}$ $\rightsquigarrow (CI, \mathbf{2})$ $\{(a,3),(b,2),(c,3),(d,3)\} \rightsquigarrow (CI, \mathbf{3})$ $\{(a,3),(b,2),(d,6)\} \rightsquigarrow (\supset)$ $\{(a,3),(b,2),$ $(d,5)\}$. The labels beside the symbols for flips ($\rightsquigarrow$) indicate the type of flip and, for CI flips, the C^mI-statement justifying the flip.

3.2 Reduction of C^mI-nets to CI-nets

We present a reduction of C^mI-nets to CI-nets in Appendix A. Given a multiset M on O and a C^mI-net $\mathcal{N}_M$ on M we there define a CI-net $\mathcal{N}_{S_M}$ on a set S_M and a mapping of every $M' \subseteq M$ to an $\widetilde{M'} \subseteq S_M$ s.t. propositions 2 and 3 (also proved in the appendix) hold.

Proposition 2. Let $\mathcal{N}_M$ be satisfiable and $M_a, M_b \subseteq M$. Then $M_a <_{\mathcal{N}_M} M_b$ iff $\widetilde{M_a} <_{\mathcal{N}_{S_M}} \widetilde{M_b}$.

Proposition 3. $\mathcal{N}_M$ is satisfiable iff $\mathcal{N}_{S_M}$ is satisfiable.

Example 3. The following is the CI-net corresponding to the C^mI-net from Example 1: **(4)** $\{a_6\} \rhd \{b_1, \ldots, b_6, c_1, \ldots, c_6, d_1, \ldots, d_6\}$; **(5)** $\{a_1\}, \emptyset : \{b_6\} \rhd \{c_1, c_2, c_3, d_1, d_2, d_3\}$; **(6)** $\{a_1, a_2, a_3\}, \{b_3, b_4, b_5, b_6\} : \{c_4, c_5, c_6\} \rhd \{d_1, d_2, d_3\}$; **(7)** $\big\{\{a_i\} \rhd \{a_{i+1}\} \mid 1 \leq i \leq 5\big\}$; **(8)** $\big\{\{b_i\} \rhd \{b_{i+1}\} \mid 1 \leq i \leq 5\big\}$; **(9)** $\big\{\{c_i\} \rhd \{c_{i+1}\} \mid 1 \leq i \leq 5\big\}$; **(10)** $\big\{\{d_i\} \rhd \{d_{i+1}\} \mid 1 \leq i \leq 5\big\}$. Here $S_M = \{a_1, \ldots, a_6, b_1, \ldots, b_6, c_1, \ldots, c_6\}$. The sequence of flips corresponding to that of Example 2 is: $\{a_1, a_2, a_3, b_1, b_2, b_3\} \ldots (CI, \mathbf{8})$ $\{a_1, a_2, a_3, b_1, b_2, b_6\} \rightsquigarrow (CI, \mathbf{5})$ $\{a_1, a_2, a_3, b_1, b_2, c_1, c_2, c_3, d_1, d_2, d_3\} \ldots (CI, \mathbf{9-10})$ $\{a_1, a_2, a_3, b_1, b_2, c_4, c_5, c_6, d_4, d_5, d_6\}$ $\rightsquigarrow (CI, \mathbf{6})$ $\{a_1, a_2, a_3, b_1, b_2, d_1, d_2, d_3, d_4, d_5, d_6\} \rightsquigarrow (\supset)$ $\{a_1, a_2, a_3, b_1, b_2, d_1, d_2, d_3, d_4, d_5\}$.

4 Encoding preferences on multisets with arbitrary multiplicities

4.1 $C^{\aleph_0}I$-nets: definition & extensional semantics

Although C^mI-nets are a straightforward generalisation of CI-nets they are somewhat artificial for modelling purposes. This is reflected in the complicated constraints on C^mI-statements (Definition 1) and is a consequence of the restriction to fixed multiplicites (see the discussion in the introduction). $C^{\aleph_0}I$-nets overcome this limitation and provide a more natural representation.

Let again O be a set of objects and $\mathcal{M}_O$ denote all finite multisets on O. $C^{\aleph_0}I$-nets consist of a set of $C^{\aleph_0}I$-statements which have a "precondition" and a "comparison expression". A *precondition* on o_i $(1 \leq i \leq n)$ is of the form $o_1R_1a_1, \ldots, o_nR_na_n$ where $o_i \in O$, $R_i \in \{\geq, \leq, =\}$, the a_i are integers ≥ 0. A multiset $M' \in \mathcal{M}_O$ *satisfies* the precondition, $M' \models o_1R_1a_1, \ldots, o_nR_na_n$, iff $m_{M'}(o_i)R_ia_i$ *for every* $1 \leq i \leq n$. A precondition P^+ is *satisfiable* if there is some $M' \in \mathcal{M}_O$ s.t. $M' \models P^+$; if P^+ is empty it is satisfied by any multiset. Comparison expressions on the other hand involve *update patterns* of the form $o_1 + +a_1, \ldots, o_n + +a_n$ with each $o_i \in O$ appearing at most once, the $a_i \geq 1$. Again, such an update pattern is defined *on* the objects o_i $(1 \leq i \leq n)$. The *update of a multiset* $M' \in \mathcal{M}_O$ *w.r.t. an update pattern* is $M'[o_1 + +a_1, \ldots, o_n + +a_n] := M''$ where $m_{M''}(o) = m_{M'}(o)$ for $o \in O$ but $o \neq o_i$ for every $1 \leq i \leq n$, and $m_{M''}(o_i) = m_{M'}(o_i) + a_i$ for $1 \leq i \leq n$.

Definition 2 ($C^{\aleph_0}I$-statement)**.** A $C^{\aleph_0}I$*-statement* on O is an expression $P^+ : P_1 \rhd P_2$ where P^+ is a precondition on a $O' \subseteq O$ and P_1, P_2 are update patterns defined on non-empty, disjoint subsets of O. The $C^{\aleph_0}I$-statement is *satisfiable* if the precondition P^+ is.

We often write $\{o_1, \ldots, o_n\}Ta$ for $o_1Ta, \ldots, o_nTa$, $T \in \{\geq, \leq, =, ++\}$. Informally $P^+ : P_1 \rhd P_2$ with $P^+ = \{o_i^+R_i^+a_i^+\}_{1 \leq i \leq n_+}$, $P_1 = \{o_j^1 + +a_j^1\}_{1 \leq j \leq n_1}$, $P_2 = \{o_k^2 + +a_k^2\}_{1 \leq k \leq n_2}$ means: "if I have $R_i^+a_i$ of o_i $(1 \leq i \leq n_+)$, I prefer having a_j^1 more of o_j^1 $(1 \leq j \leq n_1)$, than having a_k^2 more of o_k^2 $(1 \leq k \leq n_2)$, ceteris paribus".

Definition 3 (Semantics of $C^{\aleph_0}I$-statements)**.** A preference relation $>$ over $\mathcal{M}_O$ *satisfies* a $C^{\aleph_0}I$ statement $P^+ : P_1 \rhd P_2$ if for every $M' \in \mathcal{M}_O$ s.t. $M' \models P^+$, we have $M'[P_1] > M'[P_2]$.

Alternatively, abusing notation we define $P^+ := \{M' \in \mathcal{M}_O \mid M' \models P^+\}$ and for an update pattern $P = \{o_i + +a_i\}_{1 \leq i \leq n}$, $M_P := \{(o_i, a_i) \mid 1 \leq i \leq n\}$. Then $>$ satisfies $P^+ : P_1 \rhd P_2$ if for every $M' \in P^+$, we have $(M' \cup M_{P_1}) > (M' \cup M_{P_2})$. Note that if P^+ is unsatisfiable, then the $C^{\aleph_0}I$-statement $P^+ : P_1 \rhd P_2$ is satisfied by any preference relation. The notions of a preference relation *satisfying* a $C^{\aleph_0}I$-net , a $C^{\aleph_0}I$-net being *satisfiable* and the *preference relation induced* by a satisfiable $C^{\aleph_0}I$-net $\mathcal{N}$ $(>_{\mathcal{N}})$ are, again, defined as for CI-nets.

Example 4. The following $C^{\aleph_0}I$-net $\mathcal{N}'$ re-states the C^mI-net from Example 1, but now for arbitrary multisets over $\{a, b, c, d\}$: **(11)** $a + +1 \rhd \{b, c, d\} + +6$; **(12)** $a \geq 1 : b + +1 \rhd \{c, d\} + +3$; **(13)** $a \geq 3, b \leq 2 : c + +3 \rhd d + +3$. We will later be able to show that $\mathcal{N}'$ is also satisfiable. Moreover, also $\{(a, 3), (b3)\} >_{\mathcal{N}'} \{(a, 3), (b, 2), (d, 5)\}$.

The proof of Proposition 5 in [BEL09] which states that CI-nets on O are able to express all monotonic preferences over 2^O can be easily adapted to show that all monotonic preferences over $\mathcal{M}_O$ can be captured via $C^{\aleph_0}I$-nets on O, but does not exclude the need for an infinite number of $C^{\aleph_0}I$-statements. We leave it as an open question whether there is any useful alternative characterisation of the preference relations that can be captured efficiently (hence, also finitely) by $C^{\aleph_0}I$ (and, for that matter, CI) nets.

4.2 Operational semantics & confined reasoning for $C^{\aleph_0}I$-nets

We turn to giving an operational semantics for $C^{\aleph_0}I$-nets in terms of "worsening flips".

Definition 4 (Worsening flips for $C^{\aleph_0}I$-nets)**.** Let $\mathcal{N}$ be a $C^{\aleph_0}I$-net on O and $M_a, M_b \in \mathcal{M}_O$. Then $M_a \rightsquigarrow M_b$ is called a *worsening flip* w.r.t. $\mathcal{N}$ if either (i) $M_a \supset M_b$ ($\supset$ flip), or (ii) there is a $C^{\aleph_0}I$ statement $P^+ : P_1 \rhd P_2 \in \mathcal{N}$ and an $M' \in \mathcal{M}_O$ s.t. $M' \models P^+$, $M_a = M'[P_1]$, and $M_b = M'[P_2]$ (CI flip). Alternatively, $M_a = M' \cup M_{P_1}$, $M_b = M' \cup M_{P_2}$ for an $M' \in P^+$ or operationally: (i) $M_{P_1} \subseteq M_a$, (ii) $M_{P_2} \subseteq M_b$, (iii) $(M_a \setminus M_{P_1}) = (M_b \setminus M_{P_2})$, and (iv) if $M' = (M_a \setminus M_{P_1}) = (M_b \setminus M_{P_2})$, then $M' \in P^+$ (i.e. $M' \models P^+$).

Again, $M_a \hookrightarrow_{\mathcal{N}} M_b$ denotes there exists a sequence of worsening flips from M_a to M_b w.r.t. the $C^{\aleph_0}I$-net $\mathcal{N}$. Proposition 4 can also be proven analogously to Theorems 7,8 in [Bo04].

Proposition 4. Let $\mathcal{N}$ be a satisfiable $C^{\aleph_0}I$-net defined on O, $M_a, M_b \in \mathcal{M}_O$. Then $M_a >_{\mathcal{N}} M_b$ iff $M_a \hookrightarrow_{\mathcal{N}} M_b$. Also, $\mathcal{N}$ is satisfiable iff there is no $M_a \in \mathcal{M}_O$ s.t. $M_a \hookrightarrow_{\mathcal{N}} M_a$.

Note that there is a sequence of flips for $\{(a,3),(b3)\} >_{\mathcal{N}'} \{(a,3),(b,2),(d,5)\}$ where $\mathcal{N}'$ is the $C^{\aleph_0}I$-net from Example 4 that mirrors the sequence of flips in Example 3 and makes use of the $C^{\aleph_0}I$-statements **12** and **13**. Proposition 5, Corollary 1 and 2 give a straightforward characterisation of reasoning about $C^{\aleph_0}I$-nets in terms of "confined reasoning" as defined via $\hookrightarrow_{\mathcal{N},M}$ (for an $M \in \mathcal{M}_O$) in Definition 5.

Definition 5 (Confinement of sequences of worsening flips)**.** Let $M \in \mathcal{M}_O$. A sequence of worsening flips $M_a = M_1 \ldots M_n = M_b$ w.r.t. a $C^{\aleph_0}I$-net $\mathcal{N}$ on O is *confined to* M if each flip $M_i \rightsquigarrow M_{i+1}$ (for $1 \leq i < n$) in the sequence is s.t. $M_i, M_{i+1} \subseteq M$. $M_a \hookrightarrow_{\mathcal{N},M} M_b$ denotes there being a sequence of worsening flips from M_a to M_b confined to M. Finally, $\mathcal{N}$ is *c-consistent* w.r.t M if there is no $M_a \subseteq M$ s.t. $M_a \hookrightarrow_{\mathcal{N},M} M_a$.

Proposition 5. Let $\mathcal{N}$ be a $C^{\aleph_0}I$-net on O. $M_a \hookrightarrow_{\mathcal{N}} M_b$ iff $M_a \hookrightarrow_{\mathcal{N},M} M_b$ for an $M \in \mathcal{M}_O$.

Corollary 1. *If $\mathcal{N}$ is satisfiable, then $M_a >_{\mathcal{N}} M_b$ iff $M_a \hookrightarrow_{\mathcal{N},M} M_b$ for an $M \in \mathcal{M}_O$.*

Corollary 2. *$\mathcal{N}$ is satisfiable iff $\mathcal{N}$ is* c-consistent *w.r.t every $M \in \mathcal{M}_O$.*

Now usually one will only be interested in determining whether $M_a >_{\mathcal{N}} M_b$ for some $(M_a, M_b) \in U$ where $U \subseteq \mathcal{M}_O \times \mathcal{M}_O$ is what we called an *evaluation context* in the introduction (in particular, $|U| = 1$). Hence one would also like to know some (small) $M \in \mathcal{M}_O$ s.t. $\hookrightarrow_{\mathcal{N},M}$ *captures* $\hookrightarrow_{\mathcal{N}}$ for U, i.e. $M_a \hookrightarrow_{\mathcal{N}} M_b$ iff $M_a \hookrightarrow_{\mathcal{N},M} M_b$ for every $(M_a, M_b) \in U$.

Example 5. Consider again the $C^{\aleph_0}I$-net $\mathcal{N}'$ from Example 4. This $C^{\aleph_0}I$-net also has the analogue to an acyclic dependency graph for CI-nets. This means that given an initial multiset M_a, lets say $M_a = \{(a, 3), (b, 3)\}$, one can compute an upper bound on the number of instances of each object one will be able to add to the objects in M_a via worsening flips. Let $\#o$ denote this number for each $o \in O$. Then $\#a = 3$, $\#b = 3 + (\#a * 6) = 21$, $\#c = (\#a * 6) + (\#b * 3) = 81$, $\#d = (\#a * 6) + (\#b * 3) + (\#c * 3) = 324$, and therefore $\hookrightarrow_{\mathcal{N},M}$, with $M = \{(a, 3), (b, 21), (c, 81), (d, 324)\}$, *captures* $\hookrightarrow_{\mathcal{N}}$ for $U = \{(M_a, M') \mid M' \in \mathcal{M}_O\}$.

Example 6 (CI-nets as $C^{\aleph_0}I$-nets)**.** A CI-statement $c = S^+, S^- : S_1 \rhd S_2$ in a CI-net $\mathcal{N}$ can be written as the $C^{\aleph_0}I$-statement $\hat{c} := P^+ : C, P^+ := P_1^+ \cup P_2^+ \cup P_3^+, P_1^+ := \{s^+ = 1 \mid s^+ \in S^+\}$, $P_2^+ := \{s = 0 \mid s^- \in (S^- \cup S_1 \cup S_2)\}$, $P_3^+ := \{s \leq 1 \mid s \in (O \setminus (S^+ \cup S^- \cup S_1 \cup S_2))\}$, and $C := \{s_1 + +1 \mid s_1 \in S_1\} \rhd \{s_2 + +1 \mid s_2 \in S_2\}$. The CI-flips w.r.t. $\mathcal{N}$ and $\hat{\mathcal{N}} := \{\hat{c} \mid c \in \mathcal{N}\}$ are exactly the same and hence $\hookrightarrow_{\hat{\mathcal{N}},O}$ captures $\hookrightarrow_{\hat{\mathcal{N}}}$ for $U = (O \times O)$.

We sketch a translation of confined reasoning about $C^{\aleph_0}I$-nets to C^mI-nets in Appendix B. The C^mI-net from Example 1 is, in fact, the C^mI-net that results when applying this translation for confined reasoning w.r.t. $\{(A, 6), (B, 6), (C, 6)\}$ and the $C^{\aleph_0}I$-net in Example 4. The satisfiability of the $C^{\aleph_0}I$-net in Example 4 follows from Corollary 2 and the fact that the translation of confined reasoning w.r.t. this $C^{\aleph_0}I$-net and *any* $M \in \mathcal{M}_O$ produces a C^mI-net which can be reduced to a CI-net with an acyclic preference graph.

5 Encoding preferences in evidence aggregation

In this Section we show how $C^{\aleph_0}I$-nets can be applied in the context of the system for aggregating evidence presented in [HW12] (see [Wi15] for a recent use). In this system evidence from clinical trials is initially collected in the form of tables of which Table 1 could be an extract (our example is based on Table 3 in [HW12]). Table 1 summarises possible results from meta-analyses ("MA") for patients who have raised pressure in the eye and are at risk of glaucoma. The results of the studies ("Outcome value") have been normalised so that the values are desirable, i.e. they indicate the degree to which the treatment which has fared better in the study presents an improvement (column "Net outcome"; in Table 1 ">","<" means the study speaks for PG, BB resp.). Given the evidence in Table 1, the question is whether PG or BB are better to treat glaucoma.

A first step towards a solution of this problem is determining what sets of evidence items that can be used to argue in favour of the treatments are of more value in terms of preferences over "*benefits*": outcome indicator - normalised outcome value pairs. More to the point, since for methodological reasons (mainly, to avoid bias and for purposes of reuse), preferences need to be determined independently of the available evidence, the preference relation is in terms of *possible* sets of benefits, i.e. all possible sets of pairs of (normalised) outcome indicator-value pairs. Specifying preferences over "benefit sets" allows for a personalised dimension in the decision process, i.e. of considerations which have to do with, e.g., a specific patient or the experience of the medical professional. Other more "objective" elements (like statistical significance - column "Sig" in Table 1) can be incorporated in further stages of the decision process as outlined in [HW12].

ID	Left	Right	Outcome indicator	Outcome value	Net outcome	Sig	Type
e_{01}	PG	BB	change in IOP (SO)	-2.32 (m)	>	no	MA
e_{02}	PG	BB	acceptable IOP (SO)	1.54 (s)	>	yes	MA
e_{03}	PG	BB	respiratory prob	0.9 (s)	>	yes	MA
e_{04}	PG	BB	respiratory prob	0.85 (s)	>	yes	MA
e_{05}	PG	BB	cardio prob	0.82 (s)	>	no	MA
e_{06}	PG	BB	hyperaemia	0.61 (m)	<	yes	MA
e_{07}	PG	BB	drowsiness	0.58 (m)	<	yes	MA
e_{08}	PG	BB	drowsiness	0.71 (m)	<	yes	MA
e_{09}	PG	BB	drowsiness	0.62 (m)	<	yes	MA

Tab. 1: Normalised results of several meta-analysis studies comparing prostaglandin analogue (PG) and beta-blockers (BB) for patients with raised intraocular pressure.

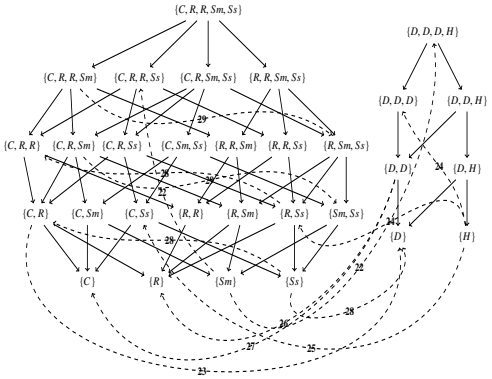


Fig. 1: Graphical representation of the preference relation induced by the $C^m I$-net in Example 8. Solid arcs are obtained by $\supset$, dotted arcs also via CI-flips.

In [HW12] the authors only consider the incorporation of preferences between *sets* of benefits in their system and for this purpose CI-nets would be a natural choice. Also allowing preferences over multisets of benefits to be stated becomes relevant when one considers that -, especially as a result of the use of some abstraction over the outcome indicators and values,- there may be more than one evidence item expressing the same benefit. Example 7 illustrates the use of $C^{\aleph_0} I$-nets for encoding preferences over multisets of benefits such as those appearing in Table 1 but where we introduce a natural abstraction. We consider both "change in IOP" and "acceptable IOP" ("IOP" = interocular pressure) as part of the "significant outcomes" which we denote "SO"; we partition the outcome indicators into "s", "m", and "l" standing for a "small", "medium", and "large" improvement respectively. The values in parentheses beside the entries for "Outcome indicator" and "Outcome value" show a possible result of applying this abstraction to the results in Table 1.

Example 7. The following is a $C^{\aleph_0} I$-net on the benefits that appear in Table 1. We use C, D, H, R for (*cardio prob*, *s*), (*drowsiness*, *m*), (*hyperaemia*, *m*), and (*respiratory prob*, *s*) respectively, while $Sm := (SO, m)$ and $Ss := (SO, s)$. $\{a_1, \ldots, a_n\}\#\#$ denotes the maximum number of each of $a_1, \ldots, a_n$ in any specific evaluation context. The $C^{\aleph_0} I$ net consists in the statements: **(14)** $Sm + +1 \rhd \{C, D, R, Ss\}\#\#, H + +1$; **(15)** $\{C, R\} + +1 \rhd D + +1$; **(16)** $C = 0 : H + +1 \rhd D\#\#, \{R, Ss\} + +1$; **(17)** $R = 0 : H + +1 \rhd D\#\#, \{C, Ss\} + +1$; **(18)** $C = 0 : D + +2 \rhd R + +1$; **(19)** $R = 0 : D + +2 \rhd C + +1$; **(20)** $Sm = 0 : Ss + +1 \rhd \{C, D, R\} + +1$; **(21)** $Sm \geq 1 : \{C, R\} + +1 \rhd Ss + +1$. $C^{\aleph_0} I$-statement **20**, for example, states that if one does not have any evidence for a modest improvement in the significant outcomes, then evidence for even a small improvement for any of the significant outcomes is preferred to evidence showing an improvement in drowsiness as well as cardio and respiratory problems.

Example 8 gives the encoding of confined reasoning for the $C^{\aleph_0} I$-net of Example 7 w.r.t. all benefits occurring in Table 1. Figure 1 shows the preference relation induced by the

$C^m I$-net in Example 8, but considering only sets of benefits which all result from the *same* treatment according to Table 1.

Example 8. The following is the encoding of confined reasoning for the $C^{\aleph_0} I$-net of Example 7 w.r.t. the multiset $M = \{(C,1),(D,3),(H,1),(R,2),(Sm,1),(Ss,1)\}$. For the encoding we interpret $o\#\#$ as the max number of occurrences of o in M. **(22)** $\{(Sm,1)\} \rhd \{(C,1),(D,3),(H,1),(R,2),(Ss,1)\}$; **(23)** $\{(C,1),(R,1)\} \rhd \{(D,1)\}$; **(24)** $\emptyset, \{(C,1)\} : \{(H,1)\} \rhd \{(D,3),(R,1),(Ss,1)\}$; **(25)** $\emptyset, \{(R,2)\} : \{(H,1)\} \rhd \{(D,3),(C,1),(Ss,1)\}$; **(26)** $\emptyset, \{(C,1)\} : \{(D,2)\} \rhd \{(R,1)\}$; **(27)** $\emptyset, \{(R,2)\} : \{(D,2)\} \rhd \{(C,1)\}$; **(28)** $\emptyset, \{(Sm,1)\} : \{(Ss,1)\} \rhd \{(C,1),(D,1),(R,1)\}$; **(29)** $\{(Sm,1)\}, \emptyset : \{(C,1),(R,1)\} \rhd \{(Ss,1)\}$.

6 Conclusion & future work

As fas as we are aware this is the first work to present a framework for encoding ordinal multiset preferences, certainly in the context of CI-nets. Our results allow for sound and complete procedures for confined reasoning, the issue of finding a multiset that captures the preference relation w.r.t. a $C^{\aleph_0} I$-net for an evaluation context remaining largely unexplored. As is determining subclasses of $C^{\aleph_0} I$-nets beyond acyclic ones where such a multiset can be found or satisfiability is guaranteed. Complexity issues remain to be explored. Finally, techniques for e.g. CP nets "in practice" [Al15] as well as algorithms and systems for CI-nets [SBH16] can be adapted and optimised for the multiset scenario.

References

[Al15] Allen, T. E.: CP-nets: From Theory to Practice. In (Walsh, T., ed.): Proceedings of ADT 2015. Vol. 9346. Lecture Notes in Computer Science, Springer, pp. 555–560, 2015.

[BEL09] Bouveret, S.; Endriss, U.; Lang, J.: Conditional Importance Networks: A Graphical Language for Representing Ordinal, Monotonic Preferences over Sets of Goods. In (Boutilier, C., ed.): Proceedings of IJCAI 2009. Pp. 67–72, 2009.

[Bo04] Boutilier, C.; Brafman, R. I.; Domshlak, C.; Hoos, H. H.; Poole, D.: CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements. JAIR 21/, pp. 135–191, 2004.

[DH16] Diller, M.; Hunter, A.: Encoding monotonic multi-set preferences using CI-nets: preliminary report. CoRR abs/1611.02885/, 2016, URL: http://arxiv.org/abs/1611.02885.

[HW12] Hunter, A.; Williams, M.: Aggregating evidence about the positive and negative effects of treatments. Artif Intell Med 56/3, pp. 173–190, 2012.

[SBH16] Santhanam, G. R.; Basu, S.; Honavar, V.: Representing and Reasoning with Qualitative Preferences: Tools and Applications. Morgan & Claypool Publishers, 2016.

[Wi15] Williams, M.; Liu, Z. W.; Hunter, A.; Macbeth, F.: An updated systematic review of lung chemo-radiotherapy using a new evidence aggregation method. Lung Cancer 87/3, pp. 290–295, 2015.

A Reduction of C^mI-nets to CI-nets

Let M be a multiset on O and a $\mathcal{N}_M$ a C^mI-net on M. We here define a CI-net $\mathcal{N}_{S_M}$ on a set S_M and a mapping of every $M' \subseteq M$ to an $\widetilde{M'} \subseteq S_M$ s.t. propositions 2 and 3 hold.

We start by introducing some notation. Given some $o \in O$ and i, j s.t. $i, j \geq 1$ we define the *forward-generated set of j indexed copies from i* of o as $[o]^F_{i,j} := \{o_i, o_{i+1}, \ldots, o_{i+(j-1)}\}$ and the *backward-generated set of j indexed copies from i* of o as $[o]^B_{i,j} := \{o_i, o_{i-1}, \ldots, o_{i-(j-1)}\}$. If $j = 0$, we define $[o]^F_{i,j} = [o]^B_{i,j} := \emptyset$. Then $S_M := \bigcup\{[o]^F_{1,m_M(o)} \mid o \in O\}$. We call $[o]^F_{1,m_M(o)} = [o]^B_{m_M(o),m_M(o)}$ for $o \in O$ the *set of indexed copies of o in* S_M.

For some $M' \subseteq M$, $*M'$ includes all sets which, for each $o \in O$, have the same *number* of elements from the set of indexed copies of o in S_M as instances of o there are in M'. Formally, we define $*M'$ to be the set $\{S \subseteq S_M \mid |S \cap [o]^F_{1,m_M(o)}| = m_{M'}(o)$ *for every* $o \in O\}$. Clearly, in particular $*M = \{S_M\}$. We also (partially) order the sets in S_M via the order $>_\flat$ defined as the transitive closure of the binary relation $\{(S_1, S_2) \mid (S_1 \cup S_2) \setminus (S_1 \cap S_2) = \{o_i, o_j\}$ *s.t.* $o \in O, o_i \in S_1, o_j \in S_2,$ *and* $j = i + 1\}$. Crucial for our purposes is that there is a unique maximal element $\bigcup\{[o]^F_{1,m_{M'}(o)} \mid o \in O\}$ w.r.t. $>_\flat$ within $*M'$ for every $M' \subseteq M$. We denote this maximal element as $\widetilde{M'}$. Note that in particular $\widetilde{M} = S_M$.

Next we proceed to define for a C^mI-statement $c \in \mathcal{N}_M$ the corresponding CI statement $\widehat{c} \in \mathcal{N}_{S_M}$. Assume c is of the form $M^+, M^- : M_1 \rhd M_2$. For simplicity we write the multiplicity functions of M^+, M^-, M_1, M_2 as m^+, m^-, m_1, m_2 respectively. Then $\widehat{c} := \widehat{M^+}, \widehat{M^-} : \widehat{M_1} \rhd \widehat{M_2}$ where $\widehat{M^+} := \bigcup\{[o]^F_{1,m^+(o)} \mid o \in O\}$, $\widehat{M^-} := \bigcup\{[o]^B_{m_M(o),m^-(o)} \mid o \in O\}$,$\widehat{M_1} := \bigcup\{[o]^B_{m_M(o)-m^-(o),m_1(o)} \mid o \in O\}$, $\widehat{M_2} := \bigcup\{[o]^F_{m^+(o)+1,m_2(o)} \mid o \in O\}$. We denote the set of CI statements corresponding to the $c \in \mathcal{N}_M$ as cs_1. Apart from the CI-statements in cs_1, $\mathcal{N}_{S_M}$ also contains the set of CI-statements $cs_2 := \{o_i \rhd o_j \mid o \in O, 1 \leq i < m_M(o), j = i + 1\}$.

Lemma 1. *If $M_a \rightsquigarrow M_b$ is a CI-flip w.r.t $c \in \mathcal{N}_M$ then there is a $S_{M_a} \in *M_a$ s.t. $S_{M_a} \rightsquigarrow \widetilde{M_b}$ is a CI-flip w.r.t. $\widehat{c} \in \mathcal{N}_{S_M}$. Also, if there are $S_{M_a} \in *M_a$, $S_{M_b} \in *M_b$ s.t. $S_{M_a} \rightsquigarrow S_{M_b}$ is a CI-flip w.r.t. $\widehat{c} \in \mathcal{N}_{S_M}$, then $M_a \rightsquigarrow M_b$ is a CI-flip w.r.t. $c \in \mathcal{N}_M$.*

Proof. Let $c = M^+, M^- : M_1 \rhd M_2$. The set of all CI-flips "induced" by $\widehat{c} \in \mathcal{N}_{S_M}$ are of the form $(S' \cup \widehat{M^+} \cup \widehat{M_1}) \rightsquigarrow (S' \cup \widehat{M^+} \cup \widehat{M_2})$ with $\widehat{M^+}, \widehat{M_1}, \widehat{M^+}, \widehat{M_2}$ defined as above and $S' \subseteq \bigcup\{[o]^F_{X_o,Y_o} \mid o \in O\}$ where $X_o = m^+(o)+m_2(o)+1, Y_o = m_M(o)-(m^+(o)+m^-(o)+m_1(o)+m_2(o))$ for each $o \in O$. Then for each M' s.t. $M_a = M' \cup M^+ \cup M_1$, $M_b = M' \cup M^+ \cup M_2$, and $M_a \rightsquigarrow M_b$ is a CI-flip w.r.t. $c \in \mathcal{N}_M$, there is a $S' \subseteq \bigcup\{[o]^F_{X_o,Y_o} \mid o \in O\}$, $S_a \in *M_a$, $S_b \in *M_b$ s.t. $S_a = S' \cup \widehat{M^+} \cup \widehat{M_1}$, $S_b = S' \cup \widehat{M^+} \cup \widehat{M_2}$ and $S_a \rightsquigarrow S_b$ is a CI-flip w.r.t. $\widehat{c} \in \mathcal{N}_{S_M}$. In particular, by construction one can pick $S' = \bigcup\{[o]^F_{X_o,m_{M'}(o)} \mid o \in O\}$ and then $S_b = \widetilde{M_b}$. Also, for $S_a = S' \cup \widehat{M^+} \cup \widehat{M_1}$, $S_b = S' \cup \widehat{M^+} \cup \widehat{M_2}$ s.t. $S_a \rightsquigarrow S_b$ is a CI-flip w.r.t. $\widehat{c} \in \mathcal{N}_{S_M}$, $M_a \rightsquigarrow M_b$ is a CI-flip w.r.t $c \in \mathcal{N}_M$, where $M_a = (M' \cup M^+ \cup M_1)$, $M_b = (M' \cup M^+ \cup M_2)$ and $m_{M'}(o) = |S' \cap [o]^F_{1,m_M(o)}|$ for each $o \in O$. □

Lemma 2. *If $S, S' \subseteq S_M$ and $S >_\flat S'$, then there is a sequence of cs_2 flips from S to S' w.r.t. $\mathcal{N}_{S_M}$.*

Proof. (sketch) This lemma follows from the fact that $>_\flat$ is equivalent to the transitive closure (within S_M) of the CI-flips induced by cs_2. □

Lemma 3. *Let $M_a, M_b \subseteq M$. If $M_a \hookrightarrow_{\mathcal{N}_M} M_b$, then $\widetilde{M_a} \hookrightarrow_{\mathcal{N}_{S_M}} \widetilde{M_b}$. Also, let $S_{M_a} \rightharpoonup_{\mathcal{N}_{S_M}} S_{M_b}$ for some $S_{M_a} \in *M_a$, $S_{M_b} \in *M_b$ denote that there exists a sequence involving at least one non-cs_2 flip w.r.t $\mathcal{N}_{S_M}$. We call such a sequence* non-trivial. *Then, if $S_{M_a} \rightharpoonup_{\mathcal{N}_{S_M}} S_{M_b}$, $M_a \hookrightarrow_{\mathcal{N}_M} M_b$ also is the case.*

Proof. We start by proving by induction on $k \geq 0$, that if there exists a sequence $M_a, \ldots, M_b$ w.r.t. $\mathcal{N}_M$ with k CI flips, then there is a sequence $\widetilde{M_a}, \ldots, \widetilde{M_b}$ w.r.t. $\mathcal{N}_{S_M}$ with k cs_1 flips. The base case ($k = 0$) follows from the fact that if $M_a \supset M_b$ then $\widetilde{M_a} \supset \widetilde{M_b}$ and hence there is a sequence $\widetilde{M_a}, \ldots, \widetilde{M_b}$ consisting only of $\supset$ flips w.r.t. $\mathcal{N}_{S_M}$.

For the inductive case assume that there exists a sequence $M_a, \ldots, M_b$ w.r.t $\mathcal{N}_M$ with $k + 1 \geq 1$ CI flips. Consider the last M_c, M_d in the sequence s.t. $M_c \rightsquigarrow M_d$ is a CI flip. By inductive hypothesis then there is a sequence of flips $\widetilde{M_a}, \ldots, \widetilde{M_c}$ w.r.t. $\mathcal{N}_{S_M}$ with k cs_1 flips. By Lemma 1 there exists a $S_{M_c} \in *M_c$ s.t. $S_{M_c} \rightsquigarrow \widetilde{M_d}$ is a CI-flip w.r.t. $\widehat{c} \in \mathcal{N}_{S_M}$. Hence $\widetilde{M_a}, \ldots, \widetilde{M_c}, \ldots, S_{M_c}, \widetilde{M_d}, \ldots, \widetilde{M_b}$ is a sequence w.r.t. $\mathcal{N}_{S_M}$ with $k+1$ cs_1 flips. Here $\widetilde{M_c} = S_{M_c}$ or $\widetilde{M_c}, \ldots, S_{M_c}$ is a sequence of cs_2 flips (that such a sequence exists follows from Lemma 2). Also $\widetilde{M_d} = \widetilde{M_b}$ or $\widetilde{M_d}, \ldots, \widetilde{M_b}$ is a sequence consisting only of $\supset$ flips.

We now prove by induction on k, that if there exists a non-trivial sequence $S_{M_a}, \ldots, S_{M_b}$ w.r.t. $\mathcal{N}_{S_M}$ with k cs_1 flips for some $S_{M_a} \in *M_a$, and $S_{M_b} \in *M_b$, then there exists a sequence $M_a, \ldots, M_b$ w.r.t. $\mathcal{N}_M$ with the k CI flips. If $k = 0$, then the sequence $S_{M_a}, \ldots, S_{M_b}$ must have at least one $\supset$ flip, i.e. $S_{M_a} \supset S_{M_b}$; therefore also $M_a \supset M_b$, and hence there is a sequence $M_a, \ldots, M_b$ consisting only of $\supset$ flips w.r.t. $\mathcal{N}_M$.

For the inductive case assume that there exists a sequence $S_{M_a}, \ldots, S_{M_b}$ w.r.t. $\mathcal{N}_{S_M}$ with $k + 1 \geq 1$ cs_1 flips for some $S_{M_a} \in *M_a, S_{M_b} \in *M_b$. Consider the last cs_1 flip $S_{M_c} \rightsquigarrow S_{M_d}$ in the sequence, with $S_{M_c} \in *M_c, S_{M_d} \in *M_d$ for $M_c, M_b \subseteq M$. If the sequence $S_{M_a}, \ldots, S_{M_c}$ is trivial we have that $M_a = M_c$. Otherwise, by inductive hypothesis there is a sequence $M_a, \ldots, M_c$ w.r.t. $\mathcal{N}_M$ with k CI flips. Moreover, by Lemma 1 also $M_c \rightsquigarrow M_d$ is a CI flip w.r.t. $\mathcal{N}_M$. Finally, either $M_d = M_b$ (i.e. $S_{M_d}, \ldots, S_{M_b}$ is a trivial sequence) or $S_{M_d} \supset S_{M_b}$ (i.e. $S_{M_d}, \ldots, S_{M_b}$ involves $\supset$-flips) in which case $M_d \supset M_b$. In all cases we have a sequence $M_a, \ldots, M_c, M_d, \ldots, M_b$ w.r.t. $\mathcal{N}_M$ with $k + 1$ CI flips. □

Proposition 2. Let $\mathcal{N}_M$ be satisfiable and $M_a, M_b \subseteq M$. Then $M_a <_{\mathcal{N}_M} M_b$ iff $\widetilde{M_a} <_{\mathcal{N}_{S_M}} \widetilde{M_b}$.

Proof. If $M_a <_{\mathcal{N}_M} M_b$, then $M_a \hookrightarrow_{\mathcal{N}_M} M_b$. Hence, from Lemma 3 it follows that $\widetilde{M_a} <_{\mathcal{N}_{S_M}} \widetilde{M_b}$. Assume now $\widetilde{M_a} <_{\mathcal{N}_{S_M}} \widetilde{M_b}$ and, therefore, $\widetilde{M_a} \hookrightarrow_{\mathcal{N}_{S_M}} \widetilde{M_b}$. Note that then in fact $\widetilde{M_a} \rightharpoonup_{\mathcal{N}_{S_M}} \widetilde{M_b}$ (for any sequence $S' \hookrightarrow_{\mathcal{N}_{S_M}} S''$ consisting only in cs_2 flips it holds

that $S_c, S_d \in *M_c$ for some $M_c \subseteq M$ and by assumption $M_a \neq M_b$.). Hence, from Lemma 3 it follows that $M_a \hookrightarrow_{\mathcal{N}_M} M_b$. □

Proposition 3. $\mathcal{N}_M$ is satisfiable iff $\mathcal{N}_{S_M}$ is satisfiable.

Proof. We prove that $\mathcal{N}_M$ is unsatisfiable iff $\mathcal{N}_{S_M}$ is unsatisfiable. Assume first that $\mathcal{N}_M$ is unsatisfiable. This means that there is an $M_a \subseteq M$ s.t. $M_a \hookrightarrow_{\mathcal{N}_M} M_a$. Hence, from Lemma 3 it follows that $\widetilde{M_a} \hookrightarrow_{\mathcal{N}_{S_M}} \widetilde{M_a}$, i.e. $\mathcal{N}_{S_M}$ is unsatisfiable. Assume now that $\mathcal{N}_{S_M}$ is unsatisfiable. Then there is a $S_{M_a} \in *M_a$ with $M_a \subseteq M$ s.t. $S_{M_a} \hookrightarrow_{\mathcal{N}_{S_M}} S_{M_a}$. In fact $S_{M_a} \rightharpoonup_{\mathcal{N}_{S_M}} S_{M_a}$ since $\mathcal{N}_{S_M}$ without the CI-statements has an acyclic dependency graph and is, therefore, satisfiable. Hence, from Lemma 3 it follows that $M_a \hookrightarrow_{\mathcal{N}_M} M_a$, i.e. $\mathcal{N}_M$ is unsatisfiable. □

B Translating confined reasoning about $C^{\aleph_0}I$-nets to reasoning about C^mI-nets

Let $\mathcal{N}$ be a $C^{\aleph_0}I$ net on O, $M \in \mathcal{M}_O$. We here sketch a translation of confined reasoning w.r.t. $\mathcal{N}$ and an $M \in \mathcal{M}_O$ to reasoning about a C^mI-net $\mathcal{N}'$ on M. Concretely, let $c = P^+ : P_1 \rhd P_2$ be a $C^{\aleph_0}I$ statement in $\mathcal{N}$. c is *meaningful w.r.t. M* if there is an $M' \in P^+$, s.t. $(M' \cup M_{P_1}) \subseteq M$, and $(M' \cup M_{P_2}) \subseteq M$. For our translation we rewrite each such $c \in \mathcal{N}$ into a C^mI-statement $c' \in \mathcal{N}'$ that is *equivalent to c for M*, i.e. the CI flips w.r.t. c' are exactly those in $\{(M' \cup M_{P_1}) \rightsquigarrow (M' \cup M_{P_2}) \mid (M' \cup M_{P_1}) \subseteq M, (M' \cup M_{P_2}) \subseteq M\}$. This means, the CI flips w.r.t. the resulting C^mI-net $\mathcal{N}' = \{c' \mid c \in \mathcal{N}\}$ are exactly those CI flips $M_a \rightsquigarrow M_b$ w.r.t. $\mathcal{N}$ s.t. $M_a, M_b \subseteq M$. As a consequence, $M_a \hookrightarrow_{\mathcal{N},M} M_b$ iff $M_a \hookrightarrow_{\mathcal{N}'} M_b$.

So assume $c = P^+ : P_1 \rhd P_2$ is a $C^{\aleph_0}I$- statement in $\mathcal{N}$ that is meaningful w.r.t. M. Then, since c is also satisfiable note that the precondition and comparison expressions can be written in the form $P^+ = \{o_i^+ \geq a_i^+\}_{1\leq i\leq p} \cup \{o_j^- \leq a_j^-\}_{1\leq j\leq q}$, $P_1 = \{o_k^1 + +a_k^1\}_{1\leq k\leq r}$, $P_2 = \{o_l^2 + +a_l^2\}_{1\leq l\leq s}$ where each $o \in O$ appears at most once in a sub-expression of the form $o_i^+ \geq a_i^+$ and at most once in a sub-expression of the form $o_j^- \leq a_j^-$ in the precondition. Let $O^* := \{o_i^+\}_{1\leq i\leq p} \cup \{o_j^-\}_{1\leq j\leq q} \cup \{o_k^1\}_{1\leq k\leq r} \cup \{o_l^2\}_{1\leq l\leq s}$. We re-label the objects in $O^* \subseteq O$ to $\{o_1, \ldots, o_m\}$ ($m = p + q + r + s$). We now define for each $1 \leq h \leq m$, $A_h^x := a_i^x$ if there is a $t \in \{1, \ldots, y\}$ s.t. $o_h = o_t^x$, $A_h^x := 0$ otherwise for $x = +$, $x = 1$, $x = 2$ and $y = p$, $y = r$, $y = s$ respectively. Also, $A_h^- := a_j^-$ if there is a $j \in \{1, \ldots, q\}$ s.t. $o_h = o_j^-$, $A_h^- := m_M(o_h)$ otherwise. Finally, for each $1 \leq h \leq m$ we define $B_h^- := max\{I \mid A_h^+ \leq I \leq A_h^- \text{ and } I + A_h^1 + A_h^2 \leq m_M(o_h)\}$. Then $c' \in \mathcal{N}'$ is the C^mI-statement $M^+, M^- : M_1 \rhd M_2$ where $M^+ := \{(o_h, A_h^+) \mid 1 \leq h \leq m, A_h^+ > 0\}$, $M^- := \{(o_h, X_{o_h}) \mid 1 \leq h \leq m, X_{o_h} > 0)\}$, $X_{o_h} := m_M(o_h) - B_h^- - A_h^1 - A_h^2$, $M^1 := \{(o_h, A_h^1) \mid 1 \leq h \leq m, A_h^1 > 0\}$, $M^2 := \{(o_h, A_h^2) \mid 1 \leq h \leq m, A_h^2 > 0\}$.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 181

Personalized Stream Analysis with PreferenceSQL

Lena Rudenko[1] and Markus Endres[2]

Abstract: In this paper we present our demo application which allows preference-based search for interesting information in a data stream. In contrast to existing stream analysis services, the application uses the attributes of the stream records in combination with soft conditions to achieve the best possible result for a query.

Keywords: Stream analysis, preferences, personalization

1 Introduction

Stream query processing is becoming increasingly important as more time-oriented data is produced and analyzed nowadays. Existing approaches for stream analysis have to consider the special characteristics of data streams which do not take the form of persistent database relations, but rather arrive in continuous, rapid and time-varying data objects. Hence, analyzing streams is a difficult and complex task which is in the focus of many researchers all over the world, cp. for example [ScZ05].

Due to the increasing amount of data – often produced by humans in social networks via mobile devices – such an endless flow of stream objects contains important and interesting records as well as spam and information trash. In order to distinguish between important and unimportant information one has to observe the stream objects which have attributes with additional information. These attributes can be used to analyze a stream with the goal to find the most relevant and personalized records.

In this demo paper we present an application which exploits *user preferences* to evaluate queries on various data streams. These user preferences are like soft constraints, requiring a match-making process: *If my favorite choice is available in the dataset, I will take it. Otherwise, instead of getting nothing, I am open to alternatives, but show me only the best ones available*.

We want to show that our preference-based approach is an alternative to the common used attribute-based stream analysis which follows a conditional filtering based on hard constraints. Our application allows to build a query in an intuitive and flexible way to search for the best matches in a stream. This prevents information flooding and unimportant data as well as the empty result effect.

[1] University of Augsburg, Institute for Computer Science, Universitätsstr. 6a, 86159 Augsburg, Germany, lena.rudenko@informatik.uni-augsburg.de
[2] University of Augsburg, Institute for Computer Science, Universitätsstr. 6a, 86159 Augsburg, Germany, markus.endres@informatik.uni-augsburg.de

2 Preference SQL

In our demo application we use Preference SQL which is a declarative extension of SQL by preferences, behaving like soft constraints under the Best-Matches-Only query model, cp. [KEW11]. Syntactically, Preference SQL extends the `SELECT` statement of SQL by an optional `PREFERRING` clause, cp. Figure 1a. The keywords `SELECT`, `FROM` and `WHERE` are treated as the standard SQL keywords. The `PREFERRING` clause specifies a preference which is evaluated after the `WHERE` condition.

```
SELECT <projection>
FROM   <table_reference>
WHERE  <hard_conditions>
  PREFERRING <soft_conditions>
```

(a) PreferenceSQL syntax.

```
SELECT STREAM *
FROM TwitterStream
  PREFERRING tweet_language
  IN ('de') ELSE ('en')
  PARETO followers_count HIGHEST;
```

(b) PreferenceSQL example.

Fig. 1: PreferenceSQL query block.

[KEW11] proposes several preference constructors to specify user preferences on *numerical*, *categorical*, *temporal*, and *spatial* domains. In Figure 1b for example, the first part after the `PREFERRING` keyword is a *POS/POS* preference, which means, that a desired value of some attribute (`tweet_ language`) should be amongst a finite first set (`IN ('de')`). Otherwise it should be from a second disjoint set of attributes (`ELSE ('en')`). If this is also not feasible, better than getting nothing any other value is acceptable. The second part of the user preference is a *numerical* `HIGHEST` preference. It expresses the wish to consider authors having the *most followers* in the social network Twitter if `followers_count` is an attribute of a stream object. These preferences can be combined into more *complex preferences*. For example, the *Pareto preference* (combined by the `PARETO` keyword in the `PREFERRING` clause, cp. Figure 1b) treats two or more preferences as equally important, whereas in a Prioritization (`PRIOR TO`) one preference is more important than another one. For more details we refer to [KEW11].

3 Showcase Application

In our demo application we show that preference-based stream analysis has several advantages over current methods. For example, the evaluation conditions are more flexible, because it is possible to determine the relative importance of the user wishes. Moreover, preference-based approaches always provide the best result w.r.t. the user preferences.

Figure 2 shows the main view of our application, which allows users to construct queries and to evaluate them on data streams. Thereby one can build preference-based queries as well as "standard" queries involving hard constraints. This allows us to present the advantage of personalized query evaluation on data streams in contrast to a conditional filtering as in the `WHERE` clause of stream-based languages like CQL (Continuous Query Language) or StreamSQL.

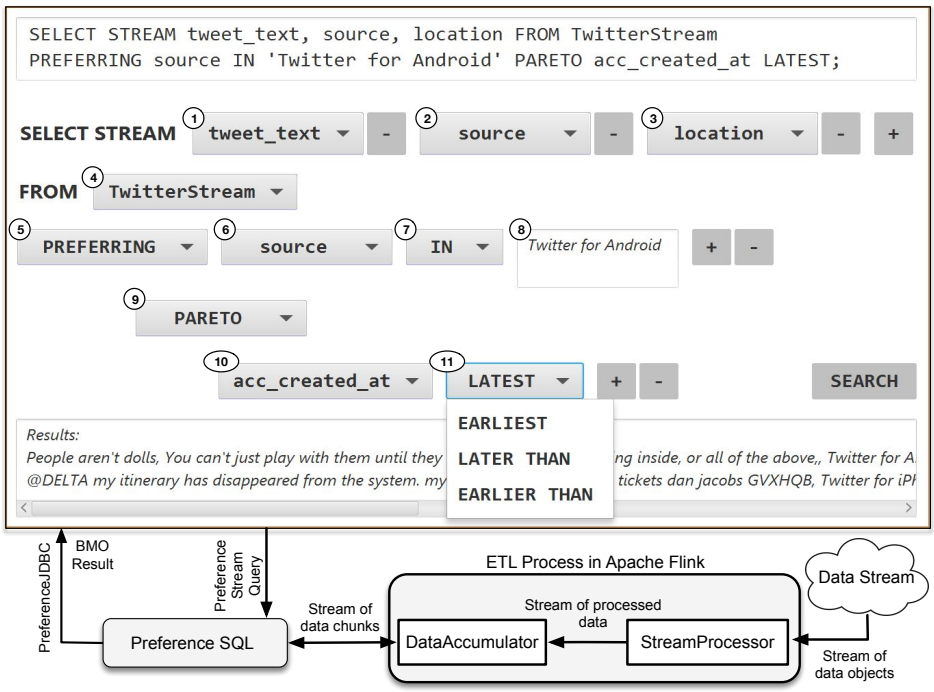


Fig. 2: Architecture of our application.

A user can build queries using *hard* or *soft constraints* (preferences). For example, in Figure 2 one wants to analyze data provided by Twitter. First, a desired stream source has to be selected from a drop-down menu list in ④. In our application a new stream connector can easily be implemented using a pluggable interface. The next step is to choose the projection attributes the user wants to see in the result. One can do it using drop-down menus where the list items are the attributes available in the stream objects (①, ② and ③ in Figure 2). The kind of attributes is different for each data source and is dynamically generated by the corresponding pre-implemented stream interface. In ⑤ the query selection mode can be specified, i.e., preference-based (`PREFERRING`) or based on hard constraints (a SQL `WHERE` clause). In ⑥ it is possible to select the attribute on which a preference or the hard condition ⑦ should be evaluated. The preference choice is determined by the data type of the selected attributes. For some preferences it is necessary to specify attribute values, such as in the POS/POS preference in Section 2. For this we provide a text field as depicted in ⑧. Such simple preferences can be combined to `PARETO` or `PRIORITIZATION` in ⑨ with additional preference conditions added by "+". In our example the preference in ⑥, ⑦ and ⑧ is equally important to a `LATEST` preference, cp. ⑩ and ⑪, which expresses that we prefer tweets posted in accounts created most recently. For hard conditions (`WHERE` in ⑤) the mode in ⑨ refers to `AND` or `OR`. Finally, the constructed query can be seen at the top of our application. Afterwards, the generated query is sent to our *preference-based stream processing framework* for evaluation.

The current University prototype of the Preference SQL system adopts a (Java 1.7-based) middleware approach as depicted at the bottom of Figure 2. For a seamless application integration we have extended standard JDBC technology towards a Preference SQL / JDBC package. This enables the client application to submit Preference SQL queries through familiar SQL clients. The Preference SQL middleware parses the query and performs preference query optimization. The final result will be computed in the execution engine by preference evaluation algorithms. Preference SQL works on well structured finite data but streams are endless and have diverse formats. Therefore, we have to preprocess the stream objects to get an attribute-based format like *attributeName* = *attributeValue* and to split the stream into finite chunks. This ETL process happens in Apache Flink[3] (see Figure 2). The transformed data can now be evaluated with Preference SQL, and the result is depicted at the bottom of the application. For more technical details we refer to [Ru16].

To illustrate the advantages of our approach we consider the following example: *the user wants to read tweets from Washington ("Washington (US)") about Donald Trump ("Trump") or US presidential election ("ElectionDay")*, cp. Figure 3.

```
SELECT STREAM *
FROM TwitterStream
 WHERE hashtags
 IN ('Trump', 'ElectionDay')
 AND place IN ('Washington(US)');
```

(a) SQL query.

```
SELECT STREAM *
FROM TwitterStream
 PREFERRING hashtags
 IN ('Trump', 'ElectionDay')
 PARETO place IN ('Washington(US)');
```

(b) PreferenceSQL query.

Fig. 3: Example query.

The evaluation of the query can provide different results depending on the evaluation mode. To satisfy the `WHERE` condition in the query both constraints must be fulfilled. Otherwise the *hard condition* mode provides an empty result set. In contrast, the *preference*-based query looks for best possible matches w.r.t. the query conditions. Therefore, results are also achieved when not all provided tweets correspond with the user's wish, but contain relevant and interesting information. Our demo application shows the advantage of preference-based stream analysis and that one gets personalized and interesting information instead of spam and information trash.

References

[KEW11] Kießling, W.; Endres, M.; Wenzel, F.: The Preference SQL System - An Overview. Bulletin of the Technical Commitee on Data Engineering, IEEE CS, 34(2):11–18, 2011.

[Ru16] Rudenko, L.; Endres, M.; Roocks, P.; Kießling, W.: A Preference-based Stream Analyzer. In: Workshop on Large-scale Learning from Data Streams in Evolving Environments, STREAMVOLV. 2016.

[ScZ05] Stonebraker, M.; Çetintemel, U.; Zdonik, S.: The 8 Requirements of Real-time Stream Processing. SIGMOD Rec., 34(4):42–47, December 2005.

[3] `https://flink.apache.org/`

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 185

Possible Voter Control in k-Approval and k-Veto Under Partial Information

Gábor Erdélyi[1] Christian Reger[2]

Abstract: We study the complexity of possible constructive/destructive control by adding voters (PCCAV/PDCAV) and deleting voters (PCCDV/PDCDV) under nine different models of partial information for k-Approval and k-Veto. For the two destructive variants, we can further settle a polynomial-time result holding even for each scoring rule. Generally, in voter control, an external agent (called *the chair*) tries to change the outcome of the election by adding new voters to the election or by deleting voters from the election. Usually there is full information in voting theory, i.e., the chair knows the candidates, each voter's complete ranking about the candidates and the voting rule used. In this paper, we assume the chair to have partial information about the votes and ask if the chair can add (delete) some votes so that his preferred (despised) candidate is (not) a winner for at least one completion of the partial votes to complete votes.

Keywords: computational social choice, voting, control, algorithms, complexity, partial information

1 Introduction

For a long time voting has been used to aggregate individual preferences and has applications in informatics, politics or economy (e.g., design of recommender systems [Gh99] or machine learning [Xi13]). In computer science applications we are often dealing with huge data volumes and hence the numbers of candidates and voters may be exorbitantly large. Thus it makes sense to study the computational complexity of (decision) problems related to voting. Such problems are *winner*, which asks if a given candidate c is a winner of an election or not, *manipulation* and *bribery*. In manipulation a group of strategic voters try to coordinate their votes to make a candidate c win, whereas in bribery an external agent, called the *briber*, alters some votes to reach his aim. The groundbreaking papers of Bartholdi et al. [BTT89a, BTT89b, BTT92] suggest that computational hardness provides a (worst case) barrier against manipulative attacks. Unfortunately various voting problems tend to be easy for frequently used voting rules like scoring rules [Li11, BTT89a, FHH09]. On the other hand, a manipulative agent traditionally has full information in voting theory, i.e., he knows the preference orders of each voter about all candidates. In many real-world settings, however, this assumption is not realistic. There is already a bunch of literature dealing with problems under uncertainty.

In this paper, we analyze the complexity for constructive control by adding (CCAV) and deleting voters (CCDV) in k-Approval and k-Veto under partial information, as these families of voting rules belong to the most prominent voting rules. Moreover, we study destructive control by adding (DCAV) and deleting voters (DCDV) under partial information. In

[1] University of Siegen, erdelyi@wiwi.uni-siegen.de
[2] University of Siegen, reger@wiwi.uni-siegen.de

CCAV (DCAV), the chair tries to add a limited number of new voters to an election to make a given candidate c win (prevent c from winning). In CCDV (DCDV), the chair deletes some voters from the election in order to make c (not) a winner. Natural examples are political elections where the voting age is lowered or raised. For CCAV, three natural generalizations of full information arise. We study if it makes a difference if the previous (*registered*) voters or the new (*unregistered*) voters are partial (or both). On the one hand, it seems natural that the chair has (due to surveys or former elections) full (or at least enough) knowledge about the registered voters, but only partial knowledge about the unregistered voters. On the other hand, the chair may personally know the unregistered voters (at least some of them) or has a belief about them, but has only little knowledge about the registered voters. We especially regard an optimistic variant of control under partial information, i.e., we ask if c can be made a winner for *at least one* way to complete the partial votes to complete rankings in the resulting election.

Related Work First of all, CCAV and CCDV under *full* information were introduced by Bartholdi et al. [BTT92]. Hemaspaandra et al. [HHR07] extended this research for destructive control. Control results especially for k-Approval and k-Veto were published in [Li11]. The first work about voting under *partial* information is by Konczak and Lang [KL05]. They introduced the possible/necessary winner problems and allowed votes to be partial orders instead of linear orders as before in literature. They studied under which conditions a candidate is a winner for *at least one* (*possible winner*) or for *all* (*necessary winner*) ways to complete the partial votes to full rankings. Our problem is related to possible winner in a sense that we consider nine different ways to reflect partial information and partial orders is one of them, and we ask if the chair can make his favorite candidate a winner under *at least one* completion of the partial votes. Possible winner has also been studied in [XC11, BD10, Ch12]. Finally we mention [CWX11] where a manipulator has partial information in form of a general information set (instead of partial orders) containing all possible profiles that can be achieved by completing the partial profile. Basically, their model is a generalization of all the models considered by us. (Necessary and possible) bribery and (necessary) voter control under these models have recently been studied in [BER16, ER16, Re16]. In these papers, one could observe that PC (aka partial orders) often yields hardness results where other (very general) partial models produce P results for the same problem. This work thus often refers to some previous works about partial information. One goal of us is to perform an extensive complexity study for different structures of partial information and this paper is one important part of this study. Another aim of this paper is to check if the possible winner variant of control often increases the complexity compared to control under full information. On the one hand, possible winner is only easy for Veto and Plurality, but hard for k-Approval and k-Veto ($k \geq 2$) given partial orders [XC11, BD10]. Thus at most these two voting rules can yield P results at all for PCCAV and PCCDV as possible winner is a special case of possible control. (Notice that this is not true for PCCAV with complete registered voters (see Section 4 for further information) although we can actually reduce some hardness results *indirectly* from possible winner.). On the other hand, we know from [BER16] that (necessary) bribery often makes the complexity jump from P to hardness by combining necessary winner and bribery.

Organization This paper is organized as follows. In Section 2, we provide some preliminaries and briefly introduce nine models of partial information before we define our problems studied in this paper in Section 3. Finally, we give our complexity results for possible control in Section 4. In Section 5, we provide results on possible destructive control under partial information. Section 6 gives a short conclusion.

2 Preliminaries

An *election* is defined as a pair $E = (C,V)$ where C is a finite *candidate set* (with $|C| = m$) and V a finite *voter set*. Usually, each voter v_i is given as a strict linear order (i.e., total, transitive and asymmetric) $\succ_{v_i}$ over C representing his preferences, i.e., there is full information. In this paper, however, votes are often given partially in form of some model which is specified later. An *n-voter profile* $P := (v_1,\ldots,v_n)$ on C consists of n voters $v_1,\ldots,v_n$ given as strict linear orders or (for a *partial profile*) partial in terms of a certain model of partial information. A *completion* or *extension* of a partial profile P is a complete profile P' not contradicting P (in other words, each vote is completed in a way that its partial structure is preserved.) We also borrow the notion *information set* from game theory and say that $P' \in I(P)$ where $I(P)$ is the information set of P containing all complete profiles not contradicting P. A *voting rule* (more precisely, a *voting correspondence*) $\mathscr{E}$ maps an election $E = (C,V)$ to a subset of C which is called the *winner set*. As we use the *non-unique winner model*, we allow a voting rule to have exactly one, more than one or no winner at all. In this paper, we restrict ourselves to *scoring rules* defined by a vector $\alpha := (\alpha_1,\ldots,\alpha_m)$ with non-negative and non-increasing entries. Each voter assigns α_1 points to his favorite candidate, α_2 points to his second most preferred candidate and so on. The candidate(s) with the highest score are the winner(s). Important special cases are k-Approval and k-Veto. In k-Approval each voter assigns one point to his k most favorite candidates and zero points to the remaining candidates, whereas in k-Veto each voter gives zero points to his k least favorite candidates and one point to the remaining ones. 1-Approval is also known as Plurality, we further say Veto instead of 1-Veto. Note that for a fixed number m of candidates, k-Approval equals $(m-k)$-Veto. In our analysis, however, only k is fixed, but m is variable.

We regard the following nine models of partial information which can be found in [BER16] and the references therein. Besides, we briefly point out their interrelations.

Gaps (GAPS) [BER16] and One Gap (1GAP) [Ba12]. For each vote v, there is a partition $C = C_1^v \dot\cup \ldots \dot\cup C_{2m+1}^v$, a total order for each C_k^v (k even) and no information at all for odd k. Note that possibly $C_k^v = \emptyset$ for some k. We further have $c_i \succ c_j \; \forall c_i \in C_i^v, c_j \in C_j^v$ with $i < j$. A special case is 1GAP, where in each vote some candidates are ranked at the top and at the bottom of the votes, and there is at most one gap. Formally, 1GAP is a special case of GAPS with $C_k^v = \emptyset$ for each $k \in \{1,5,6,\ldots,2m+1\}$ and each voter v.

Top-/Bottom-Truncated Orders (TTO,BTO) [Ba12]. TTO equals GAPS with $C_1^v = C_4^v = \ldots = C_{2m+1}^v = \emptyset$ for each voter v. BTO refers to the special case of GAPS with $C_3^v = \ldots = C_{2m+1}^v = \emptyset$ for each voter v.

Complete or empty votes (CEV) [KL05]. Each vote is either empty or complete.

Fixed Positions (FP) [BER16]. For each vote v we have a subset of candidates C^v with distinct positions in range between 1 and m assigned.

Pairwise Comparisons (PC) [KL05]. For PC (aka *partial orders*) – more or less the standard model – for each vote v there is a transitive and asymmetric subset $\Pi^v \subseteq C \times C$.

(Unique) Totally Ordered Subset of Candidates ((1)TOS) [BER16, KL05, Ch12]. In TOS, for each vote v, there is a complete ranking about a subset $C^v \subseteq C$. An important special case of TOS, 1TOS, requires that $C^v = C'$ for each voter $v \in V$.

Briskorn et al. gave a complete picture of the interrelations of these nine partial information models [BER16]. The interrelations can be visualized with a Hasse diagram, see Figure 1. For a detailed overview, motivation, and examples for these models, we refer to [BER16].

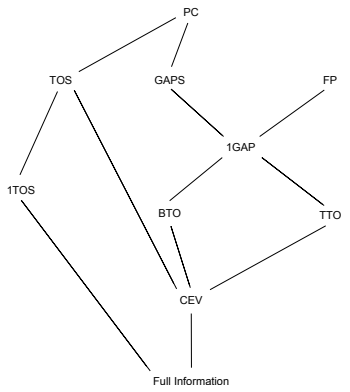


Fig. 1: Hasse diagram of the nine partial information models and full information.

3 Problem Settings

In the following, we let PIM := $\{\text{PC}, \text{GAPS}, \text{1GAP}, \text{FP}, \text{TOS}, \text{BTO}, \text{1TOS}, \text{CEV}, \text{TTO}\}$ be the set of all nine partial information models defined above and define $\overline{\text{PIM}} := \text{PIM} \cup \{\text{FI}\}$ where FI is the standard model of full information. The first problem defined in the following asks if the chair can add some new voters to the election such that c is a winner in the resulting election for at least one completion of the partial votes.

$\mathscr{E}$-(X,Y)-POSSIBLE CONSTRUCTIVE CONTROL BY ADDING VOTERS (PCCAV)	
Given:	An election $(C, V \cup W)$ with registered voters V according to model $X \in \overline{\text{PIM}}$, unregistered voters W according to model $Y \in \overline{\text{PIM}}$, a designated candidate $c \in C$, a non-negative integer $\ell \leq \lvert W \rvert$, and a partial profile P according to (X,Y)
Question:	Is it possible to choose a subset $W' \subseteq W$, $\lvert W' \rvert \leq \ell$ such that c is a winner of the election $(C, V \cup W')$ under $\mathscr{E}$ for at least one complete profile $P' \in I(P)$?

This way, one could combine all models of full/partial information. Our complexity analysis however concerns only the problems (FI,X), (X,FI) and (X,X) for $X \in$ PIM. (FI,FI) refers to both V and W representing full information which has been widely studied up to now and which is a special case of the other three problems. E.g., (FI,X)-PCCAV equals possible constructive control by adding voters where all registered votes are complete and all unregistered votes are partial according to X. The other problem studied asks if the chair can delete some votes such that c is a winner for at least one completion of the (remaining) partial profile, i.e., a possible winner for the residual partial election.

$\mathscr{E}$-X-POSSIBLE CONSTRUCTIVE CONTROL BY DELETING VOTERS (PCCDV)	
Given:	An election (C,V), a designated candidate $c \in C$, a non-negative integer $\ell \leq \|V\|$, and a partial profile P according to X.
Question:	Is it possible to choose a $V' \subseteq V$, $\|V \setminus V'\| \leq \ell$ such that c is a winner of the election (C,V') under $\mathscr{E}$ for at least one complete profile $P' \in I(P)$?

We obtain the destructive versions by replacing *a winner* by *not a winner*.

4 Results for Possible Constructive Control

In this section, we study the complexity of PCCAV and PCCDV. The question arises if there are problems for which possible winner and control are in P, but their hybridization is hard.

	FI	GAPS	FP	TOS	PC	CEV	1TOS	TTO	BTO	1GAP
Plurality	P	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
2-Approval	P	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
3-Approval	P	**P**	**P**		**NPc**	**P**	**P**	**P**	**P**	**P**
4-Approval	NPc	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*
Veto	P	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
2-Veto	P	**P**	**P**		**NPc**	**P**	**P**	**P**	**P**	**P**
3-Veto	NPc	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*

Tab. 1: (FI,X)-PCCAV

	FI	GAPS	FP	TOS	PC	CEV	1TOS	TTO	BTO	1GAP
Plurality	P	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
2-Approval	P	**P**	**P**		*NPc.*	**P**	**P**	**P**	**P**	**P**
3-Approval	P	**P**	**P**	*NPc*	*NPc*	**P**	*NPc*	**P**	**P**	**P**
4-Approval	NPc	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*
Veto	P	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
2-Veto	P	**P**	**P**		*NPc*	**P**	**P**	**P**	**P**	**P**
3-Veto	NPc	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*

Tab. 2: (X,FI)-PCCAV/ (X,X)-PCCAV

Theorem 4.1 *The complexity results are given as follows. Table 1 gives an overview over the results for* (FI,X)-PCCAV *(i.e., complete registered voters and unregistered voters partial according to model X ∈ PIM). The results both for* (X,FI)-PCCAV *and* (X,X)-PCCAV *(i.e., for partial V, complete W respectively V and W partial according to the*

	FI	GAPS	FP	TOS	PC	CEV	1TOS	TTO	BTO	1GAP
Plurality	P	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
2-Approval	P	**P**	**P**		*N*Pc	**P**	**P**	**P**	**P**	**P**
3-Approval	NPc	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*
Veto	P	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**	**P**
2-Veto	P	**P**	**P**		*N*Pc	**P**	**P**	**P**	**P**	**P**
3-Veto	P	**P**	**P**		*N*Pc	**P**	**P**	**P**	**P**	**P**
4-Veto	NPc	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*	*NPc*

Tab. 3: X-PCCDV

*same model $X \in PIM$) are summarized by Table 2. Table 3 displays the results for X-*PCCDV *for $X \in PIM$.*

We leave out all proofs due to space restrictions. In all tables of results, hardness entries in italic directly follow from hardness results for control under full information [Li11] or the respective possible winner problem. In particular, for the hardness result for 3-Approval-1TOS-POSSIBLE WINNER, we refer to [Ch12] (the TOS result follows immediately as TOS is a generalization of 1TOS). Moreover, we know that k-Approval-POSSIBLE WINNER ($k \geq 2$) [XC11] and k-Veto-POSSIBLE WINNER ($k \geq 2$) are hard [BD10]. Results in boldface are new. Column FI displays the results of control under full information following from [BTT92, Li11].

We point out that Plurality and Veto are the only voting rules yielding only easiness results. 2-Approval preserves polynomial-time decidability for PCCAV given complete registered voters and unregistered voters according to arbitrary partial model.

In contrast to (necessary) control in [Re16], PCCAV for partial registered voters and complete unregistered voters is never easier than for partial unregistered voters and complete registered voters. On the contrary, we have found two problems where the problem with complete registered voters and incomplete unregistered voters is even easier than with incomplete registered voters and complete unregistered voters. As an example, 3-Approval-(FI,1TOS)-PCCAV is in P and 3-Approval-(1TOS,FI)-PCCAV is NP-complete. We further mention 2-Approval-(FI,PC)-PCCAV which is in P whereas 2-Approval-(PC,FI)-PCCAV is hard. From [Re16], we even know that the corresponding problem (for necessary control) is in P for (PC,FI) and hard for (FI,PC).

Interestingly, possible winner for a given voting rule may be hard whereas the corresponding possible control problem is easy: As mentioned, 2-Approval-(FI,PC)-PCCAV is in P, but 2-Approval-PC-POSSIBLE WINNER is NP-complete [XC11]. Likewise, 3-Approval-(FI,1TOS)-PCCAV is in P, but 3-Approval-1TOS-POSSIBLE WINNER is NP-complete [Ch12]. The reason for this paradox is that only the problems X-PCCDV, (X,FI)-PCCAV and (X,X)-CCAV reduce to X-POSSIBLE WINNER, but not (FI,X)-CCAV.

We obtain only two hardness results with non-trivial reductions. Nevertheless, in both cases, hardness for possible winner almost directly implies hardness for PCCAV, i.e., for the proofs for PCCAV, we can embed the original hardness proofs for possible winner. This way 3-Approval-(FI,PC)-PCCAV is hard due to the possible winner hardness for 2(!)-

Approval [XC11], and 2-Veto-(FI,PC)-PCCAV is hard due to the hard possible winner problem for 2-Veto [BD10]. (On the contrary, given partial registered voters, CCAV in 2-Veto is hard under the PC model reducing directly from 2-Veto-POSSIBLE WINNER.)

We point out that six models (GAPS, FP, 1GAP, BTO, TTO and CEV) never increase the complexity of possible control whenever the corresponding standard control problem and possible winner are easy. Note that GAPS and FP are two general partial models and our initial expectation was that – similar to PC – they yield many hardness results.

5 Results for Possible Destructive Control

In contrast to constructive control, we achieve a global P result for all scoring rules for the destructive version both for PCCAV and PCCDV. The main idea is to check for any model if there is a non-distinguished candidate d beating the despised candidate c for at least one extension. Roughly speaking, we rank d as high as possible and c as low as possible in each vote and derive control problems under full information concerning these two candidates.

6 Conclusion

We studied the complexity of PCCAV and PCCDV in k-Approval and -Veto under nine different models of partial information. Surprisingly, there are only two hardness entries not directly reduced from possible winner or control under full information (although we can almost directly adopt the possible winner proofs). We further have that – in contrast to necessary control in [Re16] – CCAV with complete registered voters and partial unregistered voters is never harder than the other way around. For two instances, (X,FI) is really harder than (FI,X), namely for 2-Approval-PCCAV (PC) and 3-Approval-PCCAV (1TOS). Note also that particularly for 2-Approval-CCAV, (PC,FI) is easier than (FI,PC) for necessary control, but harder for PCCAV. For destructive control, we could settle a polynomial-time result holding for all scoring rules. Open questions for future research include the open problems in this paper or the extension of our study to other voting rules or control types.

References

[Ba12] Baumeister, D.; Faliszewski, P.; Lang, J.; Rothe, J.: Campaigns for Lazy Voters: Truncated Ballots. In: Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems. IFAAMAS, pp. 577–584, June 2012.

[BD10] Betzler, N.; Dorn, B.: Towards a Dichotomy for the Possible Winner Problem in Elections Based on Scoring Rules. Journal of Computer and System Sciences, 76(8):812–836, 2010.

[BER16] Briskorn, D.; Erdélyi, G.; Reger, C.: Bribery in k-Approval and k-Veto Under Partial Information (Extended Abstract). In: Proceedings of The 15th International Joint Conference on Autonomous Agents and Multiagent Systems. IFAAMAS, May 2016.

[BTT89a] Bartholdi, J.; Tovey, C.; Trick, M.: The Computational Difficulty of Manipulating an Election. Social Choice and Welfare, 6(3):227–241, 1989.

[BTT89b] Bartholdi, J.; Tovey, C.; Trick, M.: Voting schemes for which it can be difficult to tell who won the election. Social Choice and Welfare, 6(2):157–165, 1989.

[BTT92] Bartholdi, J.; Tovey, C.; Trick, M.: How Hard is it to Control an Election? Mathematical and Computer Modeling, 16(8/9):27–40, 1992.

[Ch12] Chevaleyre, Y.; Lang, J.; Maudet, N.; Monnot, J.; Xia, L.: New candidates welcome! Possible Winners with respect to the addition of new candidates. Mathematical Social Sciences, 64(1):74–88, 2012.

[CWX11] Conitzer, V.; Walsh, T.; Xia, L.: Dominating Manipulations in Voting with Partial Information. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence. AAAI Press, pp. 638–643, August 2011.

[ER16] Erdélyi, G.; Reger, C.: Possible Bribery in k-Approval and k-Veto Under Partial Information. In: Proceedings of The 17th International Conference on Artificial Intelligence: Methodology, Systems, Applications. Springer, September 2016.

[FHH09] Faliszewski, P.; Hemaspaandra, E.; Hemaspaandra, L. A.: How Hard is Bribery in Elections? Journal of Artificial Intelligence Research, 35:485–532, 2009.

[Gh99] Ghosh, S.; Mundhe, M.; Hernandez, K.; Sen, S.: Voting for Movies: The Anatomy of Recommender Systems. In: Proceedings of the 3rd Annual Conference on Autonomous Agents. ACM Press, pp. 434–435, May 1999.

[HHR07] Hemaspaandra, E.; Hemaspaandra, L.; Rothe, J.: Anyone but him: The complexity of precluding an alternative. 171(5-6), 2007.

[KL05] Konczak, K.; Lang, J.: Voting procedures with incomplete preferences. In: Proceedings of IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling. pp. 124–129, 2005.

[Li11] Lin, A.: The Complexity of Manipulating *k*-Approval Elections. In: Proceedings of the 3rd International Conference on Agents and Artificial Intelligence. pp. 212–218, 2011.

[Re16] Reger, C.: Voter Control In k-Approval And k-Veto Under Partial Information. In: Proceedings of the 14th International on Artificial Intelligence and Mathematics. January 2016.

[XC11] Xia, L.; Conitzer, V.: Determining Possible and Necessary Winners Given Partial Orders. Journal of Artificial Intelligence Research, 41:25–67, 2011.

[Xi13] Xia, L.: Designing Social Choice Mechanisms Using Machine Learning. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems. IFAAMAS, pp. 471–474, May 2013.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 193

Ranking Specific Sets of Objects

Jan Maly[1] Stefan Woltran[2]

Abstract: Ranking sets of objects based on an order between the single elements has been thoroughly studied in the literature. In particular, it has been shown that it is in general impossible to find a total ranking – jointly satisfying properties as dominance and independence – on the whole power set of objects. However, in many formalisms from the area of knowledge representation one does not need to order the entire power set, since certain sets are already ruled out due to hard constraints or are not satisfying some background theory. In this paper, we address the question whether an order on a given subset of the power set of elements satisfying different variants of dominance and independence can be found. We first show that this problem is tractable when we look for partial rankings, but becomes NP-complete for total rankings.

Keywords: Ranking Sets. Complexity.

1 Introduction

The problem of lifting rankings on objects to ranking on sets has been studied from many different view points – see [BBP04] for an excellent survey. Several properties (also called axioms) have been proposed in order to indicate whether the lifted ranking reflects the given order on the elements, among them dominance and independence. Roughly speaking, dominance ensures that adding an element which is better (worse) than all elements in a set, makes the augmented set better (worse) than the original one. Independence, on the other hand, states that adding an element a to sets A and B where A is already known to be preferred over B, must not make $B \cup \{a\}$ be preferred over $A \cup \{a\}$ (or, in the strict variant, $A \cup \{a\}$ should remain preferred over $B \cup \{a\}$). It is well known that constructing a ranking on the whole power set of objects which jointly satisfies such properties is in general not possible.

However, in many situations one does not need to order the entire power set. Just take preference-based formalisms from the area of knowledge representation. In fact, there exists a wide variety of approaches that proceed in the following way: in a first step, a set S of models that satisfy some hard constraints is obtained; then the most preferred models out of S are identified from an order on S that stems from some simpler principles, for instance, from an order of atomic elements. Such formalisms come in different veins. Qualitative Choice Logic [BBB04], for instance, adds a dedicated connective to standard propositional logic to express preferences between formulas; other formalisms like Answer-Set Optimization Problems [BNT03] or Qualitative Optimization Problems [FTW13] separate the specification of hard constraints and soft constraints. However, in both cases an implicit

[1] TU Wien, Institute of Information Systems, Austria
[2] TU Wien, Institute of Information Systems, Austria

order between models is constructed that is used to deliver the preferred models. The relation to the problem of ranking sets is evident. Atoms in the specification play the role of elements; models (containing elements that are set to true) of the theories can be seen as the sets to be ranked. Typically these formalisms allow a more general specification of the order than just ranking the atoms. However, the latter can be done in these formalisms as well. To the best of our knowledge, there are no investigations whether the implicit ordering of models satisfies principles like dominance or independence. More generally, the problem of lifting rankings to such specific sets of elements seems to be rather neglected, so far. The only exception we are aware of deals with subsets of a fixed cardinality [Bo95].

This indicates that is worth studying under which conditions rankings of elements can be lifted to arbitrary subsets of the power set of elements such that desirable properties hold. In order to do so, we first give a new definition for dominance which appears more suitable in such a setting. Then, we consider the following problem: Given a ranking on elements, and a set S of sets of elements, does there exist a strict (partial) order on S that satisfies D and I (where D is either standard dominance or our notion of dominance and I is independence or strict independence)? We show that the problem is either trivial or easy to solve for the case of strict partial orders. Our main result is NP-completeness for the case when strict orders are considered.

The remainder of the paper is organised as follows. In the next section, we recall some basic concepts. In Section 3 we discuss why standard dominance can be seen as too weak in our setting and propose an alternative definition. Section 4 contains our main results. Due to the space restriction, proofs are omitted.

2 Background

The formal framework we want to consider in the following consists of a finite[3], nonempty set X, equipped with a linear order $<$ and a subset $\mathscr{X} \subseteq \mathscr{P}(X) \setminus \{\emptyset\}$ of the power set of X not containing the empty set. We want to find an order $\prec$ on $\mathscr{X}$, that satisfies some niceness conditions. We will consider several kinds of orders. We recall the relevant definitions.

Definition 1. A binary relation is called a *strict partial order*, if it is irreflexive and transitive. A *strict order* is a total strict partial order. A binary relation is called a *preorder*, if it is reflexive and transitive. A *weak order* is a total preorder. If $\preceq$ is a weak order on a set X, for all $x, y \in X$, the *corresponding strict order* $\prec$ is defined by $x \prec y \leftrightarrow (x \preceq y \wedge y \not\preceq x)$.

Definition 2. Let $A \in \mathscr{X}$ be a set of elements of X. Then we write $\max(A)$ for the (unique) element of A satisfying $\max(A) > a$ for all $a \in A \setminus \{\max(A)\}$. Analogously, we write $\min(A)$ for the (unique) element of A such that $\min(A) < a$ holds for all $a \in A \setminus \{\min(A)\}$. Furthermore, we say a relation R on a set $\mathscr{X}$ *extends* a relation S on $\mathscr{X}$ if xSy implies xRy for all $x, y \in \mathscr{X}$. Finally, we say a relation R on $\mathscr{X}$ is the *transitive closure* of a relation S

[3] In the literature, infinite sets of alternatives are also considered. The results presented in the background section hold also for this case. For the results on computational complexity in the main part, finiteness of the (infinitely many) instances is obviously crucial.

on $\mathscr{X}$ if the existence of a sequence $x_1 S x_2 S \ldots S x_k$ implies $x_1 R x_k$ for all $x_1, x_k \in \mathscr{X}$ and R is the smallest relation with this property. We write $trcl(S)$ for the transitive closure of S.

Many different axioms a good order should satisfy are discussed in the literature (an overview over the relevant interpretations and the corresponding axioms can be found in the survey [BBP04]). The following axioms "have very plausible intuitive interpretations" [BBP04, p.11] for decision making under complete uncertainty and belong to the most extensively studied axioms in the literature. We added conditions of the form $X \in \mathscr{X}$ that are not necessary if $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$ holds.

Axiom 1 (Extension Rule)**.** For all $x, y \in X$ if $\{x\}, \{y\} \in \mathscr{X}$ then,

$$\{x\} \prec \{y\} \text{ iff } x < y$$

Axiom 2 (Dominance)**.** For all $A \in \mathscr{X}$ and all $x \in X$ if $A \cup \{x\} \in \mathscr{X}$ then,

$$y < x \text{ for all } y \in A \text{ implies } A \prec A \cup \{x\}$$

$$x < y \text{ for all } y \in A \text{ implies } A \cup \{x\} \prec A$$

Axiom 3 (Independence)**.** For all $A, B \in \mathscr{X}$ and for all $x \in X \setminus (A \cup B)$, if $A \cup \{x\}, B \cup \{x\} \in \mathscr{X}$ then

$$A \prec B \text{ implies } A \cup \{x\} \preceq B \cup \{x\}$$

Axiom 4 (Strict Independence)**.** For all $A, B \in \mathscr{X}$ and for all $x \in X \setminus (A \cup B)$ if $A \cup \{x\}, B \cup \{x\} \in \mathscr{X}$ then

$$A \prec B \text{ implies } A \cup \{x\} \prec B \cup \{x\}$$

Every reasonable order should satisfy the extension rule. However, in the case $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$, the extension rule is implied by dominance [BBP04]. A natural task is to find an order on $\mathscr{P}(X) \setminus \{\emptyset\}$ that satisfies dominance together with (some version of) independence. However, in their seminal paper [KP84], Kannai and Peleg have shown that this is impossible for regular independence and dominance if $|X| \geq 6$ and $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$ hold. Barberà and Pattanaik [BP84] showed that for strict independence and dominance this is impossible even for $|X| \geq 3$ and $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$.

If we abandon the condition $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$, the situation is not as clear. There are, obviously, sets $\mathscr{X}$ such that there is an order on $\mathscr{X}$ satisfying strict independence and dominance – for example, $\mathscr{X} = \{A\}$ for some set $A \in \mathscr{P}(X) \setminus \{\emptyset\}$. Every order on $\mathscr{X}$ satisfies all axioms proposed in this section by trivial means.

3 Setting the Stage

Many useful results regularly used in the setting of $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$ are not true in the more general case. For example, in contrast to the result stated above, the extension rule is, in general, not implied by dominance.

Example 3. Consider the example $X = \{x_1, x_2\}$ with $x_1 < x_2$ and $\mathscr{X} = \{\{x_1\}, \{x_2\}\}$ with $\{x_2\} \prec \{x_1\}$. The extension rule is clearly violated in this example, however dominance is vacuously true as no set with two elements exists.

It could be argued that dominance should imply the extension rule. However the regular formulation of dominance is not strong enough in our setting because the intermediate set $\{x_1, x_2\}$ is missing from $\mathscr{X}$. In general, some natural consequences of dominance in the case of $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$ rely on the availability of intermediate sets. Therefore, it is reasonable to ask for stronger versions of dominance that behave nicely in the general case. We observe that $x < y$ for all $y \in A$ implies obviously $\max(A \cup \{x\}) = \max(A)$ and $\min(A \cup \{x\}) < \min(A)$; whereas $x > y$ for all $y \in A$ implies $\max(A \cup \{x\}) < \max(A)$ and $\min(A \cup \{x\}) = \min(A)$. We can use this property to define another notion of dominance.

Axiom 5 (Maximal Dominance)**.** For all $A, B \in \mathscr{X}$,

$$(\max(A) \leq \max(B) \wedge \min(A) < \min(B)) \vee$$
$$(\max(A) < \max(B) \wedge \min(A) \leq \min(B)) \rightarrow A \prec B$$

This axiom trivially implies the extension rule. Additionally, if $\mathscr{X}$ is sufficiently large, for example $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$, dominance and independence imply maximal dominance. It would be possible to define several other versions of dominance of intermediate strength. In this work we will make do with dominance and maximal dominance. We will see that the results justify this approach.

Furthermore, we also want to look at the case where additionally the requirement that $\prec$ has to be total is dropped. If we consider (maximal) dominance and strict independence we can look at partial orders to treat this problem. However, dominance talks about strict preferences while independence produces weak preferences. Therefore, neither a partial order, nor a preorder can satisfy dominance and independence for syntactical reasons. The natural solution, using a preorder and defining the corresponding partial order by $a < b$ iff $a \leq b$ but not $a \geq b$ holds, is not possible because we want to define orders recursively. Especially, we want to add weak and strict preferences independently. To solve this problem, we define a new type of order (or, more accurately, a pair of orders). This definition is supposed to emulate the interplay between an order and its corresponding strict order in partial and preorders.

Definition 4. A pair of relations $(\leq, <)$ on a set $\mathscr{X}$ is an *incomplete order*, if (1) $\leq$ is a preorder; (2) $<$ is a partial order; and (3) $x < y \leq z$ implies $x < z$ and $x \leq y < z$ implies $x < z$ for all $x, y, z \in \mathscr{X}$.

Every incomplete order is part of a order and its corresponding strict order.

Proposition 5. *For every incomplete order $(\leq_i, <_i)$, there is an order $\leq$ with a corresponding strict order $<$ such that $\leq$ extends $\leq_i$ and $<$ extends $<_i$.*

	Dom + Ind	Max Dom +Ind	Dom + Strict Ind	Max Dom + Strict Ind
Not total	always	always	in P	in P
Total	NP-c.	NP-c.	NP-c.	NP-c.

Tab. 1: Main results

4 Main Results

We studied in total 8 problems as defined below.[4] Our results are summarized in Table 1.

The Partial (Max)-Dominance-Strict-Independence Problem. *Given a linearly ordered set X and a set $\mathscr{X} \subseteq \mathscr{P}(X) \setminus \{\emptyset\}$, decide if there is a partial order on $\mathscr{X}$ satisfying (maximal) dominance and strict independence.*

The Partial (Max)-Dominance-Independence Problem. *Given a linearly ordered set X and a set $\mathscr{X} \subseteq \mathscr{P}(X) \setminus \{\emptyset\}$, decide if there is an incomplete order on $\mathscr{X}$ satisfying (maximal) dominance and independence.*

The (Max)-Dominance-(Strict)-Independence Problem. *Given a finite linearly ordered set X and a set $\mathscr{X} \subseteq \mathscr{P}(X) \setminus \{\emptyset\}$, is there a (strict) total order on $\mathscr{X}$ satisfying (maximal) dominance and (strict) independence.*

4.1 Partial Orders

First we want to classify sets that allow for a partial order satisfying dominance and strict independence. To do so, we build a minimal transitive relation satisfying dominance and strict independence. First, we build a minimal transitive relation satisfying dominance. It is worth noting that a very similar relation can be defined for maximal dominance. Using this relation, all results in this section can be proven for maximal dominance the same way.

Definition 6. The relation $\prec_d$ is defined as follows. For all $A, A \cup \{x\} \in \mathscr{X}$, (1) $A \prec_d A \cup \{x\}$ if $y < x$ for all $y \in A$; and (2) $A \cup \{x\} \prec_d A$ if $x < y$ for all $y \in A$. The relation $\prec_d^t$ is defined as $\prec_d^t := \mathrm{trcl}(\prec_d)$.

It is easy to see that $\prec_d^t$ is a partial order and that a partial order on a set $\mathscr{X}$ satisfies dominance if and only if it extends $\prec_d^t$. We want to extend this relation to a minimal relation for strict independence and dominance.

Definition 7. We build a relation $\prec_\infty$ by induction. We start with $\prec_0^t := \prec_d$. For $\prec_{n+1}$ we select sets $A, B, A \setminus \{x\}, B \setminus \{x\} \in \mathscr{X}$ with $x \in X$ and $A \setminus \{x\} \prec_n^t B \setminus \{x\}$ and set $A \prec_{n+1} B$. Then we define $\prec_{n+1}^t := trcl(\prec_{n+1})$ and $\prec_\infty = \bigcup_n \prec_n^t$.

[4] Definitions that only differ in one or two words are combined into one definition using brackets.

This relation satisfies dominance, strict independence and transitivity by construction. However, in general, $\prec_\infty$ is not irreflexive. If $\prec_\infty$ is not irreflexive no strict partial order can extend it. This rules out the existence of a partial order satisfying dominance and strict independence, as every strict partial order on $\mathscr{X}$ satisfying dominance and strict independence is an extension of $\prec_\infty$. Hence, if we want to decide if a set allows a partial order that satisfies strict independence and dominance we only have to check if $\prec_\infty$ is irreflexive on $\mathscr{X}$. This can clearly be done in polynomial time.

Theorem 8. *The partial Dominance-Strict-Independence problem is in P.*

Additionally, we can characterize the sets allowing such partial orders in a more combinatorial fashion using the following definition of links. We then show that links indeed characterize $\prec_\infty$.

Definition 9. A $\prec_\infty$-*link* from A to B in $\mathscr{X}$ is a sequence $A =: C_0, C_1, \ldots, C_n := B$ with $C_i \in \mathscr{X}$ for all $i \leq n$ such that, for all $i < n$, either $C_i \prec_d C_{i+1}$ holds or there is a link between $C_i \setminus \{x\}$ and $C_{i+1} \setminus \{x\}$ for some $x \in X$.

Lemma 10. *For $A, B \in \mathscr{X}$, $A \prec_\infty B$ holds if and only if there is a $\prec_\infty$-link from A to B.*

This gives us an easy characterization of sets $\mathscr{X}$ with irreflexive $\prec_\infty$

Corollary 11. *$\prec_\infty$ is irreflexive if and only if there is no set $A \in \mathscr{X}$ such that there is a $\prec_\infty$-link from A to A.*

We would like to produce a similar result for dominance and (non-strict) independence. As already discussed, we have to build an incomplete instead of partial order.

Definition 12. We build a pair of relations $(\preceq^\infty, \prec^\infty)$ by induction. We start with $\prec_t^0 := \prec_d$ and $A \preceq_t^0 A$ for all A. For $\preceq^{n+1}$ we select sets $A, B, A \setminus \{x\}, B \setminus \{x\} \in \mathscr{X}$ with $x \in X$ and $A \setminus \{x\} \prec_t^n B \setminus \{x\}$ and set $A \preceq^{n+1} B$. Then we set

$$\prec_t^{n+1} := trcl(\prec^{n+1}) \cup \left\{(A,B) \mid \exists C (A \prec^{n+1} C \preceq^{n+1} B \vee A \preceq^{n+1} C \prec^{n+1} B)\right\}$$

and $\preceq_t^{n+1} = trcl(\preceq_t^{n+1})$. Finally we set $\preceq^\infty = \bigcup_n \preceq_t^n$ and $\prec^\infty = \bigcup_n \prec_t^n$.

It is clear that this pair satisfies dominance and independence. Furthermore, it is obvious that $\prec_\infty$ and $\preceq_\infty$ are transitive, that $\preceq$ is reflexive and that the pair satisfies condition (3) of an incomplete order. Furthermore, it is easy to see that $A \preceq^\infty B$ implies $\max(A) \leq \max(B)$ and $\min(A) \leq \min(B)$ possibly without strict preference and $A \prec^\infty B$ implies $\max(A) \leq \max(B)$ and $\min(A) \leq \min(B)$ with at least one strict preference. This property clearly implies the irreflexivity of $\prec^\infty$. Therefore we have an incomplete order satisfying dominance and independence on all sets.

Theorem 13. $(\preceq_\infty, \prec_\infty)$ *is an incomplete order satisfying dominance and independence on all sets $\mathscr{X} \subseteq \mathscr{P}(X) \setminus \{\emptyset\}$.*

This includes the case $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$. Here, we even get maximal dominance for free.

Proposition 14. *Let $\mathscr{X} = \mathscr{P}(X) \setminus \{\emptyset\}$. Then $(\preceq^\infty, \prec^\infty)$ is the minimal incomplete order satisfying maximal dominance and independence.*

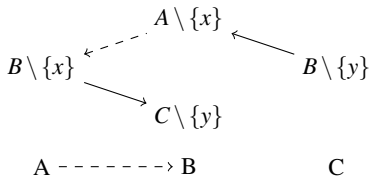


Fig. 1: Family that forces that $A \prec B$ leads to $B \prec C$

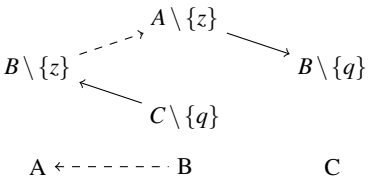


Fig. 2: Family that forces that $A \succ B$ leads to $B \succ C$

Fig. 3: Sketch of the sets V_1, V_2 and V_n

4.2 Total Orders

We show that it is, in general, not possible to construct a (strict) total order satisfying (maximal) dominance and (strict) independence deterministically in polynomial time. We do this by a reduction from betweenness.

Problem (Betweenness)**.** Given a finite set $V = \{v_1, v_2, \ldots, v_n\}$ and a set of triples $R \subseteq V^3$, find a strict total order on V such that $a < b < c$ or $a > b > c$ holds for all $(a,b,c) \in R$.

Betweenness is known to be NP-hard [Op79]. We use this result to show NP-hardness for all four versions of the of the (Max)-Dominance-(Strict)-Independence problem. We sketch the reduction for the maximal dominance and strict independence case.

We represent a triple (a,b,c) in a Max-Dominance-Strict-Independence instance with sets A,B,C and force $A \prec B \prec C$ or $A \succ B \succ C$ by adding two families of sets $A \setminus \{x\}, B \setminus \{x\}, B \setminus \{y\}, C \setminus \{y\}$ and $A \setminus \{z\}, B \setminus \{z\}, B \setminus \{q\}, C \setminus \{q\}$ where q,x,y,z are elements of X. Furthermore, we add sets that enforce $B \setminus \{x\} \prec C \setminus \{y\}$, $B \setminus \{y\} \prec A \setminus \{x\}$, $C \setminus \{q\} \prec B \setminus \{z\}$ and $A \setminus \{z\} \prec B \setminus \{q\}$ by dominance and strict independence. Figure 1 shows the case that $A \prec B$ holds for a strict total order $\prec$ on $\mathscr{X}$ satisfying maximal dominance and strict independence. We show that $B \prec C$ has to hold in this case. $A \setminus \{x\} \succ B \setminus \{x\}$ would contradict strict independence, therefore $A \setminus \{x\} \prec B \setminus \{x\}$ has to hold. Hence, $B \setminus \{y\} \prec C \setminus \{y\}$ has to hold because otherwise, $A \setminus \{x\} \prec B \setminus \{x\} \prec C \setminus \{y\} \prec B \setminus \{y\} \prec A \setminus \{x\}$ would be a circle contradicting the irreflexity of $\prec$. This implies $B \prec C$ by strict independence. Figure 2 shows, that $A \succ B$ leads to $B \succ C$ analogously.

We code the elements $v_1, \ldots, v_n$ of V for a betweenness instance (V,R) by sets $V_1, V_2, \ldots, V_n$ such that such that all sets have the same maximal and minimal element and the second largest elements are decreasing and second smallest elements are increasing (see Figure 3). Then we can enforce $V_i \prec V_j$ by adding $V_i \setminus \{\max(V_i)\}$ and $V_j \setminus \{\max(V_j)\}$ if $i < j$ holds or $V_i \setminus \{\min(V_i)\}$ and $V_j \setminus \{\min(V_j)\}$ if $i > j$ holds.

The actual construction goes like this: Let (V,R) be an instance of betweenness with $V = \{v_1,\dots,v_n\}$. We construct an instance of the Max-Dominance-Strict-Independence problem. We set $X = \{1,2,\dots,N\}$ equipped with the usual linear order, where $N = 8n^3 + 2n$. We construct $\mathscr{X}$ according to the idea discussed above. $\mathscr{X}$ contains sets representing $v_1,\dots,v_n$ of the following form:

$$V_i := \{1,N\} \cup \{i+1, i+2, \dots, N-i\}$$

Furthermore, for every triple we add the following sets: Pick a triple $(v_i,v_j,v_k) \in R$ and set $k = n+1+8i$ for the i-th triple. Let $(A,B,C) = (V_i,V_j,V_k)$ be the triple of sets coding the triple of elements (v_i,v_j,v_k). We add the following sets:

$$A \setminus \{k\}, B \setminus \{k\}, B \setminus \{k+1\}, C \setminus \{k+1\}, A \setminus \{k+2\}, B \setminus \{k+2\}, B \setminus \{k+3\}, C \setminus \{k+3\}$$

These sets correspond to the sets $A \setminus \{x\}, B \setminus \{x\}, \dots, C \setminus \{q\}$ in Figure 1 and Figure 2. Observe that the inductive construction guarantees that every constructed set is unique. We now have to force the preferences $B \setminus \{k+1\} \prec A \setminus \{k\}$, $B \setminus \{k\} \prec C \setminus \{k+1\}$, $A \setminus \{k+2\} \prec B \setminus \{k+3\}$ and $C \setminus \{k+3\} \prec B \setminus \{k+2\}$.

Firstly, for technical reasons[5], we add sets $A \setminus \{k,k+4\}, B \setminus \{k+1,k+4\}$. Then, observe that, by construction, either $A \setminus \{1,k,k+4\} \succ B \setminus \{1,k+1,k+4\}$ or $A \setminus \{k,k+4,N\} \succ B \setminus \{k+1,k+4,N\}$ is implied by maximal dominance. We add $A \setminus \{k,k+4,1\}$ and $B \setminus \{k+1,k+4,1\}$ in the first case and $A \setminus \{k,k+4,N\}$ and $B \setminus \{k+1,k+4,N\}$ in the second case. This ensures $B \setminus \{k+1\} \prec A \setminus \{k\}$ by strict independence. In the same way, we can force the other preferences using $k+5, k+6$ and $k+7$ instead of $k+4$.

Finally, we pick a new triple $(v'_i,v'_j,v'_k) \in R$ until we treated all triples in R. Observe that there are at most n^3 triple, thus, for every triple, the values $k,\dots,k+7$ lie between $n+1$ and $N-n$, hence are element of every V_i. In total, we add 24 sets per triple. Therefore, $\mathscr{X}$ contains $n+24n^3$ sets. It can be checked that (V,R) is a positive instance of betweenness if and only if $\mathscr{X}$ allows for a total strict order that satisfies maximal dominance and strict independence.

Theorem 15. *The Max-Dominance-Strict-Independence problem, the Max-Dominance-Independence problem, the Dominance-Strict-Independence problem and the Dominance-independence problem are NP-complete.*

5 Conclusion

We have shown that the problem of deciding whether a linear order can be lifted to a ranking of sets of objects satisfying a form of dominance axiom and a form of independence is in P or trivial if we do not expect the ranking to be total and NP-complete if we do. In order to prove P-membership or triviality we constructed rankings. These rankings could be used in applications e.g to eliminate obviously inferior sets of objects from a set of options. On the

[5] This makes sure that we don't accidentally force a preference between $A \setminus \{k\}$ and $B \setminus \{k\}$.

other hand, one could argue that the NP-hardness result shows that, in general, an ordering on objects does not give sufficient information to decide preferences between some sets of objects and therefore we are forced to guess preferences.

Future work concerns the complexity of the studied problem if the sets are given in a compact way, for instance in terms of a formula such that the models characterize the sets to be ranked. Another item on our agenda is to investigate whether the proposed logic in [GE11] can be used for specific sets of objects as well.

Acknowledgments. This research has been supported by the Austrian Science Fund (FWF) through projects P25207, I2854 and Y698. The authors want to thank Ulle Endriss for his helpful remarks on an earlier version of this work.

References

[BBB04] Brewka, Gerhard; Benferhat, Salem; Berre, Daniel Le: Qualitative choice logic. Artif. Intell., 157(1-2):203–237, 2004.

[BBP04] Barberà, Salvador; Bossert, Walter; Pattanaik, Prasanta K: Ranking sets of objects. In: Handbook of utility theory, pp. 893–977. Springer, 2004.

[BNT03] Brewka, Gerhard; Niemelä, Ilkka; Truszczynski, Miroslaw: Answer Set Optimization. In (Gottlob, Georg; Walsh, Toby, eds): Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence. Morgan Kaufmann, pp. 867–872, 2003.

[Bo95] Bossert, Walter: Preference extension rules for ranking sets of alternatives with a fixed cardinality. Theory and decision, 39(3):301–317, 1995.

[BP84] Barberà, Salvador; Pattanaik, Prasanta K: Extending an order on a set to the power set: some remarks on Kannai and Peleg's approach. Journal of Economic Theory, 32(1):185–191, 1984.

[FTW13] Faber, Wolfgang; Truszczynski, Miroslaw; Woltran, Stefan: Strong Equivalence of Qualitative Optimization Problems. J. Artif. Intell. Res. (JAIR), 47:351–391, 2013.

[GE11] Geist, Christian; Endriss, Ulrich: Automated Search for Impossibility Theorems in Social Choice Theory: Ranking Sets of Objects. J. Artif. Intell. Res. (JAIR), 40:143–174, 2011.

[KP84] Kannai, Yakar; Peleg, Bezalel: A note on the extension of an order on a set to the power set. Journal of Economic Theory, 32(1):172–175, 1984.

[Op79] Opatrny, Jaroslav: Total ordering problem. SIAM Journal on Computing, 8(1):111–114, 1979.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 203

Towards Complex User Feedback and Presentation Context in Recommender Systems

Ladislav Peska[1], Peter Vojtas[1]

Abstract: In this paper, we present our work in progress towards employing complex user feedback and its context in recommender systems. Our work is generally focused on small or medium-sized e-commerce portals. Due to the nature of such enterprises, explicit feedback is unavailable, but implicit feedback can be collected in both large amount and rich variety. However, some perceived values of implicit feedback may depend on the context of the page or user's device (further denoted as *presentation context*). In this paper, we present an extended model of presentation context, propose methods integrating it into the set of implicit feedback features and evaluate these on the dataset of real e-commerce users. The evaluation corroborated the importance of leveraging presentation context in recommender systems.

Keywords: Recommender systems, implicit feedback, presentation context, user preference, e-commerce.

1 Introduction

Recommender systems belongs to the class of automated content-processing tools, aiming to provide users with unknown, surprising, yet relevant objects without the necessity of explicitly query for them. The core of recommender systems are machine learning algorithms applied on the matrix of user to object preferences. The user preference is usually derived from explicit user rating (also referred as *explicit feedback*), but this is not possible e.g. for small or medium-sized e-commerce enterprises, where user ratings are extremely scarce. Instead, one can focus on collecting specific features of user behavior (*implicit feedback*) and estimate user preference from it [Cl01], [Pe14], [Pe16], [Yi14].

Our approach is to focus on a complex model of the implicit feedback and learn user preference via some machine learning method. Our working hypothesis is that by collecting more informative description of user behavior, we shall be able to better estimate user's preferences and thus provide him/her with better recommendations. Similar approach are not so common. We can mention e.g. Yang et al. [Ya12], or Claypool et al. [Cl01]. However, in both cases the domain and thus the set of collected feedback significantly differs from our approach.

Further, the key part of our approach is to relate implicit feedback features to the relevant context. We consider several *presentation context* features (e.g. related to page

[1] Charles University in Prague, Faculty of Mathematics and Physics, peska@ksi.mff.cuni.cz

complexity or user's device capability). Closest to our work is the approach by Yi et al. [Yi14], proposing several presentation context features (e.g. content type or article length) to be used while estimating user preference from *dwell time*. We differ from this approaches in both the usage of extended set of context (e.g. features related to the browser visible area), the method of incorporating *presentation context* and considered implicit feedback features.
This work follows to our previous paper [Pe16]. We extend it by presenting improved method for context incorporation and improved recommending algorithms.

2 Materials and Methods

In traditional recommender systems, user u rates some small sample $\boldsymbol{S}$ of all objects $\boldsymbol{O}$, which is commonly referred as user preference $r_{u,o}$: $o \in \boldsymbol{S} \subset \boldsymbol{O}$. The task of traditional recommender systems is to build suitable user model, capable to predict ratings $\hat{r}_{u,o'}$ of all objects $o' \in \boldsymbol{O}$. In domains without explicit feedback, user preference $\bar{r}_{u,o}$ can be learned from the set of implicit feedback $L(f_1, \dots, f_i) \rightarrow \bar{r}_{u,o}$. Then, traditional recommending algorithms can be used. There are in principle three possible approaches to construct the preference learning function L:

- We can suppose that the higher value of each feedback feature implies the higher user preference. Based on this hypothesis, the estimated rating $\bar{r}_{u,o}$ can be defined as the average of all feedback values. However, as the feedback features distribution varies greatly, the feedback values must be normalized in order to be comparable. In this study, we used standardization of features (denoted as *STD* in evaluation), and their empirical cumulative distribution (denoted as *CDF*). The estimated rating $\bar{r}_{u,o}$ can be then defined as the mean of *STD* or *CDF* values of all feedback features for the respective user and object.

$$\bar{r}_{u,o} = \sum_{l=1}^{i} CDF(f_{l,u,o}) \,/ i$$

- Another option is to consider some feedback feature as a golden standard $f_{k,u,o} \approx \bar{r}_{u,o}$. The obvious candidate in e-commerce are *purchases*. However, as they are quite sparse[2], we can hypothesize that also other visited objects were preferred to some extent. One way to derive such preference is to employ supervised machine learning aiming to learn the probability that the object was purchased, based on the other feedback feature values (*J48* decision tree was used in this study).
- A baseline option is to use binary *visits* (i.e. suppose that users equally prefer all visited objects). Such approach is also quite common in the literature (e.g. [Os13], [Re09]).

[2] Less than 0.4% of the visited objects were purchased in our dataset.

The implicit feedback features used in this paper are *view count*, *dwell time*, *travelled mouse cursor distance*, *cursor in-motion time*, *scrolled distance*, *time of scrolling*, *clicks count* and *purchases*. More details can be found in [Pe14] and [Pe16].

However, the perceived values of implicit feedback features might be highly biased by the way, how the object is presented to the user. Suppose for example that the content of a webpage fully fits into the browser visible area. Then no scrolling is necessary and thus we receive zero values of *scrolled distance* and *time of scrolling*. Similarly, if the page contains mostly text (e.g. news or tour domains), then the time spend by reading the page is mostly determined by the length of the text itself.

There can be more such factors related to the features of the object itself, or its presentation, which, altogether, can be denoted as the *presentation context*. We propose several presentation context variables, which should be generally observable on the webpages. These are *volume of text*, *links* and *images*, *page* and *browser dimensions*, *page visible area ratio* and *hand-held device* indicator. Furthermore, we propose two approaches to incorporate *presentation context* into the process of user rating $\bar{r}_{u,o}$ estimation.

- Extend the dataset of implicit feedback features by the presentation context features (denoted as *FB+C* in the evaluation). This approach leaves the context incorporation on the preference learning method.
- Use presentation context as a *baseline* value *predictor* and subtract these from the perceived feedback values. We can either derive an average baseline predictor based on all contextual features (*AVGBP*), or create a separate baseline predictor of each presentation context feature and use the Cartesian product of implicit feedback features and baseline predictors in the preference learning step (*CBP*).
- We further evaluate two baseline approaches: usage of all feedback features disregarding of any context features (*FB*) and usage of binary feedback based on visited objects (*Binary*).

After the computation of $\bar{r}_{u,o}$ ratings, these are supplied to the recommender system, which computes the final list of recommended objects. In this study, the *Vector Space Model* (*VSM*) algorithm was adopted [LGS11], with binarized content-based attributes serving as a document vector and cosine similarity over TF-IDF weights as objects' similarity measure. We further enhance the *VSM* algorithm by multiplying its results by the general objects popularity (in terms of total $\bar{r}_{u,o}$) and thus prioritize more popular objects (*popVSM*).

To sum-up, the presented approach works in three steps. In the first step, the implicit feedback and contextual features are collected and combined into a set of feedback features $\{f_1, \dots, f_i\}$. Those features are used in the second step – user preference learning, where either machine learning algorithms or simple estimators are employed to derive estimated ratings $\bar{r}_{u,o}$. The estimated ratings are forwarded to the recommending

algorithm, which proposes the final list of top-k recommended objects to the user.

3 Evaluation and Discussion

We evaluated the proposed approach on a Czech travel agency dataset [Pe16], aiming to predict objects purchased by the user. We adopted the leave-one-out cross-validation applied on purchases and considered the problem as ranking (i.e. the object purchased by the user should appear on top of the recommended objects). Due to the evaluation protocol, the dataset was restricted only to users with purchased objects and more than one visited object, resulting into 405 purchases from 253 users. Table 1 contains the results w.r.t. the normalized discounted cumulative gain (nDCG).

Table 1. Evaluation results in terms of average nDCG. Baseline methods are depicted in grey italics, the best result in bold. Results outperformed by the best result according to the binomial significance test [Sa97] w.r.t recall@top-10 are marked with * ($p < 0.05$) or ** ($p<0.01$).

Processing method	**Feedback and Context composition**				
	Binary	***FB***	**FB+C**	**AVGBP**	**CBP**
STD + popVSM	*0.255**	*0.174***	0.197**	0.161**	0.158**
CDF + popVSM	*0.255**	*0.257**	0.253*	0.258*	0.257
J48 + popVSM	*0.255**	*0.256**	**0.274**	0.240**	0.247**
J48 + objects popularity	*0.180***	*0.205***	0.211**	0.168**	0.186**
J48 + VSM	*0.222**	*0.224**	0.233	0.225*	0.224*

3.1 Discussion and Conclusions

In this work in progress report, we presented our approach towards employing complex user feedback and its context in recommender systems. Our approach works in three steps: feedback and context collection and combination, preference learning and recommendation. The evaluation shown capability of such model to improve over both binary feedback baseline and usage of complex feedback without contextual information. Adding contextual features as further feedback features before the preference learning step (*FB+C*) generally achieves the best results, combining it with *J48*-based preference learning and *popVSM* recommender provided the overall best result so far. Using J48 preference learning outperformed both *STD* and *CDF* heuristics. While the results of *STD*-based methods was clearly inferior, the *CDF* results were quite close to the *J48* ones, so we can recommend such approach in situations, where purchases or similar feedback features are not available. Using *popVSM* recommender significantly outperformed individual usage of both of its components (*VSM* and *object popularity*).

There is substantial amount future of work to be done in both algorithm design and its evaluation. In particular, we would like to focus more on feedback and context incorporation, e.g., proposing some context-based feature weighting. The results should

be further validated by using additional preference learning methods, recommending algorithms and evaluation datasets. Also, there is a space for improvement of evaluation scenario itself, e.g., evaluate recommendations for new sessions. Successful candidates from the offline evaluation should be also validated via on-line A/B testing.

Acknowledgments. The work on this paper was supported by the Czech grant P46. Supplementary materials can be obtained from: http://bit.ly/2g79VVO.

References

[Cl01] Claypool, M.; Le, P.; Wased, M.; Brown, D.: Implicit interest indicators. In: IUI '01. ACM, pp. 33–40, 2001.

[LGS11] Lops, P.; de Gemmis, M.; Semeraro, G.: Content-based Recommender Systems: State of the Art and Trends. In: Recommender Systems Handbook. Springer, 2011, pp. 73–105.

[Os13] Ostuni, V. C.; Noia, T. D.; Sciascio, E. D.; Mirizzi, R.: Top-N recommendations from implicit feedback leveraging linked open data. In: RecSys 2013. ACM, pp. 85–92, 2013.

[Pe14] Peska, L.: IPIget – The Component for Collecting Implicit User Preference Indicators. In: ITAT 2014, Ustav informatiky AV CR. Pp. 22–26, 2014, url: http://itat.ics.upjs.sk/workshops.pdf.

[Pe16] Peska, L.: Using the Context of User Feedback in Recommender Systems. In: MEMICS 2016. (to appear), EPTSC, 2016.

[Re09] Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L.: BPR: Bayesian Personalized Ranking from Implicit Feedback. In: UAI 2009. AUAI Press, pp. 452–461, 2009.

[Sa97] Salzberg, S. L.: On Comparing Classifi ers: Pitfalls to Avoid and a Recommended Approach. Data Mining and Knowledge Discovery 1/3, pp. 317–328, 1997.

[Ya12] Yang, B.; Lee, S.; Park, S.; Lee, S.-g.: Exploiting Various Implicit Feedback for Collaborative Filtering. In: WWW 2012. ACM, pp. 639–640, 2012.

[Yi14] Yi, X.; Hong, L.; Zhong, E.; Liu, N. N.; Rajan, S.: Beyond Clicks: Dwell Time for Personalization. In: RecSys 2014. ACM, pp. 113–120, 2014.

Scalable Cloud Data Management Workshop
(SCDM 2017)

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 211

Scalable Cloud Data Management Workshop 2017

Felix Gessert[1], Norbert Ritter[2]

The Fifth Workshop on Scalable Cloud Data Management (SCDM 2017) is co-located with the BTW 2017 conference and tackles the manifold new topics in the area of cloud data management. The workshop is focused on new research challenges for scalable databases and data processing in the context of cloud computing.

Motivation

The increasing adoption of cloud computing for databases and data services introduces a variety of new research challenges. To leverage elastic cloud resources, scalability has to be a fundamental architectural design trait of these new cloud databases. This challenge has manifested in new data models, replication, caching and partitioning schemes, relaxed consistency and transaction guarantees as well as new protocols, APIs and storage services.

This workshop invites submissions about new technologies enabling databases for cloud environments and offering them as services (Database-as-a-Service). We solicit research on novel database designs and their distribution and storage mechanisms as well as scalable data and data-processing services. The application side of cloud databases is equally important, as it might impose new programming models, APIs and web-enabled interfaces. We are convinced that bringing together cloud computing, service design and data architectures in this workshop will contribute to this exciting new field.

Topics of interest include, but are not limited to:

- Database as a Service, Multi-tenancy
- Elasticity and Scalability for Cloud Data Management Systems
- New Protocols, Service Interfaces and Data Models for Cloud Databases
- Polyglot Persistence, NoSQL, Schemaless Data Modeling, Integration
- Data-Centric Web-Services, RESTful Data Services
- Database Architectures for Mobile and Web Clients
- Content Delivery Networks, Caching, Load-Balancing, Web-scale workloads
- Virtualization for Cloud databases, Storage Structures and Indexing
- Frameworks and Systems for Parallel and Distributed Computing
- Scalable Machine Learning, Analytics and Data Science
- Resource and Workload Management in Cloud Databases
- Tunable and Eventual Consistency, Latency
- High Availability, Reliability, Failover

[1] Universität Hamburg, gessert@informatik.uni-hamburg.de
[2] Universität Hamburg, ritter@informatik.uni-hamburg.de

- Transactional Models for Cloud Databases
- Query Languages and Processing, Programming Models
- Consistency, Replication and Partitioning
- CAP, Data Structures and Algorithms for Eventually Consistent Stores

The goal of this first German version of the Scalable Cloud Data Manamgement workshop is, to particularly bring together the community of European researchers that are concerned with building, improving and evaluating cloud-based databases and data processing systems. The workshop offers the opportunity to present current research, exchange ideas, discuss trends and form new cooperations. Five high quality full papers were accepted for publication in SCDM 2017 that range from surveys to the design and implementation of modern cloud databases.

We would like to thank all members of the program committee that made this workshop possible and provided insightful reviews to the authors. We are looking forward to a successful workshop with interesting presentations and lively discussions.

1 Workshop Organizers

Felix Gessert, University of Hamburg
Norbert Ritter, University of Hamburg

2 Program Committee

Andreas Thor, University of Leipzig
Armin Roth IBM
Ching Hsien (Robert) Hsu, Chung Hua University
Eiko Yoneki, University of Cambridge
Fabian Panse, Universität Hamburg
Haopeng Chen, Shanghai Jiao Tong University
Harald Kosch, Universität Passau
Holger Schwarz, Universität Stuttgart
Jiannan Ouyang, University of Pittsburgh
Keke Chen, Wright State University
Liqiang Wang, University of Wyoming
Meike Klettke, Universität Rostock
Nils Gruschka, Kiel University of Applied Science
Russel Sears, Pure Storage
Sameh Elnikety, Microsoft Research
Sébastien Mosser, Université Nice-Sophia Antipolis
Sherif Sakr, University of New South Wales
Shigeru Hosono, NEC Knowledge Discovery Research Lab.
Shiping Chen, CSIRO

Stefan Dessloch, University of Kaiserslautern
Uta Störl, Hochschule Darmstadt

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 215

Coordinated Omission in NoSQL Database Benchmarking

Steffen Friedrich[1] Wolfram Wingerath[2] Norbert Ritter[3]

Abstract: Database system benchmark frameworks like the Yahoo! Cloud Serving Benchmark (YCSB) play a crucial role in the process of developing and tuning data management systems. YCSB has become the de facto standard for performance evaluation of scalable NoSQL database systems. However, its initial design is prone to skipping important latency measurements. This phenomenon is known as the coordinated omission problem and occurs in almost all load generators and monitoring tools. A recent revision of the YCSB code base addresses this particular problem, but does not actually solve it. In this paper we present the latency measurement scheme of NoSQLMark, our own YCSB-based scalable benchmark framework that completely avoids coordinated omission and show that NoSQLMark produces more accurate results using our validation tool SickStore and the distributed data store Cassandra.

Keywords: Database Benchmarks, Coordinated Omission, NoSQL

1 Introduction

In recent years, a lot of new distributed database management technologies broadly classified under the term NoSQL databases have emerged for large data intensive applications. Since the NoSQL landscape is very diverse [Ge16], decision makers have to rely on performance benchmarks in addition to functional requirements to select the appropriate data management solution for their application needs. Traditional relational database systems are evaluated with industry standard benchmarks like TPC-C [Tr05] and TPC-E [Tr07]. These are distinguished by the fact that they model particular applications and run workloads with queries and updates in the context of transactions, whereas the data integrity is supposed to be verified during the benchmark. Because NoSQL databases sacrifice consistency guarantees, transactional support and query capabilities, a new benchmark, the Yahoo! Cloud Serving Benchmark (YCSB) [Co10], has become the de facto standard, which does not model a real-world application, but examines a wide range of workload characteristics with customisable mixes of read/write operations. To support any NoSQL database, YCSB is limited to a CRUD interface only.

Nowadays, almost every new research on the NoSQL domain has been experimentally evaluated with the help of YCSB and there have been many extensions developed for it [Fr14]. Often overlooked was the underlying system model of its load generator, until recently when the *coordinated omission problem* attracted great interest among practitioners. The problem was named by Gile Tene, CTO and co-founder at Azul Systems, who describes

[1] University of Hamburg, ISYS, Vogt-Kölln-Straße 30, Hamburg, Germany, friedrich@informatik.uni-hamburg.de
[2] University of Hamburg, ISYS, Vogt-Kölln-Straße 30, Hamburg, Germany, wingerath@informatik.uni-hamburg.de
[3] University of Hamburg, ISYS, Vogt-Kölln-Straße 30, Hamburg, Germany, ritter@informatik.uni-hamburg.de

it in his talks on "How not to measure latency" [Te16]. We explain the coordinated omission problem in Section 2. Its great attention led to an extension of YCSB that tries to counteract the problem. Therefore, in addition to the normal latency values, corrected, *intended* latency values are computed. We describe this correction in Section 2.1 and explain how we can avoid the problem at all with our scalable benchmark framework NoSQLMark in Section 3. To investigate the effect of the coordinated omission problem, we extend our database system SickStore in Section 4 and evaluate how the intended latency values differ from those with coordinated omission avoidance in Section 5. We finish the paper with a summary and concluding remarks in Section 6.

2 The Coordinated Omission Problem

The Java code snippet in Listing 1 shows how YCSB generates load and measures response times. To issue requests with a given target throughput, the required wait time between two operations is computed (1). After measuring the latency for one request (5-8), the measurement thread sleeps until the computed deadline is reached (12-16). Since the call to the database system under test is done synchronously, this methodology only works well as long as the response time falls within the desired time window. For illustration, consider the example measurements depicted in Figure 1.

```
1  _targetOpsTickNanos = (long) (1_000_000_000 / target)
2  long overallStartTime = System.nanoTime();
3  while (_opsdone < _opcount) {
4    long startTime = System.nanoTime();
5    Status status = _db.read(table,key,fields,result);
6    long endTime = System.nanoTime();
7    _measurements.measure("READ",
         (int)((endTime-startTime)/1000));
8
9    _opsdone++;
10
11   long deadline = overallStartTime + _opsdone *
         _targetOpsTickNanos;
12   long now = System.nanoTime();
13   while((now = System.nanoTime()) < deadline) {
14     LockSupport.parkNanos(deadline - now);
15   }
16 }
```

List. 1: A code snippet, similar to YCSBs source code, that shows how load generators typically measure latency.

To illustrate the problem, we run a workload with one client thread, a target throughput of 10 ops/sec and configure our database SickStore (introduced in Section 4) to simulate a *hiccup* (disruptions and delays caused by garbage collection, context switches, cache

buffer flushes to disk, etc.) of one second after 10 seconds run time. According to Listing 1, we want the client thread to issue one request each 100 milliseconds. Request 1 to 9 take less than 50µs each, request 10 takes one second. The subsequent request again takes only around 50µs because it does not happen at its designated time. In this manner, the database system under test delays requests that would have been made during the synchronous call, which leads to the coordinated omission of relevant measurements such that the overall measurement results do not reflect database hiccups very well. This obscuration becomes worse if the results are reported as the mean latency only.

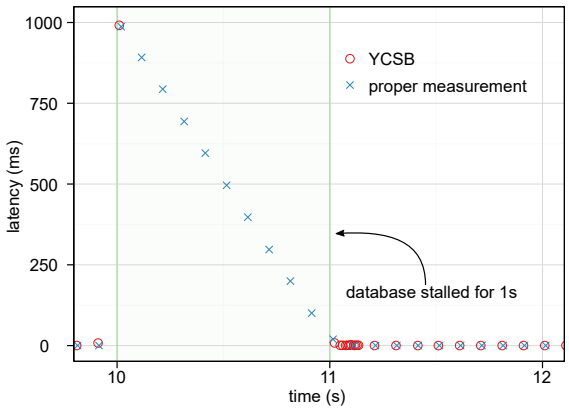
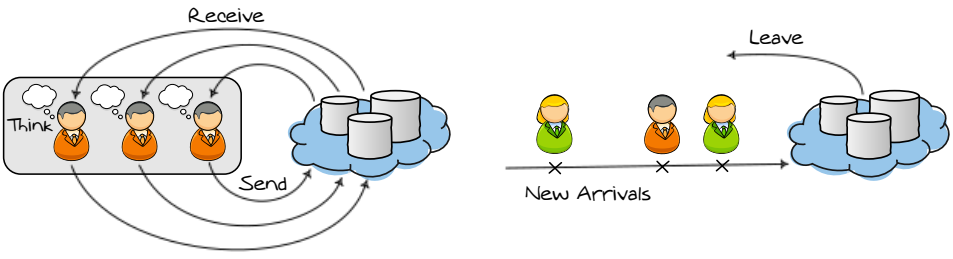


Fig. 1: Measurements that illustrate the coordinated omission problem.

According to Schröder et. al. [SWHB06], YCSB follows a *closed system model*: There is some fixed number of users that repeatedly request the system, receive the response and then "think" for some amount of time. A new request is only sent after the completion of the previous one. However, YCSB focuses on workloads web applications place on cloud data systems. Therefore, we actually want to measure the response time for user requests that appear independently such that requests arrive in an unbounded fashion even if the database system stalled for a longer period of time. This corresponds to an *open system model*, where users arrive with an average arrival rate λ and a new request is only triggered by a new user arrival, independently of request completions.



(a) Closed system model (b) Open system model

Fig. 2: Illustrations of the closed and open system models.

We assume that many developers do not consciously take this design difference into account. Schröder et. al. came to a similar conclusion and found that most of the benchmark systems that they investigated, including the relational database system benchmarks TPC-C and TPC-W, follow a closed system model. But developers in general often do not mention anything about the underlying system model in their documentations. An exception is Sumita Barahmand who states in her PhD thesis on benchmarking interactive social networking actions [Ba14] that her framework BG implements a closed system model, even though she, too, considers the open model more realistic for a social networking site (or websites in general). But she leaves this implementation open for future research.

2.1 Coordinated Omission Correction in YCSB

With the release 0.2.0 RC1 of YCSB in June 2015, some changes regarding the coordinated omission problem were introduced by Nitsan Wakart[2]. First of all, the histogram library called HdrHistogram[3] was added. In contrast to the standard histogram implementation with a fixed number of buckets, the HdrHistogram can measure a dynamic range of measurement values with configurable value precision. Additionally, the library provides a loss-free compressed serialization which allows the reconstruction of the complete histogram, for example to export data for plotting percentile curves. However, to overcome the coordinated omission problem the most important change is the implementation of the intended measurement interval. The idea is to correct the latency values by setting the measurement start time to the designated time of the request. Consider again Listing 1, the intended latency of the actual request can be measured by using the deadline of the preceding request as start time:

```
measure("Intended-READ", (int)((endTime - _precedingDeadline)/1000));
```

This may appear to solve the problem, but the database is still able to influence the actual time of the request which may affect the measurement results. Because all pending requests (i.e. their calculated deadlines lie in the past) are sent immediately after a delayed response, a very high load is generated temporarily. The big difference to an open system model is that the requests are queued on the client side, whereas in the open model a request could be sent to and processed by the database, even if the database is still processing the previous one (e.g. due to multithreading or load balancing across multiple nodes). Intuitively, we would therefore suspect that the corrected latency values tend to be higher than with an open system model. We examine this hypothesis experimentally in Section 5.

3 Coordinated Omission Avoidance in NoSQLMark

During our research on staleness-based consistency measurement methods [Wi15], we decided to develop our own benchmarking framework NoSQLMark on top of YCSB, to

[2] Changes in YCSB regarding the coordinated omission problem: https://github.com/brianfrankcooper/YCSB/blob/0.2.0-RC1/core/CHANGES.md
[3] HdrHistogram library for Java: https://github.com/HdrHistogram/HdrHistogram

overcome several of it's shortcomings. NoSQLMark is written in Scala with the Akka toolkit and thus allows to easely scale a workload on different benchmark nodes. You no longer have to start YCSB manually on different servers nor do you have to aggregate the different measurement results by your own. Since Java libraries can be used directly in Scala code we can depend on YCSB to support all of its database bindings as well. Furthermore, the actor model allows us to implement more complex workloads that require communication between actors like for example in our proposed consistency measurement method [Fr14]. To avoid coordinated omission, we implement an open system model by sending our requests in an asynchronous fashion. The Scala code snippet in Listing 2 shows how an actor generates load and measures latency.

```
1  implicit val executionContext = context.system.
2          dispatchers.lookup("blocking-io-dispatcher")
3
4  case DoOperation => {
5    val operation: Operation = workload.nextOperation
6    val startTime = System.nanoTime
7    val future = Future {
8      sendRequest(operation)
9    }
10   future.onComplete {
11     case Success(status) => {
12       val endTime = System.nanoTime
13       measurementActor ! Measure(operation.name,
             status, (endTime - startTime) / 1000)
14     }
15     case Failure(ex) => {
16       log.error(ex, "Error occured during operation
             {}", operation.name)
17     }
18   }
19   _opsdone += 1
20   if (_opsdone < _opcount) scheduleNextOperation()
21   else supervisor ! WorkIsDone(workerID, jobID)
22 }
```

List. 2: Scala code snipped for asynchronous load generation to implement an open system model in NoSQLMark.

Since an actor processes information by means of handling asynchronous messages, each Akka actor has to implement the receive method which should define a series of case statements that defines which messages an actor can handle. Thus the code snippet shows how our actor (called *worker*) handles the message of type `DoOperation`. The critical difference to Listing 1 is that we use a scala `Future` to send the actual request asynchronously (7-9). Note that we create a dedicated execution context for blocking i/o and have tried various

executor implementations / configurations[4] (1-2). Next, we register a callback on the future by using the `onComplete` method, which is applied to the value of type `Success[Status]` if the future completes successfully, or to a value of type `Failure[Throwable]` otherwise (10-18). If successful, we send the computed latency to the measurement actor (11-14). The `scheduleNextOperation` method schedules a new message of type `DoOperation` to be send to the actor himself (20). Thus, the actor remains more or less reactive for another message, e.g. an abort message by the user. At the end, a `WorkIsDone` message is sent to the supervisor actor (21), who still have to wait for the outstanding measurements by the measurement actor. The entire source code will be released in open source and we will publish further details on NoSQLMark's architecture and future development at the official project page[5].

4 Designing SickStore for Coordinated Omission

We introduced SickStore, the **s**ingle-node **inc**onsistent **k**ey-value **store** to validate existing staleness benchmarks [Wi15]. For this purpose, SickStore's design enables it to simulate a replica set consisting of multiple virtual storage nodes that produce stale values, but at the same time, it provides consistent knowledge about the system state at any point in time. This is simplified by running the server in a single thread. Nevertheless, in order to allow the simulation of latencies, the server returns the calculated latency values to the client library, which blocks the calling thread for the corresponding remaining time.
In order to investigate the problem of coordinated omission, we have extended SickStore to simulate a maximum throughput λ_{max} and database hiccups, both configurable per virtual node. Again, the server process should not really be slowed down or stopped. Therefore, we simulate a request queue by introducing a counter C for the number of outstanding requests in the system. Let

$$\begin{aligned} t_i &= \text{ timestamp at which request } i \text{ was received by the server;} \\ \Delta_{i,j} &= t_j - t_i \text{ the time period between request } i \text{ and } j \text{ where } t_i < t_j; \\ C(t_i) &= \text{ the number of outstanding requests at time } t_i;. \\ T_i &= \text{ the waiting time of request } i \text{ in the system} \end{aligned}$$

In order to simulate the maximum throughput, for each request i we compute

$$C(t_i) = max(0\ ,\ C(t_{i-1}) - \lambda_{max} \times \Delta_{i-1,i})\ , \tag{1}$$

where $i-1$ is the direct predecessor to i. Note if i is the first request then C must be equal to 0. Then we compute $T_i = C(t_i)\ /\ \lambda_{max}$ and finally add one to the counter C before returning T_i to the client library. To also simulate a hiccup, we only have to increase the counter C by

[4] We ended up with a similar configuration as suggested in the Akka docs, but with a high pool-size-factor instead of a fixed-pool-size, to get a cached thread pool with a high limit and an unbounded queue: http://doc.akka.io/docs/akka/current/scala/dispatchers.html#More_dispatcher_configuration_examples

[5] NoSQLMark can be obtained from http://nosqlmark.informatik.uni-hamburg.de/

the appropriate number of requests. Let

$$t_h = \text{ the desired start time of the hiccup;}$$
$$H = \text{ the desired hiccup duration.}$$

Then equation (1) becomes

$$C(t_i) = \begin{cases} \max(0\ ,\ C(t_{i-1}) - \lambda_{max} \times \Delta_{i-1,i} + \lambda_{max} \times H), & \text{if } t_i < t_h <= t_j \\ \max(0\ ,\ C(t_{i-1}) - \lambda_{max} \times \Delta_{i-1,i}), & \text{otherwise} \end{cases} \quad (2)$$

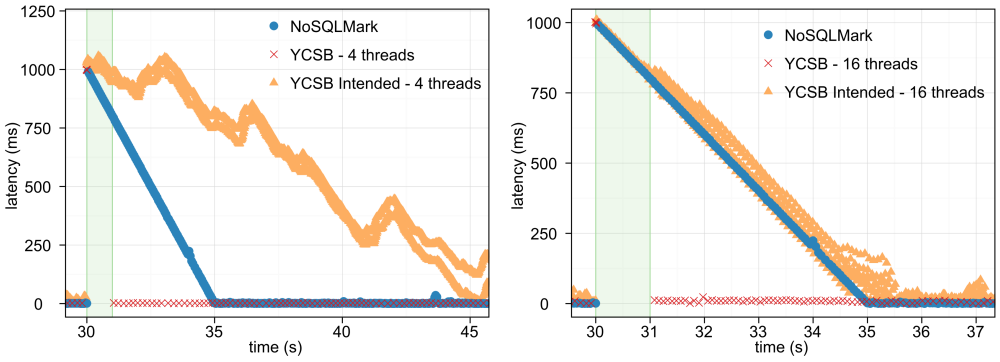
Finally, each calling client thread has to pause for the duration $l + T_i$, where l is the calculated network latency by SickStore's latency generator.

5 Experimental Evaluation

We proceed to experimentally demonstrate that the latency values corrected by the intended measurement interval in YCSB greatly differ from those measured by our implementation of an open system model. This is confirmed by the exemplifying simulation of a hiccup with SickStore, as well as in our experiments with Cassandra.

5.1 Coordinated Omission Simulation with SickStore

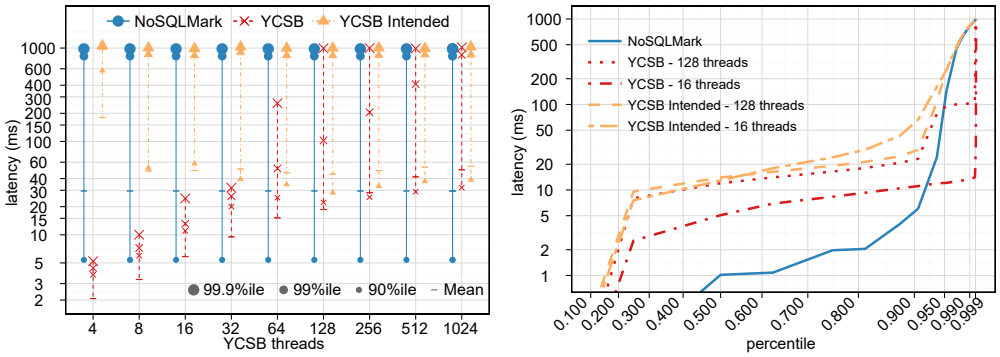
To get an insight into how the measurement results behave in a hiccup situation, we have conducted a series of micro-benchmarks with SickStore. For the simulation, we used one machine with 2.60 Ghz Quad-Core Intel i7-6700HQ processor and 16 GB memory.We run a workload with a target throughput of 1000 ops/sec, 90000 operations and configure SickStore to simulate a hiccup of one second after 30 seconds run time. Thus, each micro-benchmark runs 90 seconds, which is sufficient for each run to stabilize the latency values to a level as before the hiccup. In the first series of experiments we increase the number of YCSB client threads and set the maximum throughput at SickStore to 1250 ops/sec. The two time series plots in Figure 3 illustrate how the number of YCSB client threads affects the measurement behavior. We see that with NoSQLMark, the latency values are exactly stabilized at the point as we would expect with the configured maximum throughput and hiccup duration. For YCSB, the number of requests during the hiccup is limited by the number of threads and many latency values are omitted, which are finally to be corrected by the intended measurement interval. Surprisingly, the intended measurements stabilize very late for 4 threads, which is already much better with 16 threads. In Figure 4(a), we see some percentiles and mean values for the latencies with different YCSB thread numbers. Even if the number of threads increases further, the intended values are higher than those of NoSQLMark. This is particularly true for the mean value and the 90-percentiles, which are approaching up to 128 threads ever closer to the results of NoSQLMark but increase again as of 256 threads. It is noticeable that the uncorrected mean value approaches the mean value of NoSQLMark with increasing thread count as well. In addition, as the number of threads increases, the corrected and uncorrected latency values of YCSB approach each



(a) 4 YCSB threads (b) 16 YCSB threads

Fig. 3: Time series plots of two runs with different numbers of YCSB threads.

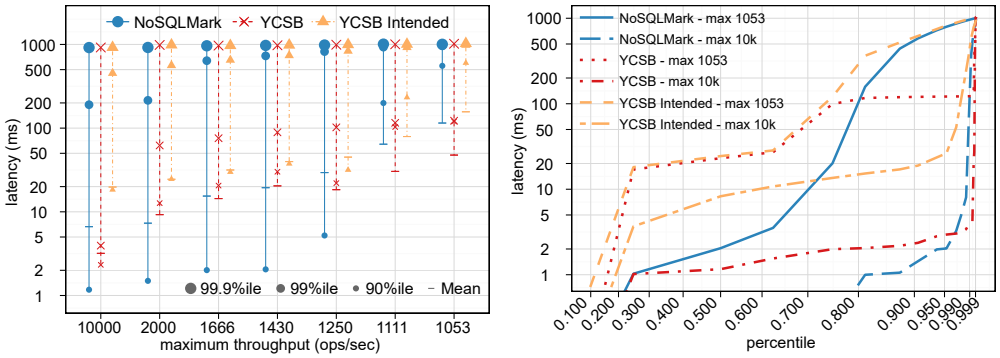
other, which becomes even more apparent when viewing the entire percentile spectrum in Figure 4(b). Incidentally, it also shows that the latencies of the lower percentiles are one order of magnitude higher than those of NoSQLMark. Next, we set the number of threads



(a) Overview (b) Percentile spectrum for 16 and 128 YCSB threads.

Fig. 4: Simulation with different numbers of YCSB threads.

to 128 and vary the maximum throughput. Figure 5 shows again the overview and the entire percentile spectrum for two selected experiments. For all three measurement approaches, it can, of course, be observed that the latency values increase with higher system load. But in principle, the measurement results differ considerably, even when the system is under different load conditions. In Figure 5(b) one can see how all percentile curves shift to the upper left corner of the spectrum, with higher system utilization. Although the difference in the 90th and higher percentiles appears to be smaller, it is significant for smaller percentiles (e.g. 200ms in the 80th percentile between NoSQLMark and YCSB Intended). In summary, our simulation of a single hiccup shows that the fundamentally different system models of load generation also result in a completely different percentile spectrum of the latency values.



(a) Overview

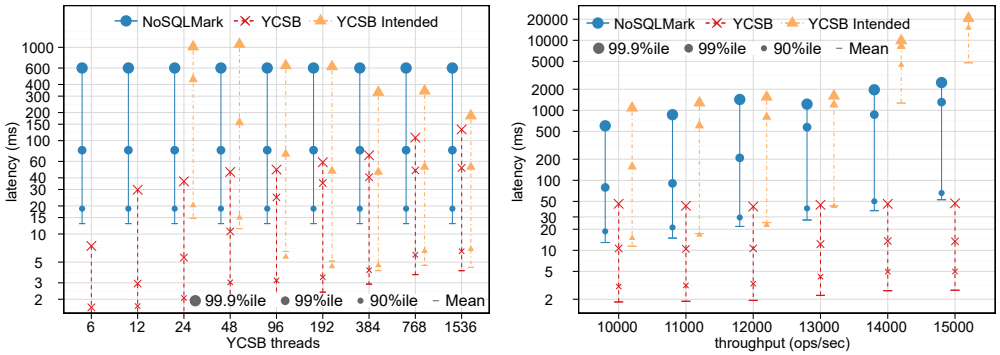
(b) Percentile spectrum for max 1053 and 10k ops/sec.

Fig. 5: Changing SickStore's maximum throughput.

5.2 Elasticity Benchmark with Cassandra

Kuhlenkamp et. al. have recently repeated scalability and elasticity experiments of previous research with YCSB [KKR14]. Since their results do not include the intended latency values of YCSB, it is a good idea to repeat an experiment to see the difference between the two system models of the load generators and the intended latencies. We performed the benchmark with a simpler configuration by running the workload on a single Cassandra node and adding another node to the cluster after 5 minutes. For each node, we used instances of our private OpenStack cloud with 4 vCPUs and 16 GB of memory and one instance with 6 vCPUs and 63 GB memory for the benchmark. Each instance runs on a separate server (2,1 GHz Hexa-Core Intel Xeon E5-2620v2, 15MB Cache, 64 GB memory, 6 disk RAID-0 array and gigabit ethernet). The Cassandra node is initially loaded with 10 million rows, i.e., the node manages 10 GB. We perform the experiments with YCSB workload A with a Zipfian request distribution and add the second node to the cluster after 5 minutes of runtime without a data streaming throughput limit. At a target throughput of 10 000 ops/sec, the second node starts serving after roughly 5 minutes, i.e. after 10 minutes run time the latency stabilizes, which is why we run each experiment for a maximum duration of 15 minutes.

Figure 6 shows the results at a glance, (a) for an increasing number of threads (adapted to 6 cores) with a target throughput of 10k ops/sec, and (b) for 48-thread increasing target throughput. 15k ops/sec was the maximum throughput that could be achieved. We can again observe that the measurement results of YCSB and YCSB Intended are becoming more and more similar with increasing number of threads. The intended latency values for the represented percentiles even fall below the values of NoSQLMark, starting at 96 threads. However, 10k ops/sec is also only 66.6% of the maximum achievable load. On the other hand, we see that the difference between the three approaches increases drastically as the load increases. Nevertheless, the same observations as in our simulation with SickStore are also reflected in these results.



(a) Different numbers of YCSB threads (10k ops/sec) (b) Different target throughputs (48 YCSB threads)

Fig. 6: Percentiles and mean value for the elasticity benchmark with Cassandra.

6 Concluding Remarks

In this paper, we described the coordinated omission problem and how it relates to the system model of load generators. Just about all benchmark frameworks are based on a closed system model and thus have the coordinated omission problem. We described how YCSB tries to correct the latency values by introducing the intended measurement interval. Afterwards we showed how an open system model was implemented in NoSQLMark and how we extended the simulation tool SickStore to investigate the coordinated omission problem. Finally, we conducted several experiments which showed that the different system models lead to completely different measurement results, even the intended latency values greatly differ from those of the open system model.
From our results, we conclude that developers of a benchmark framework should be more concerned about the underlying system model, because the type of load generation can not be corrected afterwards. For this reason, we plan to refine the load generator of NoSQLMark in the future, by generating the load using different distributions such as Possion.

References

[Ba14] Barahmand, Sumita: Benchmarking Interactive Social Networking Actions. PhD thesis, USC Computer Science Department, Los Angeles, California 90089-0781, 2014.

[Co10] Cooper, Brian F.; Silberstein, Adam; Tam, Erwin; Ramakrishnan, Raghu; Sears, Russell: Benchmarking Cloud Serving Systems with YCSB. In: Proceedings of the 1st ACM Symposium on Cloud Computing. SoCC '10, ACM, New York, NY, USA, pp. 143–154, 2010.

[Fr14] Friedrich, Steffen; Wingerath, Wolfram; Gessert, Felix; Ritter, Norbert: NoSQL OLTP Benchmarking: A Survey. In: 44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Big Data - Komplexität meistern, 22.-26. September 2014 in Stuttgart, Deutschland. pp. 693–704, 2014.

[Ge16] Gessert, Felix; Wingerath, Wolfram; Friedrich, Steffen; Ritter, Norbert: NoSQL database systems: a survey and decision guidance. Computer Science - Research and Development, pp. 1–13, 2016.

[KKR14] Kuhlenkamp, Jörn; Klems, Markus; Röss, Oliver: Benchmarking Scalability and Elasticity of Distributed Database Systems. Proc. VLDB Endow., 7(12):1219–1230, August 2014.

[SWHB06] Schroeder, Bianca; Wierman, Adam; Harchol-Balter, Mor: Open Versus Closed: A Cautionary Tale. In: Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3. NSDI'06, USENIX Association, Berkeley, CA, USA, pp. 18–18, 2006.

[Te16] Tene, Gil: , How NOT to Measure Latency. https://www.infoq.com/presentations/latency-response-time, March 2016.

[Tr05] Transaction Processing Performance Council: , TPC-C Benchmark Specification. http://www.tpc.org/tpcc, 2005.

[Tr07] Transaction Processing Performance Council: , TPC-E Benchmark Specification. http://www.tpc.org/tpce, 2007.

[Wi15] Wingerath, Wolfram; Friedrich, Steffen; Gessert, Felix; Ritter, Norbert: Who Watches the Watchmen? On the Lack of Validation in NoSQL Benchmarking. In: Datenbanksysteme für Business, Technologie und Web (BTW), 16. Fachtagung des GI-Fachbereichs "Datenbanken und Informationssysteme" (DBIS), 4.-6.3.2015 in Hamburg, Germany. Proceedings. pp. 351–360, 2015.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 227

Preserving Recomputability of Results from Big Data Transformation Workflows

Matthias Kricke [1], Martin Grimmer [1], Michael Schmeißer [2]

Matthias Kricke, Martin Grimmer und Michael Schmeißer

Abstract: The ability to recompute results from raw data at any time is important for data-driven companies to ensure data stability and to selectively incorporate new data into an already delivered data product. When external systems are used or data changes over time this becomes even more challenging. In this paper, we propose a system architecture which ensures recomputability of results from big data transformation workflows on internal and external systems by using distributed key-value data stores.

Keywords: BigData, recomputability, bitemporality, time-to-consistency

1 Introduction

For data-driven organizations, the possibility to selectively incorporate new data into an already calculated data product (like a chart, report or recommendation, etc.) can be required. Hence, the option to recompute information from their raw data is needed, even if this data comes from several external systems. It is obvious that temporal features are necessary for this kind of recomputability. In the past, a lot of research has been done on temporal databases[OS95]. Temporal features have also been incorporated in the SQL:2011 standard[KM12]. Problems like concurrency control[GR92] have been solved for modern, distributed systems of e.g. Microsoft[Dr15], Google[Co13] and SAP[Le13]. However, those systems either can't be used on-premises or have high licensing costs. In addition, recomputability of data products in big data systems with dependencies to external systems has not been addressed in recent research.

However, this is a problem which mgm technology partners GmbH has been asked to solve for a customer. The recomputability of data products enables mgm's customer to reconstruct earlier data versions, reports and analysis results to compare them with newer ones. Moreover, it is now possible to incorporate data changes only from specific external systems. To deal with the requirements of low license costs and an on-premises system, mgm's customer has decided to use a scalable and distributed key-value data store like

[1] Universität Leipzig, Fakultät für Mathematik und Informatik, Institut für Informatik, Augustusplatz 10, 04109 Leipzig, {kricke, grimmer}@informatik.uni-leipzig.de
[2] mgm technology partners GmbH, Neumarkt 2, 04109 Leipzig, michael.schmeisser@mgm-tp.com

Apache Accumulo[3], Apache HBase[4] or Apache Cassandra[5]. Nonetheless, those stores do not offer a proper solution for recomputability.

In this paper, we present ELSA (**E**xterna**L** **S**ystem **A**daptor), a big data system architecture which ensures recomputability of data products with dependencies to external systems by using a variant of the concept of bitemporality[JSS94]. We show an efficient way to recompute data products even in scenarios where the external systems aren't versioned.

Customer specific application details are confidential, yet we will provide simple examples to follow our explanations in the next sections.

2 Requirements

Mgm's customer wants a cost efficient system which is capable of handling external systems from either other company departments or companies as data sources. There is a strong requirement for distributed data transformation processes[RD00]. Those processes are incorporating data from external systems. This leads to the demand to relieve the external systems from high-frequent distributed requests. A solution has to be linearly scalable with low to none license costs and must support many simultaneous, distributed data transformation processes. In addition, it has to be able to selectively incorporate new data into an already calculated data product by recomputing it. Hence, all records are immutable and each version of a record has to be stored. On the one hand, a high volume of external data is sent to the system. On the other hand, the system has to store all versions of this data, which leads to a data base increasing by several terabytes a month.

As mentioned before, recomputability can be ensured by versioning the data.

Definition 1 (Versioning)
A record is versioned if each of its occurred states is accessible with the corresponding timestamp. A system is versioned if each of its records is versioned.

Reality shows that full versioning in external systems is nothing that can be relied on. Furthermore, it is possible that external systems do not meet the latency or throughput requirements of the distributed data transformation process. Therefore, the system has to ensure scalability, recomputability, high throughput and low latency itself which leads to the necessity of a suitable big data system architecture.

3 External System Adaptor

To deal with the requirements stated in Section 2 a decoupling of external systems and the data transformation process is necessary. The **E**xtern**l** **S**ystem **A**daptor (ELSA) is the

[3] https://accumulo.apache.org/
[4] https://hbase.apache.org/
[5] https://cassandra.apache.org/

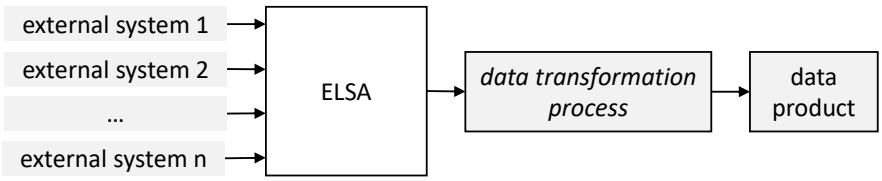


Fig. 1: Decoupling of external systems and the data transformation process with ELSA.

junction shown in Figure 1, which fulfills all requirements. However, some demands to an external system are inevitable. Every external system used in the data transformation process needs a defined interface which provides a change history on their data. Furthermore, each change needs to be well defined as operation as stated in Definition 2.

Definition 2 (External key-value Operations)
There are insert and delete operations. An insert operation is defined as a tuple

$$insert = (k, t_e, v).$$

Where k is the key, t_e is the event timestamp and v is the value of the external record.

The delete operation is defined as a tuple

$$delete = (k, t_e)$$

which invalidates every record with an event timestamp lower or equal to t_e.

Assuming an external system inserted the tuples $t_1 = (a, 1, v)$, $t_2 = (a, 2, v')$ and $t_3 = (a, 4, v'')$, a delete operation defined as $delete = (a, 3)$ would remove t_1 and t_2 in the external system and retain t_3. However, to preserve recomputability, in ELSA data is not deleted but invalidated.

To fulfill the aforementioned requirements ELSA consists of:

- the ELSA Store, which is replacing the external systems functionality by providing a queryable data history and
- the ELSA Data Synchronization, which is used for keeping the ELSA Store synchronized with the external systems.

Hence, distributed data transformation processes are now using the ELSA Store instead of the external systems, which removes the logic of handling external systems from them. Moreover, the ELSA Data Synchronization highly reduces the throughput and latency requirements for an external system since data is only read once by it. Since those requirements now has to be handled by the store it is an internal, distributed, scalable, multi-version key-value data store. To ensure recomputability, records in the store are immutable and never overwritten or deleted. In addition the store is underlying a strict back up process to avoid data loss.

3.1 ELSA Get Requests and Bitemporality

Assuming that ELSA is using the timestamp an external system provides, for storing and querying data, a significant problem for consistency and production arises. There is always a delay between the time when an event happens and when it is visible in an external system. This time further increases when ELSA gets the change history. An external system may only send unregular updates to ELSA. In that case two get requests for the same record with the same timestamps may lead to different results.

For example a car sends its current GPS position to its manufacturer at 9:00 a.m., who is sending the position to ELSA at 9:30 a.m.. At 9:15 a.m. a get request to ELSA wants to know the last position of the car five minutes ago (9:10 a.m.). The request will return no results. If the request for the last car position at 9:10 a.m. is send again at 9:45 a.m. the result would be the stored value. Hence, the result of the same request is inconsistent and thus not recomputable.

To ensure recomputability, get request result consistency is necessary. For achieving this the concept of bitemporality, as stated in Definition 3, is used.

Definition 3 (Bitemporality)
A data record is bitemporal if it has two decoupled timestamps which distinguish between the event time t_e and the ingest time t_i.

Be $q(k_q, t_E, t_I) = r$ a get request to the ELSA Store, where k_q is the key to be queried. t_E is the maximum event timestamp and t_I the maximum ingest timestamp. The resulting r is an ELSA Record or empty, as defined in Definition 4.

Definition 4 (ELSA Record)
An ELSA record r is created from an external key-value operation, see Definition 2, and defined as $r = (k, t_i, o, t_e, v)$. Where k is the key of an external key-value operation and t_i is the time the key-value operation was ingested into ELSA. The type of the operation is given in o and could either be insert or delete. The event timestamp t_e is derived from the insert or delete operation. The value v is set for an insert operation or empty for a delete operation.

Let R_q be a set of ELSA records for a certain external system

- whose keys are equal to k_q
- whose ingest times are smaller than t_I
- whose event times are smaller than t_E

The ELSA get request takes the record r with the maximum event time of R_q. If operation o of r is $delete$, the get has no result. Otherwise the ELSA record r is the result of the get request.

Regarding the car manufacturer example, t_E is set to 9:00 a.m. and t_I is set to 9:30 a.m.. The ELSA get request has to use both timestamps to identify the correct value which will make the both aforementioned queries two distinct queries. While the first request is using a maximum ingest timestamp of 9:15 a.m. the second one is using 9:45 a.m.. Thus the first get will return no results while the second will return a GPS location.

A more complex example which contains delete operations is shown in Figure 2.

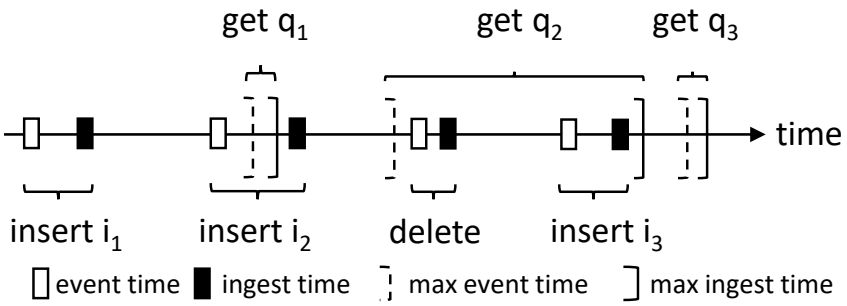


Fig. 2: All data versions of a record with different ingest and event times with several get requests.

During get q_1 the event time of *insert* i_2 is matching the query but the queries maximum ingest time is not. Hence, the result of query q_1 is the record inserted by *insert* i_1. For request q_2 the event times of the *delete* and *insert* i_3 are too large and they are filtered although the ingest time requirements where met. This leads to the record of *insert* i_2 as result for the query. In the case of q_3 the result is i_3 since it has the largest event timestamp and the ingest time requirements are met.

3.2 ELSA Data Synchronization & Store

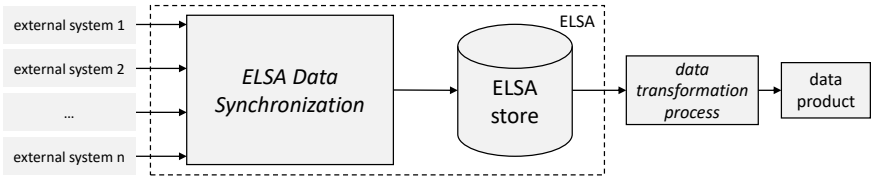


Fig. 3: The ELSA dataflow.

Figure 3 shows a more detailed view of ELSA. The ELSA Data Synchronization subscribes to all changes done in the external systems and writes them into a queue. This queue is a replicated queue which is able to handle peaks. An entry in the queue is an external key-value operation specified in Definition 2. A write process pulls the external key-value operation from the queue and transforms it into an ELSA record as defined in Definition 4, which is written into the ELSA Store.

The ELSA Store is a Big Table[Ch08] like persistent, distributed, scalable, multi-version key-value data store. It is able to handle massive, parallel requests from data transformation processes. Furthermore it is target of backup processes to ensures that no data is lost and recomputability stays intact even in the case of a system failure.

4 Implementation

ELSA was brought into production by mgm and its customer. In this section some of the pitfalls will be described and solved. For a better understanding those pitfalls will be described on specific technologies. Therefore the following explanations will be done by assuming that the software for distributed computing is Apache Hadoop[6] and the distributed database is Apache Accumulo which stores its data into the Hadoop Distributed File System (HDFS).

4.1 Time-to-Consistency

In databases there is a small delay between the ingest time given by the store and the moment the value is written. To write a record a database has to determine the ingest timestamp e.g. by using the system clock, and only then it is able to finally write it. This problem even increases in distributed databases because of network latency and communication for replication and distribution. This may lead to the situation where data with a certain ingest time t_i is written at a later time t_y, which breaks the recomputability.

The *time-to-consistency* t_{con} defines an upper bound for how long it may take from the determination of the ingest timestamp of a record till the record is written. A system fulfills the time-to-consistency if its writing processes at minimum use the current time t_{now} as ingest timestamp. Furthermore read operations may only choose a maximum ingest timestamp t_I for their read requests which is at most $t_{now} - t_{con}$.

The concept of time-to-consistency is applied to all read operations in ELSA. An example value for t_{con} could be:

$$t_{con} = \text{write timeout} \cdot (\text{write retries} + 1)$$

In an Apache Hadoop environment with configured write timeout of 30 seconds and a maximum of 3 write retries $t_{con} = 120s$.

4.2 Schema and Server-Site-Iterator

Table 1 shows the ELSA schema of record r for the distributed and sorted key-value data store Apache Accumulo.

[6] https://hadoop.apache.org/

Record r	Row-Key k	Column Family External Store	Column Qualifier t_e	Version t_i	Value operation & v
r_1	x	ext_1	5	10	insert & v_1
r_2	x	ext_1	10	30	delete
r_3	x	ext_1	12	20	insert & v_2
r_4	x	ext_1	35	40	insert & v_3

Tab. 1: Example ELSA Store table instance.

The row-key is the key k of an ELSA record r. The data of external systems is stored in the same Apache Accumulo table but distinguished by the column family. However, by using locality groups, only data for a given external system is considered during a get request. The column qualifier contains the event time of r and the version is set by Apache Accumulo and resembles t_i. Operation and value of r are encoded into the value field.

When a request is send to Apache Accumulo, by default it returns the latest value of a record. Since ELSA is using a bitemporal approach for the get requests (see Section 3.2) this is not feasible. Therefore, the version iterator had to be removed and the problem has been solved by using custom server-site iterators. This is possible since an iterator processes all records of the same key in ascending order by column family and column qualifier and in descending order by version. Even in the case that data is inserted out-of-order, like it is shown in Table 1 the algorithm returns the correct result.

For a given get request $q = (k, t_I, t_E)$ and store *ext* the server-side iterator iterates the records R with row-key equal to k and column family equal to *ext*.

Algorithm 1: Costum Server-Side Iterator Algorithm

```
result ← NULL;
foreach r ∈ R do
    if r.t_e > q.t_E then
        if result.o = insert then return result;
        else return NULL;
    if r.t_i > q.t_I then
        continue with next r;
    else
        result ← r;
return result;
```

The following examples are describing the Algorithm 1 used by the server-site iterator on Table 1. For a request $q_1 = (x, 15, 35)$ the iterator is undergoing the following steps:

1. evaluate r_1: $5 < 15 \wedge 10 < 35 \rightarrow result = r_1$
2. evaluate r_2: $10 < 15 \wedge 30 < 35 \rightarrow result = r_2$

3. evaluate r_3: $12 < 15 \wedge 20 < 35 \rightarrow result = r_3$
4. evaluate r_4: $35 \geq 15 \wedge result.o = insert \rightarrow return\ result = r_3$

For example request $q_2 = (x, 11, 40)$ the iterator is undergoing the following steps:

1. evaluate r_1: $5 < 11 \wedge 10 < 40 \rightarrow result = r_1$
2. evaluate r_2: $10 < 11 \wedge 30 < 40 \rightarrow result = r_2$
3. evaluate r_3: $12 \geq 11 \wedge result.o = delete \rightarrow return\ NULL$

During the last example request $q_3 = (x, 15, 15)$ the iterator is undergoing the following steps:

1. evaluate r_1: $5 < 15 \wedge 10 < 15 \rightarrow result = r_1$
2. evaluate r_2: $10 < 15 \wedge 30 \geq 15 \rightarrow continue$
3. evaluate r_3: $12 < 15 \wedge 20 \geq 15 \rightarrow continue$
4. evaluate r_4: $35 \geq 15 \wedge result.o = insert \rightarrow return\ result = r_1$

5 Conclusion & Future Work

We have defined the ELSA architecture which can reliably recompute data products even when data changes or external systems are used. We have managed this by keeping all versions of the data in a bitemporal and consistent way. In the former system the used state of the database was not configurable and depended on the time a get request was sent. Which led to unrecomputable results. In an ELSA get request the used state of the system is configurable and defined by the ingest timestamp. Therefore, the ELSA get request allows flexible recomputations of data transformation processes.

Besides recomputability, the architecture has several other benefits. It is lineary scalable, has a low latency and can handle massive parallel requests by using distributed technologies like Apache Hadoop and Apache Accumulo. Furthermore, temporary connection issues regarding the external systems are hidden from the data transformation process. Hence, we can even execute data transformation processes while an external system is not accessible. In addition, there is no possibility of overwhelming the external system with distributed queries from the data transformation processes.

Some data transformation processes of mgm's customer contain manual interactions of business experts. In this case, automated data transformation processes which are recomputing data products are blocked by this manual interactions. To make this case non-blocking and recomputable, we would like to enhance ELSA by a manual action registry. In this case, manual interactions are done using rich tooling and are required to result in a data transformation process themselves. One can design a system which automatically applies

those manual interactions while recomputing a data product. To bring this even further, we plan to investigate whether it is possible to reuse those manual interactions in other data transformation processes.

Additional future work may consider the version of the software used by data transformation processes since old versions may no longer be available to recompute results. We believe this is quite challenging since used framework versions, the runtime environment and hardware may change significantly. However, it might be possible by the use of recomputable virtualized environments.

6 Acknowledgements

This work was partly funded by the German Federal Ministry of Education and Research within the project Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B) and Explicit Privacy-Preserving Host Intrusion Detection System EXPLOIDS (BMBF 16KIS0522K).

References

[Ch08] Chang, Fay; Dean, Jeffrey; Ghemawat, Sanjay; Hsieh, Wilson C; Wallach, Deborah A; Burrows, Mike; Chandra, Tushar; Fikes, Andrew; Gruber, Robert E: Bigtable: A distributed storage system for structured data. ACM Transactions on Computer Systems (TOCS), 26(2):4, 2008.

[Co13] Corbett, James C; Dean, Jeffrey; Epstein, Michael; Fikes, Andrew; Frost, Christopher; Furman, Jeffrey John; Ghemawat, Sanjay; Gubarev, Andrey; Heiser, Christopher; Hochschild, Peter et al.: Spanner: Google's globally distributed database. ACM Transactions on Computer Systems (TOCS), 31(3):8, 2013.

[Dr15] Dragojević, Aleksandar; Narayanan, Dushyanth; Nightingale, Edmund B; Renzelmann, Matthew; Shamis, Alex; Badam, Anirudh; Castro, Miguel: No compromises: distributed transactions with consistency, availability, and performance. In: Proceedings of the 25th Symposium on Operating Systems Principles. ACM, pp. 54–70, 2015.

[GR92] Gray, Jim; Reuter, Andreas: Transaction processing: concepts and techniques. Elsevier, 1992.

[JSS94] Jensen, Christian S; Soo, Michael D; Snodgrass, Richard T: Unifying temporal data models via a conceptual model. Information Systems, 19(7):513–547, 1994.

[KM12] Kulkarni, Krishna; Michels, Jan-Eike: Temporal Features in SQL:2011. SIGMOD Rec., 41(3):34–43, October 2012.

[Le13] Lee, Juchang; Muehle, Michael; May, Norman; Faerber, Franz; Sikka, Vishal; Plattner, Hasso; Krueger, Jens; Grund, Martin: High-Performance Transaction Processing in SAP HANA. IEEE Data Eng. Bull., 36(2):28–33, 2013.

[OS95] Ozsoyoglu, Gultekin; Snodgrass, Richard T: Temporal and real-time databases: A survey. IEEE Transactions on Knowledge and Data Engineering, 7(4):513–532, 1995.

[RD00] Rahm, Erhard; Do, Hong Hai: Data Cleaning: Problems and Current Approaches. 2000.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 237

Privatsphäre-schützende Bereichsanfragen in unsicheren Cloud-Datenbanken

Daniel Janusz[1] Jochen Taeschner[2]

Abstract: Online-Dienste benutzen zunehmend cloud-basierte Datenbanken. Wenn dabei sensible personenbezogene Informationen gespeichert werden, müssen die Daten vor unberechtigten Zugriffen geschützt werden. Dabei dürfen nur berechtigte Personen für vorher definierte Zwecke Daten abfragen. Durch Verschlüsselung kann der Zugriff eingeschränkt werden. Ferner lassen sich mittels Datenschutzpräferenzen für jedes Individuum in der Datenbank Zugriffszwecke speichern und validieren. Eine Bereichsanfrage auf verschlüsselte Attributwerte unter Beachtung aller individuellen Präferenzen wird bisher von keinem Datenbanksystem zufriedenstellen bearbeitet. In unserem Ansatz verzichten wir auf oft benutzte „Tricks“, wie die Verwendung von deterministischen Verschlüsselungen, die Angriffe auf die Kryptographie zulassen. Stattdessen benutzen wir zwei getrennte Komponenten für die beiden Aufgaben: einen verschlüsselten Index und eine Anfragebearbeitung, die tupelweise die individuellen Präferenzen auswertet.

Keywords: Range Queries, Encrypted Cloud Database, Privacy Policies.

1 Einleitung

Bei der Administration eines Datenbankmanagementsystems können viele Fehler gemacht werden. Das richtige Backup-Management oder das Einspielen von Softwareaktualisierungen stellen essentielle Aufgaben dabei dar. Die Expertise für alle anfallenden Aufgaben besitzen vor allem kleine Unternehme in der Regel nicht. Deshalb lagern viele Firmen ihr Datenmanagement zunehmend in cloud-basierte Datenbanksystem aus, welche sich um alle anfallenden administrativen Aufgaben kümmern.

Wenn die Speicherung von sensiblen personenbezogene Informationen an einen Cloud-Anbieter ausgelagert wird, sollten die Daten vor unberechtigten Zugriffen besonders geschützt werden. Da die Daten in einer potentiell angreifbaren Umgebung gespeichert werden, muss sichergestellt sein, dass nur berechtigte Personen für vorher definierte Zwecke Daten abfragen dürfen. Durch Verschlüsselung kann der Zugriff eingeschränkt werden, sodass nicht einmal der Cloud-Anbieter die Daten im Klartext lesen kann. Ferner lassen sich mittels Datenschutzpräferenzen für jedes Individuum in der Datenbank Zugriffszwecke speichern und validieren. Es gibt bereits viele Ansätze, die Lösungen für dieses Szenario vorschlagen, z.B. [Ne15] und [BL08].

Eine Bereichsanfrage auf verschlüsselte Attributwerte unter Beachtung aller individuellen Datenschutzpräferenzen wird bisher von keinem Datenbanksystem zufriedenstellen bearbeitet. In unserem Ansatz verzichten wir auf oft benutzte „Tricks“, wie die Verwendung von

[1] Humboldt-Universität zu Berlin, Institut für Informatik, janusz@informatik.hu-berlin.de
[2] Humboldt-Universität zu Berlin, Institut für Informatik, taeschnj@informatik.hu-berlin.de

homomorpher oder deterministischer Verschlüsselung, die ineffizient ist bzw. Angriffe auf die Kryptographie zulässt. Stattdessen benutzen wir zwei getrennte Komponenten für die beiden Aufgaben: einen verschlüsselten Index und eine Anfragebearbeitung, die tupelweise die individuellen Präferenzen für ein angefragtes Attribut auswertet. Der verschlüsselte Index basiert auf einem klassischen B-Baum, der effiziente Algorithmen zur Berechnung des Anfrageergebnisses benutzt. Die Daten bleiben hierbei jederzeit verschlüsselt. Vor der Rückgabe des Ergebnisses muss tupelweise die Einhaltung der Präferenzen sichergestellt werden, die von den Individuen für die Verwendung ihrer Daten angegeben wurden.

Das in dieser Arbeit vorgestellte Anfragebearbeitungssystem wurde prototypisch implementiert. Es bettet sich in eine größere Plattform zur sicheren und datenschutzkonformen Benutzerverwaltung als Online-Service [JTD16] ein. Bei der Implementierung des letzten Teils der Plattform zeigte sich, dass es keine zufriedenstellende Lösung für Bereichsanfragen auf ein einzelnes Attribut in dem angestrebte Szenario gibt. Den dabei entwickelten Ansatz stellt dieser Beitrag vor. Der Überblicksbeitrag aus dem Datenbankspektrum 02/2016 [JTD16] enthält als Lösung bisher noch einen Ansatz, der auf einem Kryptoprozessor basiert, was das Gesamtsystem jedoch teurer und plattformabhängig macht.

Im nachfolgenden Abschnitt 2 werden wir das genaue Anfrageszenario beschreiben und die dabei auftretenden Herausforderungen verdeutlichen. Ein Literaturüberblick in Abschnitt 3 wird erörtern, warum sich bisherige Ansätze nicht für das Szenario eignen. Danach präsentieren wir in den Abschnitten 4 und 5 unseren Lösungsansatz für die beiden Hauptherausforderungen. Der vorletzte Abschnitt 6 stellt das Gesamtsystem vor, wobei insbesondere auf das Zusammenspiel der zuvor vorgeschlagenen Einzelkomponenten eingegangen wird. Der Beitrag endet mit einer Zusammenfassung und einem Ausblick auf offene Fragestellungen.

2 Anfrageszenario

Wenn Nutzer eines Online-Dienstes sich neu registrieren, geben Sie in der Regel personenbezogene Daten wie z.B. Name oder E-Mailadresse an. Wird der Onlinedienst in einem Cloud-System betrieben, wie es z.B. von Amazon AWS angeboten wird, befinden sich die Nutzerdaten anschließend in der dort laufenden Datenbank. An dieser Stelle hat nicht nur der Service-Anbieter Zugriff auf die Daten, sondern auch der Cloud-Anbieter. Den Nutzern eines Online-Dienstes ist selten ersichtlich, wo seine Daten letztlich gespeichert werden. An dieser Stelle wäre es wünschenswert, dass die Daten vor der Speicherung so verschlüsselt werden, dass der Cloud-Anbieter die Daten nicht mehr im Klartext lesen kann. Das Passwort zum Entschlüsseln der Daten sollte nur durch den Onlinedienst benutzt werden. Dadurch bleiben die Daten selbst bei potentiellen unberechtigten Zugriffen durch Hacker oder auch Administratoren vor einer Kompromittierung geschützt.

Über die Verwendung und Weitergabe von personenbezogenen Daten erteilen Online-Dienste in ihrer Datenschutzerklärung Auskunft. Idealerweise sollten Benutzer jeden Verwendungszweck und jeder Weitergabe eines Datums feingranular zustimmen oder ablehnen können. Diese individuellen Datenschutzpräferenzen können zusammen mit den Daten

in Form von *Sticky Policies* [PM11] gespeichert werden. Somit kann bei jedem Zugriff auf ein Datum die Einhaltung der Präferenzen für die geplante Verwendung überprüft werden. Ganz allgemein sollten personenbezogene Daten nach den Prinzipien der *Hippocratic Databases* [Ag02] verwaltet werden. Einen Überblick über daraus entstandene konkrete Anforderungen und Ansätze findet sich in [BBL05]. Im Datenbankspektrum 02/2016 [JTD16] stellen wir eine Nutzerdatenverwaltung vor, welche versucht diesen Anforderungen gerecht zu werden, indem z.B. Nutzerdaten verschlüsselt und zusammen mit Nutzer-Präferenzen speichert werden. Für einen Nutzer, der einen Online-Dienst mit dieser Nutzerdatenverwaltung verwenden möchte, können seine verschlüsselten Daten über eine anonyme Tupel-ID aus der Datenbank abgefragt werden. Die Tupel-ID für einen Nutzer bekommt der Online-Dienst bei die Anmeldung des Nutzers übermittelt.

Der Online-Dienst kann in der Nutzerdatenverwaltung unter Angabe eines Zweckes einfache Punktanfragen auf einzelne verschlüsselte Tupel stellen oder alternativ die komplette verschlüsselte Tabelle abfragen. Hierbei wird immer sichergestellt, dass für jedes angefragte Tupelattribut eine Präferenzenauswertung bzgl. des angegebenen Zweckes stattfindet. Durch die Angabe eines Zwecks in einer Anfrage kann der Online-Dienst sich selbst absichern, dass die in der Antwort enthaltenen Nutzerdaten für den beabsichtigten Verwendungszweck wirklich verwendet werden dürfen. Es geht hierbei nicht darum den Online-Dienst am Datenzugriff allgemein zu hindern – diese Erlaubnis hat er vom Nutzer bereits vor der Datenspeicherung bekommen –, sondern es wird lediglich eine zweckgebundene Verwendung und Weitergabe der Nutzerdaten sichergestellt.

Für viele Prozesse eines Onlinedienstes genügen diese Anfragemöglichkeiten bereits, z.B. eine Bestellung durchführen. Sollen die Daten jedoch effizient ausgewertet werden, z.B. Analysen über einen Teil der Nutzerbasis, werden weitere Anfragetechniken benötigt. Insbesondere Bereichtsanfragen auf Attributwerte werden dabei benötigt. Da die Werte in den Tupel-Attributen jedoch verschlüsselt sind, werden dafür besondere Verfahren benötigt.

3 Private Database Outsourcing

Es gibt viele Ansätze, die den Zugriff auf sensible Daten verhindern und es trotzdem erlauben eine Datenbank in ein Cloud-System auszulagern. In dieser Abschnitt werden Techniken diskutiert, die Daten innerhalb einer Datenbank von unberechtigten Zugriff schützen können und dennoch effiziente Bereichsanfragen ermöglichen.

Die existierenden kryptographischen Verfahren bieten unterschiedliche Garantien. Sie lassen sich in Kategorien einteilen:

- **Homomorphe Verschlüsselung**: Diese Verfahren erlauben es Funktionen auf den verschlüsselten Daten zu berechnen. Bisher existiert allerdings keine *vollhomomorphe Verschlüsselung*, mit der beliebige Funktionen berechenbar sind.
- **Durchsuchbare Verschlüsselung**: Durch diese Verfahren können verschlüsselte Dokumente nach einem Auftreten vor Schlüsselwörtern durchsucht werden.

- **Geordnete Verschlüsselung**: Eine Menge von verschlüsselten Werten besitzt bei diesen Verfahren eine Reihenfolge mit der Eigenschaft: Wenn $x \leq y$, dann gilt $Enc(x) \leq Enc(y)$.
- **Deterministische Verschlüsselung**: Hierbei wird ein Wert bei der Verwendung eines festen Schlüssels immer in den gleichen Ciphertext überführt.
- **Probabilistische Verschlüsselung**: Hierbei wird ein Wert bei der Verwendung eines festen Schlüssels immer in den neuen Ciphertext überführt.

Die Verschlüsselungsverfahren aus den genannten Kategorien werden in diverse Arbeiten in der Datenbankliteratur benutzt. TrustedDB [BS11] und Cipherbase [Ar13] erlauben SQL-Anfragen an die probabilistisch-verschlüsselte Datenbank. Sie bieten den vollen Umfang von Anfragemöglichkeiten einer typischen SQL-Datenbank an. Beide Systeme greifen dabei auf einen Kryptoprozessor (engl. secure co-processor) zurück, der die verschlüsselten Daten in der Cloud auswertet. Dieses Verfahren hat mindesten zwei entscheidende Nachteile: Einerseits müssen alle Nutzer darauf vertrauen, dass der Kryptoprozessor bei der Verarbeitung ihre sensiblen Daten keine Fehler macht und nicht geknackt werden kann. Andererseits lässt sich schwer abschätzen wie gut die Benutzung solcher Prozessoren bei einer steigenden Nutzeranzahl und Datenmenge skaliert. Dies dürfte außerdem auch die Kosten der Datenhaltung erheblich steigern. CryptDB [Po11] verzichtet auf zusätzliche Hardware und verwendet alternativ ein zwiebelartiges Verschlüsselungssystem. Die Entwickler setzen auf eine Reihe unterschiedlicher Verschlüsselungsverfahren unter Ausnutzung ihrer Stärken und Schwächen. Die äußeren Hüllen der Zwiebel beruhen auf besonders sicheren kryptographischen Verfahren. Während die inneren Schichten bestimmte Rückschlüsse und damit Auswertungsmöglichkeiten zulassen, z.B. eine Ordnung auf den Daten. Die Zwiebelhüllen werden jeweils nur soweit entschlüsselt, wie es für die Anfrage notwendig ist. Durch diese Entschlüsselung in der Cloud werden die Daten jedoch angreifbar.

Eine weitere Klasse von Anfragetechniken versucht völlig ohne eine Entschlüsselung der Daten auszukommen. Bisher gibt es allerdings kein Verfahren, das die vollständige SQL-Anfragevielfalt auf probabilistisch-verschlüsselten Daten bietet. Effiziente Algorithmen existieren nur für Teilbereiche. Zuerst entstanden Forschungsansätze für eine Schlüsselwortsuche [Sh05]. Hierzu gibt es effiziente Verfahren, die eine genaue Übereinstimmung des gesuchten Schlüsselwortes mit einem verschlüsselten Attributwert identifizieren können [YZW06]. Um jedoch Bereichsanfragen ausführen zu können, müssen alle bisher existierenden Verfahren einige „Tricks“ anwenden. Wie bei CryptDB werden die Daten mit nicht-probabilistischen kryptographischen Verfahren verschlüsselt, die Rückschlüsse auf die Daten zulassen. Die meisten dieser Verfahren sind effizient, lassen sich aber mittels statistischer Analyse oder der Auswertung von Zugriffsmustern angreifen. Alternativ kann ein Angriff vermieden werden, wenn dafür die Effizienz verringert wird. Beispielsweise kann die von Rivest et al. [RAD78] vorgeschlagene homomorphe Verschlüsselung Berechnung eines Schaltkreises durchführen und damit jedes Computerprogramm nachbilden. Jedoch sind die Implementierungen ([SWP00] [Ge09]) bisher extrem rechenaufwendig und nicht praktikabel.

In der Regel benutzen Datenbanken für die effiziente Berechnung von Bereichsanfragen einen Index. Index-basierte Anfragebearbeitung wurden für probabilistisch-verschlüsselte Daten adaptiert [Sh05]. Durch *Private Indexing* [Go03] kann ein nicht-vertrauenswürdiger Server zur Berechnung einer Bereichsanfrage benutzt werden. Wang et al. [WAE11] präsentieren in ihrer Arbeit einen sicheren B+-Baum für eine effiziente Bearbeitung von Datenbankanfragen. Verfahren, die auf verschlüsselten Indizes basieren, benötigen keine Spezialhardware oder vertrauenswürdige Dritte und bieten somit einen geeigneten Ansatzpunkt für den in dieser Arbeit beabsichtigten Einsatzbereich. Jedoch kann der Index nur auf Basis der unverschlüsselten Daten erstellt werden. Mittels durchsuchbarer Verschlüsselung können Indizes entworfen werden, die auch verschlüsselte Werte in den Index einfügen können. Neuhaus et al. [Ne15] haben die für einen Einsatz in Cloud-Systemen optimierte Datenbank MongoDB mit einem solchen Index für eine effiziente Stichwortsuche erweitert. Obwohl alle genannten Verfahren die Daten probabilistisch verschlüsseln, werden jedoch die Metadaten in Form der entworfenen Indizes nicht probabilistisch verschlüsselt. Es bleibt also auch hier ein potentieller Angriffsvektor übrig, durch den die Nutzer-Daten kompromittiert werden könnten. Sollen alle Daten probabilistisch verschlüsselt werden, kann keines der existierenden Anfrageverfahren benutzt werden.

4 Bereichsanfragen auf verschlüsselte Daten

Es gibt kein effizientes Verfahren, um Bereichsanfragen auf probabilistisch verschlüsselte Daten innerhalb einer Clouddatenbank zu bearbeiten. Allerdings kann durch eine alternative Anfragebearbeitung das Anfrageszenario mit cloud-seitig gespeicherten verschlüsselten Daten dennoch ausgeführt werden. Die vorgestellten index-basierten Verfahren wären effizient und sicher, wenn auch der Index probabilistisch verschlüsselte wäre. Jedoch kann dadurch ein Cloud-Server den Index nicht mehr auswerten. Im Falle eines B-Baumes kann insbesondere die Entscheidung, in welchem Kindknoten sich der Suchschlüssel befindet, nicht mehr getroffen werden. Wenn genau diese Entscheidung an den Anfragenden delegiert würde, wäre das Problem gelöst und die Bereichsanfrage könnte beantwortet werden. Der Anfragenden muss den Schlüssel besitzen, da er sonst die Antwort nicht entschlüsseln könnte. Also kann er auch die an ihn delegierte Entscheidung berechnen. Dabei kann der Index nicht kompromittiert werden, weil der Anfragende grundsätzlich die verschlüsselten Daten zugreifen darf. *ZeroDB* [EW16] implementiert genau den skizzierten Ansatz. In Abbildung 2 wird der Suchablauf beispielhaft für den in Abbildung 1 dargestellten B-Baum für den Wert „11“ illustriert.

Durch die Auslagerung von Teilschritten aus der server-seitigen Datenbank an den anfragenden Client wird der Angriffsvektor auf den Index beseitigt. Dies geschieht auf Kosten der Anfragekomplexität, die um teure Kommunikationsschritte erweitert werden muss. Da B-Bäume in der Regel aber keine große Höhe besitzen, sind in der Praxis nur wenige Kommunikationsrunden (Anzahl = Höhe) notwendig. Die sich daraus ergebene erhöhte Latenz der Anfrage bleibt selbst bei einem B-Baum mit 1 Millionen Einträgen unter einer Sekunde [EW16]. Für einen Online-Dienst bleibt die Antwortzeit auf eine Anfrage damit in einem akzeptablen Rahmen.

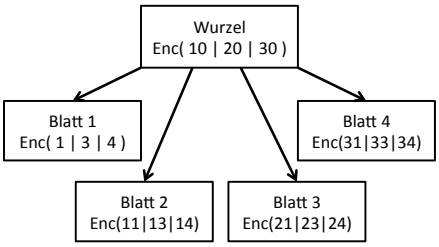


Abb. 1: Beispiel für einen verschlüsselten B-Baum

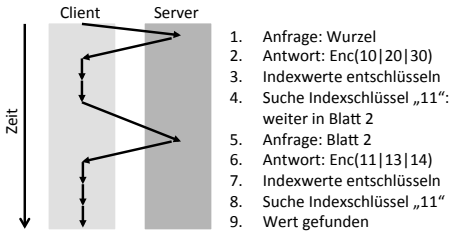


Abb. 2: Indexsuche nach Wert 11 (vgl. [EW16])

5 Tupelweise Auswertung von Nutzer-Präferenzen

ZeroDB bietet keine Möglichkeit zur Verarbeitung von Nutzer-Präferenzen. Bei jedem Datenzugriff soll jedoch ein Zweck zusammen mit der Anfrage angegeben und mit den Präferenzen abgeglichen werden. In [JTD16] wird gezeigt, wie Nutzer-Präferenzen modelliert und gespeichert werden können. Bei Datenbankanfragen steht jedoch die Auswertung von Präferenzen bzgl. der Verwendung einzelner Attribute im Vordergrund. In diesem Abschnitt liegt der Fokus daher auf der Speicherung und Auswertung von Präferenzen für einzelne Nutzerdaten eines Online-Dienstes.

Häufig besitzen Zwecke in Unternehmensabläufen eine hierarchische Abhängigkeit. Damit ein Zweck erreicht werden kann, müssen dafür entweder ein einzelner oder mehrere andere Zwecke erreicht werden. Die Zweckhierarchie eines Online-Dienstes wird durch einen Baum angegeben, wobei jeder Knoten einen Zweck des Online-Dienstes repräsentiert und jede Kante einen übergeordneten Hauptzweck und einen untergeordneten Teilzweck verbindet [BL08]. Abb. 3 zeigt eine beispielhafte Zweck-Hierarchie eines Reiseportals. Darin kann beispielsweise der Zweck *Zusendung von Informationen* per Post oder per E-Mail realisiert werden. Die Überprüfung der Nutzer-Präferenzen für jedes Tupel in einer Antwortmenge einer Datenbankanfrage erhöht den Umfang der Anfragebearbeitung. Der Aufwand zum Zeitpunkt der Anfrage kann reduziert werden, indem die Auswertung vorberechnet wird. In der Tabelle mit den verschlüsselten Werten wird hierfür zu jedem

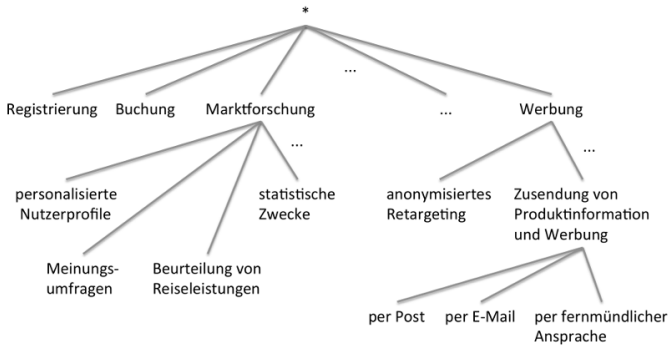


Abb. 3: Zweck-Hierarchien eines Reiseportals

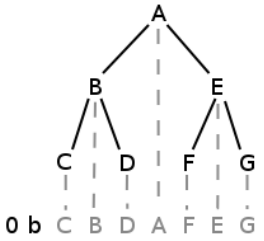


Abb. 4: Transformation einer Zweck-Hierarchien

purpose	code
A	0b0001000
B	0b0100000
C	0b1000000
D	0b0010000
E	0b0000010
F	0b0000100
G	0b0000001

Abb. 5: Zuordnung von Zwecken zu Bitfolgen

Attribut ein weiteres Feld hinzugefügt, welches die von dem jeweiligen Nutzer erlaubten Zugriffszwecke speichert. Zur schnelleren Auswertung dieser Zugriffszwecke, werden sie in Form von Bitfolgen gespeichert.

Die Darstellung der erlaubten Zwecke kann auf eine Bitfolge reduziert werden [BL08], sofern die Menge der Zwecke für eine Tabelle fixiert wurde. In Abb. 4 wird das Konzept dargestellt. Dabei wird jeder Knoten der Hierarchie mit einem Bit assoziiert (Position im String). Es kann für jeden Zweck die Frage beantwortet werden, ob ein Dienst zu genau diesem Zweck auf ein Datum zugreifen darf, daraus ergibt sich für das zugehörige Bit eine 1 (Zugriff gestattet) oder eine 0 (Zugriff verboten). In Abbildung 5 werden Bitfolgen für erlaubte Einzelzwecke dargestellt. Diese lassen sich durch den bitweisen *UND*-Operator leicht kombinierten.

Bei der Anfrage eines Dienstes kann nun für jedes angefragte Attribut eine Bitfolge der gewünschten Zwecke – mithilfe der Zuordnung: Zweck $\rightarrow$ Bit (vgl. Abbildung 4) – ermittelt werden und gegen die gespeicherte Bitfolge verglichen werden. Die Zugriffs-Bitfolge muss dabei komplett in der Attribut-Bitfolge des Wertes „enthalten“ sein. Dazu muss gelten:

$$\textit{Zugriffs-Bitfolge} \;\&\; \textit{Attribut-Bitfolge} = \textit{Zugriffs-Bitfolge}.$$

Hierbei bezeichnet & den bitweisen *UND*-Operator. Diese Eingenschaft lässt sich leicht in der WHERE-Klausel einer SQL-Anfrage formulieren, dadurch kann eine Datenbank effizient die Tupel aus einer Antwort filtern, die mit dem übergebenen Anfragezweck nicht vereinbar sind. Abbildung 6 zeigt ein Beispiel für eine erfolgreiche Anfrage, während die

	Abb. 6: Erfolgsbeispiel	Abb. 7: Fehlerbeispiel 1	Abb. 8: Fehlerbeispiel 2
Zugriffs-Bitfolge	00001010	00001010	00001010
Attribut-Bitfolge	00011110	00010100	00011100
AND	00001010	00000000	00001000
AND=Anfr.	true	false	false

Abbildungen 7 und 8 Beispiele zeigen, bei denen der Zugriff aufgrund des Anfragezwecks fehlschlägt.

Die Bitfolgen zu jedem Attribut eines Tupels können zu einem beliebigen Zeitpunkt vor einer Anfrage berechnet werden – im besten Falle zum Zeitpunkt des Einfügens in die Datenbank – was den Aufwand zum Anfragezeitpunkt stark reduziert. Bei diesem Vorgehen gibt es zwei Nachteile. Um sicherzustellen, dass die Bitfolgen stets aktuell sind, müssen sie bei jeder Änderung von Präferenzen oder der Datenschutzerklärung des Onlinedienstes aktualisiert werden. Dabei ist eine Änderung der Präferenzen von geringem Aufwand begleitet, hier müssen nur die Bitfolgen der Einträge des Individuums geändert werden. Eine Änderung der Datenschutzerklärung hingegen zieht eine Aktualisierung aller Bitfolgen der Tabelle nach sich. Bei sich häufig ändernden Datenschutzerklärungen wäre das Verwenden der Bitfolgen daher kritisch zu betrachten.

Die Anfragebearbeitung mittels Bitfolgen wurde bereits in [BL08] ausführlich evaluiert, daher verweisen wir für eine Übersicht auf die dort gewonnenen Erkenntnisse. Zusammenfassend können wir festhalten, dass sich mit dem gewählten Ansatz Nutzer-Präferenzen effizient auswerten lassen und dass dabei der Zusatzaufwand in einem akzeptablen Rahmen bleibt.

6 Übersicht der Anfragebearbeitung

Dieser Abschnitt beschreibt das Gesamtsystem. Insbesondere wird dabei auf das Zusammenspiel der zuvor vorgeschlagenen Komponenten bei der Beantwortung von Bereichsanfragen auf Attributwerte eingegangen. In Abschnitt 2 wurde bereits festgestellt, dass bei zweckgebundenen Punktanfragen auf einzelne verschlüsselte Tupel mittels Tupel-ID die Nutzerpräferenzen leicht überprüft werden können. Durch die Angabe einer Menge an Tupel-IDs bei einer Datenbankanfrage können daher ebenfalls für die resultierende Antwort tupelweise die Nutzerpräferenzen ausgewertet werden. Wird dabei die im letzten Abschnitt vorgestellte Speicherung von Bitfolgen verwendet, entsteht dadurch nur ein geringer zeitlicher Zusatzaufwand.

Bei einer Bereichsanfrage wird auch eine Menge von Tupeln ausgewählt. Die Herausforderung bei verschlüsselten Daten liegt darin, zu entscheiden welche Tupel in dem gesuchten Bereich liegen. Für diese Aufgabe kann der Index aus der ZeroDB verwendet werden. Wenn ein neues Tupel mit der ID t_{neu} samt Präferenzen für Attributnutzungen in die Nutzerdatenbank geschrieben wird, speichert der verschlüsselte Index gleichzeitig den Wert für das zu indizierende Attribut aus dem Tupel zusammen mit t_{neu} ab. Der so erstellte Index kann nun benutzt werden, um für einen gesuchten Bereich alle Tupel-IDs zu finden. Für diese Art der Datenspeicherung werden zwei Komponenten benötigt.

Abbildung 9 illustriert eine für Bereichsanfragen geeignete Systemarchitektur. Die ZeroDB enthält nur die Indizes für vorab festgelegte Attribute. Die zweite Komponente speichert die Nutzerdaten und -Präferenzen. Sie basiert auf einer angepassten PostgreSQL-Datenbank. Auf der Client-Seite läuft ein spezieller Treiber, der sich um die Anfrageadaption sowie um die benötigten Einfüge-, Aktualisierungs- und Lösch-Operationen in der jeweiligen Daten-

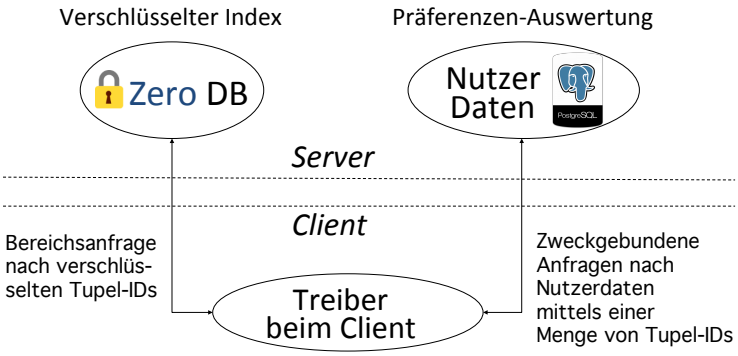


Abb. 9: Systemarchitektur für Anfragen an verschlüsselte Nutzerdaten in der Cloud

bank kümmert. Eine Bereichsanfrage auf ein Attribut wird in zwei Schritten ausgeführt. Zuerst stellt der Treiber eine Anfrage an die ZeroDB, mit der alle Tupel-IDs aus diesem Bereich gefunden werden. Im zweiten Schritt werden aus der PostgreSQL-Datenbank die entsprechenden Tupel angefragt. Die Tupel in der Antwortmenge enthalten jedoch nur Daten, für die ein zweckbezogener Zugriff durch entsprechende Nutzer-Präferenzen autorisiert wurde.

7 Zusammenfassung und Ausblick

Als Ergebnis dieser Arbeit steht die Erkenntnis: Die Aufgabe Bereichsanfragen auf verschlüsselte Daten in unsicheren Cloud-Datenbanken ausführen zu können, lässt sich durch die geschickte Kombination eines verschlüsselten Indexes in Form eines B-Baums in der ZeroDB und eines effizienten Verfahrens zur Auswertung von durch Bitfolgen kodierte Nutzerpräferenzen bewerkstelligen. Dabei kann zu keinem Zeitpunkt der Klartextwert eines Nutzerattributes in der Cloud gelesen werden. Im Gegensatz zu den meisten bisher existierenden Verfahren, die auf schwächere kryptographische Algorithmen zurückgreifen, bleiben die Nutzerdaten als auch der Index mit dem präsentierten Ansatz immer vollständig probabilistisch verschlüsselt.

Wir haben uns in dieser Arbeit darauf beschränkt das System zu motivieren und einzuführen. Daher fehlen Performanzmessungen, die genauer untersuchen, wie gut das System bei einer großen Anzahl von Nutzern funktioniert. Für bis zu 10000 Datensätzen ließ sich in unseren bisherigen Experimenten praktikable Anfragezeiten nachweisen. Eine genaue Performanzanalyse wird in einer zukünftigen Arbeit erfolgen.

Als weitere verbleibende Aufgabe müssten neben den Nutzerdaten auch die gespeicherten Bitfolgen für die Nutzer-Präferenzen verschlüsselt gespeichert und ausgewertet werden. Diese Informationen lassen zwar keine direkten Rückschlüsse auf die verschlüsselten Nutzerdaten zu, allerdings könnten einige Nutzer diese Metadaten dennoch als sensibel

einstufen. Da es für den bitweisen *UND*-Operator jedoch effiziente Algorithmen für homomorphe Verschlüsselung gibt, kann das hier vorgeschlagen Verfahren zukünftig dafür umgestellt werden.

Literatur

[Ag02] Agrawal, Rakesh; Kiernan, Jerry; Srikant, Ramakrishnan; Xu, Yirong: Hippocratic Databases. In: Proceedings of the 28th International Conference on Very Large Data Bases. VLDB '02. VLDB Endowment, S. 143–154, 2002.

[Ar13] Arasu, Arvind; Blanas, Spyros; Eguro, Ken; Joglekar, Manas; Kaushik, Raghav; Kossmann, Donald; Ramamurthy, Ravi; Upadhyaya, Prasang; Venkatesan, Ramarathnam: Secure Database-as-a-service with Cipherbase. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. SIGMOD '13, ACM, New York, USA, S. 1033–1036, 2013.

[BBL05] Bertino, Elisa; Byun, Ji-Won; Li, Ninghui: Privacy-Preserving Database Systems. In (Aldini, Alessandro; Gorrieri, Roberto; Martinelli, Fabio, Hrsg.): Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 178–206, 2005.

[BL08] Byun, Ji-Won; Li, Ninghui: Purpose based access control for privacy protection in relational database systems. VLDB J., 17(4):603–619, 2008.

[BS11] Bajaj, Sumeet; Sion, Radu: TrustedDB: A Trusted Hardware Based Database with Privacy and Data Confidentiality. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data. SIGMOD '11, ACM, New York, USA, S. 205–216, 2011.

[EW16] Egorov, Michael; Wilkison, MacLane: ZeroDB white paper. CoRR, 2016.

[Ge09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st annual ACM symposium on Theory of Computing. ACM, S. 169–178, 2009.

[Go03] Goh, Eu-Jin: Secure Indexes. IACR Cryptology ePrint Archive, 2003:216, 2003.

[JTD16] Janusz, Daniel; Taeschner, Jochen; Dimitrov, Dimitar: Sichere und datenschutzkonforme Benutzerverwaltung als Online-Service. Datenbank-Spektrum, 16(2):157–166, 2016.

[Ne15] Neuhaus, Christian; Feinbube, Frank; Janusz, Daniel; Polze, Andreas: Secure Keyword Search over Data Archives in the Cloud - Performance and Security Aspects of Searchable Encryption. In: Proceedings of the 5th International Conference on Cloud Computing and Services Science. S. 427–438, 2015.

[PM11] Pearson, S.; Mont, M. C.: Sticky Policies: An Approach for Managing Privacy across Multiple Parties. Computer 09/11, 44:60–68, 2011.

[Po11] Popa, Raluca Ada; Redfield, Catherine M. S.; Zeldovich, Nickolai; Balakrishnan, Hari: CryptDB: Protecting Confidentiality with Encrypted Query Processing. In: Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles. SOSP '11, ACM, New York, NY, USA, S. 85–100, 2011.

[RAD78] Rivest, R. L.; Adleman, L.; Dertouzos, M. L.: On data banks and privacy homomorphisms. Foundations of secure computation, 32(4):169–178, 1978.

[Sh05] Shmueli, Erez; Waisenberg, Ronen; Elovici, Yuval; Gudes, Ehud: Designing Secure Indexes for Encrypted Databases. In: Proceedings of the 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security. DBSec'05, Springer-Verlag, Berlin, Heidelberg, S. 54–68, 2005.

[SWP00] Song, D. X.; Wagner, D.; Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of 2000 IEEE Symposium on Security and Privacy. S &P 2000, S. 44–55, 2000.

[WAE11] Wang, Shiyuan; Agrawal, Divyakant; El Abbadi, Amr: A Comprehensive Framework for Secure Query Processing on Relational Data in the Cloud. In: Secure Data Management - 8th VLDB Workshop, SDM 2011, Seattle, WA, USA, September 2, 2011, Proceedings. S. 52–69, 2011.

[YZW06] Yang, Zhiqiang; Zhong, Sheng; Wright, Rebecca N.: Privacy-Preserving Queries on Encrypted Data. In (Gollmann, Dieter; Meier, Jan; Sabelfeld, Andrei, Hrsg.): Computer Security – ESORICS 2006: 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006. Proceedings. Springer Berlin Heidelberg, Berlin, Heidelberg, S. 479–495, 2006.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 249

Survey and Comparison of Open Source Time Series Databases

Andreas Bader[1] Oliver Kopp[2] Michael Falkenthal[3]

Abstract: Time series data, i.e., data consisting of a series of timestamps and corresponding values, is a special type of data occurring in settings such as "Smart Grids". Extended analysis techniques called for a new type of databases: Time Series Databases (TSDBs), which are specialized for storing and querying time series data. In this work, we aim for a complete list of all available TSDBs and a feature list of popular open source TSDBs. The systematic search resulted in 83 TSDBs. The twelve most prominent found open source TSDBs are compared. Therefore, 27 criteria in six groups are defined: (i) Distribution/Clusterability, (ii) Functions, (iii) Tags, Continuous Calculation, and Long-term Storage, (iv) Granularity, (v) Interfaces and Extensibility, (vi) Support and License.

Keywords: Time Series Databases, Survey, Comparison

1 Introduction and Background

The importance of sensors has been growing in the last years. Thereby, IoT technologies gained access to industrial environments to enable intensive metering of production steps, whole manufacturing processes, and further parameters. One key challenge of these endeavors is to efficiently store and analyze huge sets of metering data from many different sensors, which are typically present in the form of time series data. These principles are currently also applied to energy grids, where increasing amounts of dynamic and flexible power generation units, such as solar panels and thermal power stations, respectively, along with energy storages, such as batteries or pumped storage hydro power stations, require to intensively meter the parameters of the energy grid [Ko15]. At this, metered data is analyzed to smartly control the energy grid with respect to balancing volatile amounts of generated and consumed electricity, which also results in changes of the existing energy market [BF12]. To conceive the volume of time series data that may be collected in such scenarios, the equipping of a smaller city with smart meters can lead up to 200,000 million voltage measurements each month—which all have to be stored and processed [St15]. Real-world metering is also done in the context of our BMWi-funded project NEMAR (03ET4018), where Time Series Databases (TSDBs) play a crucial role to develop a Decentralized Market Agent (DMA) [Th15]. The goal of a DMA is to form a new role in the energy market to gain a global price and energy transfer optimum. This should be done efficiently, but also the TCO for the infrastructure should reasonable.

To handle such amounts of data, it is necessary to scale the Database Management System (DBMS) accordingly, which means distribution across several nodes with the possibility to

[1] University of Stuttgart, IPVS, Universitätsstr. 38, Stuttgart, Germany, andreas.bader@ipvs.uni-stuttgart.de
[2] University of Stuttgart, IPVS, Universitätsstr. 38, Stuttgart, Germany, oliver.kopp@ipvs.uni-stuttgart.de
[3] University of Stuttgart, IAAS, Universitätsstr. 38, Stuttgart, Germany, michael.falkenthal@iaas.uni-stuttgart.de

increase further when the city grows is inevitable. There are different views on the question if traditional Relational Database Management Systems (RDBMS) can handle such amounts of data. On the one hand, distributing writes in a relational model, whilst keeping full consistency and the same level of query latencies, is not easily possible [PA13]. On the other hand, VividCortex, a SaaS platform for database monitoring, uses MySQL Community Server [Or16a], a traditional RDBMS, in combination with InnoDB as storage subsystem to store 332,000 values per second while running on three Amazon Web Services (AWS) Elastic Compute Cloud (EC2) servers [Vi14]. Most of the used queries are Insertions [Sc14] and basic SQL queries like SUM queries [Sc15]. To ingest into MySQL with that ingest rate, several trade-offs are required [Sc15]: 1) Performing batch-wise ingestion into vectors[4] that are stored as delta values and grouped by a clustered primary key[5], 2) due to the clustered primary key, performing ad-hoc SQL queries is not possible, instead a time series service had to be used for querying, 3) while building the cluster, it must be decided and manually implemented how data is sharded, grouped, and how indexes are created. Scalable RDBMS which use small-scope operations and transactions are trying to achieve better scalability and performance than traditional RDBMS [Ca11]. This shows that, with enough effort and by considering the specific use case at hand, RDBMS can be used for storing time series data. However, it is not clear if this is still possible if the queries are more complex or algorithmic.

NoSQL DBMS provide a solution with more possibilities for distribution by weakening relations and consistencies [Gr13]. Knowing that sensor data has a timestamp attached to it, a specific type of NoSQL DBMS for that type of data arose: Time Series Databases (TSDBs). This new type provides the scalability to store huge amounts of time series data, ingested with an high ingest rate, and perform SCAN queries on it [Te14]. The boundaries between NoSQL DBMS and TSDBs are fluent, as there exists no precisely defined boundary between them.

However, since a vast amount of TSDB solutions have emerged in the last years, a clear overview is missing as of today. Especially the fact that TSDBs (i) are capable of diverse functionalities regarding operations on time series data, while (ii) building upon completely different technologies motivates to establish a set of criteria in order to make them objectively comparable. Such criteria can be used for architectural decisions, which may lead to other results than a plain performance comparison [Zi15].

Therefore, the contribution of this paper is threefold: (i) the main contribution is to present a survey and comparison on existing TSDB solutions, which (ii) builds upon a comprehensive list of found TSDBs, and (iii) a set of measurable comparison criteria. This survey presents a popularity ranking of all found TSDBs whereas the feature comparison regards the subset consisting of popular open source TSDBs and can be used as a basis for deciding which open source TSDBs to select for specific use cases at hand.

The remainder of this paper is structured as follows: Sect. 2 defines what a TSDB is and how TSDBs can be grouped. Sect. 3 presents related work. Sect. 4 defines the comparison criteria. Sect. 5 presents the search process and the found TSDBs. Sect. 6 introduces the

[4] Vectors help to hold multiple values per row, so that there is not one row for each second (using a distance of one second between values) [Sc15].

[5] A clustered primary key of VividCortex consists of the host name, the time series name, and a timestamp [Sc15].

chosen TSDBs more detailed. The main contribution of this paper, the feature comparison, is done in Sect. 7. Sect. 8 concludes and describes future work.

2 Definitions

For comparing TSDBs, it is necessary to define which databases belong to this group as the term "TSDB" is not consistently defined or used. Traditionally, a Database System (DBS) consists of a Database (DB) and a DBMS. While "DB" describes a collection of data, "DBMS" describes the software that is used to define, process, administrate, and analyze the data in a "DB" [Th01]. Most literature uses the term "DBMS" (like RDBMS) instead of "DBS". Therefore TSDB is mostly used in the meaning of "time series database management system". In this paper, DBMS, RDBMS, and TSDB will be used in this non-accurate way.

For this paper, a DBMS that can (i) store a row of data that consists of timestamp, value, and optional tags, (ii) store multiple rows of time series data grouped together (e. g., in a time series), (iii) can query for rows of data, and (iv) can contain a timestamp or a time range in a query is called TSDB. This definition is more precise than defining a TSDB as a database that can contain historical data besides current data [DDL14]. This means that the "time series"-character of the DBMS must inherently exist, it must be possible to do queries by timestamps and time ranges. Storing time series data in a RDBMS can be done in two ways: timestamp-sorted or series-sorted. Depending on the indexes and queries used, this can impact performance of the queries [Sc15].

Queries of TSDBs Nine different queries will be used for comparison and explanation: Insertion (INS), Updating (UPD), Reading (READ), Scanning (SCAN), Averaging (AVG), Summarization (SUM), Counting (CNT), Deletion (DEL), Maximization (MAX), and Minimization (MIN). The resulting data inside a TSDB of one single INS, that was successfully executed, is called row (of time series data). A row of time series data consists of a timestamp, a value (usually floating point with double or single precision), and optional tag names and values. A TSDB can store several time series, whereby each time series consists of a name and several rows of time series data. A time series is used to group rows of time series data together on a higher level than tags, comparable to a table in traditional RDBMS (e. g., "voltage measurements" as name of the time series, whereby the tag names and values define from which sensor in which building the measurement came). A timestamp expresses a specific point in time, usually in milliseconds starting from 1970-01-01. A granularity defines the smallest possible distance between two timestamps that can be stored. The granularity used to store data and the granularity used for querying can be different. A time range describes a period of time between two timestamps and can also be zero. A tag consists of a tag name and a tag value (both usually alphanumeric). A tag can be used to group rows together (e. g., each row consists of sensor values and a tag is named "room" and the tag values are room numbers). Data can be aggregated using the aggregation queries AVG, SUM, CNT, MAX, and MIN. These queries can also be used to group the results together in time ranges, which are also called "buckets" (e. g., querying the maximum value of each day in a year).

The queries are defined as follows: INS is a query that inserts one single row into a DBMS. UPD updates one or more rows with a specific timestamp. READ reads one or more rows

with a specific timestamp or in the smallest time range possible. SCAN reads one or more rows that are in a specific time range. AVG calculates the average over several values in a specific time range. SUM summarizes several values in a specific time range. CNT counts the amount of existing values in a specific time range. DEL deletes one or more rows with a specific timestamp or in a specific time range. MAX and MIN search for the maximum, respective minimum, value of several values in a specific time range.

Grouping of TSDBs Existing TSDBs can be subdivided into four groups. The first group exists of TSDBs that depend on an already existing DBMS (e. g., Cassandra, HBase, CouchDB, MySQL) to store the time series data. If another DBMS is only required to store meta data, the TSDB is not in this group. In the second group are TSDBs that can store time series data completely independent of other DBMS despite using DBMS for storing meta data or other additional information. RDBMS are not in this group, even if they do not require other DBMS. The third group encloses RDBMS that can store time series data. Independent from their requirements on other DBMS, RDBMS cannot be in group 1 or 2. As last group, Proprietary contains all commercially or freely available TSDBs that are not open source. Independent from their requirements on other DBMS and from their type, proprietary DBMS cannot be in group 1, 2, or 3.

3 Related Work

Previous publications are motivated by releases of new TSDBs or the search for a DBMS for processing or storing time series data for a specific scenario. Many of these publications include a performance comparison. The two most prominent performance comparison publication are described here, an overview on other related work is presented by Bader [Ba16]. Lately, an overview over 19 existing TSDBs was published on the internet [Ac16a], which is the latest and completest overview that could be found on existing TSDBs.

Wlodarczyk [Wl12] compares four solutions for storing and processing time series data. The results are that OpenTSDB is the best solution if advanced analysis is needed, and that TempoIQ (formerly TempoDB) can be a better choice if a hosted solution is desired. No benchmark results are provided, only a feature comparison is done.

Deri et al. [DMF12] present tsdb as a compressed TSDB that handles large time series better than three existing solutions. They discovered OpenTSDB as only available open source TSDB, but do not compare it to tsdb. The reason is the architecture of OpenTSDB, which is not suitable for their setup. MySQL, RRDtool, and Redis are compared against tsdb with the result that the append/search performance of tsdb is best out of the four compared DBMS.

Pungilă et al. [PFA09] tried to find a fitting database for a set of households with smart meters. For reaching their goal, they compared three RDBMS (PostgreSQL, MySQL, SQLite3), one TSDB (IBM Informix – community edition with TimeSeries DataBlade module), and three NoSQL DBMS (Oracle BerkeleyDB, Hypertable, MonetDB) in two scenarios. The conclusion is, that if the data and therefore the queries are based on a key like client identifier, sensor identifier, or timestamp, then some DBMS result in increased performance. The performance is logarithmically decreased for these DBMS when using a combination of tags. Two of the compared DBMS are interesting for their scenario,

depending whether the focus lies on INS queries, READ queries, or both. Hypertable is the best choice if the focus lies on the highest rate of executed INS queries in combination with a worse SCAN performance. BerkeleyDB, which has a lower rate of executed INS queries but executes more SCAN queries, is the second best choice.

Acreman [Ac16a] compares 19 TSDBs (DalmatinerDB, InfluxDB, Prometheus, Riak TS, OpenTSDB, KairosDB, Elasticsearch, Druid, Blueflood, Graphite (whisper), Atlas, Chronix Server, Hawkular, Warp 10 (distributed), Heroic, Akumuli, BtrDB, MetricTank, Tgres) including a performance comparison. For measuring performance, a two node setup generated about 2.4 to 3.7 million INS queries per second on average [Ac16b]. As a result, DalmatinerDB can execute two to three times as much INS queries queries than other TSDBs in this comparison, followed by InfluxDB and Prometheus. Although Steven Acreman is the Co-Founder of Dataloop, a company that is connected to DalmatinerDB, he tries to provide as much transparency as possible by releasing the benchmark [Ac16b].

As a summary it can be concluded that most of the existing publications focus on performance comparison rather than doing a feature comparison. Furthermore, the conclusion of the presented work is that there is not one single database that suits every use case. This paper closes this gap by presenting a systematic search for TSDBs resulting in 83 found TSDBs and a feature comparison of 12 found open source TSDBs that are chosen by popularity.

4 Comparison Criteria

The comparison criteria are grouped into six groups that will be explained in the rest of this section. The criteria are derived from requirements on the time series database in the context of the NEMAR project [Th15].

Criteria Group 1: Distribution/Clusterability *High Availability (HA)*, *scalability*, and *load balancing* features are compared in this group. *HA* gives the possibility to compensate unexpected node failures and network partitioning. To compensate means that a query must be answered under the mentioned circumstances, but it is not expected that the given answer is always consistent. It is expected that the DBMS uses eventual consistency at least, but since some time series databases do not give any explicit consistency guarantee or depend on DBMS that have configurable consistency guarantees, consistency levels are not further considered. It is also expected that a client needs to know more than one entry point (e. g., IP address) to compensate a node failure of an entry node. *Scalability* is the ability to increase storage or performance by adding more nodes. The ability must exist in the server part of the TSDB, otherwise adding a second TSDB and giving the client application the possibility to use two TSDBs in parallel for its queries would also result in scalability and a increased performance. *Load balancing* is the possibility to equally distribute queries across nodes in a TSDB, so that the workload of each node has just about the same level. If a TSDB uses another DBMS, it is also fulfilled if only the DBMS or the TSDB or both have these features.

Criteria Group 2: Functions The availability of *INS*, *UPD*, *READ*, *SCAN*, *AVG*, *SUM*, *CNT*, *DEL*, *MAX*, and *MIN* functions is compared in this group. Sect. 2 presents an explanation of these queries. All of these functions are specific to time series. There are

more complex queries like changing the granularity, grouping by time spans, or performing autoregressive integrated moving average (ARIMA) time series analysis, but due to the experiences in the NEMAR project [Th15] and the fact that VividCortex (see Sect. 1) also uses mostly INS queries [Sc14] and other simpler query types (such as SUM queries) in their setup [Sc15], simple query types are considered sufficient for a comparison.

Criteria Group 3: Tags, Continuous Calculation, Long-term Storage, and Matrix Time Series *Continuous calculation*, *tags*, *long-term storage*, and the support of *matrix time series* are compared this group. *Continuous calculation* means that a TSDB, having this feature, can continuously calculate functions based on the input data and stores the results. An example is the calculation of an average per hour. It is also checked if *tags* are available as these are needed to differentiate different sources (e. g., different sensors). A solution for *long-term storage* is needed for huge amounts of data, as it is challenging to store every value in full resolution over a longer period, considering the city from Sect. 1 with 360,000 values per second (on a full rollout) [St15] as an example. Solutions could range from throwing away old data to storing aggregated values of old data (e. g., storing only an average over a minute instead of values for every millisecond). Solutions that are running outside the TSDB are not considered (e. g., a periodic process that runs queries to aggregate and destroy old data). Most TSDBs support time series with one timestamp per value, so called vector time series. If, for example, forecasts (e. g., weather forecast) are stored in a TSDB, time series that support two or more timestamps per value (more than one time dimension) are needed, because a value then has two timestamps (e. g., a timestamp on which the forecast was created and a timestamp for which the forecast was made). These time series are called *matrix time series*. Solutions that can be implemented with other existing functions (e. g., tags) for storing matrix time series are not considered. The available time domain function for the first timestamp must be available for the second timestamp as well.

Criteria Group 4: Granularity *Granularity*, *downsampling*, *the smallest possible granularities* that can be used *for functions and storage*, as well as the *smallest guaranteed granularity for storage*, are compared in this group. When using queries with functions, most TSDBs have the ability to use *downsampling* for fitting the results when a result over a greater period of time is wanted (e. g., an average for every day of a month and not every millisecond). Downsampling does not mean that you can choose a period of time and get one value for that period. For downsampling two periods must be chosen and a value for each smaller period within the bigger period must be returned. The smaller period is called sample interval. *Granularity* describes the smallest possible distance between two timestamps (Sect. 2). When inserting data into a TSDB, the *granularity* of the *input* data can be higher than the *storage* granularity that a TSDB *guarantees* to store safely. Some TSDB accept data in a smaller granularity than they can store under all circumstances, which leads to aggregated or dropped data. For TSDBs that use other DBMS for storing their data, some of the compared aspects can be implemented manually with direct queries to the DBMS. Such solutions are not considered.

Criteria Group 5: Interfaces and Extensibility *Application Programming Interfaces (APIs)*, *interfaces*, *client libraries* that are not third-party, are listed in this group. It is also

compared if an interface for *plugins* exist. *APIs* and *interfaces* are used to connect to a TSDB to execute queries. Interfaces can be non-graphical (e. g., User Interface (UI)) or graphical (e. g., Graphical User Interface (GUI)). APIs (e. g., HTTP (using REST and JSON)) are used by programming languages to connect to a TSDB, execute queries, and retrieve query results. *Client libraries* encapsulate the connection management and the implementation of an API to provide easier access to a TSDB for a specific programming language. *Plugins* are used to extend the capabilities (e. g., to add new functions) of a TSDB. To develop plugins, a specific interface is required.

Criteria Group 6: Support and License The availability of a *stable version (Long Term Support (LTS))* and *commercial support*, as well as the used *license* are compared in this group. *"Stable" versions* are helpful to minimize maintenance time for updating a TSDB by only releasing updates that do not support the newest functionality but instead are considered free of bugs. If a issue in a TSDB is encountered, it needs to be solved by internal or external developers. In a situation where defined reaction times are required, that cannot be achieved internal, or when an internal development team is "stuck", *commercial support* is helpful. Only commercial support from the developer(s) of a TSDB are considered. When developing or using an open source TSDB, it is important with which *license* the TSDB is released. A license regulates how and by whom a product can be used and what modifications are allowed to it, which gives more long term safeness than without having a license.

5 Search for TSDBs

This section presents the procedure taken of searching for TSDBs. This was done by searching on Google, in ACM Digital Library, in IEEE Xplore / Electronic Library Online (IEL). Each result was individually considered and searched for TSDBs. The search terms and result numbers are presented in Tab. 1. The search terms need to include "database", otherwise many results from time series analysis or mining are included.

Search Engine	Search Term	Results
Google	`""time series database" OR "timeseries database" OR "tsdb""`	233,000[6]
ACM Digital Library	`("time series database" "timeseries database" "tsdb")`[7]	59
IEEE Xplore / Electronic Library Online (IEL)	`(((time series database") OR "timeseries database") OR "tsdb")`[8]	59

Tab. 1: Search terms and result numbers for the procedure of searching for TSDBs.

As a result, 83 TSDBs were found, 50 of them are open source solutions and thus 33 are proprietary. The most popular TSDBs are chosen from each of the introduced group. For ranking the found TSDBs by popularity, a Google search for each TSDB was performed. Since most TSDBs do not have any scientific papers, the amount of citations is not usable as a ranking method. Google's PageRank would also be usable for ranking, but then TSDBs

[6] The first 100 results were considered.
[7] "Any field" and "matches any" operators are used.
[8] Advanced search with "Metadata Only" operator is used.

that use more than one homepage (e. g., homepage and GitHub homepage) are ranked lower than others using only one homepage. Therefore, the amount of results found by Google was used, which also represents the amount of discussion (e. g., in news groups) about a specific TSDB in the internet. The search terms were adjusted to fit the results as close as possible to only match entries that are related to the TSDBs, but it is expected that the results are not exactly precise. The TSDB name in combination with `"time series"` was used as search string, e. g., `""OpenTSDB" "time series""`. For instance, when searching for `"Arctic"`, the word `"ice"` had to be excluded, because otherwise oceanic time series data is included in the search results. Further details are described by Bader [Ba16].

The American version of Google, `http://www.google.com`, was used for each search. The search was performed on September 12, 2016 between 08:11 and 09:08 AM UTC. Filtering and automatic completion was disabled, resulted in the URL `https://www.google.com/webhp?gws_rd=cr,ssl&pws=0&hl=en&gl=us&filter=0&complete=0`. The identified groups from Sect. 2 are used to rank the found TSDBs by popularity as shown in Tab. 2.

TSDB	Search Term	Results
TSDB Group 1: TSDBs with a Requirement on other DBMS		
OpenTSDB	`"OpenTSDB" "time series"`	12,900
Rhombus	`"Rhombus" "time series"`	11,700
Newts	`"Newts" "time series"`	6,610
KairosDB	`"KairosDB" "time series"`	3,130
BlueFlood	`"BlueFlood" "time series"`	2,010
Gorilla	`"Gorilla" "time series database" -"pound"`	1,520
Heroic	`"Heroic" "time series database"`	1,490
Arctic	`"Arctic" "time series database" -"ice"`	1,330
Hawkular	`"Hawkular" "time series"`	1,220
Apache Chukwa	`"Apache Chukwa" "time series"`	858
BtrDB	`"BtrDB" "time series"`	637
tsdb: A Compressed Database for Time Serie	`"tsdb: A Compressed Database for Time Series" "time series"`	634
Energy Databus	`"Energy Databus" "time series"`	605
Tgres	`"Tgres" "time series"`	445
SiteWhere	`"SiteWhere" "time series"`	436
Kairos	`"Kairos" "time series database" -"redis" -"agoragames"`	380
Cube	`"Cube" "time series" "Square, Inc."`	266
SkyDB	`"SkyDB" "time series"`	190
Chronix Server	`"Chronix Server" "time series"`	148
MetricTank	`"MetricTank" "time series"`	21
TSDB Group 2: TSDBs with no Requirement on any DBMS		
Elasticsearch	`"Elasticsearch" "time series" -"heroic"`	38,000
MonetDB	`"MonetDB" "time series"`	37,200
Prometheus	`"Prometheus" "time series" -"IMDB" -"Movie"`	33,700
Druid	`"Druid" "time series"`	28,900

TSDB	Search Term	Results
InfluxDB	"InfluxDB" "time series"	28,900
RRDtool	"RRDtool" "time series"	22,600
Atlas	"Atlas" "time series database"	7,960
Gnocchi	"Gnocchi" "time series"	5,320
Whisper	"Whisper" "graphite" "time series"	5,210
SciDB	"SciDB" "time series"	4,140
BlinkDB	"BlinkDB" "time series"	2,250
TSDB	"TSDB" "Time Series Database" "time series" -"OpenTSDB"	1,640
Seriesly	"Seriesly" "time series"	1,330
TsTables	"TsTables" "time series"	1,100
Warp 10	"Warp 10" "time series"	1,020
Akumuli	"Akumuli" "time series"	741
DalmatinerDB	"DalmatinerDB" "time series"	527
TimeStore	"TimeStore" "time series"	443
BEMOSS	"BEMOSS" "time series"	391
YAWNDB	"YAWNDB" "time series"	385
Vaultaire	"Vaultaire" "time series"	339
Bolt	"Bolt" "time series" "Data Management for Connected Homes"	176
GridMW	"GridMW" "time series"	25
Node-tsdb	"Node-tsdb" "time series"	20
NilmDB	"NilmDB" "time series"	9
TSDB Group 3: RDBMS		
MySQL Community Edition	"MySQL" "time series"	309,000
PostgreSQL	"PostgreSQL" "time series"	131,000
MySQL Cluster	"MySQL Cluster" "time series"	23,800
TimeTravel	"TimeTravel" "time series" "dbms"	743
PostgreSQL TS	"PostgreSQL TS" "time series"	1
TSDB Group 4: Proprietary		
Microsoft SQL Server	"Microsoft SQL Server" "time series"	94,000
Oracle Database	"Oracle Database" "time series"	71,500
Splunk	"Splunk" "time series"	30,600
SAP HANA	"SAP HANA" "time series"	22,100
Treasure Data	"Treasure Data" "time series"	15,000
DataStax Enterprise	"DataStax Enterprise" "time series"	12,500
FoundationDB	"FoundationDB" "time series"	11,300
Riak TS	"Riak TS" "time series"	9,720
TempoIQ	"TempoIQ" "time series"	8,810
kdb+	"kdb+" "time series"	8,220
IBM Informix	"IBM Informix" "time series"	7,580
Cityzen Data	"Cityzen Data" "time series"	6,400
Sqrrl	"Sqrrl" "time series"	5,460

TSDB	Search Term	Results
Databus	"Databus" "time series"	5,100
Kerf	"Kerf" "time series"	3,850
Aerospike	"Aerospike" "time series"	3,740
OSIsoft PI	"OSIsoft PI" "time series"	3,200
Geras	"Geras" "time series"	3,030
Axibase Time Series Database	"Axibase Time Series Database" "time series"	2,420
eXtremeDB Financial Editio	"eXtremeDB Financial Edition" "time series"	1,660
Prognoz Platform	"Prognoz Platform" "time series"	1,440
Acunu	"Acunu" "time series"	1,360
SkySpark	"SkySpark" "time series"	1,240
ParStream	"ParStream" "time series"	1,140
Mesap	"Mesap" "time series"	741
ONETick Time-Series Tick Database	"ONETick Time-Series Tick Database" "time series"	503
TimeSeries.Guru	"TimeSeries.Guru" "time series"	464
New Relic Insights	"New Relic Insights" "time series"	233
Squwk	"Squwk" "time series"	191
Polyhedra IMDB	"Polyhedra IMDB" "time series"	149
TimeScape EDM+	"TimeScape EDM+" "time series"	43
PulsarTSDB	"PulsarTSDB" "time series"	4
Uniformance Process History Database (PHD)	"Uniformance Process History Database (PHD)" "time series"	4

Tab. 2: A list of `http://www.google.com` results for each TSDB, grouped by the groups defined in Sect. 2. The twelve compared TSDBs are marked.

6 Introduction of Compared Open Source TSDBs

This section provides an overview on the compared TSDBs and introduces two or more representatives of each group. The grouping of Sect. 2 is chosen as grouping for the presentation. In group 1 and 2, the first five most popular TSDBs are described. In group 3, the two most popular open source RDBMS are described. We exclude group 4 as we want to focus on open source TSDBs. A full list of all found TSDBs is provided in Sect. 2.

TSDB Group 1: TSDBs with a Requirement on other DBMS *Blueflood* [Ra16b] uses Cassandra for storing its time series data. Zookeeper is optionally used to coordinate locking on different shards while performing rollups. Elasticsearch is optionally used to search for time series. *KairosDB* [Ka16] uses H2 or Cassandra as a storage for time series data. H2 is considered as slow and only recommended for testing or development purposes [Me15]. *NewTS* [Ev16] uses Cassandra for storing its time series data. It consists of a server and client, which both are written in Java. In contrast to the compared TSDBs, it is not possible to query for tags and time range (or timestamp) at the same time. *OpenTSDB* [La16a] uses HBase for storing its time series data. HBase in turn uses Zookeeper for coordination between nodes. *Rhombus* [Pa16] is a Java client for Cassandra and ships a schema for

writing “Keyspace Defintions”. “Keyspace Defintions” are a set of definitions that are used by Rhombus to create a time series inside Cassandra. In comparison to the other compared TSDBs, a query on one or more keys without an existing and fitting index in Cassandra is not possible. Combining tags with Boolean algebra is poorly supported, as it is only possible to define an index on one or more fields for combining them, which can be used as a Boolean “AND”.

TSDB Group 2: TSDBs with no Requirement on any DBMS *Druid* [Ya14] uses a RDBMS like Derby, MySQL Community Server, or PostgreSQL as metadata storage and Zookeeper for coordination. It also needs a distributed storage like HDFS, S3, Cassandra, or Azure for storing its data, but it also can be run with a local filesystem for testing purposes. In contrast to the other compared TSDBs, Druid uses five different node types, each type for a specific task. In addition, self-written functions as aggregating functions in JavaScript are possible. *Elasticsearch* [El16a] is a search engine based on Apache Lucene. It stores data in JSON files and is not primarily designed to store time series data, but it can be adopted to store them [Ba15]. *InfluxDB* [In16b] is a TSDB that does not depend on any other DBMS and uses a SQL-like language called InfluxDB Query Language (InfluxQL). Using more than one instance as a cluster was an experimental feature, but is now only included in InfluxEnterprise [In16c]. HA and load balancing can be achieved using InfluxDB Relay [In16e]. *MonetDB* [Mo16a] is a column store that can use SQL as query language and does not depend on any other DBMS. In comparison to the other compared TSDBs, databases and tables are used instead of time series or any related concept. Every SQL statement is internally translated to a MonetDB Assembly Language (MAL) statement. *Prometheus* [Pr16a] is a monitoring solution based on a pull-based model, which means that Prometheus periodically asks for data. For querying Prometheus, a language called PromQL is used. This means that pull-based INS queries are mainly used. It can do push-based INS queries, but then it needs a second component, called Prometheus Pushgateway [Pr16f], from which data is regularly pulled (called “scraped”) by Prometheus. The timestamps of the pushed data are replaced with a new timestamp at the time of scraping, which can be unwanted. This can be avoided by setting an explicit timestamp field, which is considered not useful in most use cases [Pr16f].

TSDB Group 3: RDBMS *MySQL Community Server* [Or16a] is a popular RDBMS, which does not depend on any other DBMS. *PostgreSQL* [PGD16a] is another popular RDBMS that does not depend on any other DBMS. For both RDBMS, a database and a table as described in Sect. 2 must be created.

7 Feature Comparison of TSDBs

The TSDBs presented in Sect. 6 are compared in this section. The comparison results are divided into six tables, one for each criteria group from Sect. 4. The results of the feature comparison are organized regarding the criteria groups into Tab. 3 to 8. The results are described as follows: ✓ means available, (✓) means available with restrictions, and ✗ means not available. The contents of the table are obtained from the official documentation and basic usage of the described TSDBs if not otherwise noted.

From that comparison it can be concluded that there is are no features supported by all TSDBs in any of the criteria groups, besides being one millisecond the smallest storage granularity used by all TSDBs. Compared to traditional RDBMS (group 3), which provide a standardized query language (SQL) and stable versions, only one TSDB (group 1 and 2) provides a stable version. No TSDBs (group 1 or 2) provides a standardized query language or interface. In return, nine of them provide HA features, nine provide scalability features, and ten provide load balancing features. Three of them support all compared queries. It can be concluded that there is not only one specific TSDB that fits for all criteria and scenarios, due to the different features of the TSDBs. Druid the best choice if all criteria besides having a stable/LTS version and commercial support must be fulfilled. In contrast to other compared TSDBs, Druid also uses five different node types and supports self-written functions as aggregating functions in JavaScript. Other TSDBs such as InfluxDB, MonetDB, or one of the two RDBMS can be a better choice if stable/LTS versions or commercial support are required.

TSDB	HA	Scalability	Load Balancing
Group 1: TSDBs with a Requirement on NoSQL DBMS			
Blueflood	✓	(✓)[9]	(✓)[9]
KairosDB	(✓)[9]	(✓)[9]	(✓)[9]
NewTS	(✓)[9]	(✓)[9]	(✓)[9]
OpenTSDB	(✓)[10]	(✓)[10]	(✓)[10]
Rhombus	(✓)[9]	(✓)[9]	(✓)[9]
Group 2: TSDBs with no Requirement on any DBMS			
Druid	✓	✓	✓
Elasticsearch	✓	✓	✓
InfluxDB	✓[11]	✗[12]	✓[11]
MonetDB	✓[13]	(✓)[14]	(✓)[14]
Prometheus	✗[15]	(✓)[16]	(✓)[17]
Group 3: RDBMS			
MySQL Community Server	✗[18]	✗	✗[18]
PostgreSQL	✗[18]	✗	✗[18]

Tab. 3: Comparison of Criteria Group 1: Distribution/Clusterability.

[9] Only for the Cassandra part.
[10] Using a multi node HBase setup, multiple TSDs and distribution of READ and INS queries using DNS Round Robin or external tools like Varnish Cache or HAProxy.
[11] Using InfluxDB Relay [In16e].
[12] Only available in InfluxEnterprise [In16c].
[13] "Transaction Replication" is an experimental feature [Mo16d].
[14] Available with "remote tables" [Mo16c] that use "merge tables" that cannot run INS or UPD queries [Mo16b].
[15] [Pr16d].
[16] Is not built in, but can be achieved by time series design or splitting time series [Kl16] or by using Federation [Pr16e].
[17] Prometheus Servers can be duplicated [Pr16d], but it is not clear if load can be distributed automatically without manual splitting of queries, using DNS Round Robin, or Federation [Pr16e].
[18] Possible to achieve for READ queries with master to slave replication, DNS Round Robin, or external tools like Varnish Cache or HAProxy.

TSDB	INS	READ	SCAN	AVG	SUM	CNT	MAX	MIN	UPD	DEL
Group 1: TSDBs with a requirement on NoSQL DBMS										
Blueflood	✓	✓[19]	✓	✓	✗	✓	✓	✓	✗	✗
KairosDB	✓	✓[19]	✓	✓	✓	✓	✓	✓	✗	✓
NewTS[21]	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗
OpenTSDB[20]	✓	✓[19]	✓	✓	✓	✓	✓	✓	✗	✗
Rhombus[21]	✓	✓	✓	✗	✗	✓	✗	✗	✓	✓
Group 2: TSDBs with no requirement on any DBMS										
Druid[22]	✓	✓[19]	✓	✓[23]	✓	✓	✓	✓	✗	✗
Elasticsearch	✓	✓	✓[24]	✓	✓	✓	✓	✓	✓	✓
InfluxDB	✓	✓	✓	✓	✓	✓	✓	✓	✓[25]	✓
MonetDB	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Prometheus	✓	✓	✓[26]	✓	✓	✓	✓	✓	✗	✗
Group 3: RDBMS										
MySQL Community Server	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
PostgreSQL	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tab. 4: Comparison of Criteria Group 2: Functions.

TSDB	Continuous Calculation	Tags	Long-Term Storage	Matrix Time Series
Group 1: TSDBs with a Requirement on NoSQL DBMS				
Blueflood	✗	✗[27]	✓	✗
KairosDB	✗	✓	✗	✗
NewTS	✓	(✓)[28]	✗	✗
OpenTSDB	✗	✓	✗	✗
Rhombus	✗	(✓)[29]	✗	✗
Group 2: TSDBs with no Requirement on any DBMS				
Druid	✓	✓	✓[30]	✗
Elasticsearch	✓[31]	✓	✗[32]	✓[33]
InfluxDB	✓[34]	✓	✓[35]	✗
MonetDB	✓[36]	✓	✗	✓[37]
Prometheus	✓[38]	✓[39]	✗[40]	✗
Group 3: RDBMS				
MySQL Community Server [Or16a]	✓[36]	✓	✗	✓[37]
PostgreSQL	✓[36]	✓	✗	✓[37]

Tab. 5: Comparison of criteria group 3: Tags, Continuous Calculation, Long-term Storage, and Matrix Time Series.

[19] Uses SCAN with the smallest possible range.
[20] DEL can be substituted with a command-line tool and HBase functions.
[21] Some unsupported functions can be substituted with own Cassandra Query Language (CQL) statements.
[22] Missing functions can be substituted with re-ingestion through an IngestSegmentFirehose.
[23] Uses SCAN and CNT.
[24] Using a Range Query [El16d].
[25] Possible to overwrite with INS when setting resolution to milliseconds, see [Sh14].
[26] Using range vector selectors in combination with an offset modifier [Pr16h] or by using the HTTP API's range queries [Pr16g].
[27] See [Ra15].

TSDB	Down-sampling	Smallest Sample Interval	Smallest Granularity for Storage	Smallest Guaranteed Granularity for Storage
Group 1: TSDBs with a Requirement on NoSQL DBMS				
Blueflood	✗	✗	1 ms	1 ms
KairosDB	✓	1 ms	1 ms	1 ms
NewTS	✓	2 ms	1 ms	1 ms
OpenTSDB	✓	1 ms	1 ms	> 1 ms[41]
Rhombus	✗	✗	1 ms[42]	1 ms[42]
Group 2: TSDBs with no Requirement on any DBMS				
Druid	✓	1 ms	1 ms	1 ms
Elasticsearch	✗[43]	1 ms	1 ms	1 ms
InfluxDB	✓	1 ms	1 ms	1 ms
MonetDB	✓[44]	1 ms[44]	1 ms[45]	1 ms[45]
Prometheus	✓[46]	1 ms	1 ms	1 ms
Group 3: RDBMS				
MySQL Community Server	✓[44]	1 ms[44]	1 ms[45]	1 ms[45]
PostgreSQL	✓[44]	1 ms[44]	1 ms[45]	1 ms[45]

Tab. 6: Comparison of Criteria Group 4: Granularity.

[28] Filtering for tag values cannot be used in combination with time ranges or aggregation functions, which results in a limited tag functionality, see Sect. 6.
[29] Boolean algebra is only poorly supported, which results in a limited tag functionality, see Sect. 6.
[30] Druid uses an immediate “rollup” to store ingested data at a given granularity which helps for long-term storage but there are no further “rollups” after the initial “rollup”. This can be used in combination with rules for data retention and kill tasks [Ya15] to achieve long-term storage.
[31] Elasticsearch’s aggregation functions can be adopted to do this [El16b].
[32] Elasticsearch does optimize long-term storage while data is ingested, which can also be started manually by curator [Ba15]. There is no possibility to run individual long term storage functions (e. g., to reduce granularity).
[33] Assuming that “timestamp” can be used twice as properties with different names [Ba15].
[34] See [In16d].
[35] Using continuous queries for downsampling and retention policies [Be15; In16d].
[36] Via triggers or views.
[37] Using two timestamp columns.
[38] By using recording rules [Pr16c].
[39] Called “labels” in Prometheus [Pr16b].
[40] Long-term storage is not yet supported by Prometheus [Pr16i].
[41] OpenTSDB does not guarantee one millisecond or any other granularity to be stored safely [La16b]. Our measurements showed that OpenTSDB can lose values when using one millisecond as granularity and does not lose values when using 1 second as granularity.
[42] Depends on the types used in keyspace definition.
[43] Using Elasticsearch’s histogram aggregation function [El16c].
[44] Can be implemented in SQL with a WHERE ... AND ... BETWEEN query.
[45] Depends on the types used in table definition.
[46] Can only be done with HTTP API’s range queries [Pr16g].

TSDB	APIs and Interfaces	Client Libraries	Plugins
Group 1: TSDBs with a Requirement on NoSQL DBMS			
Blueflood	Command-Line Interface (CLI)[47], Graphite, HTTP(JSON), Kafka, statsD[48], UDP	✗	✗
KairosDB	CLI, Graphite, HTTP(REST + JSON, GUI), telnet	Java	✓
NewTS	HTTP(REST + JSON, GUI)	Java	✗
OpenTSDB	Azure, CLI, HTTP(REST + JSON, GUI), Kafka, RabbitMQ, S3, Spritzer	✗[49]	✓
Rhombus	✗	Java	✗
Group 2: TSDBs with no Requirement on any DBMS			
Druid	CLI, HTTP(REST + JSON, GUI), Samza[50], Spark[50], Storm[50]	Java[50], Python, R	✓
Elasticsearch	HTTP(REST+ JSON)	Groovy, Java, .NET, Perl, PHP, Python, Ruby	✓
InfluxDB	Collectd[51], CLI, Graphite[51], HTTP(InfluxQL, GUI), OpenTSDB[51], UDP	✗	✓
MonetDB	CLI	Java (JDBC), Mapi (C binding), Node.js, ODBC, Perl, PHP, Python, Ruby	✗
Prometheus	CLI[52], HTTP(JSON, GUI)	Go, Java, Python, Ruby	✗
Group 3: RDBMS			
MySQL Community Server	CLI	J, Java (JDBC), ODBC, Python, and more	✓
PostgreSQL	CLI	C, C++, Java (JDBC), ODBC, Python, Tcl, and more [53]	✓

Tab. 7: Comparison of Criteria Group 5: Interfaces and Extensibility.

[47] A set of executable Java classes.
[48] Experimental feature.
[49] There are many clients made by other users.
[50] Via Tranquility.
[51] Via plugin.
[52] Not for querying [Ra16a].
[53] Some of them are third-party software.

TSDB	LTS/Stable Version	Commercial Support	License
Group 1: TSDBs with a Requirement on NoSQL DBMS			
Blueflood	✗	✗	Apache 2.0
KairosDB	✗	✗	Apache 2.0
NewTS	✗	✗	Apache 2.0
OpenTSDB	✗	✗	LGPLv2.1+, GPLv3+
Rhombus	✗	✗	MIT
Group 2: TSDBs with no Requirement on any DBMS			
Druid	✗[54]	✗	Apache 2.0
Elasticsearch	(✓)[55]	✓[56]	Apache 2.0
InfluxDB	✗	✓[57]	MIT
MonetDB	✗	✓[58]	MonetDB Public License Version
Prometheus	✗[59]	✗	Apache 2.0
Group 3: RDBMS			
MySQL Community Server	✓[60]	✓[61]	GPLv2
PostgreSQL	✓[62]	✗[63]	The PostgreSQL License

Tab. 8: Comparison of Criteria Group 6: Support and License.

8 Conclusion and Outlook

This paper presented a systematic search for TSDBs resulting in 83 found TSDBs, whereby 50 are open source TSDBs. These are grouped into three groups and representatives of each are chosen by popularity and presented more detailed. The twelve chosen representatives are compared in 27 criteria (grouped in six criteria groups). From that comparison (Sect. 7) it can be concluded that no TSDB supports all features from any of the criteria groups. However, three of them are most promising candidates for our setting. When features do not distinguish TSDBs, the performance might. Having enterprise-ready performance is a crucial step from marked readiness to enterprise readiness. As a consequence, our next step is a repeatable, extensible, and open source benchmarking framework for arbitrary TSDBs.

[54] The “normal” releases are called “stable releases”, but are not what is considered “stable” in this paper as every release is called “stable” after it has passed several release candidate stages.
[55] After leaving development and testing phase, Elasticsearch releases are named “stable releases”. Updates for older versions are still released for an undefined amount of time after a new “stable version” was released.
[56] See [El16e].
[57] See [In16a].
[58] See [Mo16e].
[59] The API is considered stable since version 1.0.0 [Pr16d].
[60] See [Or16c].
[61] See [Or16b].
[62] See [PGD16c].
[63] Several commercial support companies are listed on the PostgreSQL homepage, see [PGD16b].

References

[Ac16a] Acreman, S.: Top10 Time Series Databases, 2016, URL: https://blog.dataloop.io/top10-open-source-time-series-databases.

[Ac16b] Acreman, S.: Write Performance Benchmark (Github Gist), 2016, URL: https://gist.github.com/sacreman/b77eb561270e19ca973dd5055270fb28.

[Ba15] Barnsteiner, F.: Elasticsearch as a Time Series Data Store, 2015, URL: https://www.elastic.co/blog/elasticsearch-as-a-time-series-data-store.

[Ba16] Bader, A.: Comparison of Time Series Databases, Diploma Thesis, Institute of Parallel and Distributed Systems, University of Stuttgart, 2016.

[Be15] Beckett, S.: InfluxDB - archive / rollup / precision tuning feature, 2015, URL: https://github.com/influxdb/influxdb/issues/1884.

[BF12] Blumsack, S.; Fernandez, A.: Ready or not, here comes the smart grid! Energy 37/1, pp. 61–68, 2012.

[Ca11] Cattell, R.: Scalable SQL and NoSQL Data Stores. SIGMOD 39/4, pp. 12–27, May 2011.

[DDL14] Date, C. J.; Darwen, H.; Lorentzos, N.: Time and Relational Theory. Morgan Kaufmann, 2014.

[DMF12] Deri, L.; Mainardi, S.; Fusco, F.: tsdb: A Compressed Database for Time Series. In: Traffic Monitoring and Analysis. Springer, 2012.

[El16a] Elasticsearch BV: Elasticsearch, 2016, URL: https://www.elastic.co/de/products/elasticsearch.

[El16b] Elasticsearch BV: Elasticsearch Reference [2.4] - Aggregations, 2016, URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html.

[El16c] Elasticsearch BV: Elasticsearch Reference [2.4] - Histogram Aggregation, 2016, URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-histogram-aggregation.html.

[El16d] Elasticsearch BV: Elasticsearch Reference [2.4] - Range Query, 2016, URL: https://www.elastic.co/guide/en/elasticsearch/reference/2.3/query-dsl-range-query.html.

[El16e] Elasticsearch BV: Elasticsearch - Subscriptions, 2016, URL: https://www.elastic.co/subscriptions.

[Ev16] Evans, E.: Newts, 2016, URL: https://github.com/OpenNMS/newts; http://opennms.github.io/newts.

[Gr13] Grolinger, K.; Higashino, W.; Tiwari, A.; Capretz, M.: Data management in cloud environments: NoSQL and NewSQL data stores. Journal of Cloud Computing: Advances, Systems and Applications 2/1, 2013.

[In16a] InfluxData: InfluxData - Services, 2016, URL: https://www.influxdata.com/services/technical-support; https://www.influxdata.com/services/consulting.

[In16b] InfluxDB: InfluxDB, 2016, URL: https://influxdb.com.

[In16c] InfluxDB: InfluxDB – Clustering, 2016, URL: https://docs.influxdata.com/influxdb/v1.0/high_availability/clusters.

[In16d] InfluxDB: InfluxDB – Continuous Queries, 2016, URL: https://docs.influxdata.com/influxdb/v1.0/query_language/continuous_queries.

[In16e] InfluxDB: InfluxDB Relay, 2016, URL: https://docs.influxdata.com/influxdb/v1.0/high_availability/relay.

[Ka16] KairosDB Team: KairosDB, 2016, URL: https://github.com/kairosdb/kairosdb; http://kairosdb.github.io.

[Kl16] Klavsen, K.: Explaining a HA + scalable setup? (Github), 2016, URL: https://github.com/prometheus/prometheus/issues/1500.

[Ko15] Kopp, O.; Falkenthal, M.; Hartmann, N.; Leymann, F.; Schwarz, H.; Thomsen, J.: Towards a Cloud-based Platform Architecture for a Decentralized Market Agent. In: Informatik. GI e.V., 2015.

[La16a] Larsen, C.; Sigoure, B.; Kiryanov, V.; Demir, B. D.: OpenTSDB, 2016, URL: http://www.opentsdb.net.

[La16b] Larsen, C.; Sigoure, B.; Kiryanov, V.; Demir, B. D.: OpenTSDB - Writing Data - Timestamps, 2016, URL: http://opentsdb.net/docs/build/html/user_guide/writing.html#timestamps.

[Me15] Merdanović, E.: How to install KairosDB time series database?, Feb. 2015, URL: http://www.erol.si/2015/02/how-to-install-kairosdb-timeseries-database.

[Mo16a] MonetDB B.V.: MonetDB, 2016, URL: https://www.monetdb.org/Home.

[Mo16b] MonetDB B.V.: MonetDB – Data Partitioning, 2016, URL: https://www.monetdb.org/Documentation/Cookbooks/SQLrecipes/DataPartitioning.

[Mo16c] MonetDB B.V.: MonetDB - Distributed Query Processing, 2016, URL: https://www.monetdb.org/Documentation/Cookbooks/SQLrecipes/DistributedQuery-Processing.

[Mo16d] MonetDB B.V.: MonetDB - Transaction Replication, 2016, URL: https://www.monetdb.org/Documentation/Cookbooks/SQLrecipes/TransactionReplication.

[Mo16e] MonetDB Solutions: MonetDB Solutions Homepage, 2016, URL: https://www.monetdbsolutions.com.

[Or16a] Oracle Corporation: MySQL Community Server, 2016, URL: http://dev.mysql.com/downloads/mysql.

[Or16b] Oracle Corporation: MySQL Services, 2016, URL: http://www.mysql.com/services.

[Or16c] Oracle Corporation: MySQL – Which MySQL Version and Distribution to Install, 2016, URL: https://dev.mysql.com/doc/refman/5.7/en/which-version.html.

[PA13] Prasad, S.; Avinash, S.: Smart meter data analytics using OpenTSDB and Hadoop. In: Innovative Smart Grid Technologies-Asia (ISGT Asia). IEEE, 2013.

[Pa16] Pardot: Rhombus, 2016, URL: https://github.com/Pardot/Rhombus.

[PFA09] Pungilă, C.; Fortiş, T.-F.; Aritoni, O.: Benchmarking Database Systems for the Requirements of Sensor Readings. IETE Technical Review 26/5, pp. 342–349, 2009.

[PGD16a] The PostgreSQL Global Development Group: PostgreSQL, 2016, URL: https://www.postgresql.org.

[PGD16b] The PostgreSQL Global Development Group: PostgreSQL - Professional Services, 2016, URL: https://www.postgresql.org/support/professional_support.

[PGD16c] The PostgreSQL Global Development Group: PostgreSQL - Versioning Policy, 2016, URL: https://www.postgresql.org/support/versioning.

[Pr16a] Prometheus Authors: Prometheus, 2016, URL: http://prometheus.io.

[Pr16b] Prometheus Authors: Prometheus - Comparison to Alternatives, 2016, URL: https://prometheus.io/docs/introduction/comparison.

[Pr16c] Prometheus Authors: Prometheus – Defining Recording Rules, 2016, URL: https://prometheus.io/docs/querying/rules/#recording-rules.

[Pr16d] Prometheus Authors: Prometheus - FAQ, 2016, URL: https://prometheus.io/docs/introduction/faq.

[Pr16e] Prometheus Authors: Prometheus Federation, 2016, URL: https://prometheus.io/docs/operating/federation.

[Pr16f] Prometheus Authors: Prometheus Pushgateway, 2016, URL: https://github.com/prometheus/pushgateway.

[Pr16g] Prometheus Authors: Prometheus - Range Queries, 2016, URL: https://prometheus.io/docs/querying/api/#range-queries.

[Pr16h] Prometheus Authors: Prometheus – Range Vector Selectors, 2016, URL: https://prometheus.io/docs/querying/basics/#time-series-selectors.

[Pr16i] Prometheus Authors: Prometheus – Roadmap – Long-Term Storage, 2016, URL: https://prometheus.io/docs/introduction/roadmap/#long-term-storage.

[Ra15] Rackspace: Blueflood - FAQ, 2015, URL: https://github.com/rackerlabs/blueflood/wiki/FAQ.

[Ra16a] Rabenstein, B.: Promtool: Add querying functionality (Github), 2016, URL: https://github.com/prometheus/prometheus/issues/1605.

[Ra16b] Rackspace: Blueflood, 2016, URL: http://blueflood.io; https://github.com/rackerlabs/blueflood/wiki.

[Sc14] Schwartz, B.: Time-Series Database Requirements, 2014, URL: https://www.xaprb.com/blog/2014/06/08/time-series-database-requirements/.

[Sc15] Schwartz, B.: How We Scale VividCortex's Backend Systems, 2015, URL: http://highscalability.com/blog/2015/3/30/how-we-scale-vividcortexs-backend-systems.html.

[Sh14] Shahid, J.: Updating an existing point end up inserting, Second comment., Apr. 2014, URL: https://github.com/influxdb/influxdb/issues/391.

[St15] Strohbach, M. o.: Towards a Big Data Analytics Framework for IoT and Smart City Applications. In: Modeling and Processing for Next-Generation Big-Data Technologies. Springer, 2015.

[Te14] Ted Dunning Ellen, M. F.: Time Series Databases – New Ways to Store and Acces Data. O'Reilly Media, Inc, USA, 2014.

[Th01] Theo Härder, E. R.: Datenbanksysteme : Konzepte und Techniken der Implementierung ; mit 14 Tabellen. Springer-Verlag GmbH, 2001.

[Th15] Thomsen, J. et al.: Darstellung des Konzeptes – DMA Decentralised Market Agent – zur Bewältigung zukünftiger Herausforderungen in Verteilnetzen. In: INFORMATIK 2015. Vol. P-246. LNI, 2015.

[Vi14] VividCortex: Building a Time-Series Database in MySQL, 2014, URL: http://de.slideshare.net/vividcortex/vividcortex-building-a-timeseries-database-in-mysql.

[Wl12] Wlodarczyk, T.: Overview of Time Series Storage and Processing in a Cloud Environment. In: CloudCom. 2012.

[Ya14] Yang, F. et al.: Druid: A Real-time Analytical Data Store. In: SIGMOD. 2014.

[Ya15] Yang, F. et al.: Druid – Retaining or Automatically Dropping Data, 2015, URL: http://druid.io/docs/latest/operations/rule-configuration.html.

[Zi15] Zimmermann, O.; Wegmann, L.; Koziolek, H.; Goldschmidt, T.: Architectural Decision Guidance Across Projects – Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge. In: WICSA. IEEE, May 2015.

All links were last followed on 2016-12-01.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 269

The Case For Change Notifications in Pull-Based Databases

Wolfram Wingerath[1], Felix Gessert[1], Steffen Friedrich[1], Erik Witt[2] and Norbert Ritter[1]

Abstract: Modern web applications often require application servers to deliver updates proactively to the client. These push-based architectures, however, are notoriously hard to implement on top of existing infrastructure, because today's databases typically only support pull-based access to data. In this paper, we first illustrate the usefulness of query change notifications and the complexity of providing them. We then describe use cases and discuss state-of-the-art systems that do provide them, before we finally propose a system architecture that offers query change notifications as an opt-in feature for existing pull-based databases. As our proposed architecture distributes computational work across a cluster of machines, we also compare scalable stream processing frameworks that could be used to implement the proposed system design.

Keywords: continuous queries, materialized view maintenance, real-time stream processing, Big Data

1 Introduction

OLTP databases traditionally only support pull-based access to data, i.e. they return data as a direct response to a client request. This paradigm is a good fit for domains where users work on a common data set, but isolated from one another. A variety of modern (web) applications like messengers or collaborative worksheets, on the other hand, target more interactive settings and are expected to reflect concurrent activity of other users. Ideally, clients would be able to subscribe to complex queries and receive both the initial result as well as result updates (i.e. change notifications) as soon as they happen. But only few OLTP database systems provide real-time capabilities beyond simple triggers and application developers often have to employ workarounds to compensate for the lack of functionality. For example, applications are often modeled in such a way that application servers can filter out relevant changes by monitoring specific keys instead of actually maintaining query results in real-time. This makes it possible to notify co-workers of recent changes in a shared document or invalidating caches for static resources [GBR14], but more complex scenarios that cannot be mapped to monitoring individual keys (e.g. maintaining user-defined search queries) are simply infeasible.

In this paper, we survey currently available systems that do provide query change notifications, discuss strengths and weaknesses of their respective designs and propose an alternative system architecture that offers a unique set of strong points. The envisioned system is built around a scalable stream processing framework and, compared to the current state of the art, provides the following main benefits:

[1] University of Hamburg, Information Systems, Vogt-Kölln-Straße 30, 22527 Hamburg, Germany: wingerath@informatik.uni-hamburg.de, gessert@informatik.uni-hamburg.de, friedrich@informatik.uni-hamburg.de, ritter@informatik.uni-hamburg.de
[2] Baqend GmbH, Vogt-Kölln-Straße 30, Room F-528, 22527 Hamburg, Germany: ew@baqend.com

1. **Opt-In Change Notifications:** Being a standalone system, our proposed architecture provides real-time change notifications as an additional feature to existing DBMSs.
2. **Linear Scalability:** The system is able to scale with the number of continuously maintained queries as well as the update throughput.
3. **Pluggable Query Engine:** Through a pluggable query engine, the approach is not system-specific, but applicable to a variety of different databases.

The rest of this article is structured as follows: In Section 2, we provide an example to illustrate what exactly query change notifications are and why providing them is a non-trivial task. We then explore the three use cases (1) real-time notifications for interactive (web) applications, (2) query result cache invalidation and (3) materialized view maintenance in Section 3 and survey existing systems that provide query change notifications in Section 4. Subsequently, we present our own architecture for opt-in query change notifications and discuss viable candidates for the underlying stream processing framework in Section 5. A conclusion and final thoughts are given in Section 6.

2 Problem Statement

In order to enable clients to define their critical data set and keep it in-sync with the server, we argue that clients should be provided with the initial result *and* updates to the result alike. For an illustration of a possible set of notifications, consider Figure 1 that shows a query for NoSQL-related blog posts and a blog post that enters and leaves the result set as it is edited.

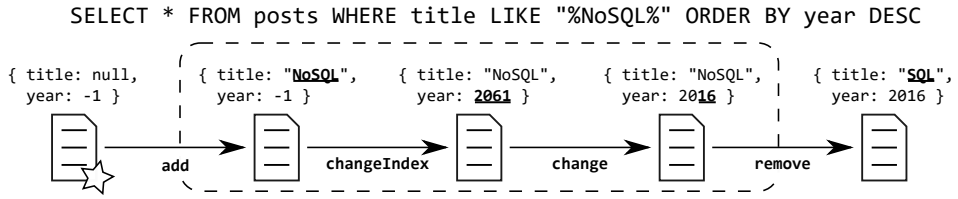


Fig. 1: An example of notifications that occur while a blog post is edited.

Initially, the blog post is created without title and without year and therefore does not match the query. When the author chooses the title to be “NoSQL”, the blog post enters the query result (dashed box) and all query subscribers have to be notified of this event through an `add` notification. Since the publication year is still set to the default value of `-1`, the blog post currently has the last position in the result set. Next, the author intends to set the publication year to the current year, 2016, but accidentally sets it to `2061`. Irrespective of this typo, the blog post moves from the last to the first position in the result set, because it now has the largest `year` value; subscribers receive a `changeIndex` notification and are thus aware of an update to the blog post that changed its position. The author becomes aware of her mistake and subsequently corrects it by setting the year to `2016`. Since there is no other more recent article, the blog post does not change its position (no `changeIndex` notification), but subscribers still receive a `change` notification. After a change of mind, the author updates the title to “SQL” and the blog post correspondingly ceases to match the

query predicate, provoking a `remove` notification. None of the following blog post updates will lead to a notification as long as the matching condition is not satisfied.

Detecting query result changes is significantly more complex than detecting updates on individual data items, because any written object (here: blog post) may or may not satisfy the matching condition of any maintained query. Further, the previous matching status of an object with respect to a query has to be known in order to be able to tell the difference between `add`, `change` and `changeIndex` events and to recognize a `remove` event.

3 Use Cases

There are many applications that rely on server-side notifications of state updates. They can be classified as follows:

1. **Notifications for Clients**: The obvious use case for change notification is forwarding them to clients, so that they are informed whenever relevant state changes occur.
2. **Cache Invalidations**: The capability of detecting result changes opens up the possibility of caching dynamic data, namely query results, with minimal staleness windows. Whenever a result is updated (i.e. becomes stale), the corresponding caches can be invalidated. Since queries are not only served by the database itself, but also by caches located near the clients, response times are reduced significantly and read workload is taken off the database.
3. **Materialized Views**: Frequently requested queries can be kept up-to-date in a separate data store to further relieve the primary database. It is even possible to amend the querying capabilities of the primary database to enable access patterns that would otherwise not be available: for example, a materialized MongoDB view could be maintained on top of a key-value store like Riak by loading all data in a bucket initially and subsequently only processing incoming updates.

4 State of the Art

The inability of traditional pull-based database systems to cope with streaming data well has been identified as a critical and mostly open challenge years ago [SC05, ScZ05] and the integration of static and streaming data has been studied for decades [BLT86, BW01, MWA+03]. While early prototypes required append-only databases [TGNO92], modern systems also consider updated and removed data and thus target more practical applications.
Complex Event Processing (CEP) engines [ACc+03, AAB+05] are software systems specifically designed to derive complex events like a sensor malfunction or an ongoing fraud from low-level events such as individual sensor inputs or login attempts. Queries do not only constrain data properties, but also temporal, local or even causal relationships between events. In contrast to databases that permanently store and subsequently update information, though, CEP engines work on *ephemeral data streams* and only retain derived state such as aggregates in memory for a relatively short amount of time.
Timeseries databases [DF14] are specialized to store and query infinite sequences of events as a function of the time at which they occurred, for example sensor data indexed by time. While they store data permanently and some of them do also have continuous

querying capabilities (e.g. InfluxDB[3]), they are typically employed for analytic queries, materialized view maintenance or downsampling streams of information and do not extend to change notifications.

In recent years, new database systems have emerged that aim to provide real-time change notifications in a scalable manner, but they provide only vendor-specific solutions. Existing applications working on a purely pull-based database have to either switch the underlying data storage system to gain real-time change notifications or have to employ workarounds to compensate for the lack of them.

Meteor[4] is a web development platform backed by MongoDB that provides real-time query change notifications using two different techniques. In principle, a Meteor server reevaluates every continuous query periodically and compares the last and the current result to detect recent changes. This "poll-and-diff" approach allows a complete coverage of the MongoDB feature set, but also adds latency of several seconds. More importantly, it puts load on the database and the application server for computing, serializing, sending and deserializing query results that is proportional to their size. Whenever possible, Meteor applies a more light-weight strategy called oplog tailing where a Meteor server subscribes to the MongoDB oplog (the replication stream) and tries to extract relevant changes from it. While oplog tailing greatly reduces notification latency and processing overhead, it still requires querying MongoDB when the information provided by the oplog is incomplete. The approach is further hard-limited by the maximum of replica set members allowed by MongoDB [Inc16], is only feasible when overall update throughput is low and prohibits horizontal scaling [Das16, met14]. **Parse**[5] is a development framework with MongoDB-like querying capabilities and change notifications for queries. The involved computation can be distributed across several machines, but is ultimately limited by a single-node Redis instance employed for messaging [Par16b]. Parse's hosted database service is going to shutdown in January 2017 and the number of people contributing to the code base has been decreasing over the last months [Par16a]. Even though other vendors have announced support of the Parse SDK [Gai16], future support for the Parse platform is uncertain. **Oracle 11g** is a distributed SQL database with complex query change notifications that supports streaming joins with certain restrictions [WBL+07]. Materialized views of the continuous queries are maintained by applying committed change operations periodically, on-demand or on transaction commit [M+08]. Due to the strict consistency requirements and the underlying shared-disk architecture, scalability is limited. **PipelineDB**[6] extends PostgreSQL by change notifications for complex queries. While the open-sourced version can only run on a single node, the enterprise version supports a clustering mode that shards continuous views and associated computation across several machines. However, since all write operations (insert, update, delete) are coordinated synchronously via two-phase commit between all nodes [Pip15], PipelineDB Enterprise is only scalable up to moderate cluster sizes. **RethinkDB**[7] is a NoSQL database that does support rich continuous query semantics, but is currently subject to a hard scalability limit [Ret16] and does not provide

[3] https://www.influxdata.com/time-series-platform/influxdb/
[4] https://www.meteor.com/
[5] https://parse.com/
[6] https://www.pipelinedb.com/
[7] https://www.rethinkdb.com/

streaming joins. It is the underlying data store for the Horizon[8] development framework. **Firebase**[9] is a cloud database that delivers notifications for changed data, but provides only limited query expressiveness due to the very restrictive underlying data model that requires information to be organized in a tree of lists and objects.

5 Vision: Scalable Opt-In Query Change Notifications

To enable query change notifications on top of systems that by themselves do not provide them, we propose amending these systems by an additional real-time subsystem.

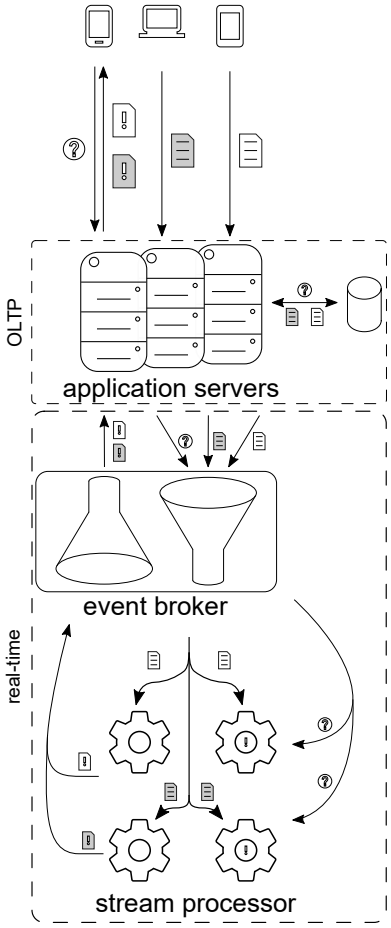


Fig. 2: An architecture that provides opt-in query change notifications on top of purely pull-based databases.

In our proposed architecture as illustrated in Figure 2, common OLTP workloads are still handled by **application servers** that interact with the **database** on behalf of clients. To cope with additional real-time workload, we introduce a new subsystem comprising an **event broker** (i.e. a streaming system like Kafka) to buffer data between application servers and a scalable **stream processor** (e.g. Storm) that maintains continuous queries and generates notifications whenever results change.

To make continuous query maintenance stand on its own, the application server has to provide the real-time subsystem with all required information, namely initial query results and complete data objects on every write. To this end, each continuous query is evaluated once upfront and then sent to the event broker along with all matching objects. Every write operation, i.e. each insert, update and delete, is sent to the event broker together with a complete after-image of the written object, i.e. with the complete data object after the operation has been executed. Besides, an application server subscribes to notifications for the continuous queries of its clients and forwards them correspondingly.

The task of matching the stream of incoming operations against all continuous queries is executed in distributed fashion and partitioned both by written objects and maintained queries. Thus, each processing node is only responsible for a subset of all queries and a subset of all operations. Changes are detected based on whether an object *used to be* a match and whether it still *is* a match for a query. For every change, a notification is sent upstream.

[8] https://horizon.io/
[9] https://firebase.google.com/

The proposed system design decouples resource requirements as well as failure domains for primary storage (persistent data, pull-based access) on the one hand and real-time features (change notifications, push-based access) on the other. Since the real-time workload is handled in a separate subsystem, resources for continuously maintaining query results can be scaled out, while persistent data may be kept in a strongly consistent single-node system. Using a shared-nothing architecture and asynchronous communication throughout the critical processing path, we avoid bottlenecks and thus achieve linear scalability and low latency.

5.1 Scalable Stream Processing Frameworks

Over the last years, a number of scalable and fault-tolerant stream processors have emerged. In the following, we briefly discuss systems that appear as viable candidates for implementing the sketched system design. We therefore do not go into detail on systems that are prone to data loss (e.g. S4 [NRK10]), have been abandoned (e.g. Muppet[10] [LLP+12] or Naiad[11] [MMI+13]), are not publicly available (e.g. Google's Photon [ABD+13] and MillWheel [ABB+13], Facebook's Puma and Stylus [CWI+16] or Microsoft's Sonora [YQC+12]) or cannot be deployed on-premise (e.g. Google's Dataflow cloud service[12] which is built on the eponymous programming model [ABC+15]). For a more detailed overview over the stream processing landscape and a discussion of the trade-offs made in the individual systems' designs, see our stream processing survey [WGFR16].

One of the oldest stream processors used today is **Storm**[13] [TTS+14]. It exposes a very low-level programming interface for processing individual events in a directed acyclic graph, the topology, with at-least-once processing guarantees and is generally geared towards low latency more than anything else. It also provides a more abstract API, Trident, that comes with additional functionality (e.g. aggregations) and guarantees (e.g. exactly-once state management), but also displays higher end-to-end processing latency than plain Storm, because it buffers events and processes them in micro-batches. Being built on the native batch processor Spark[14] [ZCD+12], **Spark Streaming**[15] [ZDL+13] also works on small batches, but usually displays even higher latency on the order of seconds. Through its integration with Spark, Spark Streaming probably has the widest user and developer base and is part of a very diverse ecosystem. **Samza**[16] [Ram15] and **Kafka Streams** [Kre16] are stream processors that are tightly integrated with the data streaming system Kafka [KNR11] for data ingestion and output. Data flow is based on individual events, but since neither Samza nor Kafka Streams have a concept of complex topologies, data has to be persisted between processing steps and latency thus adds up quickly. **Flink**[17] (formerly known as Stratosphere [ABE+14]) tries to combine the speed of a native stream processor with a rich

[10] https://github.com/walmartlabs/mupd8
[11] https://github.com/MicrosoftResearch/Naiad
[12] https://cloud.google.com/dataflow/
[13] http://storm.apache.org/
[14] https://spark.apache.org/
[15] https://spark.apache.org/streaming/
[16] https://samza.apache.org/
[17] https://flink.apache.org/

feature set comparable to that of Spark/Spark Streaming, but is not as widely adopted, yet (cf. [Fou16b, Fou16a]). Even though Flink allows to configure buffering time of individual events, it cannot be tuned as aggressively towards latency as Storm (see for example [CDE+15] or [Met16, slide 71]). **Apex**[18] is another native stream processor with similar design goals as Flink. Being relatively new on the market, Apex is still getting traction. **Heron**[19] [KBF+15] was developed by Twitter to replace Storm which had proven inefficient in multi-tenant deployments, among other reasons due to poor resource isolation. It was open-sourced recently, but has not found wide-spread use as of writing. **IBM Infosphere Streams** [HAG+13, BBF+10] is a proprietary stream processor that is bundled with its own IDE and programming language. It reportedly achieves very low latency, but performance evaluations made by IBM [Cor14] indicate it only performs well in small deployments with up to a few nodes. **Concord**[20] is a proprietary stream processing framework designed around performance predictability and ease-of-use that has just very recently been released. To remove garbage collection as a source of possible delay, it is implemented in C++. To facilitate isolation in multi-tenant deployments, Concord is tightly integrated with the resource negotiator Mesos[21] [HKZ+11].

6 Conclusion

The ability to notify clients of data changes as they happen has become an important feature for both data storage systems and application development frameworks. However, since established OLTP databases have been designed to work with static data sets, they typically do not feature real-time change notifications. The few systems that do are limited in their expressiveness, difficult to scale or they enforce a strong coupling between processing static and streaming data.

In this paper, we propose a scalable system architecture for providing change notifications on top of pull-based databases that sets itself apart from existing designs through a shared-nothing architecture for *linear scalability*, coordination-free processing on the critical path for *low latency*, a pluggable query engine to achieve *database-independence* and a *separation of concerns* between the primary storage system and the system for real-time features, effectively decoupling failure domains and enabling independent scaling for both. We are not aware of any other system or system design that makes complex query change notifications available as an opt-in feature.

While this paper only introduces the conceptual design and contrasts it to existing technology, we already have implemented a prototype that supports the MongoDB query language. So far, our prototype has been used in combination with MongoDB as primary storage system for two use cases: first, providing real-time change notifications for users of a web app and, second, invalidating cached query results as soon as they become stale. We will provide details on the implementation and performance of our prototype in future work.

[18] https://apex.apache.org/
[19] https://twitter.github.io/heron/
[20] http://concord.io/
[21] http://mesos.apache.org/

References

[AAB+05] Daniel J. Abadi, Yanif Ahmad, Magdalena Balazinska, Mitch Cherniack, Jeong hyon Hwang, Wolfgang Lindner, Anurag S. Maskey, Er Rasin, Esther Ryvkina, Nesime Tatbul, Ying Xing, and Stan Zdonik. The design of the borealis stream processing engine. In *In CIDR*, pages 277–289, 2005.

[ABB+13] Tyler Akidau, Alex Balikov, Kaya Bekiroglu, et al. MillWheel: Fault-Tolerant Stream Processing at Internet Scale. In *Very Large Data Bases*, pages 734–746, 2013.

[ABC+15] Tyler Akidau, Robert Bradshaw, Craig Chambers, Slava Chernyak, Rafael J. Fernández-Moctezuma, Reuven Lax, Sam McVeety, Daniel Mills, Frances Perry, Eric Schmidt, and Sam Whittle. The Dataflow Model: A Practical Approach to Balancing Correctness, Latency, and Cost in Massive-Scale, Unbounded, Out-of-Order Data Processing. *Proceedings of the VLDB Endowment*, 8:1792–1803, 2015.

[ABD+13] Rajagopal Ananthanarayanan, Venkatesh Basker, Sumit Das, Ashish Gupta, et al. Photon: Fault-tolerant and Scalable Joining of Continuous Data Streams. In *SIGMOD '13*, 2013.

[ABE+14] Alexander Alexandrov, Rico Bergmann, Stephan Ewen, et al. The Stratosphere Platform for Big Data Analytics. *The VLDB Journal*, 2014.

[ACc+03] Daniel J. Abadi, Don Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: A New Model and Architecture for Data Stream Management. *The VLDB Journal*, 12(2):120–139, August 2003.

[BBF+10] Alain Biem, Eric Bouillet, Hanhua Feng, et al. IBM Infosphere Streams for Scalable, Real-time, Intelligent Transportation Services. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, 2010.

[BLT86] Jose A. Blakeley, Per-Ake Larson, and Frank Wm Tompa. Efficiently Updating Materialized Views. *SIGMOD Rec.*, 15(2):61–71, June 1986.

[BW01] Shivnath Babu and Jennifer Widom. Continuous Queries over Data Streams. *SIGMOD Rec.*, 30(3):109–120, September 2001.

[CDE+15] Sanket Chintapalli, Derek Dagit, Bobby Evans, Reza Farivar, Tom Graves, Mark Holderbaugh, Zhuo Liu, Kyle Nusbaum, Kishorkumar Patil, Boyang Jerry Peng, and Paul Poulosky. Benchmarking Streaming Computation Engines at Yahoo! *Yahoo! Engineering Blog*, December 2015. Accessed: 2016-01-11.

[Cor14] IBM Corporation. Of Streams and Storms. Technical report, IBM Software Group, 2014.

[CWI+16] Guoqiang Jerry Chen, Janet L. Wiener, Shridhar Iyer, Anshul Jaiswal, Ran Lei, Nikhil Simha, Wei Wang, Kevin Wilfong, Tim Williamson, and Serhat Yilmaz. Realtime Data Processing at Facebook. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 1087–1098, New York, NY, USA, 2016. ACM.

[Das16] Dan Dascalescu. What framework should I choose to build a web app? *Dan Dascalescu's Homepage*, May 2016. https://wiki.dandascalescu.com/essays/why_meteor#BONUS_-_Scalability.

[DF14] Ted Dunning and Ellen Friedman. *Time Series Databases: New Ways to Store and Access Data*. O'Reilly Media, November 2014.

[Fou16a] The Apache Software Foundation. Apache Flink: Powered By Flink. *Apache Flink website*, 2016. Accessed: 2016-09-23.

[Fou16b] The Apache Software Foundation. Powered By Spark – Spark – Apache Software Foundation. *Apache Spark Website*, 2016. Accessed: 2016-09-23.

[Gai16] Glenn Gailey. Azure welcomes Parse developers. *Microsoft Azure-Blog*, February 2016. Accessed: 2016-09-19.

[GBR14] F. Gessert, F. Bucklers, and N. Ritter. Orestes: A scalable Database-as-a-Service architecture for low latency. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 215–222, March 2014.

[HAG+13] M. Hirzel, H. Andrade, B. Gedik, et al. IBM Streams Processing Language: Analyzing Big Data in Motion. *IBM J. Res. Dev.*, 57(3-4):1:7–1:7, May 2013.

[HKZ+11] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, and Ion Stoica. Mesos: A Platform for Fine-grained Resource Sharing in the Data Center. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 295–308, Berkeley, CA, USA, 2011. USENIX Association.

[Inc16] MongoDB Inc. Replica Sets. *MongoDB 3.0 Documentation*, 2016. https://docs.mongodb.com/manual/release-notes/3.0/#replica-sets, accessed: October 1, 2016.

[KBF+15] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy, and Siddarth Taneja. Twitter Heron: Stream Processing at Scale. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, SIGMOD '15, pages 239–250, New York, NY, USA, 2015. ACM.

[KNR11] Jay Kreps, Neha Narkhede, and Jun Rao. Kafka: a Distributed Messaging System for Log Processing. In *NetDB'11*, 2011.

[Kre16] Jay Kreps. Introducing Kafka Streams: Stream Processing Made Simple. *Confluent Blog*, March 2016. Accessed: 2016-09-19.

[LLP+12] Wang Lam, Lu Liu, Sts Prasad, et al. Muppet: MapReduce-style Processing of Fast Data. *VLDB 2012*, 2012.

[M+08] S. Moore et al. Using Continuous Query Notification. In *Oracle Database Advanced Application Developer's Guide, 11g Release 1 (11.1)*. Oracle, 2008.

[met14] Large number of operations hangs server. https://github.com/meteor/meteor/issues/2668, 2014. https://github.com/meteor/meteor/issues/2668, accessed: October 1, 2016.

[Met16] Robert Metzger. Stream Processing with Apache Flink. *QCon London*, March 2016. https://qconlondon.com/london-2016/system/files/presentation-slides/robertmetzger.pdf.

[MMI+13] Derek G. Murray, Frank McSherry, Rebecca Isaacs, Michael Isard, Paul Barham, and Martín Abadi. Naiad: A Timely Dataflow System. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 439–455, New York, NY, USA, 2013. ACM.

[MWA+03] Rajeev Motwani, Jennifer Widom, Arvind Arasu, Brian Babcock, Shivnath Babu, Mayur Datar, Gurmeet Singh Manku, Chris Olston, Justin Rosenstein, and Rohit Varma. Query Processing, Approximation, and Resource Management in a Data Stream Management System. In *CIDR*, 2003.

[NRK10] Leonardo Neumeyer, Bruce Robbins, and Anand Kesari. S4: Distributed stream computing platform. In *In Intl. Workshop on Knowledge Discovery Using Cloud and Distributed Computing Platforms (KDCloud*, 2010.

[Par16a] Parse. GitHub: contributors to ParsePlatform/parse-server. 2016. Accessed: 2016-09-19.

[Par16b] Parse. Scalability. *Parse LiveQuery documentation*, 2016. Accessed: 2016-09-20.

[Pip15] PipelineDB. Two Phase Commits. *PipelineDB Enterprise documentation*, 2015. Accessed: 2016-09-18.

[Ram15] Navina Ramesh. Apache Samza, LinkedIn's Framework for Stream Processing. *thenewstack.io*, January 2015. http://thenewstack.io/apache-samza-linkedins-framework-for-stream-processing/.

[Ret16] RethinkDB. Limitations in RethinkDB. *RethinkDB documentation*, 2016. Accessed: 2016-09-19.

[SC05] Michael Stonebraker and Ugur Cetintemel. "One Size Fits All": An Idea Whose Time Has Come and Gone. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 2–11, Washington, DC, USA, 2005. IEEE Computer Society.

[ScZ05] Michael Stonebraker, Uğur Çetintemel, and Stan Zdonik. The 8 Requirements of Real-time Stream Processing. *SIGMOD Rec.*, 34(4):42–47, December 2005.

[TGNO92] Douglas Terry, David Goldberg, David Nichols, and Brian Oki. Continuous Queries over Append-only Databases. *SIGMOD Rec.*, 21(2):321–330, June 1992.

[TTS+14] Ankit Toshniwal, Siddarth Taneja, Amit Shukla, Karthik Ramasamy, Jignesh M. Patel, Sanjeev Kulkarni, Jason Jackson, Krishna Gade, Maosong Fu, Jake Donham, Nikunj Bhagat, Sailesh Mittal, and Dmitriy Ryaboy. Storm@Twitter. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 147–156, New York, NY, USA, 2014. ACM.

[WBL+07] Andrew Witkowski, Srikanth Bellamkonda, Hua-Gang Li, Vince Liang, Lei Sheng, Wayne Smith, Sankar Subramanian, James Terry, and Tsae-Feng Yu. Continuous Queries in Oracle. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB '07, pages 1173–1184. VLDB Endowment, 2007.

[WGFR16] Wolfram Wingerath, Felix Gessert, Steffen Friedrich, and Norbert Ritter. Real-time stream processing for Big Data. *it - Information Technology*, 58(4):186–194, June 2016. http://www.degruyter.com/view/j/itit.2016.58.issue-4/issue-files/itit.2016.58.issue-4.xml.

[YQC+12] Fan Yang, Zhengping Qian, Xiuwei Chen, Ivan Beschastnikh, Li Zhuang, Lidong Zhou, and Guobin Shen. Sonora: A Platform for Continuous Mobile-Cloud Computing. Technical Report MSR-TR-2012-34, Microsoft Research, March 2012.

[ZCD+12] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, et al. Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[ZDL+13] Matei Zaharia, Tathagata Das, Haoyuan Li, et al. Discretized Streams: Fault-tolerant Streaming Computation at Scale. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 423–438, New York, NY, USA, 2013. ACM.

Studierendenprogramm

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 281

A Data Center Infrastructure Monitoring Platform Based on Storm and Trident

Felix Dreissig[1] Niko Pollner[2] (Advisor)

Abstract: Sensor data of a modern data center's cooling and power infrastructure fulfil the characteristics of data streams and are therefore suitable for stream processing. We present a stream-based monitoring platform for data center infrastructure. It is based on multiple independent collectors, which query measurements from sensors and forward them to an Apache Kafka queue. At the platform's core is a processing cluster based on Apache Storm and its high-level Trident API. From there, results get forwarded to one or multiple data sinks. Using our system, analytical queries can be developed using a collection of universal, generic stream operators including CORRELATE, a novel operator which combines elements from multiple streams with unique semantics. Besides the platform's general concept, the characteristics and pitfalls of our real-world implementation are also discussed in this work.

Keywords: Data Stream, Stream Processing, Apache Storm, Trident, Monitoring, Data Center

1 Introduction

Since the emergence of the data stream model, monitoring of sensor data has often been cited as a classical use case for data stream systems. As new measurement values arrive steadily and continuously, it is particularly fitting to view them as a stream of data. Rather than saving the values to a data store and performing periodic analysis, a data stream system can process them live and permanently.

In recent years, the data streaming paradigm has gained popularity due to the advent of new distributed stream processing systems primarily developed in the IT industry. The first and still one of the most popular systems of that kind is *Apache Storm*, which has been released as open-source software by *Twitter*. While *Storm* itself does not provide many processing guarantees, it also includes the *Trident* interface, which e. g. promises persistent state and exactly-once semantics.

In this work, we present the architecture and real-world implementation of a monitoring platform for data center infrastructure based on *Storm* and *Trident*. To the best of our knowledge, it represents the first published usage of *Trident* for monitoring purposes and one of the first publications on practical *Trident* usage.

Data center infrastructure primarily refers to cooling and power distribution systems. These are auxiliary, but critical to the actual IT equipment like servers and network devices. The

[1] Friedrich-Alexander University Erlangen-Nürnberg (FAU), Computer Science 6, Martensstrasse 3, 91058 Erlangen, felix.dreissig@fau.de
[2] Friedrich-Alexander University Erlangen-Nürnberg (FAU), Computer Science 6, Martensstrasse 3, 91058 Erlangen, niko.pollner@fau.de

particular data center covered in this work uses a modular structure for electricity supply and cooling with a multitude of sensors accessible through an IP-based network.

The platform aims to support different areas of monitoring which are usually handled by separated systems today – health monitoring with alerts in case of problems, data collection and visualization for capacity planing and performance analysis and live observation of metrics, e. g. in a schema of system components.

In the remainder of the work, we will first revisit the foundations of data stream processing and introduce basic concepts of *Storm*. Afterwards, the general architecture of the monitoring platform and its operator model are discussed. Section 4 features a presentation and evaluation of the platform's implementation. This is followed by a review of related works. Finally, we close with a conclusion and an outlook on future developments.

2 Foundations

2.1 Data Stream Systems

In contrast to database management systems, which work on a persistent collection of data, data stream processing systems execute continuous queries on sequential, append-only data streams. According to Babcock et al. [Ba02], a data stream is characterized by the online arrival of data elements, no influence for the system on the arrival order of data, potentially unbounded data and the fact that in general, elements cannot be retrieved once they are processed.

Some operations provided by data stream processing systems have counterparts in database management, while others introduce special concepts. The most prominent of such concepts are windows, which limit the number of elements to be considered for a subsequent operation like a JOIN or an aggregation. Windowing features are provided by virtually all stream processing systems, but the available characteristics can vary widely [Bo10].

Traditionally, queries for most data stream processing systems are expressed in domain-specific, declarative languages similar to SQL. *Storm*, which stems from a background in a Big Data community, is programmed using high-level procedural programming languages, mostly Java or Scala, instead.

2.2 Storm

Instead of a comprehensive set of operators, *Storm* provides a programming and runtime framework for stream processing code. Developers generally implement two basic abstractions – Spouts, which represent data sources, and Bolts, which perform processing steps on data elements. Since Bolts may emit an arbitrary number of elements, including zero, per input element, they can also act as data sinks.

Storm is distributed by nature and designed to run on a cluster of independent hosts for scalability and fault tolerance. The system will distribute the Spout and Bolt instances across the cluster and re-assign the workload upon failure of individual hosts. However, keeping persistent state in Bolts was not supported until the recent *Storm* version 1.0. Processing guarantees for the failure case are configurable between at-least-once and at-most-once semantics [To14].

2.3 Trident

Trident is an alternative, higher-level programming model building on top of *Storm*, developed and distributed as part of the *Storm* project. Its key promises are high throughput due to "micro-batching" in the order of several hundred milliseconds, a high-level interface similar to *Apache Pig*, support for persistent state and exactly-once semantics.

Operations are defined by implementing the *Trident* Java APIs, which e. g. provide support for common operations such as aggregations and filters. One or more of these operations are then mapped to a plain *Storm* Bolt for execution. With most operation types, the developer is unaware that the system combines the stream elements to micro-batches internally.

As long as processing results are not revealed to the outside world (and there are no side effects), streaming operations can be repeated an unlimited number of times without any consequences. This is why state is fundamental to *Trident*'s exactly-once support: Since true exactly-once delivery cannot be guaranteed in a distributed system, the effect of exactly-once behavior is achieved by only communicating updates to outside systems once. This generally requires those updates to be idempotent; the exact requirements are defined by *Trident*'s "transactional" and "opaque-transactional" modes [St]. It should be noted that state is always externalized in *Trident*, as it is only gets stored an external system like a database or a distributed file system and only becomes relevant when communicating results to such systems.

3 Concept

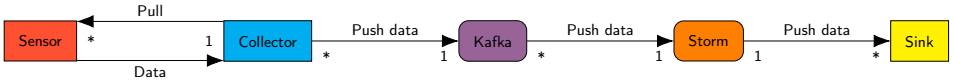


Fig. 1: High-level Architecture of the Complete Monitoring System

3.1 Components

Besides *Storm*, the complete monitoring system is made up of several other components, which are depicted in Fig. 1.

Collectors gather values from one or more sensors and forward them to the rest of the system. For data center infrastructure, they act as a gateway between sensors from the

electrical engineering domain and the software-based monitoring system. For this purpose, existing data collection software like *collectd* can be used. The collectors also mark the boundary between pull-based and push-based data transfers, as they have to explicitly request measurements from the sensors and then actively send them to the processing system.

Apache Kafka is a distributed message queue system commonly used to feed data into and move it out of *Storm* and other stream processing systems. In our architecture, it serves as a common adapter between collectors and *Storm*, for which *Kafka* Spout implementations are readily available. It also guarantees data persistency during *Storm* downtimes because of failures or maintenance. In that case, data can queue up in *Kafka* and be processed afterwards with increased latency, but without data loss.

The central processing point for measurements from all collectors is a *Storm* cluster that first normalizes the data and unifies them to common message schemas in order to simplify further processing. Afterwards, various analyses are performed on the unified data elements. The structure and characteristics of our implementation using *Storm* are covered in Sect. 4.

Finally, normalized raw measurements or processing results get stored to a data sink. Examples for data sinks include a time-series databases for historic collection or a notification system for monitoring alerts.

3.2 Operator Model

3.2.1 General Considerations

Even though *Storm* and *Trident* offer the possibility to express arbitrary queries using a high-level programming language, we developed a set of generic stream operators. These are provided by the platform and can be used to express analytical queries. The operator selection has been identified based on real-world use cases in data center infrastructure monitoring. It aims to be highly universal and reusable, so that development of custom operators is only necessary in special cases. The reusability is supported by the unification of all data to common schemas, as explained in the previous section.

The operator set contains common stream operators such as filters, windowed JOINs and mapping operators. Following the reusability paradigm, arithmetic and boolean operations are provided in form of elementary mapping operations like addition or “less than”. Mappings that work with multiple operands expect so-called Composited Records as input, which consist of multiple individual data elements. Those are generated by a JOIN or CORRELATE, a novel operator to combine elements from multiple streams.

3.2.2 CORRELATE

CORRELATE can be considered a ‘symmetrical version’ of the NEXT operator from *Cayuga* [De07]: While NEXT buffers an element from the first stream and waits for the

next one from the other stream, CORRELATE buffers the first element from either stream until an element is received on the respective other one. Upon its arrival, a Composite Record consisting of both elements is emitted. If yet another message from the first stream is received before one from the second stream, the new message replaces the buffered one. Another way to describe the CORRELATE operator is to view it as JOIN between count-based windows of size 1, which lose their content as soon as it becomes part of a Composite Record.

Primary use case for the CORRELATE operator is the combination of measurement data from various origins and of various types. In an ideal setup, all measurements would be taken simultaneously, making correlation rather simple and the derived metrics very exact. In practice, however, the frequency and exact moments of measurements are outside the control of the processing system. In contrast to windowed JOINs, CORRELATE ensures that every input element is used in at most one output element. With windows, a new result would instead be generated for every input message. Especially when combining lots of streams with different and possibly irregular message frequencies, this would weaken the significance of resulting metrics.

Correlations between more than two streams may be performed as well. In that case, the operator buffers elements from multiple streams until at least one element has been received per stream. Note that while the explanation refers to "streams", *Trident*'s standard API does not support operations with multiple input streams. Therefore, the following uses of CORRELATE expects all input elements to be in the same stream. Which elements to perform correlation on must be specified as parameter of the operator in that case.

3.2.3 Example Query

We now discuss usage of the operators for an example in data center infrastructure monitoring, the calculation of the Power Usage Effectiveness (PUE) value. Despite several criticisms, it is still the prevalent metric for a data center's energy efficiency and is defined as follows:

$$\text{PUE} = \frac{\text{Total facility power}}{\text{IT equipment power}}$$

Total facility power is usually higher than IT equipment power because of infrastructural overheads such as cooling and power distribution. Thus, the ideal PUE value of 1.0 is merely theoretic.

An implementation using our stream operators is depicted in Fig. 2. The CORRELATE operator combines the measurements required to calculate total facility power, in this case IT equipment power and one value from the main power distribution. The former consists of the power values from several sub-distributions, which are all part of the correlation. In the next step, their sum is calculated with the ADD operator. Afterwards, the facility power is divided through that sum to calculate the PUE metric.

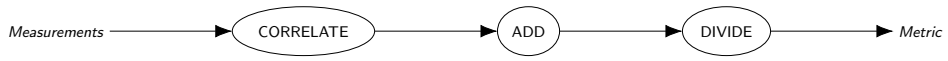


Fig. 2: Operator Graph for the PUE Calculation Query

4 Implementation

As mentioned previously, the platform has been implemented using *Storm* and *Trident*. *Storm* was selected since it is well-established, available as open-source software and enables scalability through its distributed nature. At the time of the implementation, the current *Storm* version was 0.10. It lacked recent features such as built-in windows and stateful Bolts, which made *Trident*'s state management capabilities appear very attractive. Other features like the built-in operators and exactly-once semantics also played a significant role in the decision for *Trident*. All our *Trident* code is designed for "opaque-transactional" properties [St].

4.1 Operators

While the implementation of mapping operators and filters in *Trident* was straight-forward, CORRELATE and windows provided bigger challenges. Support for filters is readily provided through a base class, with the filter condition being specified for individual implementations. Mappings correspond to "Functions" in *Trident*, another base class where only the method generating the mapping result is to be implemented.

CORRELATE and windows have to keep an internal buffer of messages for future output. However, *Trident*'s built-in stateful operations do not support operations which keep state and emit elements at the same time. Therefore, we had to use regular operations and manually perform state management. Nevertheless, our state implementation adopts *Trident*'s abstractions or at least follows similar practices. It stores data in the existing *Apache ZooKeeper* cluster already used by *Storm*, but does not use *ZooKeeper*'s coordination features.

4.1.1 State

The *ZooKeeper* State class implements *Trident*'s State interface. When a state commit begins, a new *ZooKeeper* transaction is created and stored together with the specified *Trident* transaction ID in memory. Keeping that information locally is feasible, as only one State object should be used per *Trident* micro-batch. Once the actual commit happens, that *ZooKeeper* transaction gets executed. Between commit begin and finish, storage instructions for arbitrary data can be added to the transaction. The implementation takes care of saving two versions of data for "opaque-transactional" properties. This is achieved by always storing two versions of the data to *ZooKeeper* together with a *Trident* transaction ID.

Reads from the *ZooKeeper* State happen outside of commits, but also require a *Trident* transaction ID to be specified. The recent stored version of the requested data is returned

if that transaction ID is greater than the one stored with the data; otherwise, the previous version is returned. This ensures that only confirmed state, which cannot be overwritten anymore, is ever returned from a read.

4.1.2 CORRELATE

The CORRELATE operator does not edit existing messages, but uses them to emit new Composite Records. The only *Trident* base operation supporting such behavior is the Aggregator interface. As the name suggests, it is actually designed for aggregations, but since it can emit arbitrary elements, it really is the most flexible kind of *Trident* operation.

Each CORRELATE instance has its own *ZooKeeper* State, where every new micro-batch marks the begin of a new State commit. The batch's single elements are added to an in-memory buffer if they are part of the correlation. At the end of the batch, elements missing for the correlation are retrieved from the State. At this step, it is strictly necessary that reads only return confirmed data.

When all relevant records are available locally, it is checked whether they contain records for all required types of elements. If that is the case, a Composite Record is subsequently created and emitted and the used individual elements are deleted from the State. Otherwise, those elements that were previously only kept in memory get written to the *ZooKeeper* State.

Currently, updates are committed to the *ZooKeeper* State at the end of a micro-batch. *Trident*'s built-in stateful operations actually perform commits asynchronously to processing. Ideally, our custom state updates would be triggered by these commit commands as well. However, receiving the commands is not trivially possible and requires a deep understanding of *Trident*'s undocumented internals. It would also require the *ZooKeeper* State to be adjusted to support multiple concurrent transactions.

4.2 Sinks

The *Graphite* time-series event store serves as proof of concept sink in our implementation. We implemented a special *Trident* operation, which stores selected measurement data and processing results to *Graphite*. Other sinks could be connected in a similar fashion.

For graphing of the time-series values, *Grafana* is used. It allows to display data from different sources, including *Graphite*, in a web frontend. Multiple dashboards, each containing an arrangement of graphs, are defined. An example *Grafana* dashboard is shown in Fig. 3.

4.3 Evaluation

The implementation has shown that it is generally feasible to build a monitoring platform on top of a stream processing system and the concept of collectors, processor and sinks.

Fig. 3: Excerpt of the *Grafana* Dashboard for PUE Values (Details about the specific data center obliterated)

Results, even from reasonably complex queries such as PUE calculation, are available in real-time. While the example operator graph from Fig. 2 is rather simple, the operator graphs can get very complex even for mildly more complex use cases. This a downside of the elementary, generic operator model.

On the other hand, the platform's usage can easily be expanded to further use cases from data center infrastructure monitoring or even other domains of monitoring because of the generic operators and common data schemas. Even if only the *Grafana* sink has been implemented as proof of concept, adding further sinks such as an alerting system is straightforward. Therefore, the goal of supporting different areas of monitoring is achieved as well.

One design decision that has proved particularly useful is the incorporation of *Kafka* as queue for the case of *Storm* failures. During development, it prevented several data losses due to programming errors. After bugfixing, the data could be successfully reprocessed on the then error-free code. The distributed nature of *Kafka* and *Storm* also means high scalability of the platform's core components, as both systems can be scaled to a large number of cluster nodes.

However, limitations of *Trident*'s persistent state capabilities have been revealed while implementing stateful operators such as windows or the CORRELATE operator. The current operator implementations work around these limitations, but are not suitable for large-scale production usage. *Trident*'s support for exactly-once processing is also rather limited on closer examination, as one still has to take care of idempotent state updates and a similar effect could thus be achieved with plain *Storm*'s at-least-once guarantees as well.

5 Related Works

Several authors propose monitoring systems with architectures similar to the structure of collectors, processor and sinks, but without the notion of stream processing [BF15; Ka13]. In [Si13], the stream-based *BlockMon* network monitoring system is extended by distributed processing capabilities and its performance is compared to *Storm* and the *S4* stream processing system.

GEMINI2 [BKB11], a monitoring platform for computing grids, is based on the *Esper* stream processing system. Rather than using a central processor like our system, partial queries are performed on hosts closer to the monitored system and their results are subsequently combined to the complete result.

Smit et al. [SSL13] present *MISURE*, a stream-based platform very similar to the one introduced in this work: It is also based on *Storm*, can incorporate data from various sources and makes results available to different consumers, including a time-series database. *MISURE*'s primary focus is unified processing of results from the monitoring systems of different, possibly geographically distant, cloud computing providers. Besides this difference in the area of application, *MISURE* does not use *Trident*, but plain *Storm* 0.60.

Only a few stream-based monitoring systems are known to be used in the IT industry. *Volta* [Re15], a cloud monitoring platform for systems from the whole application stack, has been developed at *Symantec* using *Storm*, *Kafka*, and *Grafana*. *Librato* is a stream-based monitoring software as a service. It started out using *Storm* as central processor, but recently switched to a custom system [Je15].

6 Conclusion and Outlook

This work showed that monitoring is not only a common use case example in the data streaming literature, but data stream processing systems can actually be utilized for practical use cases in data center infrastructure monitoring.

The architecture for a monitoring platform based on *Storm* and *Trident* has been developed as well as an operator model. The subsequent implementation revealed that while *Trident* promises support for persistent state, its API has shortcomings for stateful operators. Another contribution of the work is the CORRELATE operator, which combines elements from multiple streams with unique semantics.

As a next step, *Trident* should be replaced with a stream processing system that is better suited for the requirements of our operator model. An obvious choice would be *Storm* 1.0 with its support for persistent state and windows. Another option which we are currently exploring is *Apache Flink* (formerly *Stratosphere* [Al14]), which on top of those features also provides built-in support for application time.

With regard to more applications of the platform, additional data sinks are required. Afterwards, the platform may be expanded towards other monitoring domains such as individual hosts and network infrastructure.

References

[Al14] Alexandrov, A. et al.: The Stratosphere platform for big data analytics, The VLDB Journal 23/6, pp. 939–964, 2014.

[Ba02] Babcock, B. et al.: Models and Issues in Data Stream Systems. In: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, Madison, WI, pp. 1–16, 2002.

[BF15] Bauer, D.; Feridun, M.: A Scalable Lightweight Performance Monitoring Tool for Storage Clusters. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management, IEEE, Ottawa, ON, pp. 1008–1013, 2015.

[BKB11] Balis, B.; Kowalewski, B.; Bubak, M.: Real-time Grid monitoring based on complex event processing, Future Generation Computer Systems 27/8, pp. 1103–1112, 2011.

[Bo10] Botan, I. et al.: SECRET: A Model for Analysis of the Execution Semantics of Stream Processing Systems, Proceedings of the VLDB Endowment 3/1, pp. 232–243, Sept. 2010.

[De07] Demers, A. et al.: Cayuga: A General Purpose Event Monitoring System. In: Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, pp. 412–422, 2007.

[Je15] Jenkins, R.: SuperChief: From Apache Storm to In-House Distributed Stream Processing, Librato Blog/, Sept. 24, 2015, URL: http://blog.librato.com/posts/superchief, visited on: 2016-10-20.

[Ka13] Kai, L.; Weiqin, T.; Liping, Z.; Chao, H.: SCM: A Design and Implementation of Monitoring System for CloudStack. In: 2013 International Conference on Cloud and Service Computing, IEEE, Beijing, pp. 146–151, 2013.

[Re15] Renganarayana, L.: Volta: Logging, Metrics, and Monitoring as a Service, Talk recording and slides, Jan. 7, 2015, URL: http://www.slideshare.net/LakshminarayananReng/volta-logging-metrics-and-monitoring-as-a-service, visited on: 2016-10-20.

[Si13] Simoncelli, D.; Dusi, M.; Gringoli, F.; Niccolini, S.: Stream-monitoring with Blockmon: convergence of network measurements and data analytics platforms, ACM SIGCOMM Computer Communication Review 43/2, pp. 29–36, Apr. 2013.

[SSL13] Smit, M.; Simmons, B.; Litoiu, M.: Distributed, application-level monitoring for heterogeneous clouds using stream processing, Future Generation Computer Systems 29/8, pp. 2103–2114, 2013.

[St] Storm Project: Verison 0.10.0 Documentation, Apache Software Foundaton, chap. Trident State, URL: https://storm.apache.org/releases/0.10.0/Trident-state.html, visited on: 2016-10-19.

[To14] Toshniwal, A. et al.: Storm @Twitter. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, ACM, Snowbird, UT, pp. 147–156, 2014.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 291

Comparative Evaluation for Recommender Systems for Book Recommendations

Araek Tashkandi[1], Lena Wiese[2], Marcus Baum[3]

Abstract: Recommender System (RS) technology is often used to overcome information overload. Recently, several open-source platforms have been available for the development of RSs. Thus, there is a need to estimate the predictive accuracy of such platforms to select a suitable framework. In this paper we perform an offline comparative evaluation of commonly used recommendation algorithms of collaborative filtering. They are implemented by three popular RS platforms (LensKit, Mahout, and MyMediaLite) using the BookCrossing data set containing 1,149,780 user ratings on books. Our main goal is to find out which of these RSs is the most applicable and has high performance and accuracy on these data. We consider performing a fair objective comparison by benchmarking the evaluation dimensions such as the data set and the evaluation metric. Our evaluation shows the disparity of evaluation results between the RS frameworks. This points to the need of standardizing evaluation methodologies for recommendation algorithms.

Keywords: Recommender System, LensKit, Mahout, MyMediaLite, Book recommendations.

1 Introduction

In this age of Big Data, we face the problem of acute information overload of the available online data. Thus, it is difficult for users to find items matching their preferences. Many researchers focus on building tools to help users obtain personalized resources [Ga07]. One example of this qualified intelligent system is the recommender system (RS). Recommender systems have become a significant component in e-commerce systems since directing users to relevant content is increasingly important today. RSs have been applied in different application domains, such as entertainment, content, e-commerce and services. Some examples of these applications are Amazon.com (that recommends books, CDs and other products) and MovieLens (that recommends movies). One of the application areas where the RS shows its value is book recommendation, which is the focus of this paper. Recently, a large scale of open-source platforms has been available for the development of RSs. The selection of the foundational platform is an important step in developing the RS. Thus, there is a need to evaluate the quality in terms of predictive accuracy of such frameworks. In this paper we get inspired by [SB14], and we perform an offline comparative evaluation of commonly used recommendation algorithms of collaborative

[1] King Abdulaziz University, Faculty of Computing and Information Technology, 21589 Jeddah, Kingdom of Saudi Arabia, asatashkandi@kau.edu.sa
[2] Georg-August-University Goettingen, Institute of Computer Science, Goldschmidtstr. 7, 37077 Goettingen, Germany, wiese@cs.uni-goettingen.de
[3] Georg-August-University Goettingen, Institute of Computer Science, Goldschmidtstr. 7, 37077 Goettingen, Germany, marcus.baum@cs.uni-goettingen.de

filtering. They are implemented by three popular RS platforms (LensKit, Mahout, and MyMediaLite). We use the book ratings data set from BookCrossing (BX) from [Zi05]. The data set contains 278,858 users providing 1,149,780 ratings on 271,379 books. The evaluation is performed by internal evaluation methods (i.e. evaluation methods that are implemented by the frameworks). Our main goal is to find out which of these RSs is the most applicable and has high performance and accuracy on these data and in general for the book recommendation. Keep in mind the challenges of RSs evaluation as described by [He04] and [SB14] and the difficulties of such a result comparison due to the differences in evaluation and recommendation algorithms implementations. Therefore, in order to reach our goal, we mainly try to make a fair objective comparison of the recommendation accuracy of these RSs by considering the benchmarking of the evaluation dimensions such as the data set, data splitting, recommendation algorithms (including their parameters) and the evaluation method and metric. Our evaluation shows the disparity of evaluation results between the common RS frameworks even with benchmarking the evaluation dimensions.

2 Background

A RS learns about a user's needs and then proactively recommends information that meets his needs. Recommender systems are defined as "software agents that elicit the interests and preferences of individual users explicitly or implicitly and make recommendations accordingly" [XB07]. Three dimensions or models can describe the RS environment: users, data and application. In order to design a RS, the designer has to make choices that are related to the algorithms of the recommendation method, the system architecture and user profile. These choices are constrained by the environment models of the recommender. There are various techniques of how recommendations are generated by a RS. Depending on the information that the system gathers and the technique that is used to generate the item recommendations, the RSs can be classified into different types. The most common recommendation approaches are: content-based, collaborative filtering (CF), hybrid and knowledge-based approach [Ja11]. This paper discusses the CF recommendation approach. In the CF approach to predict the items that the current user will most probably like, the past behavior or opinions of the system's existing user community need to be exploited. Basically, its idea is based on "if users shared the same interests in the past, they will also have similar taste in the future" [Ja11]. It takes the matrix of user-item ratings as an input. It gives an output of numerical rating prediction of the degree that the current user likes or dislikes a specific item.

In order to evaluate a RS some aspects must be specified including the data set, the evaluation method and the performance metric:

- **Data Set:**
  Ekstrand et al. [ERK11] state that in order to evaluate a RS that is developed for a particular context, a data set which is relevant to that context is used. Herlocker et al. [He04] describe the properties that make a data set a best model for the tasks the RS is evaluated for. There are several publicly available data sets that are used for evaluating the RSs. These data sets are helpful in many cases: they provide a basis for

comparing the performance of a new algorithm against a known performance of an existing system, and in case of building a new system they can be used in preliminary testing. Some examples of these publicly available data sets are MovieLens, Jester, BookCrossing, and Netflix data sets.

- **Recommender System Quality Features:**
  There are some features that are considered when evaluating the RS and contribute to the system success and affect the user experience. The three commonly used RS properties from [Be13], [He04] and [SG11] are: (1) Prediction Accuracy where the RS with high accuracy recommends the most relevant items. (2) Coverage is the percentage of items in the data set that the RS is able to provide predictions for. (3) Diversity refers to the diverse features that the recommended items have. As stated by [SG11] the empirical evaluation of RSs has focused on evaluating the accuracy.

- **Evaluation Methods:**
  There are two designs or experimental settings to evaluate the RS and to compare several RSs [ERK11] and [GS09]. Online Evaluation is evaluating the RS while involving real users to use the system to perform real tasks. Offline Evaluation is based on the train-test setup in machine learning. It is performed by using a pre-collected data set of user ratings on items. By this data set we simulate the behavior of user interaction with a RS. Offline evaluation is commonly used in evaluating RSs since it is reproducible and easy to conduct and is not risky since it does not require the involvement of real users.

- **Evaluation Metrics for Performance Measurement:**
  In order to conduct a comparative evaluation for a set of candidate recommendation algorithms, some quality and performance features are measured by using evaluation metrics. We measure the prediction accuracy in evaluating the frameworks. The most popular metric that is used for evaluating accuracy of predicted ratings is Root Mean Squared Error (RMSE). RMSE measures the distance between the predicted preferences and the true preferences over items (i.e. the ones given by users in the test set). For the predicted rating $p_{u,i}$ of user u and item i, the true rating $r_{u,i}$ and data set size n it can be computed as follows [SG11]:

$$RMSE = \sqrt{\frac{1}{n}\sum_{u,i}(p_{u,i} - r_{u,i})^2} \tag{1}$$

3 Related Work

Researching evaluation of RS platforms, [Ek11] implement an evaluation of LensKit, and [SW12] perform an evaluation of Mahout. Said and Bellogin [SB14] perform a comparative evaluation of the three RS frameworks Mahout, LensKit and MyMediaLite. For each framework they perform an external controlled evaluation using a toolkit and an internal evaluation using the frameworks internal evaluation methods. They show disparity of evaluation results between the three recommendation frameworks both in the frameworks internal evaluations as well as in the external controlled evaluation.

The experimental work of evaluating the RS frameworks in this paper is based on [SB14]. They performed the evaluation to find if such a comparison is possible, while we perform a comparative evaluation of the frameworks to find out which of these RSs is the most applicable and has a high performance and accuracy on our used data and in general for book recommendations. Similar to [SB14], we use publicly available data for evaluating the RSs. However, our work differs from theirs in some aspects. First, their evaluation by framework internal evaluation methods was based on two different evaluation metrics since they state there is no common evaluation metric between the frameworks. They used Normalized Discounted Cumulative Gain (nDCG) to compare LensKit with Mahout and Root Mean Squared Error (RMSE) to compare LensKit with MyMediaLite. This results in an ambiguous cross-system evaluation of the three framework RSs. However, we identified RMSE as a common metric between the three frameworks which we use to compare all of them. Second, [SB14] show that the disparity of evaluation results occurs in both the external controlled evaluation as well as in internal frameworks evaluation; however, their internal framework evaluation was not by the same evaluation metric. Thus, we perform and focus only on an evaluation that uses the framework internal evaluation methods. Moreover, in their evaluation they use Movielens 100k data set, while we use the BX data set which is much bigger with higher sparsity. This shows the frameworks' performance, accuracy and capability in such a case using a unified evaluation metric.

4 Experiment and Evaluation

For evaluating the RS capabilities of the three frameworks, we use the internal evaluation methods. In order to perform a fair comparison, we benchmark the evaluation dimensions as will be described in this section. The experiment was run on a computer with Windows 10 Home (v.1511), 8 Gb RAM, Intel® Core(TM) i7-5500U 2.40GHz CPU, a 64-bit operating system and a x62-based processor. For Mahout and LensKit the Java JDK 1.7.0_79 is used. The LensKit evaluator groovy file was executed directly from Windows shell. Mahout was executed by NetBeans IDE 8.1 by creating a Java application in a Maven project and adding Mahout in the dependencies occurring in pom.xml file. MyMediaLite was run on a Mono version 4.2.3.

A natural starting point to evaluate recommender output is to measure the accuracy of prediction [ERK11]. To evaluate each of the three frameworks, the RMSE is computed and compared.

4.1 Data Set

As specified in [SB14], to ensure reproducibility and replicability, documenting the research, using open data, following standards and using best practice algorithms is mandatory. Therefore, we looked for publicly available data sets. As in this paper we focus on book recommendations; hence we select the BookCrossing (BX) data set. The BX data set is collected by a 4-week crawl during August-September 2004 [Zi05]. It is a collection of

book ratings containing 278,858 users providing 1,149,780 ratings (explicit and implicit) on 271,379 books. In addition, the BX data set is also used in some RS evaluations such as [Zi05] and [GS09].

The BookCrossing data set includes three files or tables. The one we use in our evaluation experiment is the Book-Ratings table including user ID, ISBN and the associated ratings in a scale from 1 to 10 for explicit ones and 0 for implicit ones. In order to use this data set in our evaluation, it should be validated with the frameworks required format. After analyzing the data set we discovered many invalid ISBNs: such as there are around 1455 ISBNs with letters and around 885 ISBNs with symbols. In total, there are 2,340 invalid ISBNs. After cleaning the invalid ISBNs out, of 1,149,780 ratings we retain 1,147,440 ratings with valid ISBNs.

Evaluating the frameworks with using this cleaned and validated data set gives a high value of RMSE around 4 which means low prediction accuracy. [Zi05] describe the BookCrossing data set as extremely sparse (i.e. there are a lot of items and few ratings per user). Therefore, we had to condense the data to reduce the sparsity. Thus, we condense the data by removing the implicit ratings with a zero value proceeding in the same way as [GS09]. The implicit ratings are basically no ratings, so we ignore them. From the data set serving only valid ISBNs, 715,019 out of 1,147,440 are implicit ratings. After removing all the implicit ratings, the final data set contains 77472 users, 183744 items and 432,421 ratings. The frameworks split the data set during the evaluation process. LensKit and MyMediaLite evaluators provide cross validation data splitting. The Mahout evaluator lacks cross validation. Instead it creates hold-out partitions according to a set of run-time parameters.

4.2 Recommendation Frameworks

- **LensKit:** LensKit (LK) is an open source RS software that was developed by researchers at Texas State University and GroupLens Research at the University of Minnesota including contributions from developers around the world [Ek11]. It can be used for two purposes: for building a recommender application or for research. It is implemented in Java. The current version at the time of writing was 2.2.1. It provides ready to use implementations for these recommender algorithms: item-based CF, user-based CF, matrix factorization and slope-one.
- **Mahout:** Apache Mahout (AM) from [Fo16], is an Apache Software Foundation project of a well-known machine learning tool. It is an open source library containing several machine learning algorithms. It mainly focuses on the following three areas of machine learning: recommender engines collaborative filtering, classification and clustering. Similar to LensKit, Mahout is developed in Java. At the time of writing it reached version 0.10.1. The algorithms provided by Mahout are available as non-distributed (i.e. executed by a single machine) as well as distributed (i.e. MapReduce, executed in parallel in large clusters of thousands of nodes). In our scenario, we use the non-distributed case for the comparison purpose with the other RS frameworks. For the collaborative filtering algorithms, Mahout implements these algorithms: user-based CF, item-based CF and matrix factorization.

- **MyMediaLite:** MyMediaLite (MML) from [Ga11], was developed at Hildesheim University. At the time of writing, it had reached version 3.11. It is an open source software that provides a library of RS algorithms for the Common Language Runtime (CLR, often called .NET). It is developed in C and it can be run on different platforms, such as Linux, Windows and Mac OS X supported by Mono. It supports ready to use implementations for the two common scenarios in collaborative filtering: rating prediction and item prediction from positive-only feedback, which are combined with various recommender methods for each.

4.3 Algorithmic Details

The main goal of this paper is a fair comparative evaluation of three selected frameworks. Thus, to achieve this goal we benchmarked our comparison by controlling the evaluation dimensions, by using the same data set as explained in the previous section, and by selecting the same recommendation methods, algorithms and metrics. The common recommendation approach implemented by the RS frameworks is the collaborative filtering. They implement memory-based CF and model-based CF methods. A full list of the common methods within the frameworks is given in [SB14].

- **Memory-based CF Algorithms:** The rating matrix is held in memory and is directly used for generating the recommendation. The recommendations are issued based on the relationship between the queried item and the rest of the rating matrix. The common implemented methods by the frameworks are neighborhood-based methods [Ja11]:
  A **User-based CF algorithm** predicts the rating to which degree a specific user u will like a certain item p by getting the rating matrix as input and identifying the nearest neighbors k or the users that had similar preferences to the active user u. Then it gives out the numerical rating prediction of an item p that has not been seen yet by the active user, based on the neighbor users' ratings of p. Similar to the user-based CF, the **Item-based CF algorithm** takes the ratings matrix as an input and it identifies the nearest neighbors k or the items that have similar rating compared to the current item p that we want to compute the prediction for. Then it gets the active user's rating of these similar items to compute his rating for item p, which is the average of them. For computing the similarity, Pearson correlation and Cosine similarity are the similarity methods that can be used [ERK11].
- **Model-based CF Algorithms:**
  The ratings matrix is first processed offline and at the run time the precomputed model makes predictions. It uses data mining and machine learning algorithms to find patterns in the training data. Then it builds the model that can be fit on the test data. The rating prediction and the recommendations are issued based on the built model. The common implemented method by the frameworks is the matrix factorization Singular Value Decomposition (SVD). It is inferred from item rating patterns which is used to characterize both items and users by vectors of factors. An

item is recommended when a high correspondence between the item and user factors exists [SB14].

5 Result and Discussion

5.1 User-based CF algorithm

While evaluating the user-based CF algorithm, we use both of the similarity methods Pearson and Cosine with various neighborhood sizes k. Table 1a shows the evaluation results in terms of RMSE, generated by the frameworks of the user-based CF algorithm with the Pearson correlation method. In general, we observe that the LensKit (LK) prediction accuracy outperforms the Apache Mahout (AM) with different neighborhood sizes by 1.4% of the difference between the averages of the RMSE of both frameworks. However, with the neighborhood size k of 5, the AM gives a lower RMSE than the LK by 0.2% for the same neighborhood size, and lower by 0.2% for the lowest RMSE of the LK. Table 1b shows the evaluation results in terms of RMSE, generated by the frameworks of the user-based CF algorithm with the Cosine similarity method. The difference between the accuracy predictions of the frameworks is not as high as in the previous case. Furthermore, the AM outperforms the LK by 0.02%; the average of the AM's RMSE is 1.9234 and the average of the LK's RMSE is 1.9457. However, there is no big difference between the lowest RMSE of the AM, which is 1.854 and of the LK, which is 1.943. Consequently, we conclude that LK performs better than AM in the user-based CF algorithm. MyMediaLite (MML) was not able to complete the user-based CF evaluation, since it requires too much memory to finish the process. In [SB14] the authors experienced the same problems.

Various neighborhood sizes are tested to identify the best neighbors to consider and to discard the remaining users. However, the best neighborhood size for the accuracy differs between the frameworks. For instance, the best neighborhood size in the user-based CF algorithm with the Pearson method of the AM is 5, whereas in the LK is 10.

(a) User-based CF with Pearson correlation

Parameters	RMSE of LK	RMSE of AM
k=5	1.942345021	1.734362382
k=10	1.938357458	2.14859848
k=30	1.94569808	3.436938497
k=50	1.943045692	3.642906765
k=100	1.941659419	3.686270042
k=200	1.944997373	3.697794514
k=400	1.944071748	3.697794514
k=600	1.94546677	3.697794514
k=1000	1.943604582	3.697794514
k=1500	1.944357718	3.697794514

(b) User-based CF with Cosine similarity

Parameters	RMSE of LK	RMSE of AM
k=5	1.949359429	1.854495873
k=10	1.94548228	2.123300997
k=30	1.94519585	1.9428385
k=50	1.945271313	1.974455281
k=100	1.944991072	1.900317615
k=200	1.947228137	1.903636759
k=400	1.945185463	1.888266977
k=600	1.944815753	1.886360028
k=1000	1.943080086	1.880583559
k=1500	1.946471742	1.880583559

Tab. 1: Results of evaluating user-based CF

5.2 Item-based CF algorithm

For evaluating the item-based CF algorithm we use both similarity methods Pearson and Cosine. As shown in Table 2, AM has the lowest RMSE and is outperforming the LK in the Cosine similarity method by 0.4%.

Algorithm	Parameters	RMSE of LK	RMSE of AM
IBPea	no parm	1.953229033	2.240947847
IBCos	no parm	1.962735886	1.579909495

Tab. 2: Results of evaluating item-based CF with Pearson correlation and with Cosine similarity

Similar to the user-based evaluation, we were not able to evaluate the item-based CF in MyMediaLite. We can relate the MML problem to the sparsity problem of the data we used. Thus, we conclude the MML is not advisable if the used data is too sparse. In general, the problem of sparsity data does not only affect the RS performance (e.g. it requires too much memory or time), it also has an impact on the accuracy of the recommendation algorithm, as observed by [He04]. Since we use a low density and high sparsity data set, we conjecture that the computation performance and the prediction accuracy of the CF recommendation algorithms are affected. On a denser data set the three frameworks might perform better.

5.3 Model-based CF Algorithm

In addition, we compare the three frameworks in terms of run time of building and evaluating the algorithm SVD. We experienced that the LK is the fastest and the MML is the slowest, as shown in Table 3. Even though the MML is the weakest with the sparse data in the user-based and the item-based CF algorithms, it gives the highest accuracy in the SVD algorithm evaluation.

SVD method takes two parameters: 1) the number of latent factors or features (f) the users and items are characterized by, and 2) the number of iterations (i) for training and computation:

Parameters	RMSE of LK	Time	RMSE of AM	Time	RMSE of MML	Time
f=50, i=10	1.926852438	8min	2.294968976	15min	1.720527	45min
f=50, i=20	1.925003523	10min	2.416660616	27min	1.69042	2h

Tab. 3: Results of evaluating Singular Value Decomposition (SVD)

5.4 Discussion of Results

To summarize, after evaluating the three RS frameworks using the sparse BookCrossing data set, we observe that the evaluation results of LK's RMSE values are more consistent in the three recommendation algorithms, and in general in most of the cases its RMSE values are lower than the ones of AM. AM's RMSE values fluctuate, but the lowest RMSEs are better than the ones of LK. LK's performance in terms of the run time is the fastest. In

addition, we observe that AM performs in the Cosine similarity better than in the Pearson correlation method. By using this sparse data we conclude that not all the RS frameworks are able to deal equally with too sparse data. Comparing to the other frameworks, the MML is the weakest in dealing with sparse data sets in the user-based and the item-based CF algorithms. It requires too much memory to finish the process and it takes the longest time to finish the SVD algorithm. Nevertheless, it gives the lowest RMSE in the SVD algorithm comparing to the other RSs. However, our presented result is limited to our used data set, which has sparsity problems that affect the prediction accuracy and the performance of the frameworks.

Even though we benchmarked the comparison dimensions such as the used data set and the algorithms with the parameters, there were difficulties to fairly compare the evaluation of the three RSs. As we mentioned, most of the RS literature points to the challenges of RS evaluation. The reason lies in the performance discrepancies and the disparity of the evaluation results of the same settings and algorithms with its parameters between the recommendation frameworks. For instance, our comparative evaluation of the SVD algorithm with the same parameters gives a 72% difference in the RMSE of the AM and the MML, and a 24% difference in the RMSE of the LK and the MML. Moreover, there is a higher than 100% difference in the RMSE of evaluating LK's and AM's user-based CF with the Pearson correlation method with the same neighborhood size parameter k of 200. Similarly, Said and Bellogin [SB14] encounter this problem of disparity of the evaluation results between the recommendation frameworks. They linked it to the differences in the implementations of the evaluation and recommendation algorithms in the frameworks.

6 Conclusion

For implementing a Recommender System (RS) by using open-source platforms, the most suitable platform has to be chosen after evaluating the considered frameworks. In this paper we perform an offline comparative evaluation of commonly used recommendation algorithms of collaborative filtering that are implemented by three popular RS platforms (Apache Mahout (AM), LensKit (LK) and MyMediaLite (MML)) using the BookCrossing (BX) book ratings data set. The performance and the accuracy of the implemented RSs differ based on the circumstances of the use case and may differ when evaluating them by using other data sets with different characteristics. Even though we used identical settings, we encountered difficulties of making a fair comparative evaluation of the three RSs. The reason for this is the performance discrepancy and the disparsity of the evaluation results. Many related research articles have discussed the RS evaluation. However, the effective evaluation of RSs is still challenging since 2004 [He04]. There is no practical rule for evaluating the RSs and the recommendation algorithms. Thus, we need a development of more standardized evaluation methodologies for RSs and for recommendation algorithms.

References

[Be13] Beel, J.; Langer, S.; Genzmehr, M.; Gipp, B.; Breitinger, C.; Nürnberger, A.: Research Paper Recommender System Evaluation: A Quantitative Literature Survey. In: Proceedings

of the Workshop on Reproducibility and Replication in Recommender Systems Evaluation (RepSys) at the ACM Recommender System Conference (RecSys). ACM, pp. 15–22, 2013.

[Ek11] Ekstrand, M.; Ludwig, M.; Konstan, J.; Riedl, J.: Rethinking The Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. In: Proceedings of 5th ACM Conference on Recommender Systems RecSys. ACM, pp. 133–140, 2011.

[ERK11] Ekstrand, M.; Riedl, J.; Konstan, J.: Collaborative Filtering Recommender Systems. Foundations and Trends in Human–Computer Interaction, 4(2):81–173, 2011.

[Fo16] Apache Mahout: Scalable Machine Learning and Data Mining, The Apache Software Foundation. https://mahout.apache.org/.

[Ga07] Gao, F.; Chunxiao, X.; Xiaoyong, D.; Shan, W.: Personalized Service System Based on Hybrid Filtering for Digital Library. Tsinghua Science and Technology. Tsinghua University Press (TUP), 12(1):1–8, 2007.

[Ga11] Gantner, Z.; Rendle, S.; Freudenthaler, C.; Schmidt-Thieme, L.: Mymedialite: A Free Recommender System Library. In: Proceedings of 5th ACM Conference on Recommender Systems, RecSys '11. ACM, pp. 305–308, 2011.

[GS09] Gunawardana, A.; Shani, G.: A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. Journal of Machine Learning Research, 10:2935–962, 2009.

[He04] Herlocker, J.; Konstan, J.; Terveen, L.; Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems TOIS, 22(1):5–53, 2004.

[Ja11] Jannach, D.; Zanker, M.; Felfernig, A.; Friedrich, G.: Recommender Systems: An Introduction. Cambridge University Press, New York, 2011.

[SB14] Said, A.; Bellogin, A.: Comparative Recommender System Evaluation: Benchmarking Recommendation Frameworks. In: Proceedings of The 8th ACM Conference on Recommender Systems RecSys'14. ACM, pp. 129–136, 2014.

[SG11] Shani, G.; Gunawardana, A.: Evaluating Recommendation Systems. Recommender Systems Handbook. Springer, pp. 257–97, 2011.

[SW12] Seminario, C.; Wilson, D.: Case Study Evaluation of Mahout as a Recommender Platform. In: Workshop on Recommendation Utility Evaluation: Beyond RMSE (RUE 2012), Proceedings of the 6th ACM Conference on Recommender Engines (RecSys). ACM, pp. 45–50, 2012.

[XB07] Xiao, B.; Benbasat, I.: Ecommerce Product Recommendation Agents: Use, Characteristics, and Impact. MIS Quarterly, 31(1):137–209, 2007.

[Zi05] Ziegler, C.; Sean, M.; Konstan, J.; Lausen, G.: Improving Recommendation Lists Through Topic Diversification. In: Proceedings of The 14th International World Wide Web Conference WWW2005. ACM, pp. 22–32, 2005.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 301

Comparing Relevance Feedback Techniques on German News Articles

Julia Romberg[1]

Abstract: We draw a comparison on the behavior of several relevance feedback techniques on a corpus of German news articles. In contrast to the standard application of relevance feedback, no explicit user query is given and the main goal is to recognize a user's preferences and interests in the examined data collection. The compared techniques are based on vector space models and probabilistic models. The results show that the performance is category-dependent on our data and that overall the vector space approach *Ide* performs best.

Keywords: Relevance Feedback, Text Mining, Filtering Systems

1 Introduction

Over the last decades personalization has become of major interest and it continually gains on popularity. Especially in information systems, involving a large amount of data, a user benefits from the presentation of only a subset of data, which is customized to the user's particular information needs or general interests.

In case of a target-oriented information need, information retrieval systems serve to satisfy this as follows: A user formulates a query that fits his or her specific need for information. The information system then returns the most appropriate documents regarding the query using some algorithm. These documents are all assumed to be relevant for the given information demand. However, this may not always be the case in real-world applications as the envisioned request usually does not completely match the phrased query. One example is the search for documents related to the animal crane. A user, with a clear objective in mind, may therefore formulate the simple query "crane". Using Google search the first results include documents that are about the animal, about the machine type called crane, or even about a paper manufacture carrying "Crane" in its name. As can be seen from this example, a semantic gap between imagination and the query can arise. To nevertheless ensure that the user's information needs will be satisfied, the formulated query can be adjusted using relevance feedback [RS65]: Following the information retrieval step, the user rates the best matching documents as relevant or non-relevant. Involving this assessment, the initial query can be modified in order to reduce the semantic gap and hence better results can be achieved.

Whereas traditional information retrieval systems serve a target-oriented information need, the more general recognition of user interests is mostly done by information filtering

[1] Heinrich Heine University Düsseldorf, Institute of Computer Science, Universitätsstraße 1, 40225 Düsseldorf, romberg@cs.uni-duesseldorf.de

systems. Information filtering systems aim to help finding user-preference-based documents by filtering unwanted information out of an overload of information. If we treat filtering as a text classification task with two classes (relevant and non-relevant), techniques such as neural networks, support vector machines, decision trees, or logistic regression, tend to perform rather well once some basis knowledge of the user's preferences is given [Zh09]. In order to collect initial training data and to handle the so-called cold start problem, target-oriented retrieval techniques may be used.

The purpose of our work is to compare different retrieval techniques that involve relevance feedback with regard to their behavior on the cold start problem and on a corpus of German news articles. The considered models are on the one hand vector space models and on the other hand models that originate in probabilistic theory. The remainder of this paper is structured as follows: In the next section we discuss important papers for ad-hoc retrieval. We then describe the data corpus, the relevance feedback techniques on which we focus, and the query formulation. Subsequently the techniques are evaluated and compared. Finally a conclusion is drawn and further work is outlined.

2 Related Work

A lot of research has been done in the field of ad-hoc retrieval and on retrieval models that involve relevance feedback. The best known approaches seem to be query modification in vector space models and improved relevance guesses in probabilistic information retrieval systems.

The first relevance feedback approach, introduced in [RS65], relates to the vector space model and gives a formula to spatial shift a vector that represents the user query towards points that represent relevant documents. In [Id71] this idea is taken up on with changed query modification formulas.

Probabilistic models estimate the relevance of documents. An overview over models such as the basis model [MK60], the binary independence model [RJ76, Ri79] and the Okapi BM25 model [RW94] is given by [JWR00a, JWR00b]. In general, relevance estimation is done by making use of a subset of previously rated documents. This intuitively leads to the application of relevance feedback. Further models that also originate in probability theory, for example Language models [HR01] and Bayesian networks [TC89], have also been applied in connection with relevance feedback.

The approaches have mostly been tested on TREC collections[3] that were released for tasks of the Text REtrieval Conference. We, however, want to evaluate on a different set of data: German news articles taken from online resources. We want to observe how the compared techniques behave on this specific data source, whether their performance is topic-dependent, and which one performs best.

[3] http://trec.nist.gov

3 Data and Relevance Feedback Methods

In this section, first, the data corpus is presented. We then illustrate the data representation we use for the comparison. Afterwards, an overview of the used relevance feedback techniques is given. Finally, we focus on the challenge of lacking an initial query.

3.1 Corpus

One basic requirement for a good comparability of different relevance feedback techniques on a quantity of data is a high linguistic quality as well as good content-related quality. This is essential as the corpus needs to be transferred in a comparable representation using a natural language processing pipeline, whereby errors should be minimized, and also as we evaluate on test persons. The German online platform *ZEIT ONLINE*[4] fulfills these conditions. Furthermore, it provides an API to get access to the meta data of all archived articles.

Our data corpus consists of 2500 German news articles which were crawled from *ZEIT ONLINE*. They were divided up in the categories *politics*, *society*, *economy*, *culture*, and *sports*, with a proportion of 500 articles each. This size was selected to fit on the one hand the need for a sufficiently large corpus to get significant results on the evaluation runs and to simultaneously reduce the impact older articles have on relevance ratings. News actuality often correlates with relevance so that the given relevance feedback could be biased by including articles that have no relation to current events.

3.2 Data Representation

In order to run our evaluation, we first need to transform the collection of documents in an applicable format with regard to the chosen relevance feedback methods. The most general approach is the use of a bag-of-words: Given an index term vocabulary $V = \{w_1, ..., w_n\}$, a document is expressed by a n-dimensional vector in which every entry is the term frequency of w_i in the regarded document. Important contents of news articles, particularly online, use to be described by key words. These key words are mostly nouns and named entities, such as person names, locations, and companies. We therefore decide to only include nouns and named entities in V. This also leeds to a remarkable dimension reduction. While in the bag-of-words model words are seen as isolated units, topic models attempt to incorporate relations between words by uncovering underlying semantic structures of the collection. One of the best known topic models is Latent Dirichlet Allocation [BNJ03]. The main idea here is that a collection covers k topics. Thereby a topic is defined as a distribution over a fixed index term vocabulary and each document belongs, to a certain proportion, to each of the k topics. A document is represented by a k-dimensional vector of length 1, which consists of the topic membership proportion.

[4] http://www.zeit.de

3.3 Vector space models

Vector space models are based on the assumption that documents and querys can be represented as vectors located in a vector space. A document's relevance concerning a given query is defined by the similiarity between the two representing vectors. We used the cosine similarity, which is the most commonly used similarity measure in this context: The smaller the angle between two vectors is, the more alike they are.

The first approach we use is *Rocchio*'s formula [RS65]. On the basis of subsets with known relevance judgments, the spatial position of an initial query vector $\vec{q}$ is improved by moving the original query vector $\vec{q}$ towards the centroid of the relevant documents D_r and away from the centroid of the non-relevant documents D_{nr}:

$$\vec{q}_m = \alpha \cdot \vec{q} + \beta \cdot \frac{1}{|D_r|} \cdot \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\forall \vec{d}_j \in D_{nr}} \vec{d}_j \tag{1}$$

$\overrightarrow{q_m}$ denotes the modified query vector which is then used to calculate the similarity to the document vectors in order to determine a more appropriate set of relevant documents. The weights $\alpha, \beta, \gamma \in \mathbb{R}_0^+$ control the impact of the single components.

One further vector space approach is *Ide* [Id71]. It removes the centroid normalization from formula 1:

$$\vec{q}_m = \alpha \cdot \vec{q} + \beta \cdot \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \cdot \sum_{\forall \vec{d}_j \in D_{nr}} \vec{d}_j \tag{2}$$

3.4 Probabilistic models

In contrast to vector space models, classic probabilistic models aim to estimate a document's relevance for a given query. For this, a term weight $\in [0,1]$ is assigned to every query term that expresses the probability of obtaining a relevant document by this term.

We used the *Binary Independence Model* and the *Okapi BM25 Model* with a smoothed term weighting proposed in [RJ76]:

$$\widetilde{w_i} = \frac{|D_{r_i}| + 0.5}{|D_r| - |D_{r_i}| + 0.5} \cdot \frac{|D| - |D_i| - |D_r| + |D_{r_i}| + 0.5}{|D_i| - |D_{r_i}| + 0.5},$$

where D_{r_i} denotes the relevant documents which contain the ith term out of all relevant documents D_r and D_i denotes the number of documents out of all documents $D = D_r \cup D_{nr}$ that contain this term.

The overall estimated relevance for a document d and for a given query $q = (w_1, w_2, ..., w_t)$ in the Binary Independence Model is, under the assumption of term independence, provided by the logarithm of the product over all weights $\widetilde{w_i}$ of the query terms that are contained in d:

$$relevance_score(d,q) = \log \prod_{i=1}^{t} \widetilde{w_i} \cdot \mathbb{1}_{i \in d} = \sum_{1=i}^{t} \log(\widetilde{w_i}) \cdot \mathbb{1}_{i \in d} \tag{3}$$

Additionally, in the Okapi BM25 Model term frequency and document length are taken into account by

$$relevance_score(d,q) = \sum_{i=1}^{t} \log(\widetilde{w}_i) \cdot \frac{\mathrm{tf}(w_i,d)\cdot(k_1+1)}{k_1\cdot((1-b)+b\cdot\frac{\mathrm{dl}}{\mathrm{avdl}})+\mathrm{tf}(w_i,d)} \tag{4}$$

The parameter k_1 controls the influence of w_i's term frequency $\mathrm{tf}(w_i,d)$ in d. The impact of the document length dl in proportion to the average document length avdl in the collection is given by $b \in [0,1]$.

Intuitively, a good way to reflect the user's information needs would be a query which consists of terms that occur frequently in relevant documents. This idea is the basis for a different approach to probabilistic models named language models. The probability that a document is relevant to a given query can, according to that, be estimated by

$$P(d|q=(w_1,w_2,...,w_t)) = P(d)\cdot P(q|d) = P(d)\cdot\prod_{i=1}^{t}((1-\lambda_i)P(w_i)+\lambda_i P(w_i|d)) \tag{5}$$

A permissible way to estimate the probabilities is [HR01]:

$$P(w_i) = \frac{\mathrm{df}(w_i)}{\sum_{j=1}^{t}\mathrm{df}(w_j)}, \quad P(w_i|d) = \frac{\mathrm{tf}(w_i,d)}{\sum_{j=1}^{t}\mathrm{tf}(w_j,d)}, \quad P(d) = \frac{\sum_{j=1}^{t}\mathrm{tf}(w_j,d)}{\sum_{j=1}^{t}\sum_{k=1}^{|D|}\mathrm{tf}(w_j,d_k)},$$

where $\mathrm{df}(w_i)$ denotes the document frequency of the query term w_i.

The parameter $\lambda_i \in [0,1]$ mirrors the importance of the ith term in q. In order to find a good value for λ_i we use an expectation-maximization algorithm as suggested in [HR01]. Expectation-maximization is a statistical method to estimate unknown parameters on incomplete data: By alternating an expectation step and a maximization step, conclusions on the probability distribution in the complete data set can be drawn. The expectation step is defined as

$$r_i' = \sum_{j=1}^{|D_r|} \frac{\lambda_i^{(p)} P(w_i|d_j)}{(1-\lambda_i^{(p)})P(w_i)+\lambda_i^{(p)}P(w_i|d_j)}$$

and the maximization step is defined as

$$\lambda_i^{(p+1)} = \frac{r_i'}{|D_r|}.$$

3.5 Query Formulation

The precondition for relevance feedback techniques to work is the existence of an initial query. In our case, there is no user formulated query to start with. To overcome this problem,

we initially define $\vec{q}$ in vector space models as zero vector, equivalent to the document representation with n dimensions in the bag-of-words format and k dimensions in the LDA format. After a first training phase, the vector is then modified into a more meaningful vector. On the other hand in probabilistic models we initially admit every index term as possible query term.

4 Evaluation

In order to run the comparison of the relevance feedback methods, the experimental setup is introduced at first. After that, different evaluation measures are discussed, followed by the evaluation results which are described and then discussed.

4.1 Experimental Setup

Table 1 gives an overview over the relevance feedback techniques that are compared. As most of the techniques' performances depend on the parameter selection, we tried different settings that were current in the literature. Furthermore the vector space model techniques were evaluated on both data representations: bag-of-words and LDA. Beside the comparison between techniques, we also wanted to observe if the techniques behave differently on various news article categories. We exemplarily took the categories *culture*, *economics*, *politics*, *society* and *sports*.

The ideal experimental setup would imply for every test person to evaluate any technique on any category and over a period of time, i.e. to run several feedback iterations. Unfortunately

Technique	Parameter Setting	BOW	LDA	Abbreviation
Rocchio (formula 1)	$\alpha = 1, \beta = 0.75, \gamma = 0.25$	✓		R
Rocchio (formula 1)	$\alpha = 1, \beta = 0.75, \gamma = 0.25$		✓	LR
Rocchio (formula 1)	$\alpha = 1, \beta = 1, \gamma = 0$	✓		R2
Rocchio (formula 1)	$\alpha = 1, \beta = 1, \gamma = 0$		✓	LR2
Ide (formula 2)	$\alpha = 1, \beta = 1, \gamma = 1$	✓		Ide
Ide (formula 2)	$\alpha = 1, \beta = 1, \gamma = 1$		✓	LI
Ide (formula 2)	$\alpha = 1, \beta = 1, \gamma = 0$	✓		I2
Ide (formula 2)	$\alpha = 1, \beta = 1, \gamma = 0$		✓	LI2
Binary Independence Model (formula 3)		✓		BIM
Okapi BM25 (formula 4)	$b = 0.75, k_1 = 1.2$	✓		B25
Okapi BM25 (formula 4)	$b = 0.75, k_1 = 1.6$	✓		252
Okapi BM25 (formula 4)	$b = 0.75, k_1 = 2.0$	✓		253
Language Model (formula 5)	EM-algorithm	✓		LM

Tab. 1: List of evaluated techniques and abbreviation names

this was not practically applicable with the test persons, which were not rewarded. The main goal of the comparison is the comparison between the techniques. For this reason we decided for every test person to evaluate all techniques but not all categories. We had 29 test persons which were randomly assigned to the five categories having five persons on culture and six persons on each other category. Instead of comparing the techniques over several iterations we chose a train-and-test setup with a bigger training phase: For the training the test persons had to give relevance feedback on 100 randomly chosen news articles. For the testing the test persons had to give relevance feedback on the top 25 results calculated by every technique-setting pair.

To observe the adaption on new data, in the test phase news articles that had already been shown and rated during the training were not shown again. The test persons were furthermore explicitly requested to rate the relevance of an article independent from up-to-dateness.

4.2 Evaluation Measures

The evaluation of relevance feedback is challenging as it is quite subjective. An appropriate way to evaluate a classification task is to calculate the *precision*:

$$\text{precision} = \frac{|\text{relevant} \cap \text{retrieved}|}{|\text{retrieved}|}$$

Precision is a set-based measure. It does not take the degree of similarity of a document and a query into account. However, in our case retrieval is done by determing the k best results, which implies a ranking. Therefore a ranking-based measure could be more informative. A widely used one is *MAP*:

$$MAP = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\frac{1}{|D_{r_i}|} \cdot \sum_{k=1}^{n} \left(precision@k \cdot \mathbb{1}_{\{k \in D_{r_i}\}} \right) \right)$$

MAP is the mean average precision of a given ranking over N instances. The average precision *AP* is calculated from the precision at any position $k \in \{1,...,n\}$ that was rated as relevant for the instance i, denoted here by the indicator function $\mathbb{1}_{\{k \in D_{r_i}\}}$. In our case, the instances i represent the test persons.

For every technique precision and *AP* are calculated for all test persons and then averaged to have a comparative value. Somewhat problematic thereby is the comparability of different test persons. Every test person has a different affinity towards the assigned topic which results in differences in distribution and proportion of relevant documents. To bypass this circumstance the application of the *argmax* may be the best evaluation measure:

$$\underset{x \in T}{\text{argmax}}\, f(x)$$

T is the set of techniques that are evaluated. The function value $f(x)$ is the total number of test persons for which technique x performed best, either by using precision or *AP*.

technique	mean precision (variance)	*MAP* (variance)
R	56.1 (4.9)	42.1 (6.7)
R2	54.1 (5.7)	39.1 (6.4)
Ide	61.4 (6.2)	49.0 (9.6)
I2	59.3 (5.4)	45.8 (7.3)
BIM	60.0 (5.1)	45.7 (7.3)
B25	61.4 (5.0)	47.2 (7.2)
252	61.2 (5.1)	47.1 (7.4)
253	61.1 (5.3)	47.3 (7.7)
LM	59.0 (4.6)	42.5 (5.8)
LR	55.6 (4.9)	38.2 (4.9)
LR2	53.5 (4.6)	35.7 (4.7)
LI	58.5 (6.1)	42.8 (6.9)
LI2	57.1 (5.5)	41.7 (5.7)

Tab. 2: mean precision and *MAP* without compliance of category

	R	R2	Ide	I2	BIM	B25	252	253	LM	LR	LR2	LI	LI2
culture	0/0	0/0	2/1	1/1	1/1	1/0	2/1	3/1	0/0	0/0	0/0	1/0	0/0
economics	2/1	0/0	2/0	1/0	1/1	2/0	2/1	1/0	1/1	2/1	2/1	2/1	2/1
politics	2/2	4/3	1/2	2/1	0/0	1/1	0/0	0/0	1/0	0/0	0/0	1/0	0/0
society	1/1	0/0	4/4	0/0	1/1	1/1	1/1	1/1	1/1	0/0	0/0	1/1	1/1
sports	0/1	1/0	0/0	0/0	1/1	0/0	0/0	0/0	0/0	1/1	2/0	3/2	1/1
all	5/5	5/3	9/7	4/2	4/4	5/2	5/3	5/2	3/2	3/2	4/1	8/4	4/3

Tab. 3: *argmax* of precision / *argmax* of *AP*

4.3 Results

We now want to present and discuss the evaluation results. Table 2 shows the mean precision and the *MAP* for every technique without compliance of the category that the test persons were split into. In addition, the variance within the techniques is stated. The best results for mean precision are reached by Ide and by B25 with 61.4, whereby B25 shows a slightly lower variance. The *MAP* shows similar results with Ide being the best, achieving a value of 49.0. B25, however, according to this measure, performs worse (47.2). If we only focus on the techniques performed on the LDA-representation, both measures determine LI as the best. Table 3 shows the results evaluated by means of the argmax-function. The first five rows stand for the categories and the columns show the techniques. The row „all“ gives a category-independent overview of the number of test persons that each technique performed best for. For mean precision, Ide and LI clearly outperform the other techniques with a total of 9 or 8. *MAP* shows the same tendency to a lesser extent.

Mean precision and *MAP* values for the category-dependent consideration of the results are shown in Table 4 and Table 5. The columns represent the different approaches while the

	R	R2	Ide	I2	BIM	B25	252	253	LM	LR	LR2	LI	LI2
culture	53.6	50.4	68.0	66.4	59.2	62.4	64.8	67.2	57.6	54.4	49.6	65.5	57.6
economics	60.7	62.0	62.0	61.3	68.7	70.0	68.7	66.7	62.0	65.3	62.0	64.7	67.3
politics	63.3	64.7	59.3	60.7	56.0	58.7	58.7	58.7	61.3	46.7	46.0	53.3	48.7
society	60.7	56.7	76.7	66.7	64.0	70.7	71.3	71.3	67.3	64.0	60.0	62.8	65.3
sports	42.0	36.0	42.0	42.7	52.0	45.3	43.3	42.7	46.7	47.3	49.3	47.3	46.7

Tab. 4: mean precision

	R	R2	Ide	I2	BIM	B25	252	253	LM	LR	LR2	LI	LI2
culture	38.1	33.4	57.8	56.4	46.6	51.4	53.7	55.5	37.6	33.3	31.3	45.3	42.7
economics	46.8	45.6	47.6	43.7	56.6	55.8	54.6	53.1	44.7	48.4	44.1	51.4	49.5
politics	50.6	51.5	46.8	48.5	43.7	45.0	44.6	44.8	43.8	30.1	30.5	34.5	31.0
society	46.4	39.8	67.3	55.2	48.0	54.6	54.7	55.0	55.4	48.3	42.9	46.2	51.0
sports	27.9	24.2	26.8	27.2	33.8	29.9	29.1	29.3	30.1	30.3	29.1	37.1	34.4

Tab. 5: *MAP*

rows represent the five categories. For the categories culture and society, Ide leeds to the best values, having mean precision= 68.0, $MAP = 57.8$ for culture and mean precision= 76.7, $MAP = 67.3$ for society. B25 obtains a mean precision of 70.0 and BIM a *MAP* of 56.6 as best for economics. In politics, R2 holds the leading position, with mean precision= 64.7 and $MAP = 51.5$. Finally, for sports BIM yields the best mean precision (52.0) and LI yields the best *MAP* (37.1). Focusing on the worst results, for any category and for both measures surprisingly Rocchio's algorithm with the proposed parameter settings performs the worst. Table 3 shows the evaluation received by the argmax. In contrast to the category-independent consideration, no clear tendency can be observed, which underlines the difference in the application of the techniques between the categories.

In summary, a dependency of category and best performing technique is apparent on our test data and test persons. Depending on the measurement, Ide's approach and the Binary Independence Model as well as the Okapi BM25 Model perform the best, whereby Ide seems to slightly lead. Rocchio's algorithm produced the worst results according to the three measures we used, although, in a few cases, it also performed well.

5 Conclusion and Future Work

In this work different approaches for the application of relevance feedback and ad-hoc retrieval techniques have been compared on a data corpus including German news articles. Thereby the focus was on the performance of the single techniques, category-dependent as well as category-independent. We showed that the techniques are category-dependent on our data representation and that Ide's formula (Formula 2) slightly outperforms the other techniques by means of the applied measures. Surprisingly Rocchio's formula (Formula 1)

did not achieve similar results to Ide's, although this is frequently stated in the literature. Nevertheless we have to admit that the determination of a appropriate measure for relevance feedback is challenging.

In future work, we want to extend the evaluation. Further parameter settings, a larger number of test persons and other data representations could lead to more meaningful results. We also plan to analyze a larger collection and to test the results we achieved on significance. Furthermore, an evaluation over all techniques and all categories for one test person, and over several iterations, would be interesting. This would also enable to evaluate approaches that refer to rank-based result sets, for example *Ide Dec-Hi* [Id71].

References

[BNJ03] Blei, David M; Ng, Andrew Y; Jordan, Michael I: Latent Dirichlet Allocation. Journal of machine Learning research, 3(Jan), 2003.

[HR01] Hiemstra, Djoerd; Robertson, Stephen E.: Relevance Feedback for Best Match Term Weighting Algorithms in Information Retrieval. In: DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries. 2001.

[Id71] Ide, Eleanor: New Experiments in Relevance Feedback. The SMART retrieval system, 1971.

[JWR00a] Jones, K Sparck; Walker, Steve; Robertson, Stephen E.: A Probabilistic Model of Information Retrieval: Development and Comparative Experiments: Part 1. Information Processing & Management, 36(6), 2000.

[JWR00b] Jones, K Sparck; Walker, Steve; Robertson, Stephen E.: A Probabilistic Model of Information Retrieval: Development and Comparative Experiments: Part 2. Information Processing & Management, 36(6), 2000.

[MK60] Maron, M. E.; Kuhns, J. L.: On Relevance, Probabilistic Indexing and Information Retrieval. J. ACM, 7(3), 1960.

[Ri79] Rijsbergen, C. J. Van: Information Retrieval. 2nd edition, 1979.

[RJ76] Robertson, S. E.; Jones, K. Sparck: Relevance Weighting of Search Terms. Journal of the American Society for Information Science, 27(3), 1976.

[RS65] Rocchio, JJ; Salton, G: Information Search Optimization and Interactive Retrieval Techniques. In: Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I. ACM, 1965.

[RW94] Robertson, Stephen E; Walker, Steve: Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In: Proceedings of the 17th annual international ACM SIGIR conference on Research and Development in Information Retrieval. Springer-Verlag New York, Inc., 1994.

[TC89] Turtle, Howard; Croft, W Bruce: Inference Networks for Document Retrieval. In: Proceedings of the 13th annual international ACM SIGIR conference on Research and development in information retrieval. 1989.

[Zh09] Zhang, Yi: Text Mining: Classification, Clustering, and Applications. RC Press, chapter Adaptive Information Filtering, 2009.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 311

Datenschutzmechanismen für Gesundheitsspiele am Beispiel von Secure Candy Castle

Corinna Giebler[1] Christoph Stach[2]

Abstract: Smartphones sind mittlerweile ein fester Bestandteil des modernen Lebens. Sie erzeugen, speichern und verarbeiten eine Vielzahl an privaten Daten. Auch im Gesundheitssektor werden sie zunehmend eingesetzt; die dabei entstehenden Daten sind besonders schützenswert. In dieser Arbeit werden daher Konzepte eingeführt, die Nutzern die Kontrolle über ihre Gesundheitsdaten geben. Zu diesem Zweck wird *Secure Candy Castle*, ein Spiel für Kinder mit einer Diabeteserkrankung, das mit einem Berechtigungsmanagementsystem verbunden ist, vorgestellt. Der Nutzer kann den Funktionsumfang des Spiels einschränken, wodurch die App weniger Berechtigungen erhält. Zusätzlich werden für SCC Komponenten entwickelt, die die Interoperabilität von Smartphones mit medizinischen Messgeräten verbessert. Die Evaluation zeigt, dass mit SCC alle aktuellen Probleme von Gesundheits-Apps adressiert werden. Die Konzepte sind generisch und lassen sich auf beliebige andere Gesundheits-Apps anwenden.

Keywords: mHealth-Apps, Datensicherheit, Datenschutz, Datenintegration, Interoperabilität.

1 Motivation

Im Zeitalter der Smartphones werden Daten mobil. Als tragbare Computer sind Smartphones und Tablets fest in den modernen Alltag integriert. Nachrichten, Dokumente und auch Fotos können überall abgerufen, bearbeitet und gespeichert werden. Sensorik ermöglicht es, die Umwelt eines Nutzers zu erfassen und zu verarbeiten. Hierfür werden auf mobile Applikationen – kurz *Apps* – zurückgegriffen. Diese erlauben es, Daten verschiedenster Art zu generieren, anzuzeigen und zu verändern.

Zu diesen Daten gehören inzwischen auch Gesundheitsdaten. Das Stichwort *mHealth* beschreibt eine Form der Gesundheitsfürsorge, bei der Patienten mittels mobiler Technologien, wie beispielsweise Smartphones oder portablen Messgeräten, weitgehend selbstständig die Diagnostik und Therapie durchführen [KP12; Ma16]. Der Patient kann angeleitet von Apps Messwerte erheben, Aktivitäten und Mahlzeiten protokollieren und mithilfe der im Smartphone vorhandenen Sensorik weitere Umwelteinflüsse erfassen. Der so gewonnenen Selbstverantwortung der Patienten wird dabei viel Bedeutung beigemessen [KP12]. Ärzte können entlastet werden und Patienten entwickeln selbst ein Gefühl dafür, was ihnen gut tut und was schadet. Diese Art der Gesundheitsfürsorge eignet sich besonders bei chronischen Krankheiten, wie Diabetes oder Asthma [AM16]. Auch Kinder können von diesem Effekt profitieren [Mi13]. Mithilfe von *Serious Games* (dt. ernsthafte Spiele) kann sowohl Wissen

[1] Universität Stuttgart, IPVS / AS, Universitätsstraße 38, 70569 Stuttgart, giebleca@studi.informatik.uni-stuttgart.de
[2] Universität Stuttgart, IPVS / AS, Universitätsstraße 38, 70569 Stuttgart, Christoph.Stach@ipvs.uni-stuttgart.de

vermittelt als auch Verhalten trainiert werden und so Kinder dazu ermutigt werden, auf sich zu achten und Umwelteinflüsse korrekt einzuschätzen [Wi10]. Martin Knöll hat bereits in verschiedenen Arbeiten den Erfolg solcher Spiele evaluiert [Kn10; Kn14].

Durch mHealth-Apps werden jedoch sensible Gesundheitsdaten verarbeitet und gespeichert. Für den Nutzer ist oftmals unersichtlich, wie mit diesen Daten verfahren wird. Ein Grund hierfür ist, dass Datenschutzerklärungen oft fehlen [Hü15]. Auch sind viele Apps überprivilegiert, wodurch sie Zugriff auf Daten erhalten können, die für ihre Funktion unerheblich sind [Fe11]. Darum sind Datensicherheit (engl. *Security*) und Datenschutz (engl. *Privacy*) wichtige Themen bei der Verwendung von Apps – besonders im Bereich der Gesundheitsfürsorge [Ba14]. Dabei bedeutet Datenschutz die Freiheit des Nutzers, über die Verwendung seiner Daten selbst zu entscheiden, während Datensicherheit die technische Sicherheit von Daten bezeichnet, erreichbar durch beispielsweise Verschlüsselung.

Ein weiteres Problem stellt die Interoperabilität zwischen App und externen Datenquellen, wie beispielsweise Messgeräten, dar [Ch12]. Um verschiedene Patienten gleichermaßen unterstützen zu können, muss eine Vielzahl von Messgeräten, beispielsweise Blutzuckermessgeräte, mit der App verknüpfbar sein. Hierfür müssen genormte Schnittstellen geschaffen werden, die eine Einbindung beliebiger Geräte ermöglichen.

In dieser Arbeit wird daher ein Konzept vorgestellt, wie einerseits Datensicherheit und Datenschutz gewährleistet und andererseits die Interoperabilität verbessert werden. Hierfür wird die mHealth-App *Candy Castle* [Kn10] mit einem Datenschutzsystem (in diesem Fall die *Privacy Management Platform*, kurz *PMP* [SM13]) verbunden, welches Mechanismen zur feingranularen Berechtigungsvergabe zur Verfügung stellt. Durch diese Verbindung kann der Nutzer selbst entscheiden, welche Daten er mit der App teilen will und die App passt ihren Funktionsumfang entsprechend an. Mit diesem Konzept wird das Problem der Datensicherheit und des Datenschutzes auf Mobilgeräten direkt adressiert. Zudem werden Erweiterungen für die PMP vorgestellt, die über eine vereinheitlichte Schnittstelle die Integration unterschiedlicher medizinischer Datenquellen (z. B. verschiedene Blutzuckermessgeräte) unterstützen. Eine weitere Erweiterung der PMP führt eine Analyse der medizinischen Daten durch, um so neues Wissen aus gesammelten Daten ableiten zu können und als Grundlage für Therapie und Behandlung genutzt werden. Die vorliegende Arbeit basiert auf den Ergebnissen meiner Bachelorarbeit [Gi16] und erweitert diese um einige Aspekte.

Der Rest dieser Arbeit ist wie folgt aufgebaut: In Abschnitt 2 werden zunächst verschiedene Gesundheitsspiele und ihr Umgang mit sensiblen Daten betrachtet. Die PMP ist in Abschnitt 3 näher beschrieben. In Abschnitt 4 werden anhand des Ablaufes einer überarbeiteten Version von Candy Castle die verwendeten Daten analysiert und die benötigten Schutzmechanismen ausgearbeitet. Die erarbeiteten Konzepte werden anschließend in Abschnitt 5 evaluiert. In Abschnitt 6 folgt eine Zusammenfassung und ein Ausblick auf zukünftige Arbeiten.

2 Verwandte Arbeiten

Mobile Geräte und Apps können für alle Arten von Gesundheitsthemen genutzt werden [Si12]. Dadurch sind mHealth-Apps in der Lage, die gesamte Spannbreite der ärztlichen Aufgaben unterstützend zu begleiten: Prävention, Diagnose, Therapie und Nachsorge. Speziell im Bereich der Diabetesfürsorge gibt es hier bereits verschiedene Apps.

In den Bereich der Prävention und Nachsorge gehören dabei Apps, die eine aufklärende Rolle übernehmen und zu den wichtigsten Fragen bezüglich Diabetes eine Antwort liefern können (z. B. *Making Chocolate-covered Broccoli* [Gl10] oder *Power Defense* [Ba12]). Bei diesem App-Typ spielen Datenintegration oder -schutz keine Rolle, da hier nur informative Lerninhalte weitergegeben und keine nutzerbezogenen Daten verarbeitet werden. Externe Geräte müssen daher nicht angebunden werden, wodurch keine Interoperabilität nötig ist.

Für Diagnose-Apps werden Gesundheitsdaten benötigt. Anhand dieser können weitere Einsichten in den Krankheitsverlauf gewonnen werden. Beispielsweise können Apps dazu genutzt werden, um das Essverhalten eines Patienten zu analysieren [Or13]. Hier werden viele gesundheitsbezogene Daten mit der App und gegebenenfalls anderen externen Instanzen ausgetauscht. Daher sollte der Datenschutz hier eine besonders wichtige Rolle einnehmen. Auch die Interoperabilität spielt hier eine Rolle, da aus Gründen der Nutzerfreundlichkeit eine Anbindung von Messgeräten von Vorteil sein kann.

Im Bereich der Therapie werden Gesundheitsdaten analysiert, um Gegenmaßnahmen einzuleiten. Ein Beispiel hierfür ist eine automatisch gesteuerte Insulinpumpe [Cu11]. Je mehr Kontextdaten dieser zur Verfügung stehen, desto besser kann die Insulinmenge bestimmt und gegebenenfalls auf Veränderungen reagiert werden. So kann beispielsweise durch hohe Lautstärken auf einen erhöhten Stresslevel geschlossen werden, welcher sich direkt auf die benötigte Insulinmenge auswirken kann. Um möglichst genaue Diagnosen stellen zu können, werden große Mengen an Gesundheitsdaten benötigt und Datenmanipulation muss ausgeschlossen werden, da dies schwerwiegende gesundheitliche Folgen haben kann. Außerdem müssen Daten korrekt in die Analyse einfließen und Analyseergebnisse korrekt an externe Geräte weitergeleitet werden. Daher ist die Lösung des Interoperabilitätsproblems hier von enormer Wichtigkeit.

Es zeigt sich, dass Datenschutz und Datensicherheit bei Diagnose- und Therapie-Apps besonderer Aufmerksamkeit bedürfen. Die mHealth-App *mySugr*[3] ist dabei eine der wenigen Apps, die dies konsequent umsetzt und dafür zertifiziert wurde. Jedoch hat auch hier der Nutzer keine Kontrolle über seine Daten und muss darauf vertrauen, dass die App sich an die gemachten Versprechen hält. Im Folgenden wird daher beschrieben, wie die PMP erweitert werden kann, damit sie als Basis für mHealth-Spiele fungiert, den Datenschutz und die Datensicherheit gewährleistet und das Interoperabilitätsproblem löst.

Dabei ist der Anwendungsfall „Diabetes-App“ beispielhaft gewählt. Auch für andere chronische Krankheiten gibt es mHealth-Apps, z. B. für Asthma [Ni12] oder Parkinson [Sa13]. Die präsentierten Konzepte lassen sich auch auf diese Bereiche anwenden.

[3] https://mysugr.com/apps

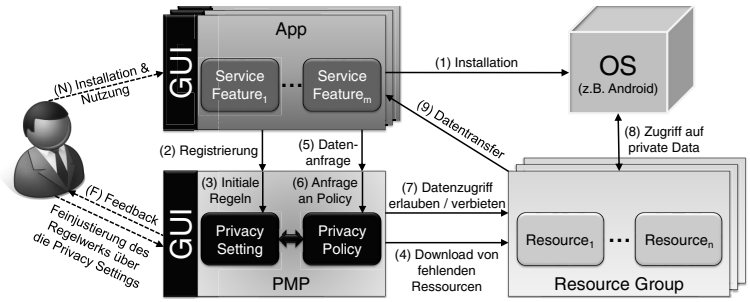


Abb. 1: Die Arbeitsweise der PMP [SM15]

3 Die PMP

Mit der PMP können Apps feingranular eingeschränkt werden. Beispielsweise kann ein Nutzer so festlegen, ob und wie eine App GPS-Daten sammeln und verarbeiten kann. Möchte er nicht, dass diese Daten gesammelt werden, kann er die entsprechende Funktion über die PMP deaktivieren. Ebenso ist es möglich, Daten mit reduzierter Genauigkeit an eine App weiterzugeben. Dies schränkt die Servicequalität der App zwar ein, der Nutzer weiß allerdings jederzeit, welche Daten von welcher App wie verwendet werden können.

Abbildung 1 stellt die Arbeitsweise der PMP dar. Damit Apps mit der PMP zusammenarbeiten können, müssen sie sich zunächst bei der PMP registrieren (1 & 2). Mithilfe von so genannten *Privacy Settings* kann der Nutzer nun beliebige Funktionen der App aktivieren, deaktivieren oder auch einschränken (3). Zu diesem Zweck bietet die App ihre grundlegenden Funktionen als *Service Features* an. Diese werden mit der PMP auf so genannte *Ressourcen* abgebildet. Ressourcen stellen dabei Schnittstellen zu Systemfunktionen oder -daten dar. Ressourcen und PMP stammen dabei aus einer vertrauenswürdigen Quelle. Zusätzliche Ressourcen können bei Bedarf nachinstalliert werden (4). Will die App ein durch die PMP verwaltetes Service Feature verwenden, muss sie die passende Methode der zugehörigen Ressource aufrufen (5). Bei einem solchen Aufruf prüft das Managementsystem der PMP, welche Privacy Settings gesetzt sind. Dies geschieht mithilfe der *Privacy Policy*, dem Regelwerk der PMP (6). Je nachdem wird die geforderte Funktion ausgeführt, Filter zur Reduktion der Datengenauigkeit angewandt oder aber Dummy-Daten an die App gesendet (7, 8 & 9). Dies verhindert einen Absturz der App, sollte ein Nutzer die angefragten Daten nicht mit der App teilen wollen. Nähere Informationen zur PMP finden sich in den zugehörigen Quellen [SM13; SM14; St15a].

4 Secure Candy Castle

Als Grundlage für den Forschungsprototypen von Secure Candy Castle (SCC), einem sicheren Gesundheitsspiel, dient das Spiel *Candy Castle* [Kn10; SS12]. Es handelt sich dabei um ein mobiles Diabetestagebuch für Kinder. Die Aufgabe des Spiels ist es, den Nutzer dazu zu motivieren, regelmäßig Messungen an möglichst verschiedenen Orten vorzunehmen. Die Verbindung zwischen Messwerten und GPS-Koordinaten kann anschließend für die Analyse

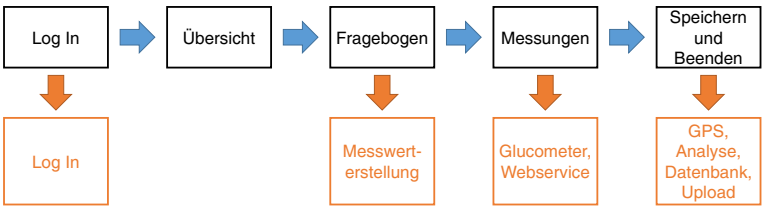


Abb. 2: Ein exemplarischer Ablauf einer Gesundheits-App

von Umwelteinflüssen auf die Behandlung von Diabetes verwendet werden [KM12]. Da es für Kinder gedacht ist, ist Candy Castle übersichtlich und verständlich gestaltet. Jede Eintragung einer Messung gibt Punkte, die den Spieler motivieren sollen. Um allerdings eine Motivation für regelmäßige Messungen zu bieten, greifen einmal täglich *dunkle Mächte* an, die dem Spieler Punkte abziehen und so neue Messungen erforderlich machen.

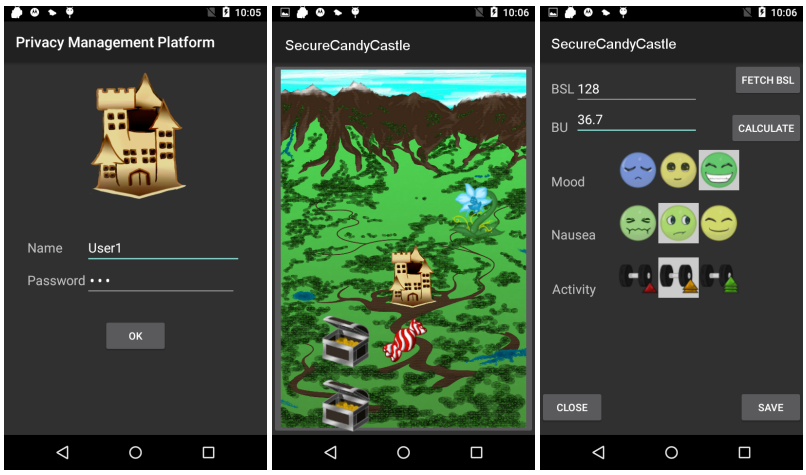
Da in einem solchen Tagebuch viele sensible Daten verwaltet werden, bietet sich der Einsatz der PMP hier an. Eltern können gemeinsam mit ihrem Kind auswählen, welche Funktionen der App erlaubt sein sollen. So kann beispielsweise die Sammlung von GPS-Daten eingeschränkt oder ganz abgeschaltet werden. Auch Berechtigungen wie zur Verbindung mit dem Internet können über die PMP bequem erteilt oder entzogen werden.

Dies bietet zwei große Vorteile: Zum einen kann die App auf den Nutzer angepasst werden. Ist z. B. die Sammlung von GPS-Koordinaten nicht erwünscht, kann diese Funktion deaktiviert werden. Auch die Verbindung zum Internet kann untersagt werden, um die Weitergabe von Daten zu verhindern. Zum anderen können potenziell bösartige Apps nicht ohne Verwendung der PMP unkontrolliert auf Daten zugreifen. Durch die Verbindung mit der PMP allerdings werden alle kritischen Funktionen, die mit sensiblen Daten arbeiten, in Ressourcen ausgelagert, die von vertrauenswürdigen Entwicklern stammen. So erhält eine App zu keinem Zeitpunkt Zugriff auf die Daten und kann auch keine Verbindung mit Dritten aufnehmen, da auch die Kommunikation mit externen oder internen Komponenten durch die PMP reglementiert wird.

Abbildung 2 zeigt den Ablauf des Prototypen. Schwarz umrahmt sind die Aktivitäten der mHealth-App dargestellt, während verwendete Ressourcen in Orange abgebildet sind. Zunächst startet die App mit einem Log In Bildschirm (siehe Abbildung 3a). Dadurch können mehrere Nutzer die App auf dem selben Gerät verwenden und es wird softwareseitig sichergestellt, dass nur autorisierte Instanzen Zugriff auf die gespeicherten Daten erhalten. Die Anmeldemaske ist als PMP-Ressource implementiert. Dies hat mehrere Vorteile: Einerseits erhält die App keinen Zugriff auf Anmeldedaten. Andererseits können Anmeldemaske und -daten dadurch in anderen Apps wiederverwendet werden, die die entsprechende Ressource einbinden. So muss sich der Nutzer nur einen Benutzernamen und ein Passwort merken, was besonders für Kinder sinnvoll ist. Dieser Ansatz ist an *OpenID*[4] angelehnt.

Nach der Anmeldung gelangt man auf den Kartenbildschirm, dargestellt in Abbildung 3b. Hier erhält der Nutzer ein visuelles Feedback zu seinem Messverhalten. Dies geschieht

[4] http://openid.net/



(a) PMP-Login (b) Spielfeld von SCC (c) Fragebogen von SCC

Abb. 3: Screenshots von SCC

mithilfe des Schlosses, das sich in der Mitte des Bildschirms befindet. Pro Messung erhält der Nutzer Punkte, die täglich in einem Angriff der dunklen Mächte um einen bestimmten Betrag verringert werden. Abhängig von dieser Punktezahl, verändert sich die Darstellung des Schlosses. Zudem geben weitere Bilder (wie Blumen, Schätze oder Bonbons) Rückmeldung über die Orte, an denen bereits Messungen vorgenommen werden. Sie stehen dabei in Relation zum Schloss, für das beliebige Koordinaten gewählt werden können. Die Verknüpfung zwischen Blutzuckermesswerten und GPS-Koordinaten ist bereits Teil des Originalkonzepts von Candy Castle und bringt neue Erkenntnisse bezüglich Umwelteinflüssen auf die Gesundheit von Diabetespatienten [KM12; Kn10].

Um Messergebnisse eintragen zu können, stellt SCC einen kindgerechten Fragebogen zur Verfügung (siehe Abbildung 3c). Der Nutzer kann hier seinen Blutzuckermesswert, die zu sich genommenen Broteinheiten sowie Laune, Übelkeit und Aktivität erfassen. All diese Werte ermöglichen es dem behandelnden Arzt, genaue Diagnosen und Prognosen zu erstellen; hierbei werden viele sensible Daten gesammelt. Dies ist durch eine sichere Messwertressource realisiert. Diese erfasst die eingetragenen Daten und fügt sie einem Messwertobjekt hinzu, welches verschlüsselt wird. Andere Ressourcen können dieses Objekt entschlüsseln und verarbeiten. Es ist somit sicher vor unbefugtem Zugriff durch die App; diese erhält ausschließlich abstrahierte Daten.

Aus dem Fragebogen heraus ist es dem Nutzer zudem möglich, ein Blutzuckermessgerät per Bluetooth zu verbinden oder sich Broteinheiten über einen Webservice berechnen zu lassen. Diese beiden Funktionen werden ebenfalls durch Ressourcen ausgeführt. Die Glucometer-Ressource regelt den Zugriff auf beliebige Bluetooth-fähige Blutzuckermessgeräte. Standardisierte Protokolle, wie das *Bluetooth SIG Health Device Profile* (siehe ISO/IEEE 11073-20601 Norm), werden von der Ressource direkt unterstützt. Für andere Messgeräte kann diese Ressource erweitert und ähnlich eines Gerätetreibers nachinstalliert

werden. Der Abruf der Broteinheiten wird über eine Webservice-Ressource durchgeführt. In der Ressource ist explizit eine externe Datenquelle angegeben, in der diese Information zu unterschiedlichen Lebensmitteln hinterlegt ist. Dadurch ist sichergestellt, dass die App zu keiner anderen Adresse eine Verbindung aufbauen kann.

Zuletzt wird der eben erstellte Messwert gespeichert. Hierfür sind verschiedene Ressourcen eingebunden. Zunächst werden GPS-Koordinaten erfasst und in den Messwert eingefügt. Hierbei kann der Nutzer in der PMP eine Genauigkeit festlegen, mit der der Ort erfasst werden soll. Anschließend wird der eben erstellte Messwert analysiert und eine Meldung an eine angegebene Kontaktperson versandt, sollte der Blutzuckerwert eine bestimmte Grenze unterschreiten. Auch diese Analyse erfolgt in einer Ressource, so dass die App weder Zugang zu den Gesundheitsdaten noch auf die Kontakte benötigt.

Der Messwert wird auf dem Gerät gespeichert und an ein Analyse-Backend für medizinische Daten wie die *ECHO-Plattform* [St15b] übermittelt, auf das der behandelnde Arzt Zugriff hat. Da in einem solchen Backend sehr viele unterschiedliche Sensordaten akkumuliert werden, ist es möglich, dass aus der Kombination dieser Daten neue Informationen über die Patienten abgeleitet werden können. Beispielsweise lassen sich aus Bewegungsprofilen mehrerer Patienten Rückschlüsse auf deren soziale Beziehung ziehen. Um die Geheimhaltung derartiger privater Informationen kümmert sich das PATRON Projekt[5], weshalb mHealth-Apps wie SCC hierfür als idealer Anwendungsfall dienen.

Da all diese Funktionen in der PMP ausgeführt werden, hat die App keinen Zugriff auf sensible Daten wie Aufenthaltsort, Kontaktperson und medizinische Daten. Auch kann sie nicht auf gespeicherte Messwerte zugreifen. Diese werden durch Verwendung des *Secure Data Container (SDC)*, einem verschlüsselten Datencontainer für die PMP [SM15; SM16], geschützt. SCC wurde in Teilen auf der IEEE MDM 2016 demonstriert [St16].

5 Diskussion des Konzeptes von Secure Candy Castle

Nachdem SCC vorgestellt wurde, wird es in dem folgenden Abschnitt evaluiert. Hierfür wird ein Anforderungskatalog von Patienten an Diabetes-Apps herangezogen [Al15] und um die Punkte Datenschutz, Datensicherheit und Interoperabilität erweitert.

Datenschutz: Durch die PMP kann der Nutzer selbst entscheiden, welche Funktionen der App er zulässt und welche er verbietet. Da die App selbst über keine Berechtigungen verfügt und die Ressourcen der PMP aus vertrauensvollen Quellen stammen, kann er sich sicher sein, dass seine Daten nur wie von ihm gewünscht verwendet werden.

Datensicherheit: Mit verschiedenen Datensicherheitskonzepten kann die Datensicherheit gewährleistet werden. Durch die Nutzung des *SDCs* [SM15; SM16] und der Verschlüsselung von Datenobjekten können unautorisierte Instanzen nicht auf die Daten zugreifen.

Interoperabilität: Da in SCC jede Kommunikation mit externen Services über Ressourcen erfolgt, können diese einfach auf die verwendeten Technologien angepasst werden. So

[5] http://patronresearch.de/

kann beispielsweise für verschiedene Messgeräte je eine Ressource zur Verfügung gestellt werden, die eingebunden werden kann. Die App bleibt hiervon unbeeinträchtigt.

Automatischer Logging Prozess: Durch die Anbindung eines externen Messgerätes kann das Erfassen der Messwerte zu einem großen Teil automatisch ablaufen,

Vorhersagen: Die App bindet eine Analyse-Ressource ein, die ein direktes Feedback zu den eingegebenen Messwerten liefern kann. Zudem werden die Daten an einen externen Service weitergeleitet, in dem der behandelnde Arzt ebenfalls Analysen durchführen und Rückmeldungen an die Patienten geben kann.

Intuitive Navigation: Da es sich bei SCC um ein Kinderspiel handelt, wurde bei der Konzeption besonderes Augenmerk auf Verständlichkeit gelegt. Viele Bilder, wenig Text und klare Funktionen bieten intuitive Möglichkeiten, die App zu bedienen.

Unterstützung für neue Geräte und Frameworks: Durch die Nutzung von erweiterbaren Ressourcen zur Kommunikation mit externen Geräten kann die App mit beliebigen Messgeräten kommunizieren.

Verständliche Daten: In SCC werden dem Patienten keine medizinische Daten wie Blutzuckerwerte angezeigt. Diese Daten werden von speziellen Ressourcen analysiert und dem Nutzer wird eine zielgruppengerechte grafische Aufbereitung präsentiert.

Haptisches oder hörbares Feedback: Um diese Funktion zu SCC hinzuzufügen, kann eine Ressource eingebunden werden, die auf die Vibrationsfunktion des mobilen Gerätes zugreift, um so zusätzlich zum visuellen auch ein haptisches Feedback geben zu können.

Positive Ermutigung: Diese geschieht im vorgestellten Konzept mithilfe des Schlosses, welches durch regelmäßige Messung intakt bleibt, sowie durch Bilder, die auftauchen, wenn neue Orte erschlossen wurden.

Erinnerungen: Die Erinnerung geschieht in SCC durch den Verfall des Schlosses. Bricht das Schloss zusammen, muss dringend eine neue Messung erfolgen. Auch könnten Push-Menüs eingebaut werden, um den Nutzer an eine Messung zu erinnern.

Integration von Sensorik: In SCC können über die Ressourcen alle in Smartphones verbauten Sensoren angesprochen werden. Zusätzlich ermöglichen Ressourcen die Kommunikation mit externen Datenquellen, wie beispielsweise tragbare Sensoren.

Launen-Erfassung: Im Fragebogen von SCC können Laune, Übelkeit und Aktivität in jeweils drei Stufen erfasst werden.

SCC erfüllt somit alle Anforderungen an eine Gesundheits-App für Diabetiker.

6 Zusammenfassung

In dieser Arbeit wird das Konzept einer sicheren mHealth-App für Kinder vorgestellt. Durch die Verbindung des Spiels Candy Castle mit der Privacy Management Plattform entsteht eine

sichere mHealth-App namens Secure Candy Castle, in welcher der Nutzer selbst entscheidet, welche Funktionen er verwenden möchte. Dadurch kann er selbstständig den Zugriff der App auf seine sensiblen Daten einschränken. Darüber hinaus kann durch die Nutzung der PMP die Interoperabilität mit externen Messgeräten sichergestellt werden, da diese über erweiterbare Ressourcen ohne das Zutun der App angesprochen werden. Die Evaluation zeigt, dass SCC alle Anforderungen an Datensicherheit, -schutz und Interoperabilität sowie die allgemeinen Anforderungen an mHealth-Apps erfüllt.

Dieses Konzept ist auf andere mHealth-Apps anwendbar. Somit können durch die hier vorgestellte Verbindung zwischen App und PMP weitere sichere Gesundheitsanwendungen erstellt werden, die der Nutzer ohne Angst um seine Daten verwenden kann.

Danksagung

Diese Arbeit wurde im Rahmen des PATRON Projekts durchgeführt, gefördert durch die Baden-Württemberg Stiftung gGmbH.

Literatur

[Al15] Alexander, S.: mHealth Technologies for the Self-management of Diabetes in the Older Population, SIGACCESS Access. Comput./111, S. 14–18, 2015.

[AM16] Alkhushayni, S.; McRoy, S.: mHealth Technology: Towards a New Mobile Application for Caregivers of the Elderly Living with Multiple Chronic Conditions (ELMCC). In: DH ’16, 2016.

[Ba12] Bassilious, E. et al.: Power Defense: A Serious Game for Improving Diabetes Numeracy. In: CHI EA ’12, 2012.

[Ba14] Bai, Y. et al.: Issues and Challenges in Securing eHealth Systems, Int. J. E-Health Med. Commun. 5/1, S. 1–19, 2014.

[Ch12] Chan, M. et al.: Smart Wearable Systems: Current Status and Future Challenges, Artif. Intell. Med. 56/3, S. 137–156, 2012.

[Cu11] Cukierman-Yaffe, T. et al.: Key elements for successful intensive insulin pump therapy in individuals with type 1 diabetes, Diabetes Research and Clinical Practice 92/1, S. 69–73, 2011.

[Fe11] Felt, A. P. et al.: Android Permissions Demystified. In: CCS ’11, 2011.

[Gi16] Giebler, C.: Privatheit im Gesundheitsspiel Candy Castle, Bachelorarbeit, Universität Stuttgart, 2016.

[Gl10] Glasemann, M. et al.: Making Chocolate-covered Broccoli: Designing a Mobile Learning Game About Food for Young People with Diabetes. In: DIS ’10, 2010.

[Hü15] Hübner, M.: Gesundheits-Apps werden für Chroniker wichtig, Studie, Ärzte Zeitung online, Juni 2015, URL: https://goo.gl/jUym5W.

[KM12] Knöll, M.; Moar, M.: The Space of Digital Health Games, Int. J. Comp. Sci. Sport 11/1, 2012.

[Kn10] Knöll, M.: "On the Top of High Towers . . . " Discussing Locations in a Mobile Health Game for Diabetics. In: MCCSIS '10, 2010.

[Kn14] Knöll, M. et al.: Wo die Monster leben? Welche Orte und Begleitung haben Einfluss auf das Blutzuckermessen und können zur Entwicklung von Serious Games für Typ-1-Diabetiker beitragen, Diabetologie und Stoffwechsel 9/1, 2014.

[KP12] Klasnja, P.; Pratt, W.: Healthcare in the Pocket: Mapping the Space of Mobile-phone Health Interventions, J. of Biomedical Informatics 45/1, 2012.

[Ma16] Matusiewicz, D.: Mobile Health, Definition, Gabler Wirtschaftslexikon, 2016, URL: https://goo.gl/AWQNJM.

[Mi13] Miller, A. D. et al.: Design Strategies for Youth-focused Pervasive Social Health Games. In: PervasiveHealth '13, 2013.

[Ni12] Nikkila, S. et al.: Wind Runners: Designing a Game to Encourage Medical Adherence for Children with Asthma. In: CHI EA '12, 2012.

[Or13] Orji, R. et al.: LunchTime: A Slow-casual Game for Long-term Dietary Behavior Change, Personal Ubiquitous Comput. 17/6, S. 1211–1221, 2013.

[Sa13] Sanders, T. H. et al.: Remote Smartphone Monitoring for Management of Parkinson's Disease. In: PETRA '13, 2013.

[Si12] Siewiorek, D.: Generation smartphone, IEEE Spectrum 49/9, S. 54–58, 2012.

[SM13] Stach, C.; Mitschang, B.: Privacy Management for Mobile Platforms – A Review of Concepts and Approaches. In: MDM '13, 2013.

[SM14] Stach, C.; Mitschang, B.: Design and Implementation of the Privacy Management Platform. In: MDM '14, 2014.

[SM15] Stach, C.; Mitschang, B.: Der Secure Data Container (SDC) – Sicheres Datenmanagement für mobile Anwendungen, Datenbank-Spektrum 15/2, S. 109–118, 2015.

[SM16] Stach, C.; Mitschang, B.: The Secure Data Container: An Approach to Harmonize Data Sharing with Information Security. In: MDM '16, 2016.

[SS12] Stach, C.; Schlindwein, L. F. M.: Candy Castle — A Prototype for Pervasive Health Games. In: PerCom '12, 2012.

[St15a] Stach, C.: How to Deal with Third Party Apps in a Privacy System — The PMP Gatekeeper. In: MDM '15, 2015.

[St15b] Steimle, F. et al.: Design and Implementation Issues of a Secure Cloud-Based Health Data Management System. In: SummerSOC '15, 2015.

[St16] Stach, C.: Secure Candy Castle — A Prototype for Privacy-Aware mHealth Apps. In: MDM '16, 2016.

[Wi10] Wiemeyer, J.: Gesundheit auf dem Spiel? – Serious Games in Prävention und Rehabilitation, Deutsche Zeitschrift für Sportmedizin 61/11, 2010.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 321

Duplikaterkennung in der Graph-Processing-Platform GRADOOP

Florian Pretzsch[1]

Abstract: Die zunehmende Bedeutung von Graphdaten im Kontext von Big Data erfordert wirksame Verfahren zur Erkennung von Duplikaten, d. h. Knoten, welche das selbe Realweltobjekt repräsentieren. Dieser Beitrag stellt die Integration von Techniken zur Duplikaterkennung innerhalb des Graphverarbeitungs-Frameworks GRADOOP vor. Dazu werden dem GRADOOP-Framework neue Operatoren zur Duplikaterkennung hinzugefügt, die u. a. in der Lage sind, Ähnlichkeiten zwischen Knoten von einem oder mehreren Graphen zu bestimmen und ermittelte Duplikate als neue Kanten zu repräsentieren. Das vorgestellte Konzept wurde prototypisch implementiert und evaluiert.

Keywords: GRADOOP, Duplikaterkennung, Similarity, Blocking, Lastbalancierung, Graphen

1 Einleitung

Die graphbasierte Speicherung und Verarbeitung großer Datenmengen gewinnt zunehmend an Bedeutung, z. B. zur Analyse großer sozialer Netzwerke [Cu13] oder im Bereich Business Intelligence [Gh15]. Die Verarbeitung großer Graphen hat eine Vielzahl von Forschungsarbeiten hervorgebracht, u. a. zur Speicherung von Graphen in Graphdatenbanken (z. B. Neo4J [neo13]) oder verteilte Systeme zur Definition und Ausführung von Graphalgorithmen (z. B. Pregel [Ma10]). Analog zum Data Warehousing benötigt die effektive und effiziente Graphanalyse jedoch die Unterstützung eines kompletten Prozess von der Erstellung eines Graphen, seiner Verarbeitung und der Analyse innerhalb eines Workflows. Das Graph-Analyse-Framework GRADOOP (Graph analytics on Hadoop) [Ju15] unterstützt die Definition solcher Workflows, welche mittels existierender Big Data Technologien (u. a. Apache Flink und HBase) effizient und verteilt ausgeführt werden können. Die Definition von Workflows erfolgt dabei unter Verwendung vordefinierter Operatoren.

Neben der Erstellung und Verknüpfung von Graphen müssen solche Workflows Vorkehrung zur Sicherung der Datenqualität treffen, damit aufbauende Analysen valide sind. Ein wesentlicher Schritt zur Sicherung der Datenqualität ist Duplikaterkennung [FS69], d. h. das automatische Identifizieren von Knoten in einem oder mehreren Graphen, welche das selbe Realweltobjekt repräsentieren. Eine Vielzahl von Forschungsarbeiten beschäftigen sich mit der automatischen Erkennung von Duplikaten (siehe z. B. [KR10] für einen Vergleich von Frameworks), u. a. in einem verteilten Shared-Nothing-Cluster [KTR12].

Dieser Beitrag stellt eine Konzept zur Duplikaterkennung innerhalb des GRADOOP-Frameworks vor. Dazu wurden neue GRADOOP-Operatoren für die Duplikaterkennung

[1] Institut für Kommunikationstechnik, HfTL, Gustav-Freytag-Str. 43-45, 04277 Leipzig, s06626@hft-leipzig.de

konzipiert und implementiert, welche in GRADOOP-Workflows (gemeinsam mit den bisherigen Operatoren) verwendet werden können. Die neuen Operatoren erweitern GRADOOP um die Möglichkeit zur Berechnung von Ähnlichkeiten zwischen Knoten, die mit Kanten verbunden werden, anhand derer dann eine Selektion von Duplikatpaaren durchgeführt werden kann. Auch soll es möglich sein, zur Ähnlichkeitsberechnung Nachbarknoten mit einzubeziehen. Der Fokus dieser Arbeit liegt jedoch auf der Ähnlichkeitsberechnung anhand von Knotenattributen. Die rechenintensive Ähnlichkeitsberechnung kann dabei analog zu [KTR12] auf verschiedene Server verteilt werden, um eine gleichmäßige Auslastung aller Server zu erreichen (Lastbalancierung). Zur Repräsentation von Duplikaten werden Kanten zwischen denen als Duplikat erkannten Knoten eingefügt, so dass das Ergebnis der Duplikaterkennung für die weiteren Schritte des Graph-Analyse-Workflows (z. B. Datenfusion, Berechnung abgeleiteter Werte) zur Verfügung steht.

Abschnitt 2 stellt die Grundlagen der Duplikaterkennung sowie von GRADOOP vor, ehe Abschnitt 3 das Konzept der neuen Operatoren vorstellt. Abschließend beschreibt Abschnitt 4 die prototypische Implementation inklusive einer kurzen Evaluation.

2 Grundlagen

2.1 Duplikaterkennung

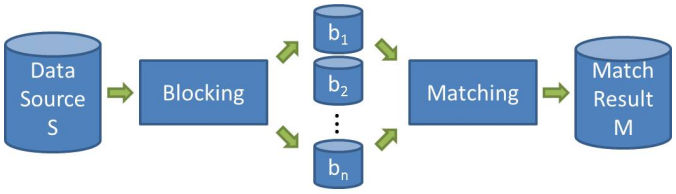


Abb. 1: Schematische Darstellung der Duplikaterkennung [Ko14]

Das Problem der Duplikaterkennung (u. a. auch bekannt als Entity/Object Matching) wurde 1969 von Felligi und Sunter formuliert [FS69]. Ziel ist es, gleiche Objekte trotz ihrer unterschiedlichen Repräsentation in einer oder mehreren Datenquellen effizient zu identifizieren. Abbildung 1 zeigt die schematische Darstellung der Duplikaterkennung innerhalb einer Datenquelle. In einem ersten Schritt (Blocking) werden aus Performancegründen die Datenobjekte in (ggf. überlappende) Blöcke eingeteilt, so dass die anschließende Duplikaterkennung (Matching) nur innerhalb der jeweiligen Blöcke realisiert wird. Beispielsweise könnten in einem Graphen, der Bücher und Autoren beinhaltet, die Datenobjekte nach diesen beiden Kriterien unterteilt und innerhalb der Blöcke nach Duplikaten gesucht werden. Die meisten Verfahren [KR10] setzen beim Matching auf Ähnlichkeitsberechnungen mittels Attributwerten sowie Kontextinformationen, anhand derer eine Gesamtähnlichkeit im Intervall [0,1] bestimmt wird. Ein Wert von *0* würde dabei völlige Unähnlichkeit ausdrücken, während *1* für totale Übereinstimmung (Gleichheit) steht. Der Einsatz eines Blocking-Verfahrens reduziert damit im Allgemeinen die Anzahl der notwendigen Ähnlichkeitsberechnung deutlich. Das finale Ergebnis (Match Result) ergibt sich aus einer

Filterung der paarweisen Ähnlichkeitswerte. Ein typisches Beispiel ist die Verwendung aller Datenobjektpaare, deren Ähnlichkeit über einem definierten Schwellwert liegt.

2.2 GRADOOP-Framework

GRADOOP unterstützt die effiziente Integration, Analyse und Repräsentation von Graphenstrukturen [Ju15]. Basierend auf dem *Extended Property Graph Data Model* (EPGM), welches Graphen mit Knoten, Kanten und Attributen abbildet, werden Operatoren zum Analysieren einzelner Graphen und Verarbeiten von Graph-*Collections* zur Verfügung gestellt. Neben einer domainspezifischen Sprache (*Graph Analytical Language* – GrALa) zur Definition von Workflows ist ein weiteres Merkmal die verteilte Speicherung der Graphen in HBase, wodurch mittels Hadoop die parallele Abarbeitung der Workflows auf unterschiedlichen Servern ermöglicht wird.

Abbildung 2 (links) zeigt die elementaren Schritte eines Ende-zu-Ende-Graphanalyse-Workflows, welche die Integration der Daten aus heterogenen Quellen in ein Graphdatenformat (*Data Integration*), die Graphanalyse mittels Operatoren (*Graph Analytics*) und die grafische Darstellung der Ergebnisse (*Representation*) umfasst. Die Duplikaterkennung wird demnach in der Phase *Data Integration* durchgeführt.

Das Framework ist in Java programmiert und nutzt Apache Flink als Datenstreaming-Engine. Die Architektur von GRADOOP ist in Abbildung 2 (rechts) dargestellt. Neben den bereits genannten Hauptkomponenten HBase (*Distributed Graph Data Store*) und EPGM stellt die *Operator Implementations*-Komponente die wesentlichen Operatoren zur Datentransformierung und Ergebnisausgabe zur Verfügung. Die *Workflow Execution* ermöglicht die parallele Ausführung von Datenintegrations- und Analyse-Workflows. Über die *Operator Implementation*-Komponente wird die Ausführung überwacht und stellt Ausführungsinformationen für die Benutzer zur Verfügung. Die oberste Schicht (*Workflow Declaration* und *Representation*) der Architektur stellt eine Umgebung zur deklarativen bzw. visuellen Erstellung und Definition von Workflows bereit. Des Weiteren werden hier die Ergebnisse als Graph visualisiert oder in Form von Tabellen und Diagrammen dargestellt.

GRADOOP stellt eine Reihe von Operatoren zur Verfügung, die ein oder zwei Graphen bzw. Graph-*Collections* verarbeiten. Beispielsweise kombiniert der Operator *Combination* zwei Graphen zu einem Graphen. In GrALa-Notation schreibt man dafür `result = graph1.combine(graph2)`, wodurch aus den zwei Graphen (`graph1` und `graph2`) der Graph (`result`) als Rückgabewert zurückgeliefert wird. Eine vollständige Liste aller Operatoren ist in [Ju15] zu finden.

3 Konzeption

3.1 Workflow

Abbildung 3 zeigt den schematischen Workflow zur Duplikaterkennung mit GRADOOP, welcher sich an dem allgemeinem Verfahren zur Duplikaterkennung (siehe Abbildung 1)

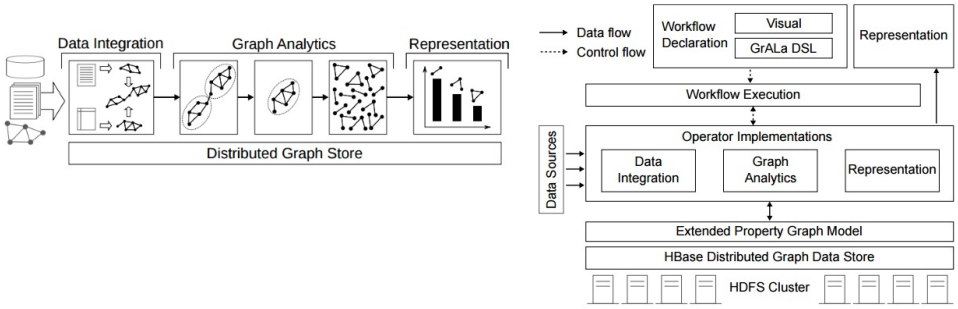


Abb. 2: GRADOOP: Schematische Darstellung eines Workflows (links) sowie der Architektur (rechts) [Ju15]

orientiert. Die Eingabe sind zwei Graphen, in denen nach Duplikaten bei den Knoten gesucht wird. Als Ergebnis entsteht ein gemeinsamer Graph, der beide Eingabegraphen enthält und um Kanten zwischen als Duplikat erkannten Knoten ergänzt wird.

Der Workflow ist modular aufgebaut und die einzelnen Schritte werden in separaten, parametrisierten Operatoren ausgeführt, welche (neben Operator-spezifischen Parametern) als Ein- und Ausgabedaten Graphen bzw. Graph-*Collections* verwenden. Dadurch wird eine nahtlose Integration in das GRADOOP-Konzept erreicht, da bestehende GRADOOP-Operatoren wiederverwendet werden können (z. B. *Union*) und neu eingeführte Operatoren auch in anderen Workflows zur Anwendung kommen können. Gleichzeitig kann der gesamte Workflow als ein *großer* Operator gekapselt werden. Die einzelnen Operatoren werden zur Laufzeit mit Hilfe von Apache Flink-Operatoren abgebildet und umgesetzt. Die effiziente und effektive Verarbeitung sowie Verteilung auf die verschiedenen Serverknoten übernimmt dabei Apache Flink abhängig von der Anzahl der Server und konfigurierten Parallelität.

Der Workflow beginnt mit dem *Blocking*-Operator, der die zwei Eingabe-Graphen entgegen nimmt. Als Rückgabe werden die anhand eines Blockschlüssels gruppierten Knoten innerhalb einer Graph-*Collection* als eigener Subgraph je Block zurückgegeben, die sich ggf. auch überlappen können. Die einzelnen Graphen der Graph-*Collection* werden im parametrisierten *Similarity*-Operator weiterverarbeitet. Die Ähnlichkeitsberechnung von Knoten, die als Entitäten angesehen werden, erfolgt z. B. anhand der String-Ähnlichkeit der Werte gleichnamiger Knotenattribute. Das Ergebnis der Ähnlichkeitsberechnung wird als Kante zwischen den zwei beteiligten Knoten dargestellt, wo der Ähnlichkeitswert als Attribut der Kante hinzugefügt wird. Die Vorteile der Kantenrepräsentation sind, dass Duplikate im weiteren Verlauf der Workflows leicht erkannt werden können und dass der Graph gesamtheitlich weiterhin besteht, ohne das Informationen verloren gehen. Im anschließenden *Selection*-Schritt werden die paarweisen Ähnlichkeitswerte gefiltert und das Match Result als Graph-*Collection* zurückgegeben. In zukünftigen Arbeiten kann hier auch eine Informationsfusion erfolgen, d. h. als Duplikat erkannte Knoten zu einem (Super-)Knoten zusammengeführt werden. Die Rückgabe-*Collection* kann durch den bereits existierenden *Union*-Operator mit den beiden Eingangsgraphen beim *Blocking*-Operator zu einem Ergebnisgraph zusammengefügt werden.

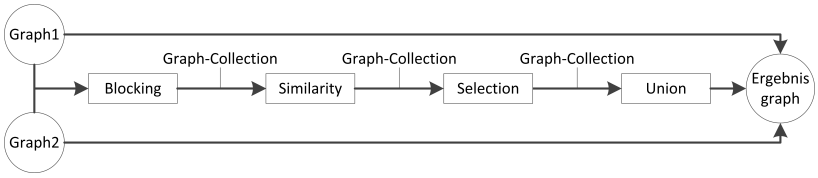


Abb. 3: Konzeptioneller Entwurf des Operator-basierten Workflows zur Duplikaterkennung

Operator	Parameter	Mögliche Werte
Blocking	Angabe des Blocking-Verfahrens	kein Blocking, Attributwert, Blockschlüssel
	Angabe eines Blocking-Attributs	*name*
	Angabe des Knotentyp-Mappings	originaler Knotentyp
	Definition Lastbalancierung je Knotentyp	ja, nein
Similarity	Ähnlichkeitsalgorithmus	Ähnlichkeitsmaß, z. B. Jaro-Winkler
	Angabe von Vergleichsattributen und Schema-Mapping je Knotentyp	Paare von Attributnamen (Standard: alle gleichnamigen Attribute)
	Anzahl Tiefe Nachbarschaftsknoten	0 (= keine Nachbarknoten) oder höher
Selection	Definition Selektionsverfahren	schwellwertbasiert (0<=s<=1), Top-k (k>0)
	Label	*duplicate*
	Kantenattribut	*value* = "Ähnlichkeitswert"
Union	*parameterlos*	

Tab. 1: Übersicht Teil-Operatoren und zugehörige Parameter

3.2 Parametrisierung

Die Operatoren können durch Parameter an die spezifischen Daten (Domäne) angepasst werden, um z. B. zu definieren, für welche Knotentypen Duplikate erkannt werden sollen und welche Ähnlichkeitsfunktion verwendet werden soll. Tabelle 1 gibt für jeden Operator mögliche Parameter sowie ggf. beispielhafte Werte an.

Der Blocking-Operator unterstützt die Bildung von Blöcken anhand eines Knotentyps und/oder eines Knotenattributs (alle Knoten mit dem gleichen Typ bzw. Attributwert bilden einen Block) sowie eines Blockschlüssels, der sich aus verschiedenen Attributwerten zusammensetzt. Dabei ist es möglich, für jeden Knotentyp ein individuelles Blocking zu ermöglichen, d. h. z. B. Autoren werden nur nach Knotentyp gruppiert, während die Publikationen mittels eines zusammengesetzten Blockschlüssels in Blöcke eingeteilt werden. Da es vorkommen kann, dass die beiden Eingangsgraphen für inhaltlich gleiche Knoten unterschiedliche Knotentypbezeichnungen verwenden, kann die Angabe des Knotentyps um ein Typ-Mapping erweitert werden. Dabei kann angegeben werden, welcher Knotentyp aus dem ersten Graphen zu welchem Knotentyp des zweiten Graphen korrespondiert. Zusätzlich kann für den Blocking-Operator eine Untergruppierung angegeben werden, welche eine Lastbalancierung ermöglicht (Details siehe Abschnitt 3.3).

Zur Ähnlichkeitsberechnung müssen die zu vergleichenden Attribute je Knotentyp angegeben werden. Dabei kann die Bezeichnung der Attribute unterschiedlich sein, wodurch ein Schema-Mapping erforderlich wird. Hierbei werden Paare von den zu vergleichenden Attributen beider Knoten angegeben. Standardmäßig sollen alle Attribute verwendet

werden, die die gleiche Bezeichnung haben und in beiden Knoten vorkommen. Auch die Angabe, welcher Vergleichsalgorithmus für welches Attribut verwendet werden soll, kann realisiert werden, da die verschiedenen Vergleichsalgorithmen [GM13] unterschiedlich gut geeignet sind. So ist z. B. für Namen die Jaro-Winkler-Ähnlichkeit [Wi06] und für längere Zeichenketten die Jaccard-Ähnlichkeit geeignet. Des Weiteren kann die Tiefe der Nachbarschaft, also alle Nachbarknoten innerhalb eines Abstands vom zu vergleichenden Knoten, die dann zur Ähnlichkeitsberechnung mit herangezogen werden sollen, angegeben werden.

Die Angabe des Selektionsverfahren ist ebenfalls über einen Parameter steuerbar. Standardmäßig wird ein schwellwertbasiertes Verfahren verwendet, welches alle Duplikatkanten mit einem Ähnlichkeitswert über einem Schwellwert (z. B. 90%) als Duplikatkante identifiziert. Die Eigenschaften der Duplikatkanten (Name, Gewicht) sind ebenfalls konfigurierbar.

3.3 Blocking & Lastbalancierung

Durch das Blocking soll der Suchraum durch Einteilung in Gruppen (Blöcke) reduziert werden, da die Anzahl der paarweisen Vergleiche quadratisch mit der Anzahl der Knoten ansteigt (siehe dazu [Ko14], S. 58). Die Aufteilung in Gruppen erlaubt es, dass die Ähnlichkeitsberechnungen nur noch zwischen Knoten der selben Gruppen durchgeführt werden müssen.

Anhand eines Attributs *gender* könnten bspw. in Personengraphen die Knoten nach dem Geschlecht gruppiert werden. Problem ist, dass nicht jeder Knoten die gleichen Attribute enthalten muss. Durch einen Knotentyp, den jeder Knoten in GRADOOP besitzt, könnten Knoten in einem Autorengraphen nach Autoren, Büchern und Verlagen gruppiert werden. Die Anzahl der Gruppen ist hier jedoch eingeschränkt. Die Blockanzahl erhöht werden kann durch einen zusammengesetzten Blockschlüssel, bestehend aus dem Knotentyp sowie einem einheitlichen Attribut innerhalb des jeweiligen Knotentyps. Zum Beispiel könnte die Autorengruppe durch Hinzufügen des Anfangsbuchstaben vom Namen um den Faktor 26 vergrößert werden. Des Weiteren könnten Knoten anhand von Nachbarknoten eingeteilt werden, um bspw. alle Bücher einer Gruppe zuzuweisen, die den gleichen Autor besitzen.

Tabelle 2 stellt die einzelnen Verfahren anhand bestimmter Merkmale gegenüber. Durch das Blocking werden potenziell weniger Duplikate gefunden, falls gleiche Knoten in unterschiedlichen Gruppen einsortiert werden. Ein auf die Daten angepasstes Blockingverfahren ist dabei entscheidend.

Neben der Reduzierung der Anzahl der paarweisen Vergleiche ermöglicht Blocking auch die effiziente Ausführung eines Workflows in einer verteilten Cloud-Umgebung. Abbildung 4 illustriert die Ausführung schematisch. Nach der Partitionierung der Eingangsdaten durch Blocking ist eine parallele (d. h. unabhängige) Bearbeitung der Blöcke voneinander möglich.

Insbesondere wenn Attributwerte sowie Knotentypen, welche zum Blocking herangezogen werden, sehr ungleich verteilt sind (Data Skew), entstehen jedoch (wenige) große Blöcke, die einer effizienten Verarbeitung in einem Cluster entgegenstehen. Da die Bearbeitung

Merkmal	Möglichkeiten Blocking			
	Knotenattribut	**Knotentyp**	**zusammengesetzter Blockschlüssel**	**ohne Blocking**
Anzahl Gruppen	mittel (abhängig von Wertebereich)	wenig	hoch	eine
Reduction Ratio	mittel	niedrig (bei Ungleichverteilung von Knotentypen sehr niedrig)	hoch	keine
Pairs Completeness	hoch, abhängig von Attribut	hoch	mittel	maximal

Tab. 2: Möglichkeiten für das Blocking im GRADOOP-Umfeld

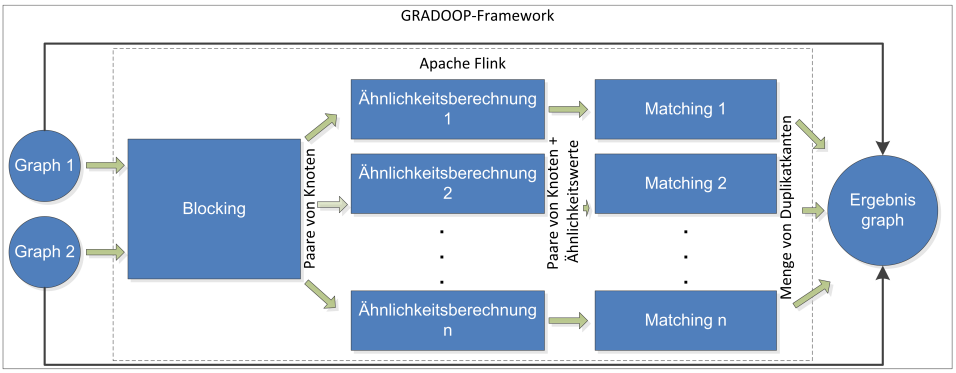


Abb. 4: Schematische Darstellung der parallelen Ausführung von Dupliklaterkennungs-Workflows mit Apache Flink

eines Blocks (paarweiser Vergleich) auf einem Serverknoten erfolgt, kann ein Data Skew zu erheblich unterschiedlichen Laufzeiten auf den einzelnen Servern führen. Basierend auf den Arbeiten zu Dedoop unterstützt der Blocking-Operator eine Untergruppierung ähnlich der in Dedoop vorgestellten Lastbalancierungsstrategie BlockSplit. [KTR12]

Dabei werden (große) Blöcke logisch in disjunkte Teilblöcke (Untergruppen) zerlegt, die dann paarweise miteinander verglichen werden. Abbildung 5 zeigt das Konzept für einen Block, welcher in vier Teilblöcke unterteilt wird. Die Knoten des Blocks aus dem ersten Graphen (A) sowie des zweiten Graphen (B) werden zunächst in insgesamt vier (gleichgroße) Hilfsgruppen zerlegt, die dann paarweise zu Untergruppen zusammengeführt werden. Damit wird sichergestellt, dass weiterhin alle Knotenpaare eines Blocks miteinander verglichen werden, aber nicht Knoten aus dem gleichen Graphen zusammengefügt werden. Das vorgestellte Verfahren wird durch die Angabe der Anzahl der Hilfsgruppen pro Block definiert und stellt eine vereinfachte Variante der BlockSplit-Strategie [KTR12] dar, welche die Zerlegung anhand der Blockgröße durchführt (und damit nur große Blöcke zerlegt). Die Übertragung der Lastbalancierungsstrategien aus [KTR12] ist Teil zukünftiger Arbeiten.

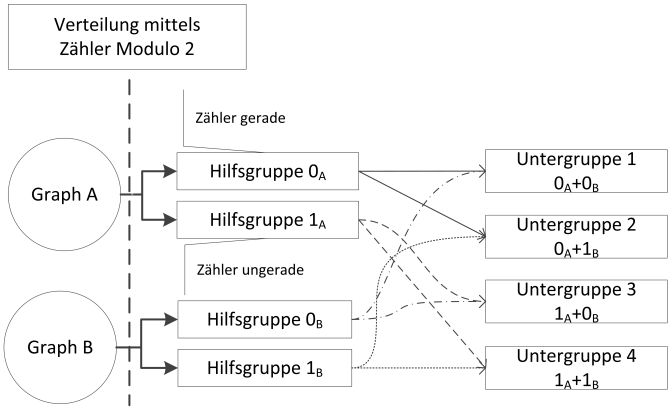


Abb. 5: Gruppierung in vier Untergruppen

4 Umsetzung & Evaluation

Die prototypische Implementierung der Konzeption aus Kapitel 3 erfolgte als *duplicate-linking*-Operator im GRADOOP-Framework. Der Operator wurde zur Demonstration der Funktionsweise monolithisch implementiert. Zur Evaluation der korrekten Funktionsweise und Skalierbarkeit des Operators wurden folgende Größen nach der prototypischen Umsetzung überprüft:

- Qualität der Duplikaterkennung (Precision, Recall und F-Measure)
- Laufzeitverhalten bei Vergrößerung der Taskanzahl sowie der Datengröße
- Laufzeitverhalten bei Untergruppierung (Lastbalancierung)

Als Datenquelle wurden analog zu [KTR10] Publikationsdaten der DBLP verwendet, welche Publikationen und Autoren im Bereich der Informatik auflistet. Für die Ermittlung von Precision und Recall wurde das manuell erstellte Match-Result[3] verwendet. Für die Messungen wurde aus DBLP für jede Publikation, wenn die Daten vorhanden waren, ein Verlags-Knotentyp (*publisher*), die beteiligten Autoren (*author*) und die Publikation selbst (*publication*) erstellt. Daraus sind zwei Graphen entstanden. Die Tabelle 3 zeigt die Eigenschaften der resultierenden Test-Graphen, welche – bis auf die Messung mit Erhöhung der Datensatzanzahl – als Testgrundlage verwendet wurden. Die Testgraphen sind so gewählt, dass nur in den Publikationen Duplikate vorhanden sind, da diese in beiden Graphen existieren. Für die Knotenvergleiche werden bei den Laufzeituntersuchungen alle gleichen Attribute verwendet.

In den folgenden Abschnitten wurden verschiedene Testläufe durchgeführt. Die Messungen sind mindestens zwei mal durchgeführt und jeweils der Durchschnitt der Laufzeiten

[3] http://dbs.uni-leipzig.de/de/research/projects/object_matching/fever/benchmark_datasets_for_entity_resolution/, Zugriff: 10.04.2016

Eigenschaft	Graph g1	Graph g2
Anzahl Knoten	35173	3616
Anzahl Autoren	19543	0
Anzahl Publikationen	15630	3597
Anzahl Verleger	0	19
Anzahl Matches	**40012**	

Tab. 3: Eigenschaften der DBLP-Graphen `g1` und `g2`

Standardmaße	Schwellwert						
	80 %	**85 %**	**90 %**	**92,5 %**	**95 %**	**97,5 %**	**100 %**
Precision	30,5 %	55,4 %	70,8 %	74,5 %	77,3 %	77,1 %	77,5 %
Recall	81,2 %	78,9 %	74,7 %	71,5 %	68,2 %	62,0 %	46,9 %
F-Measure	44,3 %	65,1 %	72,7 %	73,0 %	72,4 %	68,8 %	58,5 %
Laufzeit	85 s	85 s	85 s	82 s	82 s	82 s	83 s

Tab. 4: Standardmaße mit Blocking bei verschiedenen Schwellwerten (Standard-Hardware)

Standardmaße	Schwellwert						
	80 %	**85 %**	**90 %**	**92,5 %**	**95 %**	**97,5 %**	**100 %**
Precision	30,9 %	55,5%	70,9 %	74,5 %	77,3 %	77,1 %	77,8 %
Recall	84,9 %	79,5 %	75,1 %	71,8 %	68,4 %	62,1 %	47,8 %
F-Measure	45,3 %	65,4 %	72,9 %	73,2 %	72,6 %	68,8 %	59,2 %
Laufzeit	1091 s	1079 s	1081 s	1085 s	1067 s	1082 s	1087 s

Tab. 5: Standardmaße ohne Blocking bei verschiedenen Schwellwerten (Standard-Hardware)

bestimmt wurden. Für die einzelnen Testläufe kam zu Demonstrationszwecken die gleiche Standard-Hardware[4] mit vier CPU-Kernen(== vier Cores) zum Einsatz. Falls nicht anders angegeben, wurden die Testläufe ohne Parameter mit vorkonfigurierten Einstellungen des Operators durchgeführt.

4.1 Standardmaße

In der Tabelle 4 sind die Standardmaße mit steigendem Schwellwert abgebildet, die bei der Ausführung des Operators mit Blocking (Knotentyp+Anfangsbuchstabe vom Namen) erzielt wurden. In Tabelle 5 wurden die Untersuchungen ohne Blocking durchgeführt. Deutlich ist in den Tabellen 4 und 5 zu erkennen, dass mit steigendem Schwellwert der Precision-Wert wächst und der Recall-Wert reduziert wird. Der F-Measure-Wert hat seinen maximalen Wert von ca. 73% bei 92,5%.

Demzufolge wird dieser Schwellwert als Standard hinterlegt. Der Höchstwert bei der Precision und gleichzeitig der niedrigste Wert des Recalls liegt bei einem Schwellwert

[4] CPU: Intel Core i5-4300M, 2,6 GHz; Arbeitsspeicher: 8 Gigabyte

von 100 %. Durch den Vergleich der Attribute können nicht alle enthaltenen Duplikate gefunden werden und der Anteil falsch erkannter Duplikate ist dabei am größten. Die geringen Laufzeitschwankungen in Tabelle 4 sind Messungenauigkeiten. Die Laufzeit im Vergleich zu Tabelle 5 beträgt nur ca. 7,8 % bei fast gleichen Standardmaßen.

4.2 Skalierbarkeit

Die Skalierbarkeitsuntersuchung bei Erhöhung der Taskanzahl wurde mit den Graphen aus Tabelle 3 durchgeführt und in Abbildung 6 zusammenfassend dargestellt. Bei der Berechnung ohne Blocking wurde der *Cross*[5]-Operator verwendet und schrittweise die Taskanzahl erhöht sowie die Ausführungszeit gemessen. Dies ist die Zeit für die Erstellung des Ergebnisgraph sowie der Zählung von Kanten und Knoten. Außerdem wurde das Blocking anhand des Knotentyps und mit zusammengesetzten Blockschlüssel (Knotentyp+Anfangsbuchstabe vom Namen) untersucht.

Die Ergebnisse zeigen, dass die Laufzeit mit Erhöhung der Taskanzahl bei allen Varianten reduziert wird und damit skaliert. Annähernd ausgeglichen sind die Tasklaufzeiten ohne Blocking, sodass damit der größte Laufzeitgewinn erzielt wurde. Der *Cross*-Operator von Apache Flink skaliert dabei effizient. Bei der Variante mit einem zusammengesetzten Blockschlüssel ist die Laufzeit am geringsten, jedoch sind die Tasklaufzeiten nicht so ausgeglichen. Hier verteilt Apache Flink mittels *CoGroup*-Operator die Gruppen nicht optimal. Beim Blocking anhand des Knotentyps reduziert sich die Laufzeit durch eine geringere Paaranzahl im Vergleich zur Variante ohne Blocking. Die Vergleiche werden nur durch eine Task durchgeführt und muss daher optimiert werden. Durch das Blocking mit einem zusammengesetzten Blockschlüssel werden rund 8100 Duplikatpaare gefunden, was ca. 20% der Duplikate der anderen beiden Varianten entspricht. Ist die Anzahl von Tasks größer als die Anzahl zur Verfügung stehender CPU-Kerne, steigen durch einen Overhead die Laufzeiten wieder an. Die Anzahl der Tasks sollte im Optimalfall genauso groß eingestellt werden, wie die Anzahl zur Verfügung stehender CPU-Kerne. Des Weiteren sollte eine Untergruppierung bei beiden Blocking-Varianten angewendet werden, um die Lastbalancierung zu optimieren.

4.3 Laufzeitverhalten bei Untergruppierung

Zur Demonstration der Lastbalancierung wurde nur der parametrisierte Operator getestet. Je Durchlauf wurde die Anzahl b der Gruppen verändert sowie die Einstellung der zu untersuchenden Blockingvariante. Für die Taskanzahl von vier ist das Ergebnis in Abbildung 7 dargestellt. Als Referenzwert ist die jeweilige Laufzeit der Blocking-Varianten ohne Untergruppierung bei gleicher Taskanzahl eingezeichnet.

Erkennbar an den Verläufen ist, dass die Gesamtlaufzeit im Vergleich zur Variante ohne Untergruppierung bis auf $b = 1$ reduziert werden kann. Der Grund für die etwas größeren

[5] *DataSet Transformations - Operatorenübersicht.* https://ci.apache.org/projects/flink/flink-docs-release-1.1/apis/batch/dataset_trans-formations.html, Zugriff: 14.10.2016

Laufzeiten bei $b = 1$ ist der Umstand, dass mehr Prüfungen beim Blocking und der Ähnlichkeitsberechnung durchgeführt werden müssen, die sich jedoch bei $b > 1$ als vorteilhaft erweisen. Des Weiteren ist speziell beim Blocking anhand des Knotentyps die Gesamtlaufzeit auf ca. 33% im Vergleich zur Variante ohne Untergruppierung zurückgegangen. Beim zusammengesetzten Blockschlüssel ist der Laufzeitgewinn geringer, da der Overhead einen größeren Anteil an der Gesamtlaufzeit hat.

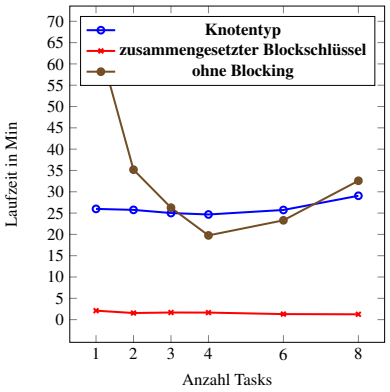


Abb. 6: Laufzeitverhalten bei Erhöhung der Taskanzahl (Standard-Hardware)

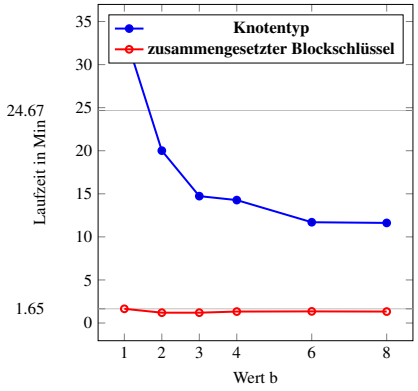


Abb. 7: Laufzeitverhalten bei Erhöhung der Blockanzahl durch Untergruppierung (Standard-Hardware, Taskanzahl = 4)

4.4 Laufzeitverhalten bei Erhöhung der Datensatzanzahl

Für die Laufzeituntersuchung mit Erhöhung der Datensatzanzahl wurden die Varianten ohne Blocking mittels *Cross*-Operator und parametrisierten Operator sowie die Varianten mit Blocking anhand des Knotentyps und mit einem zusammengesetzten Blockschlüssel jeweils mit der gleichen Anzahl an Datensätzen gegenübergestellt. Bei den Blocking-Varianten kam eine Untergruppierung mit $b = 4$ zum Einsatz, die übrigen Parameter blieben gleich. Als Datensatz sei ein Knoten innerhalb der Graphen anzusehen und die Gesamtzahl ist demnach die Summe der Knoten beider Eingangsgraphen, die jeweils ca. um den Faktor 10 vergrößert wurde. Die Ergebnisse sind in Tabelle 6 dargestellt, wobei die Werte in Klammern bei der Variante ohne Blocking die entsprechenden Ergebnisse des parametrisierten Operators sind.

Ersichtlich ist, dass die Laufzeiten unterschiedlich stark mit zunehmender Datensatzanzahl ansteigen und die Laufzeit des parametrisierten Operators etwas größer ist, als die des *Cross*-Operators. Die erhöhte Anzahl von Prüfungen der einzelnen Parameter fällt hier erkennbar auf. Bei den einzelnen Vergleichen wurden immer alle gleichen Attribute herangezogen. Die hohe Anzahl von Matches bei mehr als 100.000 Knoten kann durch den Schwellwert von $92,5\%$ begründen werden. Bei einem Schwellwert von $99,9999\%$ werden 10713 Matches gefunden und zwar genau die, die vorhanden sind. Die beiden Graphen sind so gewählt, dass nur die Schnittmenge identische Knoten besitzt, da auf der einen Seite Autoren und zugehörige Publikationen abgebildet sind und auf der anderen Seite Publikationen mit dem Verlag/Verleger. Demzufolge konnte der Verleger-Graph nur aus Publikationen

erstellt werden, die dieses Attribut gepflegt haben. Das sind im Autoren-Graph 10713 Publikationen.

Statt den *CoGroup*-Operator in Apache Flink zu verwenden, wurde bspw. auch der *Join*-Operator für einige Testläufe untersucht. Die Laufzeit lag dabei rund 25 % höher als beim Blocking anhand des Knotentyps mittels *CoGroup*-Operator. Abschließend lässt sich sagen, dass der Erhöhung der Laufzeiten in dem Fall mit mehr parallelen Servern und der damit verbundenen vergrößerten Taskanzahl entgegen getreten werden muss.

Knotenanzahl	ohne Blocking		Knotentyp		Knotentyp & 1. Buchstabe Name	
	Anz. Duplikate	Laufzeit	Anz. Duplikate	Laufzeit	Anz. Duplikate	Laufzeit
135	14	12s (13s)	14	11s	13	10s
1237	55	16s (16s)	55	15s	17	11s
11.683	830	4m21s (4m23s)	829	3m29s	63	24s
103.520	167786	4h14m (4h22m)	167752	3h18m	31210	13m53s

Tab. 6: Laufzeitverhalten bei Erhöhung der Datensatzanzahl (Standard-Hardware, Taskanzahl = 4)

5 Zusammenfassung und Ausblick

Im Rahmen dieses Beitrages wurde in Kapitel 3 ein Konzept vorgestellt, welches die Duplikaterkennung in Graphen innerhalb des GRADOOP-Frameworks ermöglicht. Der vorgestellte Workflow umfasst dabei Blocking, Lastbalancierung durch Untergruppierung sowie die Berechnung von Ähnlichkeiten zwischen Knoten. Das Konzept wurde prototypisch als ein GRADOOP-Operator implementiert und evaluiert. Die Ergebnisse der Evaluation haben gezeigt, dass der Höchstwert der Gesamtqualität bei den gefundenen Duplikaten bei ca. 80 % liegt und die Laufzeit mit Erhöhung der Task- und Gruppenanzahl skaliert. Aus Zeit- und Ressourcenmangel war eine Untersuchung auf einen Servercluster nicht möglich, jedoch ist zu vermuten, dass mit Erhöhung der Cores die Laufzeiten weiter skalieren.

Der Operator-basierte Workflow ermöglicht eine nahtlose Integration in andere GRADOOP-Workflows und damit in komplexe Datenanalyseszenarien (Ende-zu-Ende-Analysen). Weiterentwicklungen fokussieren auf die Übertragung von Lastbalancierungsalgorithmen, um die Effizienz bei der parallelen Ausführung von Duplikaterkennungs-Workflows in GRADOOP zu steigern.

Anmerkung: Die vorliegende Arbeit wurde durch Prof. Dr. Andreas Thor (Hochschule für Telekommunikation Leipzig, HfTL) und Dr. Eric Peukert (Universität Leipzig) im Rahmen des Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig (BMBF 01IS14014B) betreut.

Literatur

[Cu13] Curtiss, Michael; Becker, Iain; Bosman, Tudor; Doroshenko, Sergey; Grijincu, Lucian; Jackson, Tom; Kunnatur, Sandhya; Lassen, Soren; Pronin, Philip; Sankar, Sriram; Shen,

Guanghao; Woss, Gintaras; Yang, Chao; Zhang, Ning: Unicorn: A System for Searching the Social Graph. PVLDB, 6(11):1150–1161, 2013.

[FS69] Fellegi, I. P.; Sunter, A. B.: A Theory for Record Linkage. Journal of the American Statistical Association, 64:1183–1210, 1969.

[Gh15] Ghrab, Amine; Romero, Oscar; Skhiri, Sabri; Vaisman, Alejandro A.; Zimányi, Esteban: A Framework for Building OLAP Cubes on Graphs. In: Advances in Databases and Information Systems - 19th East European Conference, ADBIS 2015, Poitiers, France, September 8-11, 2015, Proceedings. Jgg. 9282 in Lecture Notes in Computer Science. Springer, S. 92–105, 2015.

[GM13] Getoor, Lise; Machanavajjhala, Ashwin: Entity resolution for big data. In: The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013. ACM, S. 1527, 2013.

[Ju15] Junghanns, Martin; Petermann, André; Gómez, Kevin; Rahm, Erhard: GRADOOP: Scalable Graph Data Management and Analytics with Hadoop. CoRR, abs/1506.00548, 2015.

[Ko14] Kolb, Lars: Effiziente MapReduce-Parallelisierung von Entity Resolution-Workflows. Dissertation, University of Leipzig, 2014.

[KR10] Koepcke, Hanna; Rahm, Erhard: Frameworks for entity matching: A comparison. Data & Knowledge Engineering, 69(2):197–210, 2010.

[KTR10] Köpcke, Hanna; Thor, Andreas; Rahm, Erhard: Evaluation of entity resolution approaches on real-world match problems. PVLDB, 3(1):484–493, 2010.

[KTR12] Kolb, Lars; Thor, Andreas; Rahm, Erhard: Dedoop: Efficient Deduplication with Hadoop. PVLDB, 5(12):1878–1881, 2012.

[Ma10] Malewicz, Grzegorz; Austern, Matthew H.; Bik, Aart J.C; Dehnert, James C.; Horn, Ilan; Leiser, Naty; Czajkowski, Grzegorz: Pregel: A System for Large-scale Graph Processing. SIGMOD '10, S. 135–146, 2010.

[neo13] Graph Database Applications and Concepts with Neo4j, SAIS, 2013. http://aisel.aisnet.org/sais2013/24.

[Wi06] Winkler, William E: Overview of record linkage and current research directions. Bericht, BUREAU OF THE CENSUS, 2006.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 335

Energy Efficiency in Main-Memory Databases

Stefan Noll[1]

Abstract: As the operating costs of today's data centres continue to increase and processor manufacturers are forced to meet thermal design power constraints, the energy efficiency of a main-memory database management system becomes more and more important. In this paper, we experimentally study the impact of reducing the clock frequency of the processor on the energy efficiency of common database algorithms such as scans, simple aggregations, simple hash joins and state-of-the-art join algorithms. We stress the fundamental trade-off between peak performance and energy efficiency, as opposed to the established race-to-idle strategy. Ultimately, we learn that database workloads behave considerably different than other workload types and that reducing the computing power e.g. by limiting the clock frequency can significantly improve energy efficiency.

Keywords: main-memory database systems, energy efficiency, DFVS.

1 Introduction

While a lot of current research in the domain of *main-memory database systems* (MMDBs) focuses on improving performance, only a small part of the research community focuses on improving energy efficiency. However, the energy consumption of a main-memory database system makes for an interesting research objective for several reasons. First, reducing the energy consumption also means reducing the operating costs, because the system needs less power and cooling. As electricity costs increase due to the growing demand for computing power, it is expected that the trend towards increased operating and cooling costs will intensify in the near future [PN08].
In addition, energy efficiency is a major aspect for chip design. In fact, semiconductor engineers have to meet fixed *thermal design power* (TDP) constraints. This means for example that not all cores of a processor can be fully powered at the same time without damaging parts of the integrated circuit. As a result, underutilised areas of the chip must be kept powered off (dark) or have to be operated at a low voltage and clock frequency, which is referred to as the Dark Silicon problem [Es11].
The aim of this work is to experimentally study as well as to understand the relationship between energy consumption and software to improve energy efficiency. We show that most algorithms used in a MMDB differ from algorithms used in other software. In fact, most database algorithms are memory-bound instead of compute-bound. As a result, unused computing power as well as electrical power gets wasted during the execution of database algorithms. Therefore, we propose to reduce the unused computing power of a processor for example by lowering the clock frequency and voltage which decreases the power consumption of a processor.

[1] TU Dortmund University, Databases and Information Systems Group, Otto-Hahn-Straße 14, 44227 Dortmund, stefan.noll@cs.tu-dortmund.de

We conduct a series of experiments with a compute-bound microbenchmark, with different memory-bound microbenchmarks simulating database algorithms as well as with two state-of-the-art join algorithms and analyse the impact of a limited clock frequency on the energy efficiency of the different workloads. In particular, we stress the fundamental trade-off between peak performance and energy efficiency, as opposed to the established race-to-idle strategy, which is usually used to maximise energy efficiency. Ultimately, we show that reducing unused computing power significantly improves the energy efficiency of memory-bound database algorithms.
In summary, we make the following contributions:

- We develop microbenchmarks simulating a compute-bound algorithm and different memory-bound algorithms.
- We experimentally study the impact of a limited clock frequency on the energy efficiency of the microbenchmarks and two state-of-the-art join algorithms.
- We use the results of the experiments to propose concepts for energy-aware software intended to improve the energy efficiency of a MMDB.

2 Basics

In this section we focus on the characteristics of database workloads. In addition, we introduce the power management features, which we use in the design of the experiments as well as for the reasoning in the course of the paper.

2.1 Characteristics of Database Workloads

Most database operations of a MMDB such as scans, aggregations or joins heavily depend on the memory bandwidth of the computer system. Thus, database algorithms are usually more memory-intensive than compute-intensive. In fact, most database algorithms are typically memory-bound or more specifically bandwidth-bound or latency-bound, because they perform lots of (random) memory accesses but only a few computations.
If an algorithm is memory-bound, the CPU stalls. It has to wait for data from the main memory until it can proceed with the execution of the algorithm. As a consequence, a lot of clock cycles are wasted. However, wasting computing power also means wasting electric power. Consequently, we argue that a high amount of computing power is not needed if an algorithm is memory-bound. To improve the energy efficiency of a database algorithm we propose to carefully reduce unused computing power.

2.2 Power Management Features

The *Advanced Configuration and Power Interface* (ACPI) defines several power management features of a modern computer system. This includes power and performance states

a computer system can enter to manage power consumption. In fact, the standard specifies two groups of states for a processor: the *processor power states* (C-States) and the *processor performance states* (P-States).
C-States describe the capability of an idle processor to save power by powering down unused parts of the processor. When individual processor cores or entire packages are idle, the operating system can put the hardware in a low-power state. Therefore, C-States are used to implement the **race-to-idle** strategy. Race-to-idle means executing a task as fast as possible and subsequently putting the processor in a sleep mode to save power.
P-States define the capability of an active processor to save power by lowering the clock frequency and voltage. A more common term used to describe this concept is *dynamic frequency and voltage scaling* (DFVS). In fact, a P-State can be interpreted as a pair consisting of a clock frequency and voltage. Higher (numerical) P-States result in slower processing speeds but also in a lower power consumption. The lowest P-State refers to the maximum clock frequency using turbo mode (e.g. "Turbo Boost" by Intel).

3 Experimental Design

In this section we present the experimental design. First, we introduce the database algorithms used in the experiments: one compute-bound microbenchmark, four memory-bound microbenchmarks and two state-of-the-art join algorithms. Afterwards, we introduce the hardware platform used to execute the experiments. Following this, we describe the measurement metrics and the experiments.

3.1 Benchmarks

The microbenchmark `compute` is designed to simulate a compute-bound algorithm. It induces a heavy load situation on all processor cores but avoids any accesses to the main memory. The benchmark creates several worker threads that spin on calculating the square root of a random number and on performing a multiplication.
The other four microbenchmarks are designed to simulate memory-bound, read-intensive database algorithms. They process integers with a size of 64 bits. The benchmarks `scan` and `local_scan` feature a sequential memory access pattern and resemble a scan or a simple aggregation. They create several worker threads that read a different section of a shared data array containing randomly generated integers (500 MB per thread). Each thread adds up all the integers of its section and prints the sum. In addition, we let each thread execute its work ten times to prolong the execution time in order to reduce variations and confounding factors.
The benchmarks `dira` and `local_dira` feature a *data-independent random* memory *access* (DIRA) [Ba14] pattern and resemble a simple hash join. The memory access pattern is called data-independent, because the processor can perform a random access without knowing the result of the previous random access. Each thread reads a different section of a shared data array containing randomly generated integers (500 MB per thread). These integers are used as indices to randomly access a second shared data array (8 GB).

We implement the microbenchmarks in `C++` using the program library `libnuma` to allocate the input data on specific NUMA nodes. The data is either allocated on one NUMA node only (`scan` and `dira`) resulting in both local and remote memory accesses or equally distributed across both NUMA nodes (`local_scan` and `local_dira`) resulting in local memory accesses only. We execute the microbenchmark using 32 threads.
In addition, we implement two benchmarks featuring two state-of-the-art join algorithms by using the program code of Balkesen et al. [Ba13]. We use the implementation of the optimised parallel radix hash join and the implementation of the NUMA-aware sort-merge join with either the multi-way or the multi-pass merging strategy. We slightly modify the source code to include the measurement metrics described in Section 3.3. We execute the algorithms using 16 threads and join two relations with a size of 4 GB each.

3.2 Hardware Platform

The hardware platform is configured as a dual-socket machine. It has two Intel Xeon E5-2690 processors featuring 8 processor cores (per socket), which are clocked at a base frequency of 2.9 GHz. Using turbo mode the processor can theoretically achieve a maximum clock frequency of up to 3.8 GHz. Furthermore, the system has 32 GB of DDR3 main memory per socket. The maximum memory bandwidth amounts to 51.2 GB/s for one socket or to 102.4 GB/s for both sockets.

3.3 Measurement Metrics

First, we determine the **execution time** of a benchmark to evaluate the performance. The execution time refers to the time each benchmark needs to process all of the input data. It does not include the time it takes to generate the data.
Second, we measure the total **energy consumption** of both processors by using the energy estimations provided by the *Running Average Power Limit* (RAPL) interface. The energy estimations refer to the energy consumption of the entire processor die including the core and uncore parts of the processor but excluding the attached DRAM.
Third, we compute the **average power consumption** of both processors by dividing the energy consumption by the execution time. In addition, we use an external power meter[3], which we plug between the power supply socket and the power supply unit to measure the power consumption of the entire computer system.
Fourth, we measure the **memory read bandwidth** by accessing hardware performance counters [WDF16] which provide information about the amount of bytes that are read from the memory controller of both sockets. Note that this includes additional memory traffic caused by cache misses if a new cache line has to be fetched but not all the data of the cache line is used by the program.
We do not use a specific metric to evaluate the *energy efficiency*. Instead, we argue that we improve the energy efficiency of a benchmark if the percentage decrease of the energy consumption is more significant than the percentage decrease of the performance. The point

[3] ELV Energy Master Basic 2 Energiekosten-Messgerät

of comparison is the measurement data from the execution at the highest clock frequency of 3.8 GHz, which is the default setting of the operating system.

3.4 Experiments

We conduct experiments to study the impact of a reduced computing power using the example of lowering the clock frequency of the processor. In fact, we limit the maximum clock frequency the Intel P-State driver of the Linux kernel `intel_pstate` is able to select. The driver can still select a clock frequency which lies below the limit but not above the limit. We execute every benchmark at different clock frequency limits from 1.2 GHz to 3.8 GHz while measuring the metrics described in Sect. 3.3.

4 Evaluation

Before we start evaluating the energy efficiency of the benchmarks, we first take a look at the accuracy of the energy estimations provided by the RAPL interface. Using the example of the microbenchmark `compute`, we compare the power estimations of both processors with the measurement data obtained from an external power meter. Figure 1a illustrates the experimental results. We observe that the two curves differ in approximately 65 W. Only in the frequency range of the turbo mode the two curves differ in up to 85 W. Thus, we argue that the energy estimations are reliable enough to evaluate the energy efficiency of the benchmarks. Furthermore, we verify that reducing the power consumption of both processors significantly improves the energy balance of the entire computer system. In addition, we notice that the curves break at 3.4 GHz. That is because the processor is not able to achieve a higher clock frequency using all cores due to thermal constraints.
Next, we evaluate the impact of a limited clock frequency on the energy efficiency of the benchmarks. The experimental results of the microbenchmark `compute` depicted in Fig. 1b reveal that the performance of the benchmark heavily depends on the clock frequency. Limiting the clock frequency to e.g. 1.9 GHz decreases the performance by more than 42%. At the same time the energy consumption only degrades by 26%. Thus, we observe that limiting the clock frequency does not improve the energy efficiency. Instead, it seems best to clock the processor at the highest frequency, finish the tasks as fast as possible and then put the processor in a sleep mode to save power (race-to-idle). Furthermore, we verify that the benchmark is only compute-intensive because the measured memory read bandwith is approximately 0 GB/s (cf. Table 1).
Figure 2a and Figure 2b illustrate the experimental results of the microbenchmarks `scan` and `dira`. We notice that limiting the clock frequency to e.g. 1.9 GHz only results in performance degradations of up to 4%. However, the energy consumption is reduced by 47%. Thus, limiting the clock frequency improves the energy efficiency of both benchmarks.
Moreover, we notice that the microbenchmarks `scan` and `dira` achieve a memory read bandwidth of up to 30 GB/s. The value lies below the hardware limit of 51 GB/s per socket. In fact, we determine that both benchmarks are limited by the bandwidth of the QPI links which connect both sockets. That is because all threads executed on the second

Benchmark Name	Execution Time [s]	Energy Consumption [J]	Memory Read Bandwidth [GB/s]
`compute`	16.1 / 5.9	956 / 1202	0.0 / 0.0
`scan`	6.6 / 5.4	455 / 973	24.3 / 30.2
`local_scan`	3.5 / 1.7	268 / 452	45.0 / 90.4
`dira`	5.5 / 4.5	387 / 837	25.5 / 31.2
`local_dira`	3.3 / 1.9	248 / 465	42.0 / 71.5
Parallel Radix Hash Join	4.3 / 2.5	295 / 448	15.4 / 24.8
Sort-Merge Join (multi-way)	8.5 / 3.7	611 / 842	9.9 / 21.9
Sort-Merge Join (multi-pass)	10.0 / 5.6	744 / 1233	18.5 / 32.3

Tab. 1: Execution time, energy consumption and memory read bandwidth of every benchmark at the lowest and highest clock frequency limit: 1.2 GHz / 3.8 GHz.

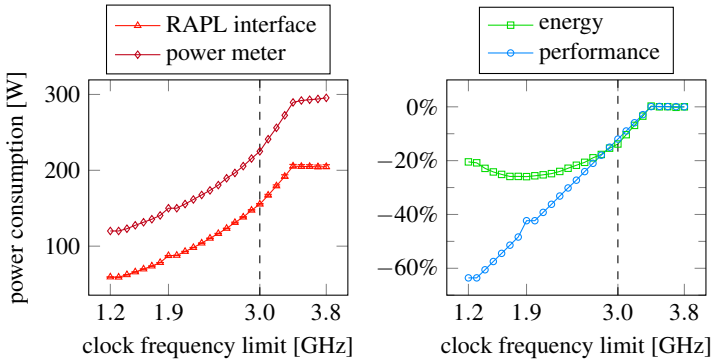


Fig. 1: Results of the benchmark `compute`: comparison of the energy estimations provided by the RAPL interface with the data obtained from an external power meter (a) and percentage decrease of the energy consumption and the runtime performance at different clock frequency limits (b). The dashed line marks the start of the turbo mode.

socket perform remote memory accesses to the input data which is located on the first socket. As a result, the execution slows down considerably.

Figure 3a and Figure 3b illustrate the results of the microbenchmarks `local_scan` and `local_dira`. We observe that limiting the clock frequency to e.g. 1.9 GHz saves more than 40% of the energy. At the same time we lose 20% of the performance. Thus, limiting the clock frequency improves the energy efficiency of both benchmarks. Furthermore, we learn that the microbenchmark `local_scan` reaches a memory read bandwidth of up to 90 GB/s (cf. Table 1), which is very close to the hardware limit of 102.4 GB/s for both sockets. In fact, the benchmark is bandwidth-bound. The microbenchmark `local_dira` on the other hand only reaches a memory read bandwidth of up to 72 GB/s (cf. Table 1). We conclude that the benchmark is latency-bound, i.e. by the time it takes to transfer data from the main memory to the processor.

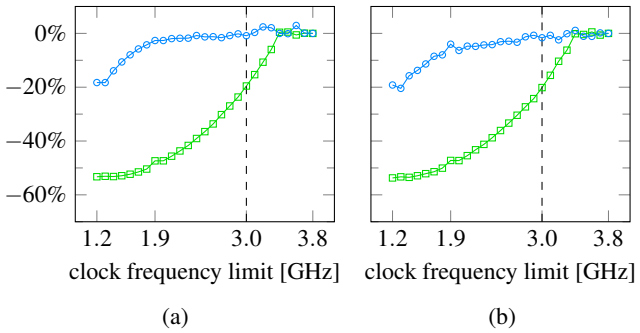


Fig. 2: Percentage decrease of the energy consumption and the runtime performance of the benchmarks `scan` (a) and `dira` (b) at different clock frequency limits.

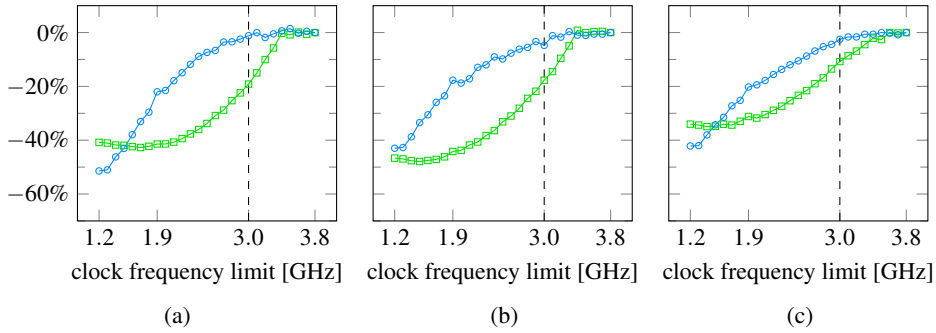


Fig. 3: Percentage decrease of the energy consumption and the runtime performance of the benchmarks `local_scan` (a), `local_dira` (b) and the parallel radix hash join (c) at different clock frequency limits.

Figure 3c depicts the measurement results of the parallel radix hash join. We learn that limiting the clock frequency to e.g. 1.9 GHz degrades the performance by 20%. At the same time the energy consumption is reduced by 31%. Thus, limiting the clock frequency improves the energy efficiency of the parallel radix hash join, too.

Figure 4a and Figure 4b illustrate the results of the sort-merge join algorithm using the multi-way and the multi-pass merging strategy. We observe that in case of the multi-way merging, limiting the clock frequency to e.g. 1.9 GHz deteriorates the performance by 34% while the energy consumption is reduced by 28%. In case of the multi-pass merging, limiting the clock frequency to 1.9 GHz degrades the performance by 21%. At the same time we save 36% of the energy. Thus, limiting the clock frequency significantly improves the energy efficiency of the sort-merge join algorithm if the multi-pass merging is used.

In order to explain the results we have to visualise the difference between the two algorithms. Figure 4c illustrates the execution times of every phase of the sort-merge join algorithm using either multi-way or multi-pass merging. We learn that the partitioning phase, the sorting phase and the joining phase all take approximately the same amount of

time. However, we observe that the merging phase lasts significantly longer if the algorithm uses the multi-pass merging technique.
Considering that the in-cache sorting is compute-intensive while merging is memory-intensive, it is plausible that the energy efficiency of the sort-merge join algorithm using multi-way merging does not improve. The algorithm is mainly compute-intensive. Thus, it heavily depends on the clock frequency of the processor. The sort-merge join using the multi-pass merging on the other hand is mainly memory-intensive. Therefore, we can reduce unused computing power of the processor to improve the energy efficiency.

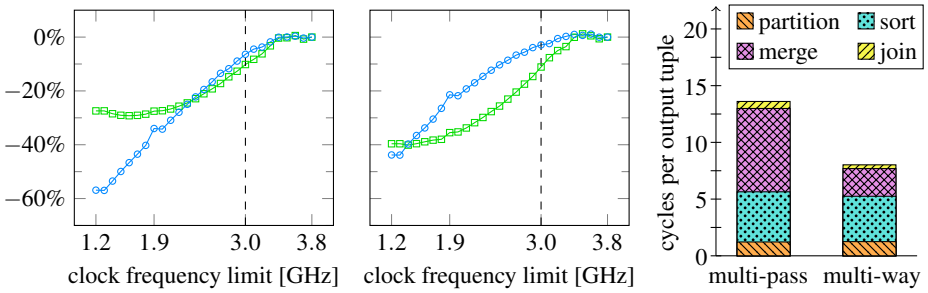


Fig. 4: Percentage decrease of the energy consumption —□— and the runtime performance —○— of the sort-merge join using either the multi-way merging strategy (a) or the multi-pass merging strategy (b) while limiting the clock frequency. Breakdown of the execution times (c) of the different phases of the sort-merge join algorithm using either multi-way merging or multi-pass merging [Ba13].

Discussion. Our evaluation reveals that the energy efficiency of the microbenchmark `compute` does not improve if we limit the clock frequency. We conclude that the most energy-efficient execution strategy for a compute-bound algorithm is the race-to-idle strategy. In contrast, we learn that limiting the clock frequency improves the energy efficiency of the memory-bound microbenchmarks.
If the input data is allocated on only one NUMA node, reducing the clock frequency hardly degrades the performance. If the input data is equally distributed across all NUMA nodes, limiting the clock frequency has a bigger impact on the performance. However, the energy savings prevail. Moreover, we observe that the impact of a limited clock frequency does not differ between a sequential memory access pattern (bandwidth-bound) and a random memory access pattern (latency-bound). In addition, we learn that reducing the clock frequency improves the energy efficiency of the parallel radix hash join, too.
In case of the sort-merge join we observe that the impact of a limited clock frequency depends on the merging strategy. If we use the multi-way merging, the algorithm is mostly compute-intensive. Reducing the clock frequency does not improve the energy efficiency. If we use the multi-pass merging, the algorithm is mostly memory-intensive. Limiting the clock frequency improves the energy efficiency.
Therefore, we conclude that we can improve the energy efficiency of a database algorithm by reducing the computing power if the algorithm is more memory-intensive than

compute-intensive. We showed that we can for example limit the clock frequency to improve the energy efficiency of database algorithms: If we execute a *compute-bound* algorithm, we should *increase* the *clock frequency* to the maximum (race-to-idle). If we execute a *memory-bound* algorithm, we should *decrease* the *clock frequency*.

Towards Energy-Efficient Software. The results of the evaluation reveal that we can reduce unused computing power to improve the energy efficiency of database algorithms. This allows us to develop ideas for energy-efficient software used in a MMDB.
First, the results can influence the design of database algorithms. If an algorithm consists of a compute-bound phase and a memory-bound phase, we propose to balance both phases in order to avoid energy waste. In case of e.g. the sort-merge join algorithm, we could overlap the compute-bound sorting phase with the memory-bound merging phase.
Second, we propose to extend the optimiser of a MMDB to avoid energy waste. When the optimiser decides upon the physical database operations of an execution plan, it should balance compute-bound operations and memory-bound operations. Thus, the optimiser needs specific cost models. These can be built based on the ratio between compute intensity and memory intensity of a database operation.
Third, we propose to extend the execution engine of a MMDB by a live monitoring component. The engine could use hardware performance counters to periodically calculate the ratio between compute intensity and memory intensity of database operations at runtime. This can be done on multiple levels: for the entire system, individual sockets or individual cores. The collected statistics can be used to dynamically change settings such as the clock frequency, thread-to-core mappings or the data placement.

5 Related Work

In this section we briefly present related work from the research community specialising on the topic of energy efficiency in DBMS. Harizopoulos et al. [Ha09] argue that hardware optimisations are only part of the solution towards improving energy efficiency. They predict that software will be the key to improve energy efficiency and propose to focus on system-wide configuration parameters and query optimisations. In fact, they propose to consolidate resource utilisation and power in order to facilitate powering down unused hardware and to redesign data-structures and algorithms. Thus, their observations align with our own results.
Tu et al. [Tu14] propose a storage manager which puts unused disk drives in a low power mode and employs energy-aware data placements. In addition, they present a prototype DBMS, which uses a feedback loop using DFVS based on the current performance and a performance threshold. They claim to achieve significant energy savings.
The work of Psaroudakis et al. [Ps14] further supports our results. They propose a fine granular approach for improving energy efficiency by dynamically scheduling tasks as well as using DVFS. They conclude that using simultaneous multithreading, a moderate clock frequency and a thread placement that fills both sockets can significantly improve the energy efficiency of memory-intensive workloads. In addition, they propose to use a memory allocation policy that uses the memory bandwidth of all available sockets.

6 Conclusion

The characteristics of typical database workloads matter. In fact, most database algorithms are memory-bound instead of compute-bound. As a result, computing power as well as electrical power gets wasted if the processor has to wait for data from the main memory. We experimentally explored the impact of reducing the computing power on the energy efficiency of several benchmarks representing common database algorithms such as scans, simple aggregations and joins. Our evaluation reveals that limiting the clock frequency deteriorates the energy efficiency of the compute-bound microbenchmark, while it improves the energy efficiency of the memory-bound benchmarks including state-of-the-art joins. We show that database workloads behave considerably differently than other workload types and that reducing the computing power e.g. by limiting the clock frequency can significantly improve energy efficiency.

Acknowledgments

This work was supported by the DFG, Collaborative Research Center SFB 876, A2.

References

[Ba13] Balkesen, C.; Alonso, G.; Teubner, J.; Özsu, M. T.: Multi-Core, Main-Memory Joins: Sort vs. Hash Revisited, Proc. VLDB Endow. 7/1, pp. 85–96, 2013.

[Ba14] Barber, R.; Lohman, G.; Pandis, I.; Raman, V.; Sidle, R.; Attaluri, G.; Chainani, N.; Lightstone, S.; Sharpe, D.: Memory-Efficient Hash Joins, Proc. VLDB Endow. 8/4, pp. 353–364, 2014.

[Es11] Esmaeilzadeh, H.; Blem, E.; St. Amant, R.; Sankaralingam, K.; Burger, D.: Dark Silicon and the End of Multicore Scaling, SIGARCH Comput. Archit. News 39/3, pp. 365–376, 2011.

[Ha09] Harizopoulos, S.; Shah, M. A.; Meza, J.; Ranganathan, P.: Energy Efficiency: The New Holy Grail of Data Management Systems Research. In (): Online Proc. CIDR, 2009.

[PN08] Poess, M.; Nambiar, R. O.: Energy Cost, the Key Challenge of Today's Data Centers: A Power Consumption Analysis of TPC-C Results, Proc. VLDB Endow. 1/2, pp. 1229–1240, 2008.

[Ps14] Psaroudakis, I.; Kissinger, T.; Porobic, D.; Ilsche, T.; Liarou, E.; Tözün, P.; Ailamaki, A.; Lehner, W.: Dynamic Fine-grained Scheduling for Energy-Efficient Main-memory Queries. In (): Proc. DaMoN, ACM, 1:1–1:7, 2014.

[Tu14] Tu, Y.; Wang, X.; Zeng, B.; Xu, Z.: A System for Energy-Efficient Data Management, SIGMOD Rec. 43/1, pp. 21–26, May 2014.

[WDF16] Willhalm, T.; Dementiev, R.; Fay, P.: Intel Performance Counter Monitor - A better way to measure CPU utilization, 2016, URL: http://www.intel.com/software/pcm, visited on: 08/01/2016.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 345

Fake war crime image detection by reverse image search

Alexander Askinadze[1]

Abstract: In the media, images of war crimes are often shared, which in reality come from other contexts or other war sites. In this paper an approach is proposed to detect duplicate or fake war crime images. For this, the bag of visual words model is used in conjunction with localized soft assignment coding and the k-nn classifier. For evaluation, a data set with 600 images of war crimes was crawled. Different distances and parameters were used for evaluation. Unmodified images can be recognized with this approach with 100% accuracy. Rotated and scaled images can also be detected with nearly 100% accuracy. Modifications like cropping or the combination of scaling and cropping ensure significantly smaller accuracy results. The run time was investigated and it was found that about 3000 images per second can be processed on an Intel Core i5 processor.

Keywords: Near-duplicate image detection, image recognition

1 Introduction

It has already been the case that images of war crimes were published and shared on mass media and social networks which originally came from a different context or were taken from previous war sites [Fu12][BB14]. This is often the case in the Near and Middle East, especially in connection with the conflicts in Iraq, Syria and Palestine. For example, an article by the BBC on the Gaza conflict in 2014 has shown that Palestinians in Gaza used fake pictures to share the suffering on social networks:

> Over the past week the hashtag #GazaUnderAttack has been used hundreds of thousands of times, often to distribute pictures claiming to show the effects the airstrikes.
>
> Some of the images are of the current situation in Gaza, but a #BBCtrending analysis has found that some date as far back as 2009 and others are from conflicts in Syria and Iraq. [BB14]

With more and more shared content in the media, it becomes increasingly important to find a way to verify the source of the image. This problem belongs to the field of duplicate image recognition. However, to the author's best knowledge, very few publications can be found in the literature that address the issue of fake war crime image detection. In order to solve this issue, this paper examines how the problem of detecting fake images of war crimes can be solved.

[1] Heinrich-Heine-Universität, Datenbanken und Informationssysteme, Universitätsstraße 1, 40225 Düsseldorf, askinadze@cs.uni-duesseldorf.de

The intended suggestion is a system where a user can query the image database with an image. This is called reverse image search. The search query can be made either as a file upload or through the URL of an image.

In Figure 1, such a scenario is presented where a user submits an image as a search input. The image is transformed into an internal representation and compared with similar representations present in the database. The best results are displayed to the user. If the user recognizes the image in one of these results, meta-information is displayed to the user. Meta-information can contain the first occurrence of this image and already existing URLs. Otherwise, the image is added to the database to participate in further search queries.

Such reverse image search systems are already available, e.g., at Google[2] or tineye[3]. These search engines already provide good search results, but soon fail with rotated images. It is possible that people who share fake images would also be willing to manipulate them so that such search engines do not recognize these images as duplicates. Therefore, a system is required that is invariant regarding rotation, scaling, brightness changes both of original images as well as of cropped images.

The paper is organized as follows. Section 2 gives a brief overview of necessary fundamentals of image processing, which are necessary to transform images into numerical vectors (image representations). Additionally, it presents how the numerical vectors can be compared with each other to recognize duplicates. The evaluation of the proposed approach on the crawled data set for various hyper-parameters is outlined in Section 4. The conclusions are drawn in the final section.

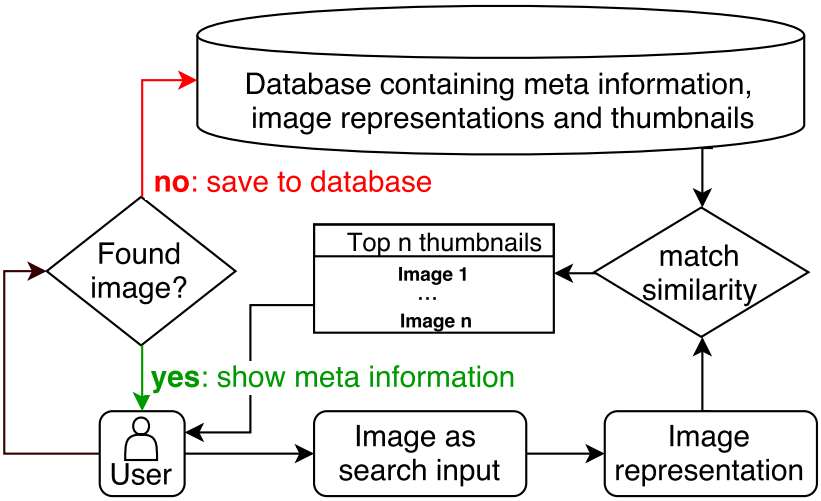


Fig. 1: Typical scenario of an reversed image search

[2] http://images.google.com/
[3] https://www.tineye.com/

2 Method

2.1 Image representation

An image is usually given as a pixel matrix with $n \times m$ pixels. Where n is the number of rows and m the number of columns of the matrix. Each image can have any size $n \times m$. Therefore, it is hard to compare two images in pixel representation. This requires an uniform image representation.

The Bag of Words model *(BoW)*, which is known from text processing, provides a way to represent images uniformly. Here, according to the words in a text, certain image features are selected as visual words. The visual words must be computed from a set of all image features. The set of visual words is called a visual codebook. The codebook creation process consists of the following three steps:

1. First of all, points are selected in an image. This can be done in a number of ways:
   (i) the points are distributed equally, (ii) only points are taken which are found by a point detector (e.g. an edge detector), or (iii) the points are selected randomly in the picture.
2. After the points have been selected, a descriptor describing the local properties of this point is computed for each point. Many methods of image processing use SIFT [Lo99] or SURF [Ba08] descriptors as image features. In [PPS13] the authors compared SIFT and SURF descriptors and came to the conclusion that SURF is faster and has a similar performance to SIFT. Therefore, SURF will be used as a descriptor.
3. Finally, the codebook is calculated from the set of all descriptors of all images. This large number of descriptors must be reduced to a small number of descriptors, which represent the entire quantity as well as possible. This is often implemented with the k-Means algorithm [Ll82]. The cluster centers are then chosen as visual words. The parameter k of the k-means algorithm thus determines the size of the codebook.

After the codebook has been created, the descriptors of each individual image are assigned to one of the visual words. A histogram is used to determine how often each visual word is taken as an assignment. This histogram thus also has the size K depending on the codebook size. This histogram is called *BovW* (bag of visual words) [Cs04]. Let h be the non-zero based BovW histogram, K the codebook size, D the set of all descriptors of an image, and $v_1, ..., v_K$ the visual words. For each descriptor $d \in D$, h is then updated as illustrated in algorithm 1.

Algorithm 1 Standard BovW Computation

for each d in D **do**
  $h[\arg\min_k ||d - v_k||] + +$
end for

Another strategy called *Soft assignment Coding* [Va10] extends this approach. Not only the one code word that is closest to the descriptor ensures the incrementation of the histogram

but all the code words. The weighting of the incrementation depends on the distance of the descriptor and the code words. Let h be the non-zero-based BovW histogram, K the codebook size, D the set of all descriptors of an image, and $v_1, ..., v_K$ the visual words. For each descriptor $d \in D$, h is then updated as illustrated in algorithm 2. The parameter β indicates how soft the assignment is.

Algorithm 2 Soft assignment Coding

for each d in D **do**

for k:=1 to K **do**

$$h[k] \leftarrow h[k] + \frac{exp(-\beta||d - v_k||)}{\sum_{j=1}^{K} exp(-\beta||d - v_j||)} \tag{1}$$

end for
end for

In [LWL11] the authors have found that it is difficult to find a good choice for β that can handle both large and small distances well. Therefore, it is suggested that not all the visual words should be used, but only l neighbors of a descriptor. This strategy is called *localized soft-assignment coding*. Let h be the non-zero-based BovW histogram, K the codebook size, D the set of all descriptors of an image, and $v_1, ..., v_K$ the visual words. For each descriptor $b \in D$ let $N_l(b)$ be the l nearest neighbors of b. The histogram h is then updated as illustrated in algorithm 3. The authors of [LWL11] say that the parameter β can be tuned

Algorithm 3 Localized soft assignment Coding

for each b in D **do**

for k:=1 to K **do**

$$\begin{aligned} \hat{d}(b, v_k) &\leftarrow \begin{cases} ||b - v_k||, & \text{if } v_k \in N_l(b) \\ \infty, & \text{otherwise} \end{cases} \\ h[k] &\leftarrow h[k] + \frac{exp(-\beta\ \hat{d}(b, v_k))}{\sum_{j=1}^{K} exp(-\beta\ \hat{d}(b, v_j))} \end{aligned} \tag{2}$$

end for
end for

by cross-validation. To create the BovW histograms, *localized soft assignment coding* and the parameters $\beta = 10$ and $l = 5$ given in [LWL11] will be used.

2.2 Similarity Search

For the classification of images, many different classifiers such as *k-NN*, *SVM*, *decision trees* or *neural networks* can be used [CHV99] [BZM07] [KSH12]. In the near-duplicate detection of images, there are just as many classes as images. With the BovW histograms a

uniform image representation has been found. Now a similarity search is enough. Therefore, the k-nearest neighbors (*k-nn*) algorithm will be used.

If the image being searched for is present in the database, the distance of the image representations would be 0 and the 1-nn classifier would recognize the correct image. If the image being searched for is a modified version of an image present in the database, the distance of the BovW histograms should be greater than 0. In this case, it would be appropriate to show the user the k nearest neighbors images whose distances are smallest from the input image.

In order to perform a similarity search, a distance must be calculated for all existing image representations. For n images in the database, a linear run time $\mathcal{O}(n)$ is obtained. Since most distance functions compare the corresponding dimension of two vectors, the run time also depends on the length of the BovW histograms. For a length m of BovW histograms and n images in the database, a $\mathcal{O}(nm)$ run time can be expected.

Because the result can depend on the selected distances, different distances will be investigated. This includes the following distances:

- Manhattan distance. For two n-dimensional vectors x, y the distance is defined by:

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{3}$$

- Euclidean distance. For two n-dimensional vectors x, y the distance is defined by:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{4}$$

- The cosine distance can be derived from the cosine similarity. For two n-dimensional vectors x, y the distance is defined by:

$$d(x,y) = 1 - \frac{\sum_i^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2}\sqrt{\sum_{i=1}^n y_i^2}} \tag{5}$$

3 Evaluation

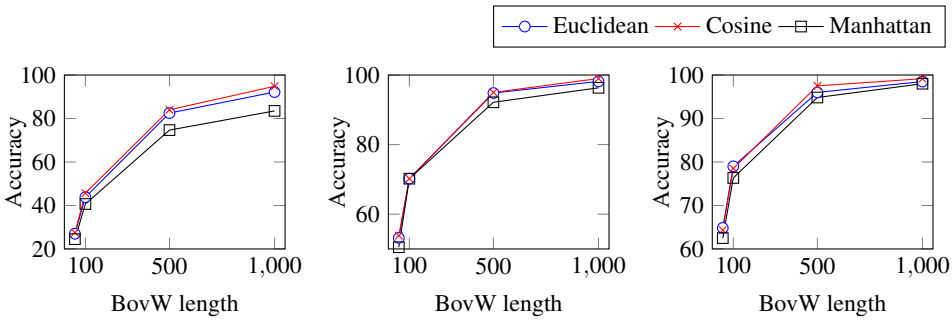
To perform an evaluation, an image data set was created by crawling Google image results for the search terms "Syria war crimes", "Iraq war crimes", and "Gaza war crimes". The search results had to be cleansed initially because they often contained images of politicians or caricatures. Of each of the search requests, 200 images were taken so that the data set consists of a total of 600 images.

It is the Intention to examine how well the war crimes can be detected in different manipulation stages. The investigated manipulation stages include: original images, scaled images, rotated images, cropped images as well as cropped and simultaneously scaled images.

To get the results for different parameters, a grid search will be performed. The examined parameters include: distances (Euclidean, Manhattan, Cosine), length of BovW histograms (50,100,500,1000) and number of search results (1,5,10) for the user in which the image could be recognized. So in total, $3 \times 4 \times 3 = 36$ different combinations. In addition, the training time for different lengths of the BovW histograms and the search run time for the 36 combinations was measured. The overall measurements results of all runs are summarized in Table 1. A detailed visualization of the results, which will be discussed in the following, can be found in Figs. 2, 3, and 4. The visualization of the training and search time can be found in Figs. 5 and 6.

As shown in Table 1, an unchanged image is recognized in almost every run with a precision of 100%. The results for rotated images (rotation around 90 degrees) are slightly less accurate but also around the 100%.

For the evaluation of scaled images, the originals were scaled to 50% of the original size. The results for different number of search results are shown in Figs. 2a, 2b, and 2c. If only a single search result is displayed to the user, the accuracy at large lengths of the BovW histograms is at both the Euclidean and Cosine distances close to 95%. If 10 possible image match candidates are displayed to the user, the accuracy increases to 100%

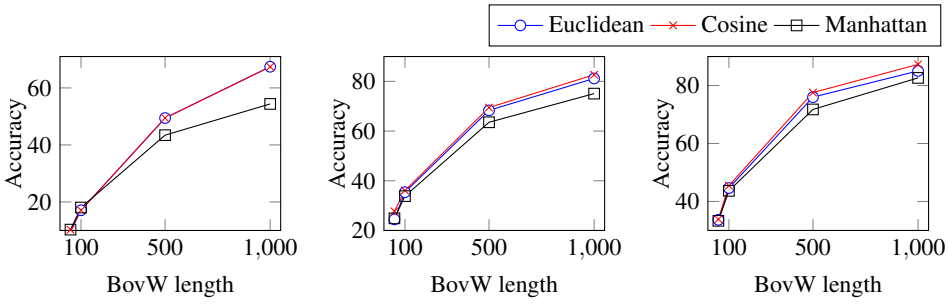


(a) 1 search result (b) 5 search results (c) 10 search results

Fig. 2: Accuracy values for **scaled** image queries for different count of search results

The results for cropped images for a different number of search results are shown in Figs. 3a, 3b, and 3c. To evaluate cropped images, the original image were divided into four equal parts. Each part was used as a search query, and so there were a total of 2400 search requests. With increasing length of the BovW vectors, the accuracy increases significantly in all three cases. While the accuracy results of all runs with Euclidean and Cosine distance are approximately equal, the results of the runs with the Manhattan distance are smaller. As shown in Figure 3a, the highest achieved accuracy value for a single search result is 67%. The tendency increases, which motivates the use of a larger length of BovW histograms. If ten search results are displayed to the user, the accuracy of the image being found is 87.29%, as shown in Figure 3c. Here too, a larger length of BovW histograms is expected to increase the accuracy.

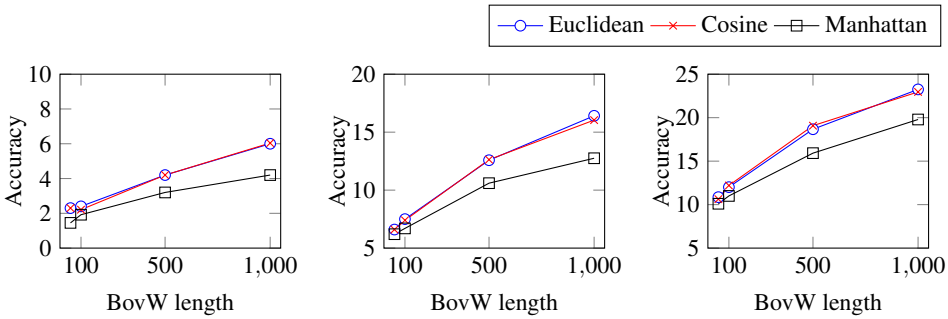
In order to test the limits of this approach, quarters were cropped from the images and then scaled to half of the original size. The accuracy results for different numbers of search



(a) 1 search result (b) 5 search results (c) 10 search results

Fig. 3: Accuracy values for **cropped** image queries for different count of search results

results are shown in Figs. 4a, 4b, and 4c. Compared to the results of scaled and cropped images, the results for the combination of the two manipulations are clearly poorer. With a single search result, only 6% accuracy can be achieved. With 10 search results, the accuracy increases to about 25%, but remains comparatively small. Here too, the two distances, Euclidean and Cosine, show significantly better values than the Manhattan distance.

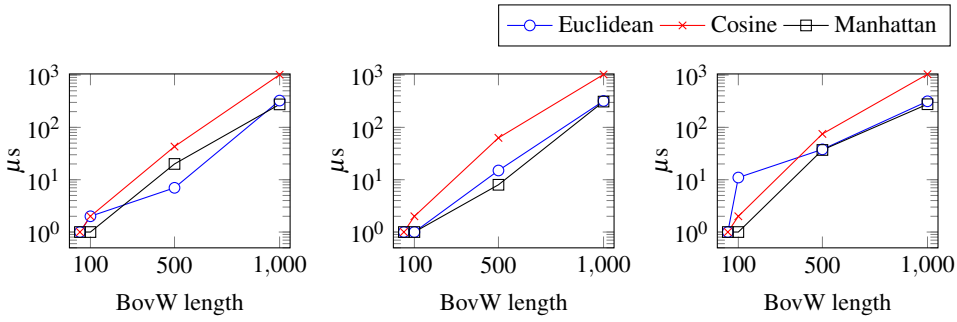


(a) 1 search result (b) 5 search results (c) 10 search results

Fig. 4: Accuracy values for **cropped and scaled** image queries for different count of search results

Search times per image for different numbers of search results are shown in Figs. 5a, 5b, and 5c. It is clear that the Cosine distance caused a significantly longer duration compared to the Euclidean and the Manhattan distance. The Manhattan and the Euclidean distance have similar values. From the fact that the Euclidean distance gives better results than the Manhattan distance, the use of the Euclidean distance is recommended. For BovW lengths of 1000 and the Euclidean distance, the search for an image takes about $\frac{1}{3}$ ms. In a database with n images, a search time of approximately $\frac{n}{3}$ ms can thus be expected on a computer with a Intel Core i5-3740 processor meaning that about 3000 images per second can be processed for example. The values were calculated during processing on a single core. Since this task is highly parallelizable, significantly faster running times can be implemented.

The training time depends on the running time of the k-means algorithm. This depends in turn on the number of data points and the number k of the clusters. In Figure 6, the training



(a) 1 search result (b) 5 search results (c) 10 search results

Fig. 5: Search time per image for different count of search results

time for different number of clusters (or the resulting BovW histograms) is illustrated. The running time increases significantly with the number of clusters. For 1000 clusters the running time is approximately 1 hour. In our investigation, all surf descriptors were used for clustering. The run time could be significantly reduced if only one random subset of descriptors is used by each image.

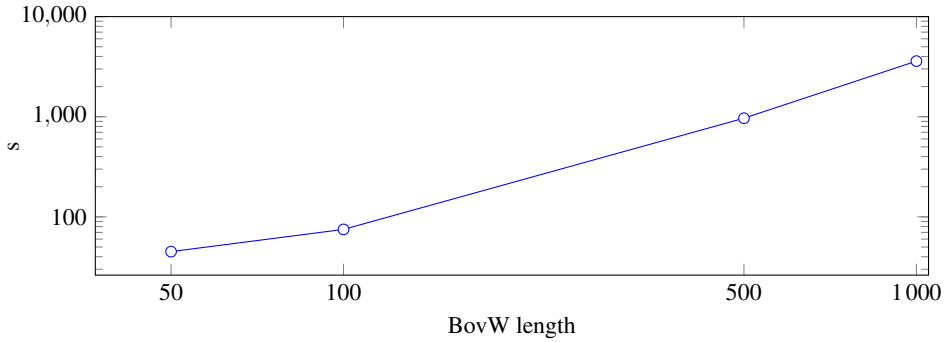


Fig. 6: Training time (k-Means Clustering) for the whole data set

4 Conclusion

In this paper, an approach to detect duplicate or fake war crime images is proposed. There is a considerable amount of literature on duplicate image detection, but no studies have been published on this special task. This approach consists of using the bag of visual words model in conjunction with localized soft assignment coding and the k-nn classifier. With this approach, unmodified images can be recognized with an accuracy of 100%. Rotated and scaled images can also be detected with an accuracy of nearly 100%. Modifications like cropping or the combination of scaling and cropping ensure significantly smaller accuracy results.

Future work will focus on improving the accuracy of strongly modified images. It also needs to be examined how new images can be used to improve the model without causing a completely new training. In a second, about 3000 images can be compared with respect to similarity. This can already be used in a web service. However, the run-time should be optimized for use with larger databases.

Tab. 1: Evaluation results (ordered by results of scaled and cropped image queries)

Run	Settings			Accuracy in %					run time	
	Distance: **E**uclidean, **M**anhattan, **C**osine	BovW length	Count search results	not scaled images			scaled images		train time in s, search time in μs per image	
				original	rotated	cropped	original	cropped	train	search
1	*E*	1000	10	100	99,5	85,04	98,5	23,25	3593	315
2	*C*	1000	10	100	100	87,29	99,16	22,96	3593	1037
3	*M*	1000	10	100	100	82,63	98	19,79	3593	277
4	*C*	500	10	100	100	77,58	97,5	19,08	967	75
5	*E*	500	10	100	99,67	76	96	18,67	967	38
6	*E*	1000	5	100	99,5	81,17	98,17	16,41	3593	320
7	*C*	1000	5	100	100	82,67	99	16,04	3593	1025
8	*M*	500	10	100	100	71,75	94,83	15,92	967	37
9	*M*	1000	5	100	100	75,08	96,33	12,75	3593	311
10	*C*	500	5	100	100	69,46	95	12,625	967	63
11	*E*	500	5	100	99,5	68,33	94,83	12,58	967	15
12	*C*	100	10	100	100	45,46	78,5	12,2	76	2
13	*E*	100	10	100	99,67	44,625	79	12,01	76	≤ 1
14	*M*	100	10	100	100	43,67	76,33	11	76	≤ 1
15	*E*	50	10	100	99,33	33,67	64,83	10,86	45	≤ 1
16	*C*	50	10	100	99,33	33,88	64,33	10,63	45	≤ 1
17	*M*	500	5	100	100	63,54	92,17	10,6	967	8
18	*M*	50	10	100	99,67	33,33	62,5	10,1	45	≤ 1
19	*E*	100	5	100	99,5	35,4	70,2	7,5	76	≤ 1
20	*C*	100	5	100	99,67	36,1	70,2	7,4	76	≤ 1
21	*M*	100	5	100	99,83	33,83	66,5	6,7	76	≤ 1
22	*E*	50	5	100	99,17	24,46	53,17	6,6	45	≤ 1
23	*C*	50	5	100	99	24,7	53,83	6,6	45	≤ 1
24	*M*	50	5	100	99,17	24,88	50,5	6,2	45	≤ 1
25	*C*	1000	1	99,5	99,17	67	94,83	6,04	3593	1015
26	*E*	1000	1	99,5	98,5	67,42	92,17	6	3593	325
27	*E*	500	1	99,33	98,5	49,42	82,5	4,2	967	7
28	*C*	500	1	99,5	99	49,2	84	4,2	967	43
29	*M*	1000	1	99,67	99,33	54,38	83,5	4,2	3593	275
30	*M*	500	1	99,33	99,17	43,42	74,67	3,2	967	20
31	*E*	100	1	99,5	97,67	17,08	43,83	2,4	75	2
32	*E*	50	1	99,67	94	9,6	27	2,3	45	≤ 1
33	*C*	50	1	99,33	94,33	10,25	27,5	2,3	45	≤ 1
34	*C*	100	1	99,67	97,5	17,63	45,66	2,21	75	2
35	*M*	100	1	99,5	98,5	18,04	40,67	1,92	75	≤ 1
36	*M*	50	1	99,83	94,17	10,29	24,5	1,45	45	≤ 1

References

[Ba08] Bay, Herbert; Ess, Andreas; Tuytelaars, Tinne; Van Gool, Luc: Speeded-up robust features (SURF). Computer vision and image understanding, 110(3):346–359, 2008.

[BB14] BBC Trending: , #BBCtrending: Are #GazaUnderAttack images accurate? http://www.bbc.com/news/blogs-trending-28198622, 2014. [Online; accessed 21-October-2016].

[BZM07] Bosch, Anna; Zisserman, Andrew; Munoz, Xavier: Image classification using random forests and ferns. In: 2007 IEEE 11th International Conference on Computer Vision. IEEE, pp. 1–8, 2007.

[CHV99] Chapelle, Olivier; Haffner, Patrick; Vapnik, Vladimir N: Support vector machines for histogram-based image classification. IEEE transactions on Neural Networks, 10(5):1055–1064, 1999.

[Cs04] Csurka, Gabriella; Dance, Christopher; Fan, Lixin; Willamowski, Jutta; Bray, Cédric: Visual categorization with bags of keypoints. In: Workshop on statistical learning in computer vision, ECCV. volume 1. Prague, pp. 1–2, 2004.

[Fu12] Furness, Hannah: , BBC News uses 'Iraq photo to illustrate Syrian massacre'. http://www.telegraph.co.uk/culture/tvandradio/bbc/9293620/BBC-News-uses-Iraq-photo-to-illustrate-Syrian-massacre.html, 2012. [Online; accessed 21-October-2016].

[KSH12] Krizhevsky, Alex; Sutskever, Ilya; Hinton, Geoffrey E: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105, 2012.

[Ll82] Lloyd, Stuart: Least squares quantization in PCM. IEEE transactions on information theory, 28(2):129–137, 1982.

[Lo99] Lowe, David G: Object recognition from local scale-invariant features. In: Computer vision, 1999. The proceedings of the seventh IEEE international conference on. volume 2. Ieee, pp. 1150–1157, 1999.

[LWL11] Liu, Lingqiao; Wang, Lei; Liu, Xinwang: In defense of soft-assignment coding. In: 2011 International Conference on Computer Vision. IEEE, pp. 2486–2493, 2011.

[PPS13] Panchal, PM; Panchal, SR; Shah, SK: A comparison of SIFT and SURF. International Journal of Innovative Research in Computer and Communication Engineering, 1(2):323–327, 2013.

[Va10] Van Gemert, Jan C; Veenman, Cor J; Smeulders, Arnold WM; Geusebroek, Jan-Mark: Visual word ambiguity. IEEE transactions on pattern analysis and machine intelligence, 32(7):1271–1283, 2010.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 355

Modellierung von relationalen Datenbanken mit UML im Round-Trip-Engineering[1]

Björn Salgert[2], Thomas C. Rakow[3]

Abstract: Die Unified Modeling Language (UML) wird mithilfe eines Profils erweitert, um den Entwurf relationaler Datenbanken zu unterstützen. Elemente von UML wie Vererbung können zur Daten-bankmodellierung genutzt werden, und datenbankspezifische Elemente wie beispielweise Indizes lassen sich in UML modellieren. Es wird eine automatisierte Abbildung vom UML-Klassendiagramm zum Datenbankschema erklärt und das Round-Trip-Engineering zwischen UML und einem Datenbankschema dargestellt. Schließlich wird der Nutzen dieser Modellierungsmethode verdeutlicht.

Keywords: UML, Schema-Evolution, Objekt-relationale Abbildung, Round-Trip-Engineering

1 Einleitung

Beim Datenbankentwurf kommen verschiedene Diagramme zum Einsatz, wie das Entity-Relationship-Model oder das Relationendiagramm. Dabei wird das ER-Diagramm auf das Relationendiagramm abgebildet, danach wird aus dem Relationenmodell das SQL-Schema erstellt. Gebräuchliche Datenbankfunktionalitäten wie Indizes, Primärschlüssel oder Constraints lassen sich mit den verwendeten Diagrammen nicht abbilden. Durch die Verwendung mehrerer Diagramme (ER-Diagramm, Relationenmodell) entsteht bei Änderungsanforderungen der Aufwand, alle Diagramme aktualisieren zu müssen, damit die vorhandenen Diagramme nicht veralten. Eine Vereinfachung dieses Prozesses ist sinnvoll. Dazu bietet es sich an, zur Modellierung die in der Software-Entwicklung weit verbreitete Unified Modeling Language (UML) zu verwenden [BQ13; Fo07; OS13; RQ12].

Durch die Verwendung der UML werden die Datenbankmodelle einem breiteren Kreis von Entwicklern zugänglich gemacht. Dies verbessert u. a. die Kommunikation in Projekten. Der Ansatz dieser Arbeit ist es, die UML zum Entwickeln von Datenbankmodellen zu nutzen. Dadurch soll ermöglicht werden, ein UML-Klassendiagramm für die Anwendungslogik und die Datenbank zu verwenden, um so eine einfachere Anwendungsentwicklung zu ermöglichen. Durch eine Abstraktion zwischen dem UML-Diagramm und dem Datenbankmodell wird eine Unabhängigkeit von einem spezifischen Datenbankmanagmentsystem erreicht. Dadurch muss die Wahl für ein konkretes Datenbankmanagmentsystem erst spät in einem Projekt getroffen werden.

[1] Diese Arbeit basiert auf der Masterarbeit des Erstautors [Sa16]

[2] Hochschule Düsseldorf, Fachbereich Medien, Münsterstraße 156, 40476 Düsseldorf, bjoern.salgert@hs-duesseldorf.de

[3] Hochschule Düsseldorf, Fachbereich Medien, Münsterstraße 156, 40476 Düsseldorf, thomas.rakow@hs-duesseldorf.de

Änderungen an der Datenbank werden oft nicht in das einmal erstellte Diagramm des konzeptionellen Entwurfs übertragen. Dieses Problem wird durch ein sog. Round-Trip-Engineering gelöst. Dabei wird das Datenbankmodell mit der Datenbank synchron gehalten und Änderungen in der Datenbank spiegeln sich im Modell wider und umgekehrt. Darüber hinaus soll eine Kontrolle über den Abbildungsprozess des konzeptionellen Entwurfs zur Datenbank gewährleistet werden, da dieser abhängig von der Anwendung aus Varianten wählen muss (s. Vererbung). Diese Funktionalität wurde in einer Erweiterung des bestehenden UML-Tools UMLet umgesetzt [ATB03; TUT16].

Beim vorgestellten Ansatz der Datenbankmodellierung wird ein weitverbreitetes und standardisiertes Modell für den kompletten Datenbankentwurf verwendet. Dadurch entfallen Modelltransformationen, und durch das Round-Trip-Engineering kann das Diagramm leicht über den kompletten Projektverlauf aktuell gehalten werden. Auch können direkte Änderungen an der Datenbankstruktur leicht ins Diagramm übernommen werden. Durch die Datenbankunabhängigkeit kann die Wahl eines konkreten DBMS im Laufe des Projektes getroffen werden und nicht bereits zu Beginn des Projektes. Durch die Nutzung der Möglichkeiten der UML sind die Modelle näher an der Anwendungsprogrammierung und werden von mehr Entwicklern direkt verstanden. Andere Ansätze für die objekt-relationale Abbildung wie objekt-relationale Mapper oder objektorientierte Datenbanken werden in dieser Arbeit nicht betrachtet.

In Abschnitt 2 werden bereits vorhandene Ansätze erläutert und evaluiert. Die Möglichkeiten, die die UML zur Datenbankmodellierung bildet, werden in Abschnitt 3 aufgezeigt. Die Erweiterungen für ein Round-Trip-Engineering werden in Abschnitt 4 beschrieben. Im darauffolgenden Abschnitt 5 wird die Softwareerweiterung von UMLet vorgestellt. Schließlich wird in Abschnitt 6 ein Fazit gezogen.

2 Verwandte Arbeiten

Die Arbeit „An interactive tool for UML class model evolution in database applications“ [MM13] verfolgt einen ähnlichen Ansatz wie die vorliegende Arbeit. Während die vorliegende Arbeit das UML-Diagramm auf ein Datenbankschema abbildet und aus diesem Schema dann das existierende Datenbankschema aktualisiert, wird in der Arbeit von Milovanovic und Milicev [MM13] das Datenbankschema aufgrund der Unterschiede des aktuellen UML-Diagramms mit dem vorherigen UML-Diagramm aktualisiert. Diese Methode kann daher allerdings keine Änderungen berücksichtigen, die in der Datenbank selbst vorgenommen wurden. Darüber hinaus erweitert die vorliegende Arbeit den Entwicklungsprozess zu einem Round-Trip-Engineering und berücksichtigt das Modellieren von datenbankspezifischen Elementen.

In „A Methodological Approach for Object-Relational Database Design using UML“ [MVC03] wird die Erweiterung der UML für die Modellierung von objekt-relationalen Datenbanken beschrieben. Als Datenbanksystem wird hier das Oracle RDBMS gewählt. Dieser Ansatz benötigt keine aufwendige Definition von Abbildungsregeln, da objekt-relationale Datenbanken objekt-orientierte Konzepte wie Vererbung beherrschen. Der

Nachteil dieser Methode ist jedoch, dass objekt-relationale Datenbanken sehr stark proprietär und weniger verbreitet sind als relationale Datenbanken. Die vorliegende Arbeit geht bei der Abbildung von UML noch einen Schritt weiter und bildet UML auf eine relationale Datenbank ab.

Der Artikel „A UML Profile for Data Modeling“ [Am03] beschreibt eine Erweiterung von UML mithilfe eines UML-Profils, das viele Aspekte relationaler Datenbanken beinhaltet. Es werden hier jedoch keine systematischen automatisierten Abbildungen beschrieben. Dies erschwert den Aufwand bei der Datenbankmodellierung, da die Abbildung des UML-Diagramms in eine Datenbank per Hand erfolgen muss.

Ein Teilaspekt des Round-Trip-Engineering ist das Reverse Engineering. Dies wird in der Arbeit „SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas“ [ACD08] beschrieben. Die Arbeit beschreibt, wie sich UML-Diagramme aus Datenbanken generieren lassen. Die vorliegende Arbeit berücksichtigt hingegen das komplette Round-Trip-Engineering. Auch werden in der Arbeit die relationalen Konstrukte nicht an die Elemente des UML-Klassendiagramms angepasst, sondern eins zu eins aus der Datenbank im UML-Diagramm dargestellt. Dadurch werden gebräuchliche UML-Elemente wie Assoziationen nicht unterstützt.

3 UML für den relationalen Datenbankentwurf

Für den Entwurf von Datenbankschemata mit UML eignet sich das Klassendiagramm. Bei diesem Diagrammtyp lassen sich die meisten Elemente für den Datenbankentwurf verwenden. Es lassen sich Klassen bidirektional auf Tabellen abbilden und auch die Assoziationen entsprechen im Wesentlichen den Schlüssel-Fremdschlüsselbeziehungen relationaler Datenbanken. UML-Diagramme eigenen sich sowohl für den konzeptionellen als auch den physischen Entwurf. Zu Beginn des Entwicklungsprozesses einer Anwendung können Klassen nur mit Attributen (auch ohne Datentypen) modelliert werden. Später können die Datentypen hinzugefügt sowie Indizes und weitere Bedingungen modelliert werden. Einige Tools verfügen über eine SQL-Generierung aus UML-Klassendiagramm, die jedoch die Modellierung vieler datenbankspezifischer Funktionalitäten wie Indizes nicht ermöglicht.

3.1 Erweiterung der UML

Die UML bietet vielfältige Möglichkeiten zur Modellierung von Anwendungen. Jedoch können die Möglichkeiten der UML nicht die Anforderungen jedes Projektes abdecken. Um die UML besser an die Anforderungen von Projekten anpassen zu können, sind in der UML Erweiterungsmöglichkeiten vorgesehen. Es werden zwei Erweiterungsmöglichkeiten unterschieden. Zum einen die leichtgewichtige Erweiterung durch UML-Profile und zum anderen die schwergewichtige Erweiterung durch Veränderungen des Metamodells. Durch das Metamodell wird die UML selbst definiert. Durch Veränderungen des Metamodells sind umfassende Veränderungen der UML möglich. Um die UML zur Datenbankmodellierung

zu verwenden, wird die UML mithilfe eines Profils erweitert. Dadurch bleiben die UML-Basiskonzepte erhalten und die UML-Notationen können weiterverwendet werden. Für Anwender entsteht daher kein zusätzlicher Lernaufwand, denn der Sinn der gemeinsamen Notationssprache bleibt erhalten. Natürlich ist die Erweiterung durch ein Profil einfacher als eine Anpassung des Metamodells.

Bei der Erweiterung der UML durch ein Profil wird die UML um neue Elemente ergänzt. Dies bedeutet, es kommen nur neue Elemente hinzu, bestehende Elemente der UML können mit einem Profil nicht verändert werden. Elemente, die mit einem Profil hinzugefügt werden können, sind hauptsächlich Stereotypen. Ein Profil selbst wird mithilfe eines UML-Profildiagramms definiert.

3.2 Modellierung datenbankspezifischer Eigenschaften

Ein Aspekt bei der Entwicklung eines Schemas sind Integritätsbedingen für einzelne Attribute (CHECK-Constraints in SQL). In UML lassen sich diese Bedingungen mithilfe der OCL [Fo07] modellieren. Viele datenbankspezifische Komponenten lassen sich nicht mit Hilfe der UML modellieren. Diese Elemente definieren wir in einem UML-Profil. Die Definition des Profils erfolgt mit folgendem UML-Profildiagramm in Abb. 1.

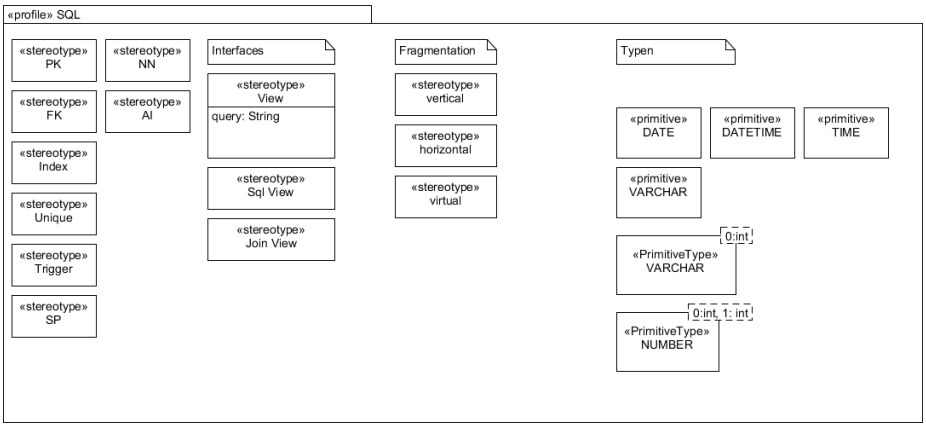


Abb. 1: UML-Profil für die Datenbankentwicklung

Im Profil sind zu meist Stereotypen für Datenbankkonstrukte, wie `NOT NULL`, `INDEX`, `PK`, `FK`, `VIEW` definiert. Darüber hinaus enthält das Profil die aus SQL bekannten Datentypen, wie z. B. `VARCHAR`. Das Profil erweitert die bisherigen Arbeiten, die UML-Profile zur Datenbank Modellierung entwickelt haben [Go03; Sp01] um Abbildungsmöglichkeiten für die Vererbung.

3.3 Vererbung

Aus dem erweitertem ER-Diagramm (EER) sind drei Strategien für die Abbildung der Vererbung bzw. ist-ein-Beziehung bekannt. Diese Strategien sind die vertikale, horizontale und virtuelle Fragmentierung [Fa07].

Eine virtuelle Fragmentierung bietet sich an, wenn die Unterklassen wenige oder kaum eigene Attribute besitzen. In diesem Fall werden alle Attribute der Unterklasse in die Oberklasse verschoben und es wird ein zusätzliches Attribut für den Typ eingeführt.

Eine horizontale Fragmentierung bietet sich an, wenn die Oberklasse abstrakt ist, siehe Abb. 2. Dieser Fall ist analog zu einer ist-ein-Beziehung mit totaler Teilnahme.

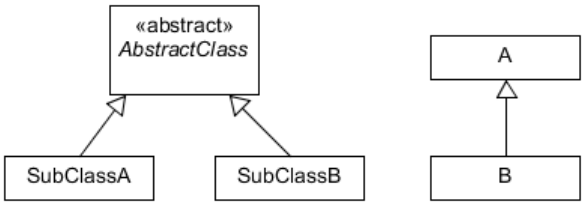


Abb. 2: Vererbung mit abstrakter Klasse

Da die Oberklasse in diesem Fall nicht abgebildet wird, müssen Beziehungen, die sich auf die Oberklasse beziehen, in allen Unterklassen berücksichtigt werden.

Eine vertikale Fragmentierung bleibt bei der Abbildung einer Vererbung sehr nah am UML-Diagramm, da alle beteiligten Klassen als Tabellen abgebildet werden. Die Tabellen der Unterklassen erhalten den Primärschlüssel der Oberklasse als Primär- und als Fremdschlüssel.

Wie im EER-Diagramm beinhaltet auch die UML einige Attribute, die die Teilnahme an der Generalisierungsbeziehung regeln. Diese Attribute sind: *complete* (total) oder *incomplete* (partiell) und *disjoint* (disjunkt) oder *overlapping* (überlappend). In Klammern sind die Entsprechungen des EER-Diagramms auf Deutsch angegeben [RQ12]. Bei einer überlappenden Teilnahme an einer Generalisierung sollte der Entwickler jedoch beachten, dass diese in Java (oder anderen OO-Sprachen) mit den Sprachkonstrukten selbst nicht umsetzbar sind. Welche Art der Vererbung bzw. Fragmentierung verwendet werden soll, kann der Modellierer per Stereotyp angeben («vertical», «horizontal» oder «virtual»), default-mäßig wird vertikal fragmentiert (s. oben).

3.4 Modellierung eines Indexes

Indizes sind für die Leistungsfähigkeit von Datenbanksystemen sehr wichtig. In einer Tabelle können mehrere Indizes definiert werden. Daher reicht es nicht aus, einen «index»-Stereotyp zu definieren, bei dem alle Attribute, die mit dem Stereotyp ausgezeichnet sind, zu einem

Index zusammengefasst werden. Aus diesem Grund wird der Stereotyp «index» für Klassen definiert. Die Semantik dieses Stereotyps bedeutet, dass alle Attribute der Klasse, die mit dem Stereotyp «index» ausgezeichnet sind, einen Index bilden. Der Index wird für die Klasse erstellt auf die die «index»-Klasse mit einer Assoziation verweist. Abb. 3 verdeutlicht diese Semantik man Beispiel eines Indizes für das TPC-H Schema:

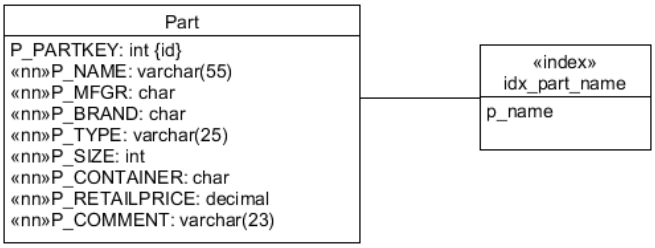


Abb. 3: Indexdefinition in UML am Beispiel TPC-H

4 Round-Trip-Engineering

Wichtig für einen effizienten Entwicklungsprozess ist das Automatisieren von Aufgaben. Aus diesem Grund wurde Umlet mit einem Round-Trip-Engineering ausgestattet und zu Umlet+RTE. Das ermöglicht es, aus dem für das Relationenschema verwendete UML-Klassendiagramm die Datenbank automatisch zu erstellen bzw. eine ältere Version der Datenbank zu aktualisieren. Auch lässt sich so aus einer Datenbank ein UML-Diagramm erstellen. Bei der Aktualisierung einer Datenbank wird die Datenbank aktualisiert und zusätzlich wird ein SQL-Script erstellt, das die Datenbank ebenfalls aktualisieren kann. Dadurch können Produktupdates einfach realisiert werden. Die Generierung der Skripte findet unter Berücksichtigung des vorhandenen Schemas statt. Durch die automatisierte Abbildung von z. B. Vererbungsstrukturen lassen sich die Möglichkeiten der UML weiterhin zur Modellierung verwenden.

4.1 Software Architektur

Im Gegensatz zu einem Round-Trip-Engineering zwischen dem UML-Klassendiagramm und beispielsweise dem Java-Quellcode gibt es beim Round-Trip-Engineering zu einer Datenbank einige Besonderheiten. Zum einen müssen verschiedene DBMS mit unterschiedlichen SQL-Dialekten berücksichtigt werden. Zum anderen gibt es drei Komponenten beim Round-Trip-Engineering: das Diagramm, die Datenbank und das SQL-Schema. Das folgende UML-Verteilungsdiagramm in Abb. 4 skizziert das Round-Trip-Engineering für die Datenbankmodellierung.

Die Datenbank wird im Diagramm durch den Knoten DatabaseLayer repräsentiert. Der DatabaseLayer enthält die Komponente DatabaseConnection. Diese Komponente benutzt

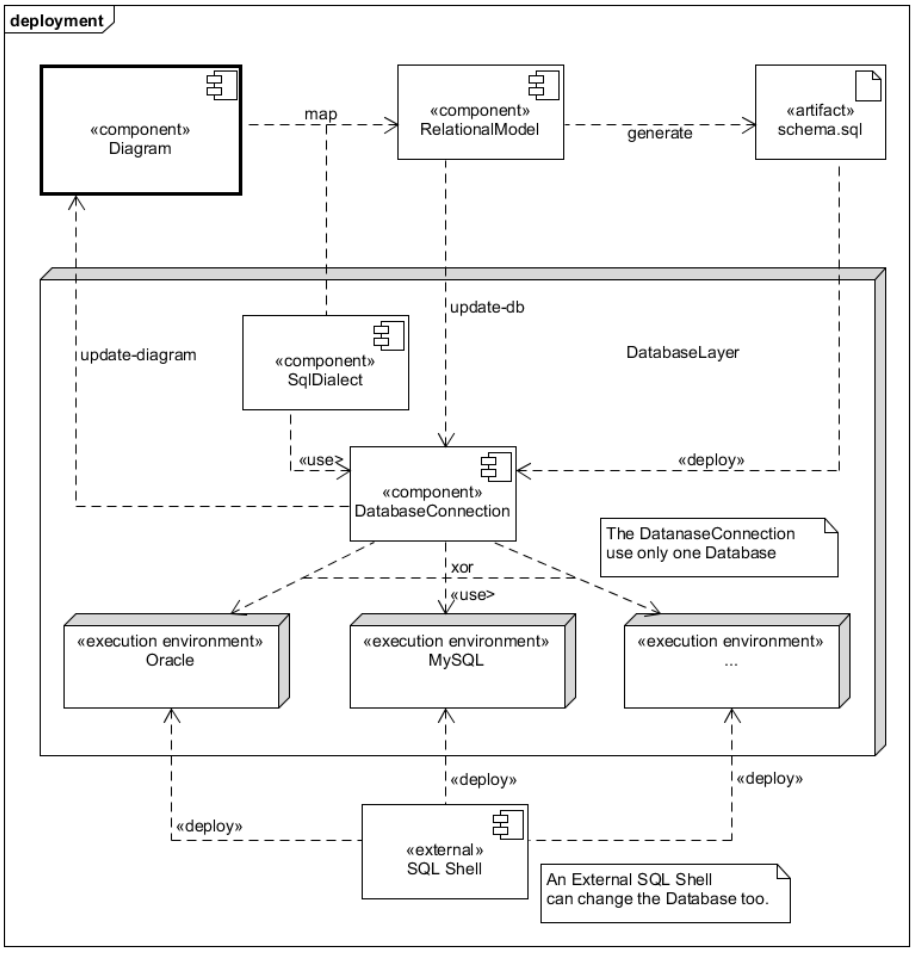


Abb. 4: Skizze Round-Trip-Engineering

genau eine Datenbank («use»). Durch diese Konstruktion wird die Unabhängigkeit von einem einzelnen DBMS umgesetzt.

Die Komponente Diagram repräsentiert das UML-Klassendiagramm zur Datenbankmodellierung. Diese Komponente bildet das Modell auf ein Relationendiagramm (RelationalModel) ab. Aus dem Relationendiagramm wird mithilfe des SqlDialect des verwendeten DBMS das SQL-Schema (schema.sql) generiert. Das SQL-Schema kann eine komplette Datenbank erzeugen. Mit der Datei schema.sql kann z. B. das SQL-Schema auf eine Produktionsumgebung übertragen werden. Durch die automatische Generierung des SQL-Schemas steht immer eine aktuelle und ausführbare SQL-Datei mit dem Schema zur Verfügung.

In dem Modell sind auch externe Änderungen der Datenbank vorgesehen. Diese Möglichkeit wird durch die Komponente SQL Shell symbolisiert. Die externen Änderungen an der Datenbank werden auch im Diagramm übernommen. Wichtig ist, dass die Komponente DatabaseConnection nur eine Datenbank verwenden kann («use»), da es nicht möglich ist, dass die Datenbank die Anwendung bei Änderungen am Schema informiert.

Das Schema (schema.sql) kann über die DatabaseConnection auf dem Datenbankserver ausgeführt werden («deploy»). Änderungen an der Datenbank werden von der Komponente DatabaseConnection erkannt und ins Modell bzw. Diagramm übertragen (update-diagram). Bei dem Update des Diagramms werden die relationalen Strukturen der Datenbank auf die Elemente des UML-Diagramms abgebildet. Umgekehrt werden Änderungen am Diagramm in der Datenbank umgesetzt (update-db). Bei diesem Schritt wird das UML-Diagramm zuerst in ein Relationenmodell umgewandelt. Nach diesem Zwischenschritt wird die Datenbank über die Database-Connection aktualisiert.

Ein wesentlicher Aspekt bei dem beschriebenen Round-Trip-Engineering sind die Daten. Neben der Notation für Klassen bietet die UML auch eine Notation für Objekte. Dadurch können mit der UML auch Datensätze bzw. Instanzen modelliert werden. Es ist sinnvoll einen initialen Datenbestand für die Produktivumgebung zu modellieren (zu diesem initialen Datenbestand kann z. B. ein erster Benutzer gehören). Der Schritt, alle Datensätze aus der Entwicklungsdatenbank ins Diagramm zu übernehmen, kann zu Problemen führen. Problematisch kann in diesem Zusammenhang sein, dass das Diagramm nach einem Testen der Anwendung mit Datensätzen überfüllt ist. Für den Umgang mit Datensätzen sollten Optionen geschaffen werden.

Das angedachte Round-Trip-Engineering weist einige Besonderheiten auf. Es findet nicht zwischen einer SQL-Datei und dem Klassendiagramm statt, sondern zwischen dem Klassendiagramm und der Datenbank. Bei der Abbildung vom Diagramm zur Datenbank im Round-Trip-Engineering wird mittels einer Modell-Transformation ein Relationenmodell erstellt, das schließlich mit der Datenbank abgeglichen wird.

5 Umsetzung der Software

Das Werkzeug zur Datenbankmodellierung wurde als Erweiterung +RTE der unter der GNU-GP-Lizenz verfügbaren Software UMLet umgesetzt [ATB03; TUT16]. Das Tool wurde wegen seiner einfachen Bedienung, seiner Verfügbarkeit im Quellcode und seiner Erweiterbarkeit ausgewählt [Sa16]. Der größte Vorteil bei dieser Vorgehensweise ist, dass viele Funktionen in der Software schon vorhanden sind. Damit ist für den eigentlichen Diagramm-Editor keine Arbeit mehr nötig und es bleibt mehr Zeit für die Implementierung des Round-Trip-Engineering übrig. Sichtbar wurde UMLet durch +RTE kaum verändert. Es gibt im Wesentlichen nur zwei zusätzliche Button: einen Button zur Aktualisierung des Diagramms aus Änderungen in der Datenbank und einen weiteren Button, um die Änderungen des Diagramms in die Datenbank zu übertragen. Bei einer Änderung der Datenbank wird zusätzlich eine SQL-Datei mit dem kompletten Schema angelegt. Der Nutzer kann wählen, ob er die Datenbank direkt aktualisieren will oder nur das SQL-Skript mit den UPDATE- und ALTER-Anweisungen erhalten will. Darüber hinaus gibt es eine zusätzliche Palette mit vorgefertigten Elementen für den Datenbankentwurf (z. B. Klasse mit Primärschlüssel, Index, etc.).

Bei der Evaluierung stellt sich die Frage: „Helfen UML-Diagramme in der Datenbankentwicklung bzw. Modellierung?“ [BQ13]. Das Tool wurde bislang einmal im Praktikum für Datenbanksysteme 2 im Studiengang Medieninformatik eingesetzt. Trotz Problemen mit der Stabilität des verwendeten Tools UMLet+RTE konnte ein Verständnis für die Modellierung in UML entwickelt und die Notwendigkeit der Konsistenz zwischen Modell und Datenbank vermittelt werden.

6 Fazit und Ausblick

Diese Arbeit beschreibt die Modellierung von Datenbankschemata mittels des UML-Klassendiagramms und der standard-konformen Erweiterung durch ein UML-Profile. Der Einsatzbereich der UML wird auf die Datenbankmodellierung ausgedehnt und durchgehend für den konzeptionellen sowie den physischen Datenbankentwurf genutzt. Änderungen erfolgen durch das Round-Trip-Engineering konsistent in Modell und Datenbanksystem.

Bei dem Thema dieser Arbeit gibt es viele Anknüpfungspunkte für künftige Entwicklungen. Als Erstes wäre eine Erweiterung des Round-Trip-Engineerings auf eine objekt-orientierte Sprache wie Java sinnvoll. Dadurch bekäme das Round-Trip-Engineering neben dem UML-Diagramm und der Datenbank einen weiteren Teilnehmer. Darüber hinaus sollte auch der XMI-Standard in der Anwendung Verwendung finden. Durch die Unterstützung des XMI-Standards könnten die UML-Diagramme auch in anderen UML-Tools genutzt werden. Auch wird die Anwendungsentwicklung weiter unterstützt, indem Formulare automatisch aus dem Modell erstellt werden können. Hierfür müssen die Eigenschaften der Klassen mit entsprechenden Annotationen versehen werden und Templates für die Formulare erstellt werden. Schließlich können auch NoSQL-Datenbanken, berücksichtigt werden. So ließe sich das UML-Klassendiagramm auch auf eine dokumentbasierte Datenbank wie z. B. MongoDB abbilden. Auch ließe sich neben dem SQL-Schema ebenfalls ein XML-Schema generieren. Dies würde den Datenaustausch mit anderen Anwendungen vereinfachen.

Literatur

[ACD08] Alalfi, M. H.; Cordy, J. R.; Dean, T. R.: SQL2XMI: Reverse Engineering of UML-ER Diagrams from Relational Database Schemas. In: 15th Working Conference on Reverse Engineering. Institute of Electrical und Electronics Engineers (IEEE), S. 187–191, Okt. 2008.

[Am03] Ambler, S. W.: A UML Profile for Data Modeling, 2003, URL: http://www.agiledata.org/essays/umlDataModelingProfile.html, Stand: 20. 10. 2016.

[ATB03] Auer, M.; Tschurtschenthaler, T.; Biffl, S.: A flyweight UML modelling tool for software development in heterogeneous environments. In: Proceedings 29th Euromicro Conference. Institute of Electrical und Electronics Engineers (IEEE), 2003.

[BQ13] Byrne, B.; Qureshi, S.: UML Class Diagram or Entity Relationship Diagram: An Object Relational Impedance Mismatch. In: Proceedings of 6th Int. Conf. of Education, Research, and Innovation. 2013.

[Fa07] Faeskorn-Woyke, H.; Bertelsmeier, B.; Riemer, P.; Bauer, E.: Datenbanksysteme. Pearson Studium, 2007.

[Fo07] Forbrig, P.: Objektorientierte Softwareentwicklung mit UML. Hanser Fachbuchverlag, 2007.

[Go03] Gornik, D.: UML data modeling profile, IBM Rational Software Whitepaper TP 162 05/02, 2003.

[MM13] Milovanovic, V.; Milicev, D.: An interactive tool for UML class model evolution in database applications. Software & Systems Modeling 14/3, S. 1273–1295, Sep. 2013.

[MVC03] Marcos, E.; Vela, B.; Cavero, J. M.: A Methodological Approach for Object-Relational Database Design using UML. Software and Systems Modeling 2/1, S. 59–72, März 2003.

[OS13] Oestereich, B.; Scheithauer, A.: Analyse und Design mit der UML 2.5. Gruyter, de Oldenbourg, 2013.

[RQ12] Rupp, C.; Queins, S.: UML 2 glasklar. Hanser Fachbuchverlag, 2012.

[Sa16] Salgert, B.: Modellierung von Datenbanken mit UML im Round-Trip-Engineering, Masterarbeit, Hochschule Düsseldorf, 2016.

[Sp01] Sparks, G.: Database modelling in UML, Sparx Systems whitepaper, 2001.

[TUT16] The UMLet Team: UMLet 14.2 – Free UML Tool for Fast UML Diagrams, 2016, URL: http://www.umlet.com, Stand: 20. 10. 2016.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 365

Serverseitige Aggregation von Zeitreihendaten in verteilten NoSQL-Datenbanken

Oliver Swoboda[1]

Abstract: Die effiziente Erfassung, Abspeicherung und Verarbeitung von Zeitreihendaten spielt in der Zeit von leistungsstarken Anwendungen eine große Rolle. Durch die schnelle und stetig wachsende Erzeugung von Daten ist es nötig, diese in verteilten Systemen abzuspeichern. Dadurch wird es nötig über Alternativen zur sequenziellen Berechnung von Aggregationen, wie Minimum, Maximum, der Standardabweichung oder von Perzentilen nachzudenken. Diese Arbeit untersucht, wie existierende Zeitreihendatenbanken im Hadoop-Ökosystem Aggregationen umsetzen und welche Probleme bei der sequenzuellen Berechnung auftreten. Um diese Probleme zu lösen, wird gezeigt, wie Aggregationen auf Zeitreihendaten verteilt und parallel in verschiedenen Systemen umgesetzt werden können und welche Herangehensweise bessere Laufzeiten liefert.

Keywords: Apache Accumulo, Apache Flink, Zeitreihendaten, Aggregation, serverseitig, verteilt, iterativ

1 Einleitung

Heutzutage werden durch leistungsstarke Anwendungen eine Vielzahl von zeitabhängigen Daten generiert, die es effizient zu erfassen, abzuspeichern und zu Verarbeiten gilt [ADEA11]. Stetig wachsende Datenmengen in Verbindung mit steigenden Anforderungen bei deren Auswertung, lassen konventionelle Herangehensweisen schnell an ihre Grenzen stoßen. Daher werden zunehmend verteilte Lösungen untersucht [ADEA11, Ch10]. Insbesondere ist die parallele und verteilte Berechnung von Aggregationen, wie exakten Perzentilen, eine Herausforderung [SS98].

Die vorliegende Arbeit untersucht einerseits, wie sich die Verwendung unterschiedlicher Schemata in Apache Accumulo[2] auf die Laufzeit von iterativen Aggregatfunktionen auswirken, um herauszufinden ob und in wie weit die Art des Abspeicherns einen Einfluss auf die Performanz der Berechnungen hat. Andererseits wird überprüft, ob eine serverseitige Umsetzung im Datenbanksystem schnellere Berechnungen als in einer Ausführungsumgebung, wie Apache Flink[3], liefert. Außerdem werden Möglichkeiten zur Umsetzung von verteilten und parallelen Aggregationsberechnungen vorgestellt und an einem realen Datensatz evaluiert.

Abschnitt 2 gibt zunächst einen Überblick über verwandte Arbeiten. Abschnitt 3 und Abschnitt 4 stellen das Konzept und die Umsetzung der Aggregationen vor, welche in Abschnitt 5 anhand von realen Daten evaluiert werden.

[1] Universität Leipzig, ScaDS Leipzig, Ritterstrasse 9-13, 04109 Leipzig, mai11brw@studserv.uni-leipzig.de
[2] https://accumulo.apache.org/
[3] https://flink.apache.org/

2 Hintergrund und verwandte Arbeiten

Die sequenzielle Berechnung von Aggregationen über Zeitreihendaten im Umfang von mehreren hundert Gigabyte und mehr, unterliegt einigen Einschränkungen. Heutzutage genutzte Mehrkernprozessoren können nicht optimal genutzt werden, da lediglich ein Thread für Berechnungen ausgenutzt wird. Weiterhin muss genug Festplattenspeicher vorhanden sein, um alle Daten auf einem Rechner ablegen und verarbeiten zu können. Die Größe des Arbeitsspeichers und der maximal mögliche CPU-Takt beeinflussen die Laufzeit der Aggregationen erheblich. Diese Faktoren sorgen dafür, dass sequenzielle Berechnungen mit großen Datenmengen im Vergleich zu parallelen und verteilten Ansätzen teurer sind, da es inzwischen billiger ist mehrere Server mit Standardserverhardware, als einen High-End-Server zu beziehen [Gr09].

Bei der Untersuchung mehrerer Zeitreihendatenbanken im Hadoop-Ökosystem zeigte sich, dass Vertreter, wie OpenTSDB[4] und KairosDB[5], Daten verteilt abspeichern und Aggregationen sequenziell berechnen [Op15, Un15]. OpenTSDB nutzt die NoSQL-Datenbank Apache HBase[6] zum verteilten abspeichern von Zeitreihendaten, wohingegen KairosDB auf Apache Cassandra[7] aufsetzt. Neben Aggregationen, wie Minimum, Maximum, Summe, Anzahl und Durchschnitt, wird auch die Berechnung der Standardabweichung und von Perzentilen unterstützt. Bei KairosDB können diese frei gewählt werden. Wenn mehr als 1028 Werte vorliegen, werden die Perzentile näherungsweise bestimmt [Pe16]. Mit OpenTSDB hat man die Möglichkeit fest vorgegebene Perzentile (50%, 75%, 90%, 95%, 99%, 99.9%) zu berechnen [Op16]. In dieser Arbeit sollen frei wählbare Perzentile, unabhängig von der Anzahl der Werte, exakt berechnet werden. Bei beiden Vertretern werden die verteilten Daten zunächst zu einem Knoten übertragen und anschließend sequenziell berechnet.

Timely[8] ist eine Zeitreihendatenbank der NSA, die Apache Accumulo zum Abspeichern der Daten und zum Berechnen von Aggregationen benutzt. Da Timely erst im Laufe dieser Arbeit veröffentlicht wurde, konnten keine weiteren Untersuchungen vorgenommen werden.

Da Perzentile in dieser Arbeit exakt bestimmt werden sollen, kommen Verfahren, die die Ergebnisse näherungsweise mit einem zufälligen Beispielsatz der Daten berechnen [BS09], nicht in Frage. Um ein Perzentil sequenziell zu bestimmen, müssen die Werte sortiert vorliegen und mit Hilfe der Gesamtanzahl kann dann, z.B. beim Median (50% Perzentil), die gesuchte Stelle $i = \frac{N}{2}$ der Menge A mit $|A| = N$ bestimmt werden, indem über A iteriert wird, bis die gesuchte Stelle erreicht wurde. Im parallelen und verteilten Fall, liegt keine Sortierung der Werte vor und es ist im Allgemeinen nicht möglich, das Ergebnis aus Teilergebnissen zu erhalten. Allerdings gibt es iterative Verfahren für die Berechnung [SS98].

[4] http://opentsdb.net/
[5] https://kairosdb.github.io/
[6] http://hbase.apache.org/
[7] http://cassandra.apache.org/
[8] https://nationalsecurityagency.github.io/timely/

3 Ansatz

3.1 Berechnung in Apache Accumulo

Accumulo bietet ein serverseitiges Programmierframework, die Iteratoren[9]. Dabei handelt es sich im Funktionen, die auf dem Server auf einem oder mehreren Key-Value-Paaren[10] ausgeführt werden. Beim Scanvorgang werden erst die Systemiteratoren, z.B. zum Filtern nach Version oder Zugriffsrechten, und dann Iteratoren vom Nutzer ausgeführt. Ein Iterator erhält die Daten seines Vorgängers. Wenn z.B. ein Iterator die Anzahl der gelesenen Werte liefert, dann kann dessen Nachfolger nicht mehr auf die ursprünglichen Werte zugreifen.

Für die Berechnung von Minimum, Maximum, Summe, Anzahl und Durchschnitt bestimmt jeder Iterator ein Teilergebnis über die gelesenen Daten. Anschließend werden alle Ergebnisse zum Master gesendet und zusammengeführt. Für die Berechnung der Standardabweichung σ müssen normalerweise alle Werte zwischengespeichert werden. Mit der modifizierten Gleichung (1) lässt sich das umgehen, indem die Summen fortlaufend für neue Werte aktualisiert werden. Bei den Perzentilen muss bei der Anwendung des iterativen Algorithmus [SS98] ein Pivotelement, im Allgemeinen der Median der Mediane, bestimmt werden. Hierfür muss jeder Iterator den Median über eine Teilmenge der Daten bilden. Ohne die Daten zwischenzuspeichern, ist das im Allgemeinen nicht möglich, da dazu erst die Anzahl der Werte bestimmt und anschließend erneut über alle Key-Value-Paare iteriert werden müsste. Dafür wäre ein weiterer Scanvorgang notwendig.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} A_i^2 - \frac{(\sum_{i=1}^{N} A_i)^2}{N}}{N-1}} \tag{1}$$

Mit Histogrammen ist es möglich, Perzentile in einem Scanvorgang zu berechnen [DZ12]. Hierbei erstellt jeder Iterator ein Histogramm der gelesenen Werte und liefert zusätzlich deren Gesamtanzahl zurück. Anschließend werden die Histogramme zusammengeführt und der gesuchte Rang k, abhängig vom Perzentil, mit der Nearest Rank Methode (Gleichung (2)) bestimmt. Abschließend kann das Ergebnis durch Aufsummieren der Wertanzahlen bestimmt werden. Sobald die Summe der gelesenen Werte größer oder gleich k ist, entspricht die aktuelle Stelle im Histogramm dem gesuchten Perzentil. Die Perzentilberechnung mit Histogrammen ist auf einen kleinen Wertebereich beschränkt. Da es sich bei den genutzten Daten lediglich um ganze Zahlen handelt, die einen Bereich von unter 2000 Werten abdecken, fällt der Speicherbedarf zum Erstellen und Übertragen der Histogramme gering aus. Bei größeren Wertebereichen

$$k = \left\lceil \frac{Perzentil}{100} * Anzahl \right\rceil \tag{2}$$

[9] http://accumulo.apache.org/1.8/accumulo_user_manual.html#_iterators
[10] http://accumulo.apache.org/1.8/accumulo_user_manual.html#_data_model

Metrik	Anzahl Werte
PRCP	907.661.688
TMAX	357.693.332
TMIN	355.857.064
SNOW	317.110.211
SNWD	257.911.061
Andere	389.867.034
Gesamt	2.586.100.390

Tab. 1: Zusammensetzung des Datasatzes

Datenmodell	Row ID	Column Family	Version
Schema1	Monat_SID	Metrik	Tage [1,31]
Schema2	Jahr_SID	Metrik	Tage [1,366]

Tab. 2: Die Schemata

3.2 Datensatz

Bei den Daten handelt es sich um Wetterdaten des Global Historical Climatology Network - Daily (GHCN-Daily) [Me12] und bestehen aus der Stationskennung, dem Datum, der Wettermetrik und dem zugehörigen Wert für den angegebenen Tag. Bei den Wettermetriken werden Niederschlag *PRCP*, Schneefall *SNOW*, Schneetiefe *SNWD*, maximale Temperatur *TMAX* und minimale Temperatur *TMIN* betrachtet, da diese den Großteil der Daten ausmachen, was Tabelle 1 verdeutlicht. Die Daten reichen vom 01.01.1763 bis zum 28.07.2016.

3.3 Datenmodell

Für das Abspeichern der Wetterdaten werden zwei Schemata, die in Tabelle 2 zu sehen sind, genutzt. Tabelle 3 und Tabelle 4 verdeutlichen den Aufbau mit Beispieldaten. *Schema1* nutzt den Monat und die Stationskennung (SID) für die Row ID und legt pro Tag des Monats eine Version an. *Schema2* hingegen nutzt das Jahr in der Row ID und legt pro Tag des Jahres eine Version in der Tabelle an. Beide Schemata speichern die Wettermetriken in der ColumnFamily und nutzen keine Column Qualifiers.

Monat_SID	Metrik	Version	Wert
199401_CA002303986	TMIN	1	-390

Tab. 3: Schema1 am Beispiel

Jahr_SID	Metrik	Version	Wert
1994_CA002303986	TMIN	1	-390

Tab. 4: Schema2 am Beispiel

Durch Locality Groups können Zugriffe auf unterschiedliche Column Families beschleunigt werden. Alle Wettermetriken wurden einer Gruppe zugewiesen, wodurch deren Daten in unterschiedlichen Dateien abgelegt werden. Dadurch müssen die Daten anderer Metriken nicht gelesen und übersprungen werden.

Tabellen werden in Accumulo in Table Splits aufgeteilt. Diese Splits können durch Angabe eines Schwellenwerts eingestellt oder direkt angegeben werden. Pro Split wird maximal ein Iterator ausgeführt. In dieser Arbeit wurde pro Jahr ein Split, also 254 Splits (Zeitraum 1763-2016), angelegt.

4 Implementierung

4.1 Apache Accumulo

In Accumulo wurden die Aggregationen mit einem Iterator in Java umgesetzt. Dieser wird einem BatchScanner[11] hinzugefügt, um die parallele Ausführung zu gewährleisten. Dem Iterator wird vor dem Starten des Scanvorgangs übergeben, welche Aggregation berechnet werden soll. Jede Aggregationsklasse implementiert eine *add*-, *merge*- und *getResult*-Funktion. Während ein Iterator über die Daten iteriert, werden neue Werte durch die Methode *add* dem Aggregator hinzugefügt und dieser letztendlich zurückgeliefert. Anschließend werden die Einzelergebnisse durch die *merge*-Funktion in einem Aggregator zusammengeführt. Das Endergebnis kann dann durch die Methode *getResult* abgerufen werden.

Algorithmus 1 : Pseudocode für die Berechnung des Minimums

```
Data : Wetterdaten verteilt auf x Iteratoren in einem Cluster
Result : Minimum aller Werte
1 min ← ∞
2 forall Iterator iter im Cluster do
3     minIt ← ∞
4     foreach Wert value in iter do
5         if value < minIt then
6             minIt ← value
7     if minIt < min then
8         min ← minIt
```

[11] http://accumulo.apache.org/1.8/accumulo_user_manual.html#_batchscanner

Algorithmus 1 zeigt am Beispiel des Minimums den Ablauf bei der verteilten Berechnung. Hierbei ist zu beachten, dass die Iteratoren parallel ausgeführt werden. Jeder Iterator bestimmt das lokale Minimum *minIt* und liefert das Ergebnis an den Master. Dieser bildet aus den erhaltenen Teilergebnissen das Endergebnis *min* sequenziell. So werden mehrere Milliarden Werte parallel und verteilt voraggregiert, so dass das Ergebnis aus wenigen Hundert Datenpunkten sequenziell bestimmt werden kann. Das Maximum und die Summe werden auf ähnliche Weise berechnet. Beim Durchschnitt wird die Summe und die Anzahl, bei der Standardabweichung zusätzlich noch die Summe der Quadrate der Werte mitgeführt. Beim Abrufen des Ergebnisses wird dann für den Durchschnitt die Summe durch die Anzahl geteilt und bei der Standardabweichung Gleichung (1) angewendet. Für die Berechnung von Perzentilen werden *Maps* mit dem Datenpunkt als Key und einem Zähler als Value genutzt um Histogramme der gelesenen Werte zu erstellen. Auf dem Master werden die *Maps* zusammengeführt um ein Histogramm der Gesamtdaten zu erhalten. Mit Gleichung (2) wird dann der Rang k bestimmt. Anschließend wird über die Werte im Histogramm iteriert, die durch Nutzung einer *TreeMap* sortiert vorliegen. Beim Iterieren werden die Anzahlen addiert und wenn diese gleich oder größer k sind, ist der aktuelle Wert das gesuchte Perzentil.

4.2 Apache Flink

Bei jedem Flinkjob[12] müssen zuerst Daten eingelesen werden. Mit dem *AccumuloInputFormat* können die Daten aus der Datenbank als *DataSet* eingelesen werden. Nach dem Einlesen werden die Daten, die als Key-Value-Paare vorliegen, zunächst mit einer *FlatMapFunction* zu Triplets, welche den Wert, die Anzahl und das Quadrat des Wertes enthalten, transformiert. Für die Verarbeitung erhält jeder Task einen Teil der Daten. Danach werden die erhaltenen Teilergebnisse durch *CombineFunctions* zusammengeführt und das Endergebnis bestimmt.

Minimum, Maximum und Summe sind in Flink bereits implementiert und können auf das erste Element des Triplets angewendet werden. Zur Bestimmung der Anzahl wird die Summe vom zweiten Element berechnet. Der Durchschnitt ergibt sich aus der Summe des ersten Elements und anschließendem summieren der Anzahlen der Werte. Abschließend werden die Teilergebnisse in einer *GroupCombineFunction* aufsummiert und das Ergebnis durch das Teilen der Summe aller Elemente durch deren Anzahl ermittelt. Bei der Standardabweichung werden alle drei Stellen des Triplets benutzt, indem alle enthaltenen Werte jeweils in ihren Tasks aufsummiert und dann wiederum in einer GroupCombineFunction vereint werden. Das Endergebnis wird danach durch Anwenden von Gleichung (1) berechnet.

Bei den Perzentilen wird ähnlich wie in Accumulo vorgegangen. Mit einer *MapPartitionFunction* wird von jedem Task unter Verwendung einer *TreeMap* ein Histogramm der gelesenen Werte erstellt. Diese liefert dann geordnete Paare von *TreeMaps* und Anzahlen der Elemente. Anschließend findet ein *GroupCombine* zum Zusammenführen der *TreeMaps* und Aufsummieren der Anzahlen statt. Außerdem wird an dieser Stelle die Nearest Rank

[12] https://ci.apache.org/projects/flink/flink-docs-release-1.2/concepts/index.html

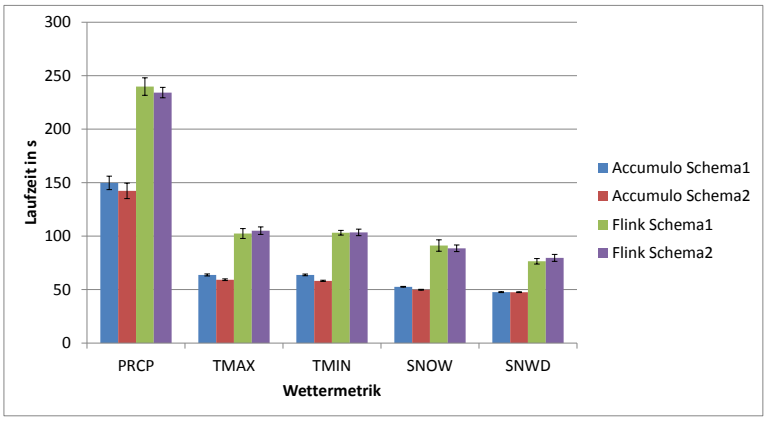


Abb. 1: Ergebnisse der Testläufe

Methode zum Bestimmen des Rangs k angewendet und damit, wie bereits bei Accumulo beschrieben, das Perzentil bestimmt.

5 Evaluation

Um die verschiedenen Schemata und Techniken zu vergleichen, wurde eine Reihe von Tests konfiguriert und ausgeführt. Zunächst wurde überprüft, wie lange ein Scan über die gesamte Tabelle für *Schema1* und *Schema2* jeweils in Accumulo und Flink dauert. Die Ergebnisse hiervon bilden die Baseline. Anschließend wurden die Laufzeiten der implementierten Aggregationen Minimum, Maximum, Summe, Anzahl, Durchschnitt, Standardabweichung und Median, als Vertreter der Perzentile, für die Wettermetriken *TMIN*, *TMAX*, *PRCP*, *SNOW*, *SNWD* gemessen.

Für die Tests stand ein Cluster bestehend aus fünf Knoten zur Verfügung, jeder mit:

- 64 GB Arbeitsspeicher
- 7 TB HDD
- Intel Core i7-6700 (4 Kerne / 8 Threads)
- zwei Netzwerkschnittstellen: extern mit 1 Gbit/s, intern mit 1 Gbit/s
- CentOS Linux release 7.2.1511 (Core)

Dabei fungiert ein Knoten als Master und vier als Worker. Bei der Konfiguration von Accumulo, wurde jedem der vier Tablet Server 16 GB und dem Master 8 GB Arbeitsspeicher zugewiesen. Flink nutzt für jeden der vier Taskmanager 25 GB Arbeitsspeicher.

In Abbildung 1 sind die vorläufigen Ergebnisse der Testläufe zu sehen. Da alle Aggregationen nahezu identische Laufzeiten aufwiesen, wird hier nur die durchschnittliche Laufzeit

des Minimums über den gesamten Zeitraum dargestellt. Für jede Aggregation wurden 10 Testläufe durchgeführt und davon der Durchschnitt und die Standardabweichung berechnet.

Bei beiden Systemen steigt die Laufzeit annähernd proportional zur Datenmenge. Es ist zu sehen, dass Accumulo bei allen Tests deutlich bessere Laufzeiten als Flink erzielt. Ein möglicher Grund dafür ist, dass Flink zuerst die Daten beziehen muss, bevor die Aggregationsberechnungen durchgeführt werden können. Welchen Anteil das Auslesen der Daten mit Flink bei den Testläufen einnimmt, wird Gegenstand zukünftiger Untersuchungen sein. Beim Vergleich der Schemata fällt auf, dass in beiden Systemen kein großer Unterschied zwischen *Schema1* und *Schema2* zu erkennen ist. In Accumulo ist *Schema2* im Durchschnitt schneller. Bei Flink ist jedoch keine klare Tendenz zu sehen.

6 Diskussion

Die ausgeführten Implementierungen in dieser Arbeit sind an die benutzten Daten angepasst. Diese enthalten ausschließlich ganze Zahlen, weswegen bei der Berechnung der Standardabweichung nicht auf Rundungsfehler geachtet werden muss, die bei Fließkommazahlen auftreten würden. Außerdem können dadurch und weil die Werte einen kleinen Wertebereich haben, Perzentile mit Histogrammen der Klassengröße Eins berechnet werden.

Enthalten die Daten Fließkommazahlen, kann es beim Nutzen von Gleichung (1) dazu kommen, dass versucht wird die Wurzel aus einer negativen Zahl zu ziehen[Kn98]. Im allgemeinen Fall muss also eine anderer Algorithmus verwendet werden. Dasselbe gilt für Perzentile, die im Allgemeinen mit einem iterativen Algorithmus berechnet werden müssen, der pro Iteration das Problem verkleinert und gegebenenfalls das Ergebnis schlussendlich sequenziell bestimmt.

7 Fazit und Ausblick

In dieser Arbeit wurde gezeigt, wie iterative Aggregationsberechnungen parallel und verteilt in unterschiedlichen Systemen, Apache Accumulo für die datenbankinterne und Apache Flink für die externe Berechnung, umgesetzt werden können. Es wurde untersucht und evaluiert, wie die implementierten Aggregationen bei steigender Datenmenge skalieren und welches System die besseren Laufzeiten erzielt. Zusätzlich wurde mit zwei Schemata überprüft, ob deren Aufbau einen Einfluss auf die Performanz der Berechnungen hat. Die Ergebnisse zeigen, dass die Laufzeit der Aggregationsberechnungen annähernd proportional zur Datenmenge steigt und Accumulo in allen Testfällen schneller ist. Beim Vergleich der Schemata wurde festgestellt, dass keine großen Unterschiede bei den Laufzeiten festzustellen sind, aber die Berechnungen mit *Schema2* in Accumulo tendenziell schneller ablaufen.

Zukünftig sind weitere Untersuchungen in Bezug auf den zeitlichen Anteil des Einlesens der Daten mit Apache Flink nötig. Weiterhin können Testläufe mit unterschiedlichen Zeiträumen Aufschluss darüber geben, ob sich ein Schema für einen bestimmten Zeitraum besser eignet.

8 Danksagung

Ich möchte mich ganz herzlich bei Matthias Kricke und Martin Grimmer für die Betreuung vor und während meiner Masterarbeit bei mgm technology partners GmbH und an der Universität Leipzig bedanken. Eric Peukert möchte ich für die Betreuung am ScaDS Leipzig und am Lehrstuhl Datenbanksysteme der Universität Leipzig danken. Die vorliegende Arbeit wurde teilweise gefördert durch das Bundesministerium für Bildung und Forschung innerhalb des Competence Center for Scalable Data Services and Solutions (ScaDS) Dresden/Leipzig. (BMBF 01IS14014B)

Literatur

[ADEA11] Agrawal, Divyakant; Das, Sudipto; El Abbadi, Amr: Big data and cloud computing: current state and future opportunities. In: Proceedings of the 14th International Conference on Extending Database Technology. ACM, S. 530–533, 2011.

[BS09] Buragohain, Chiranjeeb; Suri, Subhash: Quantiles on streams. In: Encyclopedia of Database Systems, S. 2235–2240. Springer, 2009.

[Ch10] Chen, Chun; Chen, Gang; Jiang, Dawei; Ooi, Beng Chin; Vo, Hoang Tam; Wu, Sai; Xu, Quanqing: Providing scalable database services on the cloud. In: International Conference on Web Information Systems Engineering. Springer, S. 1–19, 2010.

[DZ12] Dinkel, Kevin; Zizzi, Andrew: Fast Median Finding on Digital Images. In: AIAA Regional Student Paper Conference. April. Jgg. 4, 2012.

[Gr09] Greenhalgh, Adam; Huici, Felipe; Hoerdt, Mickael; Papadimitriou, Panagiotis; Handley, Mark; Mathy, Laurent: Flow processing and the rise of commodity network hardware. ACM SIGCOMM Computer Communication Review, 39(2):20–26, 2009.

[Kn98] Knuth, Donald Ervin: The Art of computer programming. Volume 2, Seminumerical algorithms. S. 216, 1998.

[Me12] Menne; M.J.; Durre, I.; Korzeniewski, B.; McNeal, S.; Thomas, K.; Yin, X.; Anthony, S.; Ray, R.; Vose, R.S.; Gleason, B.E.; Houston, T.G.: Global historical climatology network-daily (GHCN-Daily), Version 3.22. NOAA National Climatic Data Center, 2012. http://doi.org/10.7289/V5D21VHZ, Stand:18.10.2016.

[Op15] OpenTSDB Aggregators.java, https://github.com/OpenTSDB/opentsdb/blob/e2efe9a4875f8cddd3404e9c6b64f5aab4b72c0a/src/core/Aggregators.java#L184, Stand: 18.10.2016.

[Op16] OpenTSDB Aggregators, http://opentsdb.net/docs/build/html/user_guide/query/aggregators.html, Stand: 18.10.2016.

[Pe16] Percentile aggregator returns inconsistent values for longer sample lengths, https://github.com/kairosdb/kairosdb/issues/290,Stand:18.10.2016.

[SS98] Saukas, Einar LG; Song, Siang W: Efficient selection algorithms on distributed memory computers. In: Proceedings of the 1998 ACM/IEEE conference on Supercomputing. IEEE Computer Society, S. 1–26, 1998.

[Un15] Understanding KairosDB for production (distributed database)?, https://groups.google.com/d/msg/kairosdb-group/Pb2W9CXsSkY/0NFHOyKKCgAJ, Stand: 18.10.2016.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 375

Understanding Trending Topics in Twitter

Roland Kahlert Matthias Liebeck Joseph Cornelius

Institute of Computer Science, Heinrich Heine University Düsseldorf, Germany
{roland.kahlert,joseph.cornelius}@hhu.de
liebeck@cs.uni-duesseldorf.de

Abstract: Many events, for instance in sports, political events, and entertainment, happen all over the globe all the time. It is difficult and time consuming to notice all these events, even with the help of different news sites. We use tweets from Twitter to automatically extract information in order to understand hashtags of real-world events. In our paper, we focus on the topic identification of a hashtag, analyze the expressed positive, neutral, and negative sentiments of users, and further investigate the expressed emotions. We crawled English tweets from 24 hashtags and report initial investigation results.

Keywords: Text Mining, Topic Recognition, Sentiment Analysis, Emotion Detection, Twitter

1 Introduction

Social media sites, such as Twitter, Facebook, and Instagram, allow users to share their thoughts, feelings, journals, and travels in the form of text and images. In order to group similar content, these sites provide the functionality to label posts with so-called hashtags. A hashtag consists of a string which is preceded by the character '#', like #fifa. Hashtags can be used for a variety of functions. They may be used to group posts by topics, by emotions (for instance #joy or #love) or by events in the real world (e.g., boxing matches or scandals) which are often discussed by the users.

The growth of social media sites has been enormous over the last years. Since many people use Twitter to discuss events, a system that is able to understand hashtags about events is envisaged. The system could be used on popular hashtags to textually describe what happened in the world, who is concerned about it, and what the public opinion is. It would be timesaving to receive such information as a summary without the need to read and understand hundreds of text messages, which are called "tweets" on Twitter.

There are a few issues that have to be taken into account with such an approach:

(i) Some events may only be discussed in a language other than English. In order for such a system to work, we need language-specific models for natural language processing components and must adapt our techniques to every supported language individually, which requires the ability to understand these languages. We focus on English tweets because of the large amount of available resources for natural language processing.

(ii) Different events can be tagged with the same hashtag. For instance, tweets can be tagged with #WorstDayOfMyLife. Person A might have had a series of bad news on a particular day, whereas person B might have another series of bad things that happened to him or her. As part of their nature, a summarization of such hashtags will fail because they comprise tweets of more than one event.

(iii) Smaller events might not be frequently discussed on social media. For example, the opening of a certain restaurant might be of interest. However, if this is not frequently discussed on Twitter, our system will not process the tweets from the hashtag since we gather tweets from the most popular hashtags.

This paper focuses on our approaches for an automatic understanding of hashtags about events. The remainder of this paper is structured as follows: The next chapter discusses related work about analyzing tweets. In Section 3, we briefly describe our dataset and highlight multiple subtasks which we consider interesting in a hashtag summarization. Thereafter, we report our initial results and exemplarily show good and bad results. We conclude in Section 5 and outline future work.

2 Related Work

The analysis of tweets has gained much popularity in the last years. The topic detection of tweets has been the focus of previous analyses [HD10, OKA10].

Hong and Davison [HD10] used *Latent Dirichlet Allocation (LDA)* [BNJ03] to analyze tweets. The authors focused on two tasks: (i) the prediction of whether a tweet will be retweeted in the future and (ii) the classification of users and their messages into topical categories. For the second task, tweets were crawled from more than 250 verified users who were supposedly posting messages belonging to one of 16 topics, such as *entertainment* or *politics*. Instead of using a fixed set of topics, we focus on automatically describing topics from tweets.

TweetMotif [OKA10] focused on the summarization of tweets that were returned from user queries. The authors used frequent n-grams of length 1 to 3 to extract multiple topic labels that are frequent in the returned tweets but infrequent among other tweets. In order to refine the results, similar topics and near-duplicated tweets were grouped together. TweetMotif's final visualization of a user query consists of multiple topic labels and several exemplary tweets that contain the topic labels.

3 Data and Investigated Tasks

In this section, we describe our Twitter dataset and briefly outline the tasks we focused on for our automated analysis.

3.1 Data

In this paper, we focus on tweets from Twitter that are limited to 140 characters in length. Due to the length restriction, tweets have unique characteristics as users often do not write complete sentences or utilize abbreviations to shorten their text content. Each tweet is posted by a user. We crawled English tweets from 23 different hashtags about events in 2015 and added the non-event hashtag #love. Table 1 lists the crawled hashtags in our dataset and their respective descriptions that we created manually. In total, we crawled roughly 3.3 million tweets.

Hashtag	Tweet count	Timeframe	Event description
#bbking	46275	Jan - Dec	Blues singer B.B. King died
#blatterout	19229	Jan - Dec	People demanding the resignation of Sepp Blatter
#broner	41967	May - Jul	Boxing: Adrian Broner vs Shawn Porter
#charlestonShooting	154061	Apr - Dec	Mass shooting in Charleston, SC
#dieselgate	11344	Feb - Dec	Volkswagen emissions scandal
#endAusterityNow	25875	May - Jul	Protests against austerity in UK
#expo2015	58995	Mar - Dec	Universal Exposition hosted by Milan
#fifa	179310	May - Jul	FIFA corruption scandal
#GameOfThrones	363576	May - Jul	Popular TV series
#germanwings	59684	Jan - Dec	Germanwings flight crashed in the Alps
#heartgate	1057	Jan - Dec	Twitter replaces favorite stars with hearts
#KGL9268	1074	Oct - Dec	Metrojet flight crashed in the Sinai
#love	1043780	May - Jul	All time trending topic
#nbafinals	435817	May - Jul	NBA finals: Warriors vs Cavaliers
#nepalearthquake	121579	Apr - Dec	Heavy earthquake in Nepal
#ohNoHarry	17455	May - Jul	Harry Styles falls off stage
#plutoflyby	32764	Jul	Space probe *New Horizons* reaches Pluto
#PSYAngBatasNgApi	77038	Jun - Dec	1M tweets for #PSYAngBatasNgApi
#seppblatter	22832	Jan - Dec	Sepp Blatter resigns after corruption scandal
#uswnt	261685	Jan - Dec	US women's soccer team wins the world cup
#volkswagen	92438	Jan - Dec	VW, but mainly including #dieselgate
#windows10	241637	Mar - Dec	Release of Windows 10
#wwdc15	52440	Jan - Dec	Apple Developer Conference 2015
#WWEChamber	411	Jun - Dec	World Wrestling: Owens vs Cane

Tab. 1: List of the crawled hashtags in our dataset and descriptions of their content

3.2 Research Tasks

We now briefly describe and motivate the three analysis tasks that we focus on.

3.2.1 Topic Detection

Topic detection is the task of automatically detecting key words or key phrases that describe the topic of text content. In our specific use case, we aim to summarize the discussion

topic of each hashtag in our dataset. As our dataset is focused on events, the desired summarization should convey a general understanding of each event. We focus on extractive techniques in order to summarize the content of the hashtag.

The topic detection of a hashtag is challenging. First of all, it is difficult to formally define the term *topic*. In our work, a summarization should contain information regarding an entity that performed an action or is affected by it. For example, let us assume that the following three example sentences are tweets:

> Sebastian Vettel won the Grand Prix.
> Vettel was the winner of the last race.
> The race in Monaco was won by the German driver Sebastian Vettel.

A manual summarization of these tweets could be *"The Monaco Grand Prix was won by the German driver Sebastian Vettel."* The automatically extracted topic should at least comprise a subset of these information. We make the assumption that each hashtag in our dataset, except for #love, contains one topic which should become apparent with our approaches if we have a large enough collection of tweets per hashtag.

However, the topic detection is still challenging because:

(i) Language is versatile since the same content can be described with different words, e.g., with synonyms. Persons or locations can be mentioned with their full names or only with parts of their names, like surnames. Additionally, Twitter users tend to use abbreviations.

(ii) Is it difficult to automatically evaluate the results of an algorithm or to compare the results of two different approaches because this requires a reasonable distance function between the generated output and manually created descriptions.

3.2.2 Sentiment Analysis

Sentiment analysis is the task of identifying positive, negative, and neutral statements in text content. This task is useful in a variety of application domains. For instance, a company might be interested in their customers' opinion on social media sites. Commonly analyzed application domains include customer reviews [HL04] and film reviews [PLV02].

Depending on the individual use case, the sentiments expressed should be analyzed with their viewpoint. As an example, the statement *"The new product of company A is a bad product."* is a negative statement for company A, but it might be a good statement for its competitor.

Sentiment analysis differs between application domains because of domain-specific wordings. For instance, product reviews on Amazon often contain adjectives that describe certain product features, like *"The battery life is really good."* In movie reviews, there are wordings like *"The actor deserves an Oscar for his performance."* which are very domain-specific.

3.2.3 Emotion Detection

In addition to positive and negative sentiments, we want to capture emotions that users express in order to gain a deeper insight into their feelings. Human emotions have been extensively studied in the past. Paul Ekman proposed the existence of basic emotions [Ek92] (anger, disgust, fear, joy, sadness, and surprise) which have been observed in the context of facial expressions. There has been research about other emotion models in the past, such as *Plutchik's wheel of emotions* [Pl80] which comprises the following eight basic emotions: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

In the case of our dataset, we want to automatically identify emotions in tweets in order to provide a rough overview over the users' feelings, e.g., if they express more sadness than joy.

4 Experimental Results

After describing our dataset and interesting tasks, we now outline our approaches and experimental results.

First, we need to preprocess the tweets in our dataset with a natural language processing pipeline. We use the Twitter-specific pipeline TweeboParser [Ko14] to extract sentences, words, and their part-of-speech tags. Afterwards, we are able to extract task-specific information, such as nouns and verbs for a topic detection.

4.1 Topic Detection

One of the more intuitive solutions to the topic detection of the tweets is to simply count all occurrences of words in a bag-of-words model and report the m most commonly used words in all tweets. Table 2 lists the results of our topic detection approaches for the hashtag #KGL9268. In our case, most of these ten words (listed in the row named *Counter*) are prepositions and articles which do not add any contextual information. They can be regarded as stopwords and be filtered out using a stopword list, which we refer to as *StopWordCount*.

Another idea to express a topic is to only use nouns and names. To accomplish this, we use the part-of-speech tags to filter the words of every tweet. Thereafter, we rank the most frequent words as *NounPOSCount*. We also experiment to describe the storyline by choosing the most frequent verbs (*VerbPOSCount*).

Additionally, we examine if the first sentence of a tweet provides most of the information. We combine our approaches regarding nouns and verbs and only apply them to the first sentence in *FirstSentenceCount*. Our approach *VerbPhraseCount* considers dependencies between the root verbs of each sentence and the nouns in a bag-of-words model.

Approach	Top 10 results
Counter	#kgl9268, the, to, of, in, crash, russian, #egypt, a, plane
StopWordCount	#kgl9268, crash, russian, #egypt, plane, #russia, #7k9268, flight, egypt, sinai
NounPOSCount	egypt sinai, plane, russian spygame, crash, flight, #kgl9268, family, bomb, #egypt, airliner
VerbPOSCount	crash, say, claim, bring, know, cause, break, it', fly, ru
FirstSentenceCount	crash, plane, russian spygame, egypt sinai, flight, #kgl9268, family, #egypt, bomb, condolence
VerbPhraseCount	crash[plane], claim[video], break[plane, apart], ru[poo-poo'd], cause[flight], confirm[offical], rip[soul], mourn[today, people], kill[flight, people], fly[airline]
NGramCount ($n = 3$)	[russian, plane, crash], [sinai, plane, crash], [russian, airliner, crash], [plane, crash, survivor], [survivor, russian, airliner], [crash, russian, flight], [plane, crash, egypt], [survivor, crash, russian], [flight, sinai, egyptian], [russian, flight, sinai]
LDA 1st	plane crash, russian airliner, condolence family, flight crash, thought prayer, crash victim, bbc news, crash survivor, 224 people, family friend
LDA 2nd	russian plane, russian flight, sinai plane, crash russian, crash site, claim responsibility, crash egypt, deep condolence, crash sinai, #kgl9268 crash

Tab. 2: Overview of the topic detection for hashtag #KGL9268

In order to better understand the structure of a tweet, we also extract word sequences of length n from the tweets which were previously filtered to only contain adjectives, numbers, verbs, and nouns (*NGramCount*).

Futhermore, we use *Latent Dirichlet Allocation (LDA)* [BNJ03] to find k topics in a collection of tweets. For each word w from a vocabulary V, LDA calculates the probability $\varphi_{w,t}$ that w belongs to topic t. In our approach, we set $k = 2$, use bigrams, and extract 10 words with the highest probability from each topic. For #KGL9268, the *LDA* approach seems to provide the most information.

We now list several observations that we made during our experiments.

The hashtag #ohNoHarry is a good example for our use case because its name is not self-explanatory. The *NGramCount* approach provides us almost sentence-like n-grams, which tell us that singer Harry Styles fell off a stage and the community was amused of it.

Furthermore, #volkswagen is a good example for multiple topics inside a hashtag. Table 3 shows that LDA is capable of differentiating the 2015 emission scandal from other news regarding the company, e.g., new car models. The results for #love show us that the hashtag does not form a coherent topic. This could be due to the large amount of tweets that do not share the same topic. While the first topic of #nepalearthquake is about asking for help, the second topic deals with the victims.

We also observed for #PSYAngBatasNgApi that Twitter users express pride about one million tweets in the hashtag, but our approaches were unable to tell what exactly the hashtag is about. A reason for this might be the missing contextual information in our collected tweets because fans of a specific entity are not going to describe the entity in their tweets.

Hashtag	First topic	Second topic
#volkswagen	#volkswagen volkswagen, volkswagen beetle, emission scandal, beetle classic, #volkswagen scandal	volkswagen golf, golf gti, vw golf, #volkswagen golf, new #volkswagen
#love	good morning, #love love, #love com, celine dion, #love #quote	#love #photography, #photography #fashion, love #love, love love, #money #love
#nepalearthquake	nepal earthquake, relief effort, need help, #nepalearthquake relief, #nepalearthquake victim	death toll, affect #nepalearthquake, victim #nepalearthquake, thought prayer, people nepal

Tab. 3: Top five *LDA* results for various hashtags

4.2 Sentiment Analysis

Since our corpus does not contain sentiment annotations, we are unable to evaluate the output of any sentiment analysis approach on our dataset. However, SemEval-2016 covered a specific challenge [Na16] for sentiment analysis in Twitter. We decided to use the publically available system from [Gi16] which ranked at fifth place.

[Gi16] classifies each tweet as positive, neutral or negative by using an ensemble of two linear support vector machines (SVM). The first SVM is trained on part-of-speech tags, sentiment lexicons, negations, cluster of tweets, and morphological features. The second SVM uses the centroid of the word embeddings in GloVe [PSM14] of all words in a tweet. The embeddings were pre-trained on tweets.

The sentiment distribution of the hashtags in our dataset is illustrated in Figure 1. The hashtag #expo2015 has a significantly low amount of negative tweets. The most number of negative tweets were posted regarding #charlestonShooting while the amount of neutral tweets remains stable across all hashtags. It is quite surprising to see that more negative sentiments are expressed in the hashtag #heartgate than in #KGL9268 or in #germanwings. This prompted us to further investigate the emotions expressed in these hashtags.

4.3 Emotion Detection

In this work, we use the emotion dictionary *EmoLex* [MT13] which is based on *Plutchik's wheel of emotions* [Pl80] and the following eight basic emotions: anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

Each word in EmoLex is listed as positive or negative and has 8 binary values, which represent whether the word expresses the respective emotion. For example, the word *love* is listed as $[0,0,0,0,1,0,0,0]$. EmoLex was constructed via crowdsourcing on Amazon's Mechanical Turk. In crowdsourcing, small tasks, called *Human Intelligence Tasks* (HIT), are solved by multiple human workers at a very low price.

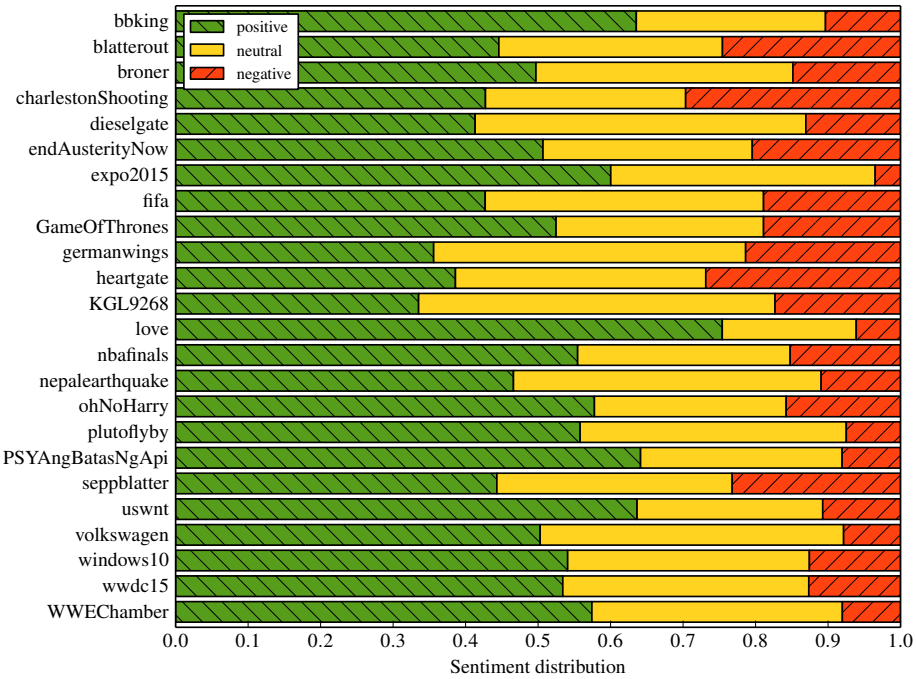


Fig. 1: Summarized sentiments per hashtag

In our analysis of the emotions, we use the lexicon to look up every word from each tweet and calculate an average emotion distribution. Our results are illustrated in Figure 2.

The hashtag with the most emotions expressed is #charlestonShooting where roughly 30% of the tweets are emotional, mostly expressing fear and sadness. The fewest emotions were expressed in the scientific hashtag #plutoflyby, which mostly consists of news-like tweets. It can be observed that fear is often expressed in hashtags that are about events with fatalities, such as airplane crashes (#KGL9268 and #germanwings). Surprisingly, joy is often expressed for #bbking and the tweets in #ohNoHarry contain a high amount of sadness.

5 Conclusion and Future Work

We have presented our approach to an automated understanding of hashtags on Twitter that deal with events. We focused on an extractive topic detection, sentiment analysis, and emotion detection. Regarding topic detection, the more intuitive methods do not provide enough information in order to understand the hashtags. Our *NGramCount* approach and LDA yield short word sequences that help us to get a rough idea of the content. Furthermore, we analyzed expressed sentiments and emotions in the dataset.

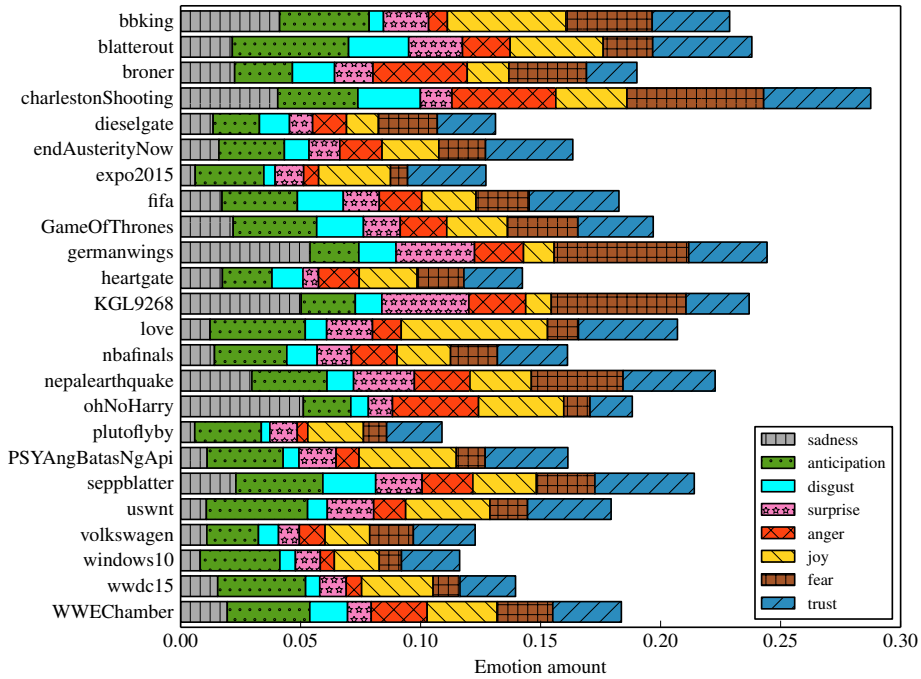


Fig. 2: Summarized emotions per hashtag

In our future work, we want to annotate a subset of our crawled tweets to be able to evaluate techniques for sentiment analysis and to tweak existing approaches for hashtags about events. Furthermore, we will use methods to predict semantic textual similarity between tweets and merge tweets that are almost identical. We aim to utilize the unsupervised *Overlap* method [Li16] and adapt it to Twitter.

Another interesting task is the prediction of user demographics, like age and gender. With such information, a middle-aged man could skip topics that are discussed by younger girls, such as fashion shows. Unfortunately, demographic information about users are not publically available. We would like to use crowdsourcing to annotate demographics for a subset of the users that have posted the tweets. Then, we could incorporate and adapt systems from the *author profiling* challenges in the PAN Series [Ra13], for instance [MLC16], to generate statistics about the users.

References

[BNJ03] Blei, David M.; Ng, Andrew Y.; Jordan, Michael I.: Latent Dirichlet Allocation. Journal of Machine Learning Research, 3:993–1022, 2003.

[Ek92] Ekman, Paul: An Argument for Basic Emotions. Cognition and Emotion, 6(3-4):169–200, 1992.

[Gi16] Giorgis, Stavros; Rousas, Apostolos; Pavlopoulos, John; Malakasiotis, Prodromos; Androutsopoulos, Ion: aueb.twitter.sentiment at SemEval-2016 Task 4: A Weighted Ensemble of SVMs for Twitter Sentiment Analysis. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). Association for Computational Linguistics, pp. 96–99, 2016.

[HD10] Hong, Liangjie; Davison, Brian D.: Empirical Study of Topic Modeling in Twitter. In: Proceedings of the First Workshop on Social Media Analytics. SOMA '10. ACM, pp. 80–88, 2010.

[HL04] Hu, Minqing; Liu, Bing: Mining and Summarizing Customer Reviews. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '04. ACM, pp. 168–177, 2004.

[Ko14] Kong, Lingpeng; Schneider, Nathan; Swayamdipta, Swabha; Bhatia, Archna; Dyer, Chris; Smith, Noah A.: A Dependency Parser for Tweets. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014. pp. 1001–1012, 2014.

[Li16] Liebeck, Matthias; Pollack, Philipp; Modaresi, Pashutan; Conrad, Stefan: HHU at SemEval-2016 Task 1: Multiple Approaches to Measuring Semantic Textual Similarity. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). Association for Computational Linguistics, pp. 607–613, 2016.

[MLC16] Modaresi, Pashutan; Liebeck, Matthias; Conrad, Stefan: Exploring the Effects of Cross-Genre Machine Learning for Author Profiling in PAN 2016. In: Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum,. pp. 970–977, 2016.

[MT13] Mohammad, Saif M.; Turney, Peter D.: Crowdsourcing a Word-Emotion Association Lexicon. 29(3):436–465, 2013.

[Na16] Nakov, Preslav; Ritter, Alan; Rosenthal, Sara; Sebastiani, Fabrizio; Stoyanov, Veselin: SemEval-2016 Task 4: Sentiment Analysis in Twitter. In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). Association for Computational Linguistics, pp. 1–18, 2016.

[OKA10] O'Connor, Brendan; Krieger, Michel; Ahn, David: TweetMotif: Exploratory Search and Topic Summarization for Twitter. In: Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010. The AAAI Press, pp. 384–385, 2010.

[Pl80] Plutchik, Robert: A general psychoevolutionary theory of emotion. Emotion: Theory, Research, and Experience, 1:3–31, 1980.

[PLV02] Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar: Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10. EMNLP '02. Association for Computational Linguistics, pp. 79–86, 2002.

[PSM14] Pennington, Jeffrey; Socher, Richard; Manning, Christopher D.: GloVe: Global Vectors for Word Representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543, 2014.

[Ra13] Rangel, Francisco; Rosso, Paolo; Koppel, Moshe; Stamatatos, Efstathios; Inches, Giacomo: Overview of the Author Profiling Task at PAN 2013. In: Working Notes for CLEF 2013 Conference. CLEF, 2013.

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 385

Vergleich und Evaluation von RDF-on-Hadoop-Lösungen

Wolfgang Amann[1]

Abstract: Mit der steigenden Anzahl von Daten, welche in Form des Resource Description Framework (RDF) veröffentlicht werden entsteht eine Menge von Daten, bei der Datenoperationen nicht mehr von einem einzelnen Rechner zu bewältigen sind. In dieser Arbeit werden Systeme vorgestellt, welche zur Lösung dieses Problems das Hadoop-Framework ausschließlich bzw. in Kombination mit anderen Big-Data-Frameworks nutzen. Danach werden mit PigSPARQL und Rya zwei dieser Ansätze, welche exemplarisch für die neuere Entwicklung dieser RDF-on-Hadoop-Systeme stehen, anhand der Benchmark-Queries der Waterloo SPARQL Diversity Test Suite auf spezifische Stärken und Schwächen analysiert.

Keywords: Semantic Web, Resource Description Framework, SPARQL, Big Data, Benchmarking.

1 Einleitung

Seit den 1990/2000er Jahren schreitet die Digitalisierung der Gesellschaft mit großen Schritten voran – durch die stetige Digitalisierung von Informations- und Kommunikationsprozessen wurde die Wissens- und Informationsproduktion immer weiter beschleunigt, was einen starken Anstieg der Daten, die elektronisch gespeichert werden, zur Folge hat. Viele dieser Informationen werden ohne formale Struktur im Internet angeboten, sodass sie von Maschinen nur schwer zu interpretieren sind. 2001 stellte Tim Berners-Lee als Lösung seine Idee des *Semantic Web* vor, wodurch Computern mithilfe des einheitlichen Datenmodells *Resource Description Framework (RDF)* das einfache automatische Ableiten von Wissen aus dem Internet ermöglicht werden soll. Die Idee des Semantic Web wird zunehmend durch ein breiteres Publikum angenommen, wodurch eine Menge an Daten entsteht, bei der Datenoperationen nicht mehr von einem einzelnen Rechner zu bewältigen sind. Eine Lösung dieses Problems bieten die in den letzten Jahren vermehrt entstanden Ansätze, welche mithilfe des Big-Data-Frameworks Hadoop die Arbeit auf skalierbare Rechnernetze auslagern. Ziel dieser Arbeit ist es, einen Überblick über diese bestehenden Ansätze hinsichtlich ihrer Eigenschaften zu schaffen und zwei Systeme zu evaluieren, welche die neueren Entwicklungen der RDF-on-Hadoop-Systeme widerspiegeln.

2 Funktionaler Vergleich von RDF-on-Hadoop-Lösungen

Da die verteilte Speicherung und Verarbeitung von RDF-Daten ein noch relativ junges Anliegen ist, gibt es anders als bei zentralisierten RDF-Stores keine etablierten Systeme

[1] Universität Leipzig, Big Data Kompetenzzentrum, Ritterstraße 9-13, 04109 Leipzig, amann@studserv.uni-leipzig.de

wie Apache Jena, Sesame oder Virtuoso. Vielmehr gibt es eine Vielzahl von forschungsorientierten Machbarkeitsstudien, die entweder neue Lösungen vorstellen oder versuchen, durch Kombination verschiedener bestehenden Ansätzen Laufzeiten zu optimieren. Analog zu [GFM14] wird eine Klassifizierung in *native* und *hybride* Systeme vorgeschlagen:

Native Lösungen verwenden ausschließlich MapReduce gemeinsam mit HDFS bzw. HBase zur Bearbeitung von SPARQL-Anfragen. Dazu wird eine Serie von MapReduce-Jobs aufgerufen – deswegen bemühen sich Ansätze dieser Kategorie vor allem darum, physische Optimierungen vorzunehmen und Strategien zu entwickeln, um die Anzahl der benötigten MapReduce-Jobs zu verringern. Die Speicherung geschieht dabei exklusiv im HDFS, teilweise auch mithilfe von NoSQL-Datenbanken, die auf dem Hadoop-Dateisystem aufbauen, um Indexierungsstrategien auszunutzen.

Hybride Lösungen benutzen im Gegensatz dazu eine Kombination aus Hadoop mit anderen Systemen. Die hybriden Lösungen wurden in drei Unterkategorien unterteilt: (1) Ansätze, die auf den Cluster-Knoten zentralisierte RDF-Stores installieren. Dadurch können Teile von SPARQL-Queries direkt auf den jeweiligen Knoten berechnet werden, auf die der RDF-Graph aufgeteilt wird. Dies macht MapReduce-Jobs nur noch notwendig, um die Ergebnisse der einzelnen Knoten miteinander zu vereinigen. (2) Systeme, welche SPARQL-Anfragen zur Bearbeitung in andere Anfragesprachen wie Pig, Impala oder HiveQL übersetzen. Die zugehörigen Publikationen behandeln vor allem die korrekte und effiziente algebraische Übersetzung von SPARQL in andere Query-Sprachen. (3) Ansätze, die zentralisierte Query-Engines basierend auf der verteilten Speicherung im HDFS nutzen. (4) Ein Ansatz, der im Gegensatz zu allen anderen Ansätzen graph-parallele Berechnungen implementiert, wodurch mehr Rücksicht auf die Struktur von RDF-Daten genommen wird.

2.1 Native Ansätze

2.1.1 HDFS-basierte Systeme

SHARD [RS10] repräsentiert die einfachste Herangehensweise an die Verarbeitung von RDF-Daten mithilfe von Hadoop: RDF-Tripel werden zeilenorientiert direkt im HDFS gespeichert. Für die Anfrageverarbeitung wird MapReduce und der „Klausel-Iterations"-Ansatz genutzt, d. h. für jedes Tripelmuster (Klausel) einer Anfrage wird ein eigener MapReduce-Job initiiert, was zu einer hohen Anzahl sequenziell auszuführender Berechnungen führt, v. a. durch die fehlende Optimierung der Anfragen.

HadoopRDF [Hu09] will diese vielen MapReduce-Jobs vermeiden, indem der optimale physische Auswertungsplan für eine SPARQL-Anfrage durch einen Greedy-Algorithmus ausgewählt wird, welcher aufgrund der Gesamtanzahl der Variablen einer Query deren Join-Selektivitäten abschätzt, um die Anzahl der nötigen MapReduce-Jobs zu verringern. Letztendlich werden Joins in der reduce-Phase bearbeitet, was einen hohen Datenverkehr in der shuffle-Phase bedeutet.

Diesen hohen Kosten für Joins widmen sich die Systeme **[MYL10]**, **MapMerge** [Pr13] und **[LYG13]**. Bei [MYL10] geschieht dies über die Implementierung eines Mehrwege-

Joins, über den mehrere Joins zusammengelegt werden und so die MapReduce-Jobs pro Iteration verringert werden. MapMerge ist eine Adaption von Sort-Merge-Joins für die Verarbeitung von RDF im HDFS. Die Berechnung der Joins wird komplett in der map-Phase vollzogen während die reduce-Phase lediglich für die Nachbearbeitung der Join-Ergebnisse für folgende Iterationen verwendet wird. Dieses Vorgehen hat eine Reduzierung des Datenaufkommens in der shuffle-Phase auf Kosten einer verlängerten Partitionierungs- und Indexierungsphase beim Einlesen der Datensätze zur Folge. [LYG13] schlägt einerseits den „Multiple-Join-With-Filter“ vor, der den SQL-Join ersetzten soll, andererseits versucht die „Select-Join“-Primitive, Projektionen nicht als eigenständigen MapReduce-Job auszuführen, sondern in die Ausführung eines Joins zu integrieren.

[NLH08] und **CliqueSquare** [Go13] wollen das Vorgehen von SHARD über verbesserte Datenspeicherung lösen. [NLH08] erweitert dazu eine RDF-Speicherungstechnik namens „RDF molecules“, welche im Umfang zwischen Tripel und Graph einen Graphen in „Moleküle“ aufteilt, von denen jedes jeweils ein verbundener Subgraph des Originals ist. CliqueSquare nutzt die dreifache Replikation innerhalb des HDFS aus, um jedes RDF-Tripel in drei verschiedenen Formen partitioniert und gruppiert nach Subjekt, Objekt und Prädikat zu speichern. Diese Partitionen sind so gewählt, dass zur Bearbeitung der Queries durch den eingeführten clique-basierten Algorithmus ein möglichst kleiner Datenaustausch in den shuffle-Phasen der MapReduce-Jobs enstehen soll und somit „partitionierte“ Joins ausgeführt werden können, d. h. Joins, die in der map-Phase evaluiert werden.

2.1.2 NoSQL-basierte Systeme

Bei der Nutzung von NoSQL-Datenbanken zur Speicherung von RDF-Daten nutzen die jeweiligen Systeme spezifische Eigenschaften der Stores für eine beschleunigte Anfragebearbeitung aus. Die frühen Ansätze **[SJ10]** und **[Fr11]** stützen sich hierzu auf die Idee, mehrere Indextabellen mit möglichen Tripelpermutationen zu nutzen. Die Tripel selbst werden direkt im Zeilenschlüssel von HBase gespeichert. Da die Anfragebearbeitung der Systeme einfach gehalten ist, resultieren aus SPARQL-Queries zahlreiche sequentielle MapReduce-Jobs.

MAPSIN *(map-side index nested loop-join)* [Sc12] will dies durch die Join-Verarbeitung komplett in der map-Phase (ähnlich wie bei MapMerge) umgehen. Der Vorteil der Nutzung von HBase ist, dass mehrere MapReduce-Jobs aneinandergereiht werden können, ohne dass die Ausgabe für den jeweils nächsten Job sortiert werden muss. Hierdurch und durch das Zusammenlegen von Joins verschiedener Tripelmuster (v. a. bei sternförmigen Anfragen) wird überflüssiger Datenfluss über das Netzwerk vermieden. **RDFChain** [CJL13] baut direkt auf MAPSIN auf und will durch eine Kombination von drei Tabellen die namensgebenden linearen Basic Graph Patterns schnell bearbeiten. Tripelterme, die nicht nur als Subjekt, sondern auch als Objekt erscheinen, werden in einer gesonderten Tabelle abgelegt, alle anderen Tripel, die dort nicht vorkommen in den von anderen Ansätzen bekannten Subjekt-Prädikat-Objekt- und Objekt-Prädikat-Subjekt-Tabellen.

H_2RDF+ [Pa13] will die Anfragegeschwindigkeit über Heuristiken beschleunigen: Zusätzlich zu den Indextabellen legen sie aggregierte Indexstatistiken in HBase an, mithilfe deren Joinkosten und Selektivität von Tripelmustern abgeschätzt werden sollen. Auf Grundlage dieser Schätzungen wird entschieden, ob Anfragen zentralisiert auf nur einem Clusterknoten berechnet oder über MapReduce an alle Knoten verteilt werden.

Einen anderen Weg schlägt **SPIDER** *(Scalable, Parallel/Distributed Evaluation of large-scale RDF data)* [Ch09] ein. Es sortiert RDF-Tripel nach ihrer URI und teilt sie in Subgraphen auf die verschiedenen Cluster-Server auf. Queries werden nach den so generierten URI-Schlüsseln in Subqueries aufgeteilt und an die Server gesendet, welche die entsprechenden Subgraphen speichern. Da jeder Knoten nur einen Teil der RDF-Daten speichert, muss SPIDER zur Ergebnisbildung die Subquery-Ergebnisse jedes Knotens per MapReduce vereinen, was viel Rechenzeit und Netzwerkkommunikation benötigt.

2.2 Hybride Ansätze

2.2.1 Ansätze mit zentralisierten Datenbanken

[Du12] greift die Idee von SPIDER auf und will die Vorteile „traditioneller“ Triple Stores mit MapReduce verbinden. Die Datensätze werden hierzu so partitioniert, dass Tripel mit dem selben Prädikat auf dem selben Knoten, auf denen jeweils ein OpenRDF-Seame-Triplestore läuft, verteilt werden können. Zur Auffindung der Tripel wird dazu eine Hash-Map angelegt. Gestellte SPARQL-Anfragen werden ebenfalls nach dieser Hash-Map partitioniert und an die verschiedenen Knoten gesendet. Die so entstehenden Einzelergebnisse werden anschließend über MapReduce-Jobs vereinigt. Auch **GraphPartition** [HAR11] und **[LL13]** versuchen, die Netzwerkkommunikation zwischen den Knoten eines Clusters zu minimieren, indem RDF-Graphen in Subgraphen unterteilt werden, von denen jeder auf einem eigenen Knoten gespeichert wird, auf dem jeweils RDF-3X läuft. Je nach System werden die Daten unterschiedlich partitioniert – einerseits basierend auf der Entfernung im Originalgraphen, andererseits über Subjekt-Objekt-Kombinationen. Bei der Queryausführung wird dann zunächst entschieden, ob eine Anfrage innerhalb einer Partition und damit ohne MapReduce ausgeführt werden kann, oder ob mehrere Partitionen und MapReduce genutzt werden müssen.

2.2.2 Ansätze basierend auf Query-Übersetzungen

RAPID+ [RKA11] fokussiert sich wie viele native Ansätze durch die Interpretation sternförmiger Joins als Tripelgruppen auf die Reduzierung von MapReduce-Zyklen bei der Join-Verarbeitung. Dazu baut das System die Sprache von Apache Pig um eine zwischenstufliche Algebra *(Nested TripleGroup Algebra)* aus, um die Tripel effizienter bearbeiten zu können. Die Daten werden direkt aus dem HDFS gelesen.

[Ko12] und **PigSPARQL** [Sc13] übersetzen SPARQL-Anfragen nach Pig Latin, der Sprache von Apache Pig. Pig übersetzt diese Anfragen wiederum in MapReduce Jobs, welche

die Daten, die direkt im HDFS gespeichert werden oder bei PigSPARQL optional vertikal partitioniert werden, anfragen. Die Systeme können hierdurch stark von der Weiterentwicklung von Pig profitieren, ohne Optimierungen am eigenen System vornehmen zu müssen. **Hive-HBase-RDF** [Ha13], **Sempala** [Sc14] und **S2RDF** *(SPARQL on Spark for RDF)* [Sc15b] gehen einen ähnlichen Weg. Die Systeme basieren auf der Übersetzung von SPARQL nach SQL (bzw. HiveQL) und dessen anschließender Bearbeitung mit Hive, Impala oder der SQL-API von Spark. Während das erste System Tripel in HBase speichert, nutzen die beiden letzteren Systeme das spaltenorientierte HDFS-Format Parquet. Hive-HBase-RDF und Sempala können durch die Nutzung von Property Tables vor allem sternförmige SPARQL-Anfragen ohne Joins bearbeiten, die Besonderheit an S2RDF ist die neu vorgeschlagene Indexierungsstrategie namens *Extended Vertical Partitioning* (ExtVP).

2.2.3 Ansätze mit zentralisierter Query-Bearbeitung

Rya *(RDF y Accumulo)* [PCR12] und **Jena-HBase** [Kh12] speichern RDF-Tripel mithilfe von Accumulo bzw. HBase und kombinieren dies mit den zentralisierten Query-Engines von OpenRDF Sesame/Apache Jena. Im Gegensatz zu Jena-HBase nutzt Rya die lexikographische Sortierung der Zeilenschlüssel in Accumulo, wodurch einzelne Tripelmuster durch einen Scan von nur einer der drei Indextabellen beantwortet werden können. Anfragen werden in Rya durch einen geschachtelten Loop-Join zentralisiert auf einem Server bearbeitet, was besonders bei nicht-selektiven Queries einen potentiell limitierenden Faktor dieses Ansatzes hinsichtlich der Skalierung von Datensätzen darstellt.

2.2.4 Ansätze mit graph-paralleler Berechnung

S2X *(SPARQL on Spark with GraphX)* [Sc15a] nutzt einen anderen Ansatz als alle anderen vorgestellten Systeme. Mithilfe der Apache-Spark-API „GraphX“, welche Werkzeuge für Graphoperationen bereitstellt, werden graph-parallele Berechnungen auf RDF-Daten ausgeführt. Diese werden dafür auf die „Property Graph“-Struktur von GraphX abgebildet, welcher im HDFS persistiert werden kann.

3 Evaluation

Die Evaluation wurde auf einem Cluster von 16 Dell PowerEdge R320 Servern durchgeführt (ein Namenode, ein Zookeeper und 14 Worker), die jeweils mit einem Intel Xeon E5-2430 v2 (2,5GHz mit 6 Cores) und 48GB RAM ausgestattet und durch 1GBit/s Ethernet mit dem Netzwerk verbunden waren. Auf den einzelnen Rechnern war jeweils openSUSE 13.2 mit Hadoop 2.6.0, Pig 0.15, Accumulo 1.7.0 und Tomcat 7 installiert.

Es wurde exemplarisch die Leistung von hybriden Ansätzen getestet, da diese die neueren Entwicklungen in den RDF-on-Hadoop-Lösungen widerspiegeln. Die Wahl fiel auf die beiden Systeme PigSPARQL (basierend auf Query-Übersetzungen) und Rya (zentralisierte

Skalierungsfaktor	25000	50000	100000
urspr. Dateigröße	142,6 GB	287,2 GB	560,5 GB
Datenbankgröße PigSPARQL (VP)	238,4 GB	480,9 GB	939,4 GB
Datenbankgröße Rya	42,3 GB	84,1 GB	166,3 GB
Ladezeit PigSPARQL (VP)	01:17:52	02:46:27	05:56:59
Ladezeit Rya	04:16:02	07:52:23	15:13:44

Tab. 1: Ladezeiten für die Datensätze des WatDiv-Benchmarks.

Queryausführung) – auf weitere Systeme wurde verzichtet, da die Quelltexte aller Ansätze mit zentralisierten Datenbanken nicht öffentlich verfügbar sind und sich S2X im Laufe der Arbeit aufgrund großem Informationsoverhead bei der Queryausführung als noch nicht geeignet für den Big-Data-Anwendungsfall herausgestellt hat. Zur Evaluation wurde die *Waterloo SPARQL Diversity Test Suite* (WatDiv) [Al14] genutzt. Aus den 17 mitgelieferten Vorlagen für lineare, sternförmige und schneeflockenförmige Anfragen wurden jeweils fünf konkrete Anfragen erstellt, wodurch sich 85 verschiedene Queries pro Datensatz ergaben – jede davon wurde einmal auf den beiden Systemen ausgeführt und aus den Laufzeitergebnissen der Median gebildet. Beim Generieren der Daten wurden sich verdoppelnde Skalierungsfaktoren von 25000, 50000 und 100000 verwendet, was in Datensätzen von ca. 2,6 bis 10,5 Milliarden Tripeln resultierte. In Tabelle 1 sind die Dateigrößen der ursprünglichen Datensätze und die sich daraus ergebenen Ladezeiten und Datenbankgrößen dargestellt. Durch die vertikale Partitionierung muss PigSPARQL einige Daten redundant abspeichern und verbraucht so mehr Speicherplatz als der ursprüngliche Datensatz. Prinizpiell doppelt auch Rya durch seinen Three Tables Index viele Tripel – durch die in Accumulo standardmäßig integrierte gzip-Kompression wird dies aber ausgeglichen und dadurch lediglich 30% der Ausgangsgröße verbraucht. Dies kommt aber zum Preis einer sehr viel längeren Vorverarbeitungszeit: Schon beim kleinsten Datensatz braucht Rya dreimal so lang zum Laden der Daten wie PigSPARQL.

3.1 Vergleich der Laufzeiten

L1–L5 Bei PigSPARQL spiegelt sich der niedrige Selektivitätsgrad der fünf linearen Anfragen direkt in den Laufzeiten nieder: Sie liegen konstant sehr niedrig, selbst beim größten Datensatz war die rechenaufwändigste Query L1 in 2:40 Minuten zu beantworten (siehe Abbildung 1). Bei den Tests mit Rya war eine deutlich langsamere Anfragezeit zu beobachten – L3 brauchte beim kleinsten Datensatz 12 Stunden und 20 Minuten (im Vergleich zu 59 Sekunden bei PigSPARQL), mit Skalierungsfaktor 50000 lag die Bearbeitungszeit bereits über der gesetzten Maximalzeit von 24 Stunden. Der Grund hierfür ist die fehlende algebraische Optimierung der SPARQL-Anfragen durch Rya. Diese wird nur durchgeführt, wenn ein zeit- und speicherintensiver Prozess namens *Prospector* aufgerufen wird, welcher die Selektivität von Tripelmustern eines Datensatzes abzählt. Die Optimierung wäre auch ohne diesen Prozess möglich, was besonders bei Anfrage L3 gut zu beobachten ist: PigSPARQL implementiert zur Optimierung u. a. eine sehr einfache Strategie zur

Laufzeit namens *Variable-Counting*-Heuristik, welche Tripelmuster mit weniger ungebundenen Bestandteilen in einer Anfrage möglichst weit nach oben verschiebt, um früh Zwischenergebnisse zu verkleinern. Bei Eingabe dieser durch PigSPARQL optimierten Tripelanordnung in Rya kann man eine starke Verringerung der Laufzeit beobachten: Mit 0,4 Sekunden Bearbeitungszeit beim größten Datensatz liegt Rya plötzlich deutlich vor PigSPARQL.

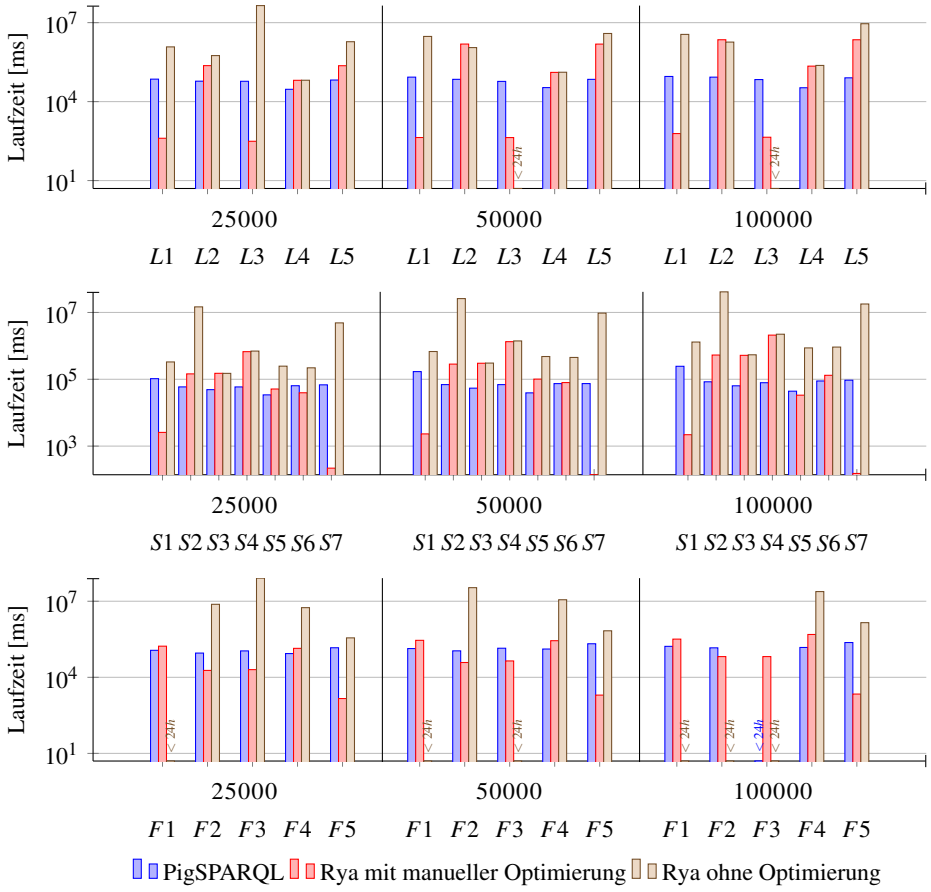


Abb. 1: Laufzeiten der getesteten WatDiv-Anfragen mit Skalierung.

F1–F5 Durch die Kombination von linearen mit sternförmigen Anfragen kommt Rya bei den schneeflockenförmigen Queries an seine Grenzen: Bereits beim kleinsten Datensatz benötigt das System mehr als 24 Stunden zur Beantwortung von F1. Hier macht sich die Vermeidung von Kreuzprodukten durch die Optimierung von PigSPARQL gemeinsam mit der Mehrwege-Join-Unterstützung von Pig stark bemerkbar (zwei Joins im Gegensatz zu ursprünglich fünf). F4 gestaltet sich ähnlich: PigSPARQL benötigt zur Bearbeitung des schneeflockenförmigen Musters zwei im Vergleich zu acht Joins bei Rya. Durch eine manuelle Neuordnung der Anfragen im Stile der Variable-Counting-Heuristik kann man die Laufzeiten der Anfragen durch Vermeidung von Kreuzprodukten nah an das Niveau

von PigSPARQL heran bringen, welches aber nicht ganz erreicht werden kann. F2, F3 und F5 werden von Rya ebenfalls langsamer bearbeitet als von PigSPARQL. Die drei Anfragen können im Gegensatz zu F1 und F2 aber so neu geordnet werden, sodass Rya schnellere Laufzeiten erreichen kann als PigSPARQL. Obwohl PigSPARQL hier wieder sieben Joins zu lediglich zwei Joins zusammenfassen kann, reicht die Neuordnung aus, dass Rya bei allen Datensatzgrößen schneller als PigSPARQL rechnet. Bei F3 und F5 erhält man durch ähnliche Verschiebungen ebenfalls bessere Laufzeiten als PigSPARQL.

S1–S7 Ohne manuelle Optimierung der Anfragen generiert Rya bei den sternförmigen Anfragen viel längere Laufzeiten als PigSPARQL. Vor allem bei S1 und S7 können durch aufsteigende Sortierung der Tripelmuster nach ungebundenen Elementen und durch Vermeidung von Kreuzprodukten ähnlich wie bei L1 und L3 stark beschleunigte Ergebnisse von Rya erreicht werden. Mit diesen kleinen Zwischenergebnissen kann Rya dank seines Three Tables Index die Ergebnisse in nahezu konstant bleibender Zeit berechnen. Bei den restlichen Queries ist PigSPARQL durch seine Integration eines Mehrwege-Joins eindeutig überlegen. Bei S2, S3 (ohne weitere statistische Informationen nicht optimierbar), S5 und S6 benötigt das System dadurch lediglich einen Join im Gegensatz zu jeweils einem Join pro Tripelmuster bei Rya und kann dadurch viel schnellere Laufzeiten erzielen.

4 Fazit

Mit der steigenden Beliebtheit von RDF geht auch ein größeres Interesse der akademischen Forschung einher. Während 2008/09 noch sehr wenige Paper zu RDF-on-Hadoop-Systemen veröffentlicht wurden, nimmt die Rate seit sechs Jahren zu. Bei der Evaluation der zwei exemplarisch ausgewählten Systeme PigSPARQL und Rya erlangte ersteres durch seine effiziente physische und logische Optimierung durchweg sehr gute Laufzeiten bei WatDiv, besonders bei komplexeren Queries, welche große Zwischenergebnisse mit sich bringen. Sobald Queries jedoch sehr selektiv sind, ist PigSPARQL sehr viel langsamer als der Konkurrent Rya – denn der größte geschwindigkeitshemmendere Faktor ist nicht die zentralisierte Anfrageauswertung, sondern die sehr rudimentäre SPARQL-Optimierung durch Rya. Das System verlässt sich vollkommen auf den sehr zeit- und speicherintensiven Prozess namens Prospector. Während dies zwar zur präzisesten Neuordnung der Tripelmuster einer SPARQL-Anfrage führt, reichen bereits sehr einfach zur Laufzeit ausführbare Strategien wie die Variable-Count-Heuristik aus, um sehr viel schnellere Ergebnisse zu erzielen. Daneben ist die Prospector-Optimierung nicht ausreichend zur vollständigen Optimierung von Anfragen – grundlegende Maßnahmen wie Filter-Substitutionen werden komplett ignoriert. Wie die Entwickler von Rya angekündigt haben, ist geplant, einen MapReduce-Job zur Bearbeitung von Queries zu implementieren – hier würde es sich anbieten, ähnlich wie H_2RDF+ auf Grundlage der durch den Prospector evaluierten Selektivitäten abzuschätzen, ob eine Anfrage schneller zentralisiert oder über den Cluster verteilt beantwortet werden kann. Als abschließende Erkenntnis dieser Arbeit bleibt die beeindruckende Fülle an angebotenen Kombinationen der RDF-on-Hadoop-Lösungen. Fast jede mögliche Kombination von Zentralisierung und Verteilung der Datenspeicherung und Anfrageausführung wurde bereits behandelt. Nach zwölf Jahren RDF sind die RDF-on-Hadoop-Anwendungen aber immer noch weit davon entfernt, für produktive Anwendungen geeignet zu sein.

5 Danksagung

Die vorliegende Arbeit wurde betreut durch Dr. Eric Peukert und gefördert durch das Bundesministerium für Bildung und Forschung innerhalb des *Competence Center for Scalable Data Services and Solutions* (ScaDS) Dresden/Leipzig (BMBF 01IS14014B).

Literatur

[Al14] Aluç, G.; Hartig, O.; Özsu, T.; Daudjee, K.: Diversified Stress Testing of RDF Data Management Systems. In: Proceedings of the 13th ISWC. Springer, Cham, S. 197–212, 2014.

[Ch09] Choi, H.; Son, J.; Cho, Y.; Sung, M. K.; Chung, Y. D.: SPIDER: A System for Scalable, Parallel / Distributed Evaluation of large-scale RDF Data. In: Proceedings of the 18th CIKM. ACM, New York, 2009.

[CJL13] Choi, P.; Jung, J.; Lee, K.-H.: RDFChain: Chain Centric Storage for Scalable Join Processing of RDF Graphs using MapReduce and HBase. In: Proceedings of the 2013 ISWC - P&D Track. 2013.

[Du12] Du, J.-H.; Wang, H.-F.; Ni, Y.; Yu, Y.: HadoopRDF: A Scalable Semantic Data Analytical Engine. In: Intelligent Computing Theories and Applications. Berlin: Springer, 2012.

[Fr11] Franke, C.; Morin, S.; Chebotko, A.; Abraham, J.; Brazier, P.: Distributed semantic web data management in HBase and MySQL cluster. In: Proceedings of the 2011 CloudCom. S. 105–112, 2011.

[GFM14] Giménez-García, J. M.; Fernández, J. D.; Martínez-Prieto, M. A.: MapReduce-based Solutions for Scalable SPARQL Querying. Open Journal of Semantic Web 1/, 2014.

[Go13] Goasdoué, F.; Kaoudi, Z.; Manolescu, I.; Quiané-Ruiz, J.; Zampetakis, S.: CliqueSquare: efficient Hadoop-based RDF query processing. In: Proceedings of the 2013 BDA. Nantes, 2013.

[Ha13] Haque, A.: A MapReduce Approach to NoSQL RDF Databases, Bachelorarbeit, University of Texas, 2013.

[HAR11] Huang, J.; Abadi, D. J.; Ren, K.: Scalable SPARQL Querying of Large RDF Graphs. Proceedings of the VLDB Endowment 4/11, S. 1123–1134, 2011.

[Hu09] Husain, M. F.; Doshi, P.; Khan, L.; Thuraisingham, B.: Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce. In: Cloud Computing. Springer, Berlin, S. 680–689, 2009.

[Kh12] Khadilkar, V.; Kantarcioglu, M.; Thuraisingham, B.; Castagna, P.: Jena-HBase: A Distributed, Scalable and Efficient RDF Triple Store. In: ISWC-PD 2012. CEUR-WS.org, Aachen, S. 85–88, 2012.

[Ko12] Kotoulas, S.; Urbani, J.; Boncz, P.; Mika, P.: Robust Runtime Optimization and Skew-resistant Execution of Analytical SPARQL Queries on Pig. In: Proceedings of the 11th ISWC. Springer, Berlin, S. 247–262, 2012.

[LL13] Lee, K.; Liu, L.: Scaling Queries over Big RDF Graphs with Semantic Hash Partitioning. Proceedings of the VLDB Endowment 6/14, S. 1894–1905, 2013.

[LYG13] Liu, L.; Yin, J.; Gao, L.: Efficient Social Network Data Query Processing on MapReduce. In: Proceedings of the 5th ACM Workshop on HotPlanet. ACM, New York, S. 27–32, 2013.

[MYL10] Myung, J.; Yeon, J.; Lee, S.: SPARQL Basic Graph Pattern Processing with Iterative MapReduce. In: Proceedings of the 2010 MDAC. ACM, New York, 6:1–6:6, 2010.

[NLH08] Newman, A.; Li, Y.-F.; Hunter, J.: Scalable Semantics - The Silver Lining of Cloud Computing. In: Proceedings of the 4th eScience. S. 111–118, 2008.

[Pa13] Papailiou, N.; Konstantinou, I.; Tsoumakos, D.; Karras, P.; Koziris, N.: H_2RDF+: High-performance distributed joins over large-scale RDF graphs. In: Proceedings of the 2013 IEEE Big Data Conference. S. 255–263, 2013.

[PCR12] Punnoose, R.; Crainiceanu, A.; Rapp, D.: Rya: A Scalable RDF Triple Store for the Clouds. In: Proceedings of the 1st Cloud-I. ACM, New York, 4:1–4:8, 2012.

[Pr13] Przyjaciel-Zablocki, M.; Schätzle, A.; Skaley, E.; Hornung, T.; Lausen, G.: Map-Side Merge Joins for Scalable SPARQL BGP Processing. In: Proceedings of the 5th CloudCom. 2013.

[RKA11] Ravindra, P.; Kim, H.; Anyanwu, K.: An Intermediate Algebra for Optimizing RDF Graph Pattern Matching on MapReduce. In: Proceedings of the 8th ESWC 2011. S. 46–61, 2011.

[RS10] Rohloff, K.; Schantz, R. E.: High-performance, Massively Scalable Distributed Systems Using the MapReduce Software Framework: The SHARD Triple-store. In: PSI EtA 2010. ACM, New York, 4:1–4:5, 2010.

[Sc12] Schätzle, A.; Przyjaciel-Zablocki, M.; Dorner, C.; Hornung, T.; Lausen, G.: Cascading Map-Side Joins over HBase for Scalable Join Processing, 2012, URL: https://arxiv.org/pdf/1206.6293v1, Stand: 12. 12. 2016.

[Sc13] Schätzle, A.; Przyjaciel-Zablocki, M.; Hornung, T.; Lausen, G.: PigSPARQL: A SPARQL Query Processing Baseline for Big Data. In: Proceedings of the 2013 ISWC. CEUR-WS.org, Aachen, S. 241–244, 2013.

[Sc14] Schätzle, A.; Przyjaciel-Zablocki, M.; Neu, A.; Lausen, G.: Sempala: Interactive SPARQL Query Processing on Hadoop. In: Proceedings of the 13th ISWC. Springer, New York, S. 164–179, 2014.

[Sc15a] Schätzle, A.; Przyjaciel-Zablocki, M.; Berberich, T.; Lausen, G.: S2X: Graph-Parallel Querying of RDF with GraphX. In: Big-O(Q) 2015. 2015.

[Sc15b] Schätzle, A.; Przyjaciel-Zablocki, M.; Skilevic, S.; Lausen, G.: S2RDF: RDF Querying with SPARQL on Spark, 2015, URL: http://arxiv.org/abs/1512.07021, Stand: 12. 12. 2016.

[SJ10] Sun, J.; Jin, Q.: Scalable RDF store based on HBase and MapReduce. In: Proceedings of the 3rd ICACTE. S. 633–636, 2010.

Tutorienprogramm

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 397

Multimedia Similarity Search

Thomas Seidl[1]

Das Tutorial zeigt aktuelle Methoden der Ähnlichkeitssuche in Multimedia-Datenbanken auf. Es bezieht sich auf verschiedene Datentypen wie Farbbilder und Videos sowie Zeitreihen, Trajektorien und geometrische Formen. Der technische Bogen spannt sich von Objektrepräsentationen über Ähnlichkeitsmodelle und effizienten Algorithmen zur Ähnlichkeitssuche bis hin zum Data Mining mit Aufgaben von Clustering und Klassifikation.

Die behandelten Ähnlichkeitsmodelle erstrecken sich auf unmittelbare Objektdarstellungen wie beispielsweise bei kernbasierten Methoden sowie auch auf Einbettungen von komplexen Objekte in Vektorräume. Die Algorithmen zur Ähnlichkeitssuche beziehen Datenbanktechniken wie mehrstufige Anfragebearbeitung, Approximationen und Indexstrukturen ein. Im Bereich der Data-Mining-Methoden werden Techniken behandelt, die auf der Verwendung der vorher genannten Ähnlichkeitsmodellierung beruhen.

Thomas Seidl ist Professor für Informatik und leitet den Lehrstuhl für Datenbanksysteme an der Ludwig-Maximilians-Universität München. Er studierte Informatik an der TU München und schloss seine Promotion 1997 und seine Habilitation 2001 an der LMU ab. Von 2002 bis 2016 führte er den Lehrstuhl für Datenmanagement und –exploration an der RWTH Aachen. Seine Grundlagenforschung in Data Mining und Datenbanktechnologien mit Anwendungen in Ingenieur-, Wirtschafts-, Lebens- und Geisteswissenschaften führte bislang zu mehr als 250 wissenschaftlichen Publikationen. Er ist Mitglied in wissenschaftlichen Beiräten und vielen Programmkomitees sowie in den Leitungsgremien des Data Science Labs und des neuen Elitestudiengangs Data Science an der LMU.

[1] LMU München, Lehrstuhl für Datenbanksysteme, seidl@dbs.ifi.lmu.de

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 399

Scalable Data Management: An In-Depth Tutorial on NoSQL Data Stores

Felix Gessert[1] Wolfram Wingerath[2] Norbert Ritter[3]

Abstract: The unprecedented scale at which data is consumed and generated today has shown a large demand for scalable data management and given rise to non-relational, distributed "NoSQL" database systems. Two central problems triggered this process: 1) vast amounts of user-generated content in modern applications and the resulting request loads and data volumes as well as 2) the desire of the developer community to employ problem-specific data models for storage and querying. To address these needs, various data stores have been developed by both industry and research, arguing that the era of one-size-fits-all database systems is over. The heterogeneity and sheer amount of these systems – now commonly referred to as NoSQL data stores – make it increasingly difficult to select the most appropriate system for a given application. Therefore, these systems are frequently combined in polyglot persistence architectures to leverage each system in its respective sweet spot. This tutorial gives an in-depth survey of the most relevant NoSQL databases to provide comparative classification and highlight open challenges. To this end, we analyze the approach of each system to derive its scalability, availability, consistency, data modeling and querying characteristics. We present how each system's design is governed by a central set of trade-offs over irreconcilable system properties. We then cover recent research results in distributed data management to illustrate that some shortcomings of NoSQL systems could already be solved in practice, whereas other NoSQL data management problems pose interesting and unsolved research challenges. In addition to earlier tutorials, we explicitly address how the quickly emerging topic of processing and storing massive amounts of data in real-time can be solved by different types real-time data management systems.

Keywords: NoSQL, Scalability, Distributed Systems, High Availability, Polyglot Persistence, Real-Time Processing, Cloud Data Management

1 Introduction

Traditional relational database management systems (RDBMSs) provide powerful mechanisms to store and query structured data under strong consistency guarantees and have reached an unmatched level of reliability, stability and support through decades of development. In recent years, however, the amount of useful data in some application areas has become so vast that it simply cannot be stored or processed by traditional database solutions [SF12]. User-driven content in social networks or data retrieved from large sensor networks are only two examples of this phenomenon commonly referred to as Big Data [La01]. A class of novel data storage products able to cope with Big Data are subsumed under the term NoSQL databases, many of which offer horizontal scalability and higher availability than traditional relational databases or other useful properties by sacrificing querying options and consistency guarantees [LS13].

[1] Universität Hamburg, gessert@informatik.uni-hamburg.de
[2] Universität Hamburg, wingerath@informatik.uni-hamburg.de
[3] Universität Hamburg, ritter@informatik.uni-hamburg.de

There are dozens of NoSQL database systems, and it is very hard to keep an overview over what these systems provide, where they fail and where they differ. Beyond a mere presentation of prominent NoSQL representatives and their respective features, this tutorial intends to give an overview over the requirements typically posed to NoSQL database systems, the techniques used to fulfill these requirements and the trade-offs that have to be made in the process.

The main problem NoSQL database systems seek to solve is providing storage and query capabilities for specific problem domains. This encompasses specialization in *data models*, *query languages*, *consistency*, *scalability* and *availability* properties, *transactional guarantees*, *schema management*, *low latency*, *analytical* and *real-time processing* as well as *durability*, *reliability* and *elasticity*. While the foundational techniques are often similar (e.g., asynchronous master-slave replication), the resulting systems exhibit very different behavior in both functional and non-functional aspects. Since very different requirements are often found in different parts of the same application, a recent trend is the consolidation of different database systems within a single application (*polyglot persistence*).

Our tutorial discusses the characteristics of NoSQL databases and introduces approaches proposed to address their challenges. To this end, we provide a classification scheme that helps to choose candidate systems for applications based on a set of data management criteria.

Furthermore, we highlight open problems that provide opportunities for contributing to the emerging area of scalable data management and polyglot persistence.

2 Tutorial Outline

Our tutorial is divided into fours parts and structured as follows.

As background, we recall the basics of distributed data management, in particular partitioning, replication, eventual consistency and the different NoSQL data models. We also present the most important impossibility results for distributed databases, e.g. the widely used CAP theorem.

The core survey of NoSQL databases covers the discussion and classification of the different systems. Each system is described in depth and relations to current research are given. We include open-source and commercial NoSQL systems, research systems as well as cloud-based database-as-a-service systems. We classify each system according to functional and non-functional properties.

The third part discusses the new challenge of real-time data management. To this end, different database systems and real-time processing frameworks are analyzed with respect to their capabilities in storing, querying and analyzing data in real-time.

The final part reviews the integration of the discussed systems in polyglot persistence environments, in particular challenges and potential benefits. A short summary of open challenges concludes the tutorial.

3 Intended Audience

We expect the tutorial to appeal to a large portion of the BTW community:

- Students who are looking for novel research topics and orientation
- Experienced researchers in the fields of database systems, cloud computing and distributed systems interested in open challenges and a comparative overview of the NoSQL landscape
- Industry practitioners tackling data management problems who are looking for a survey and classification of existing systems and their respective sweet spots

The tutorial is aimed at balancing practical aspects (e.g., APIs and deployments models) and research approaches (e.g., novel architectures and transaction protocols).

4 Relationship to Prior Tutorials

This tutorial is an extended version of our tutorial from BTW 2015 [GR15] and ICDE 2016 [GR16]. In addition to updates that reflect progress made in both research and practice, we included the topic of real-time data management to account for the quickly emerging trend of serving data for interactive applications.

5 Presenters

Felix Gessert is a Ph.D. student at the databases and information systems group at the University of Hamburg. His main research fields are scalable database systems, transactions and web technologies for cloud data management. His thesis addresses caching and transaction processing for low latency mobile and web applications. He is also founder and CEO of the startup Baqend that implements these research results in a cloud-based backend-as-a-service platform. Since their product is based on a polyglot, NoSQL-centric storage model, he is very interested in both the research and practical challenges of leveraging and improving these systems. He is frequently giving talks on different NoSQL topics.

Wolfram Wingerath is a Ph.D. student under supervision of Norbert Ritter teaching and researching at the University of Hamburg. He was co-organiser of the BTW2015 conference and has held workshop and conference talks on his published work on several occasions. Wolfram is part of the databases and information systems group and his research interests evolve around scalable NoSQL database systems, cloud computing and Big Data analytics, but he also has a background in data quality and duplicate detection. His current work is related to real-time stream processing and explores the possibilities of providing always-up-to-date materialized views and continuous queries on top of existing non-streaming DBMSs.

Norbert Ritter is a full professor of computer science at the University of Hamburg, where he heads the databases and information systems group. He received his Ph.D. from the University of Kaiserslautern in 1997. His research interests include distributed and federated database systems, transaction processing, caching, cloud data management, information integration and autonomous database systems. He has been teaching NoSQL topics in various database courses for several years. Seeing the many open challenges for NoSQL systems, he and Felix Gessert have been organizing the annual SCDM[4] workshop for three years to promote research in this area.

References

[GR15] Gessert, Felix; Ritter, Norbert: Skalierbare NoSQL- und Cloud-Datenbanken in Forschung und Praxis. In: Datenbanksysteme für Business, Technologie und Web (BTW 2015) - Workshopband, 2.-3. März 2015, Hamburg, Germany. volume 242 of LNI. GI, pp. 271–274, 2015.

[GR16] Gessert, Felix; Ritter, Norbert: Scalable Data Management: NoSQL Data Stores in Research and Practice. In: 32nd IEEE International Conference on Data Engineering, ICDE 2016. 2016.

[La01] Laney, Douglas: 3D Data Management: Controlling Data Volume, Velocity, and Variety. Technical report, META Group, February 2001.

[LS13] Lehner, Wolfgang; Sattler, Kai-Uwe: Web-Scale Data Management for the Cloud. Springer, 2013.

[SF12] Sadalage, Pramod J.; Fowler, Martin: NoSQL distilled: a brief guide to the emerging world of polyglot persistence. Pearson Education, 2012.

[4] Scalable Cloud Data Management Workshop: www.scdm.cloud

Data Science Challenge

B. Mitschang et al. (Hrsg.): BTW 2017 – Workshopband,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2017 405

BTW 2017 Data Science Challenge (SDSC17)

Tim Waizenegger[1]

Welche aufschlussreiche Datenvisualisierung oder Analyse können Sie mit ihrem eigenen Ansatz erzeugen?

Im Rahmen der Data Science Challenge haben Studierende und Doktoranden die Möglichkeit, einen eigenen Ansatz zur Cloud-basierten Datenanalyse zu entwickeln und damit gegen andere Teilnehmer anzutreten. Auf der BTW2017 in Stuttgart präsentieren die Teilnehmer Ihre Ergebnisse die von einer Fachjury aus Forschung und Industrie bewertet werden. Die Gewinner, sowie die Nächstplatzierten, werden mit einem Preisgeld honoriert.

1 Preise

Erster Platz: 500 Euro
Zweiter Platz: 300 Euro
Dritter Platz: 200 Euro

2 Vorgehen

Mit der Ausschreibung wurden Beispieldatenquellen sowie Beispielaufgaben bekanntgegeben. Die Bewerber nutzten diese Datenquellen und Aufgaben um ihr Vorgehen zu planen und sich mit einer zweiseitigen Vorstellung zu bewerben.

Einen Monat vor der BTW 2017 (am 07.02.17) werden die Datenquellen und Aufgabe für die eigentliche Challenge bekanntgegeben. Diese sind ähnlich wie die Beispiele, sodass die Teilnehmer ihre geplanten Ansätze anwenden können.

Das Ziel der Challenge ist es also, in einem Monat den gewählten Ansatz an die neuen Daten anzupassen und die Aufgabe zu lösen. Es ist also vorteilhaft, einen flexiblen, wiederverwendbaren Ansatz zu wählen.

Im Rahmen der Challenge müssen die Datenquellen integriert und ausgewertet werden. Das Ergebnis der Analyse kann eine aufschlussreiche Visualisierung oder Handlungsempfehlung sein. Die Teilnehmer haben freie Auswahl der verwendeten Cloud Plattformen und Technologien. Ihr Ansatz kann verfügbare Dienste und Werkzeuge integrieren oder neue entwickeln. Die Daten werden über die Plattform IBM Bluemix bereitgestellt. Bluemix und die dort verfügbaren Analysedienste steht des Weiteren kostenfrei zur Durchführung des Wettbewerbs zur Verfügung.

[1] Universität Stuttgart, Institut für Parallele und Verteilte Systeme, waizentm@ipvs.uni-stuttgart.de

3 Problemstellung

Die Data Science Challenge umfasst zwei Problemstellungen: Ein Beispiel, sowie die eigentliche Challenge-Aufgabe. Das Beispiel wurde von den Teilnehmern verwendet, um ihr Vorgehen zu entwickeln und zu beschreiben. Es enthielt Daten von New Yorks öffentlichem Bike-Sharing Dienst sowie Kartendaten und Wetterdaten. Die Challenge-Aufgabe wird einen Monat vor der BTW2017 veröffentlicht und ein vergleichbares Szenario und Daten enthalten.

Die Bewerber nutzten unterschiedlichste Technologien wie Apache Spark, Python, R, Graphdatenbanken, Matlab, IBM Watson oder Tableau Online. Die Herangehensweisen reichten von statische Analysen über interaktive Visualisierungen bis hin zu eigenen Web-Anwendungen.

4 Die Challenge auf der BTW2017

Auf die Ausschreibung haben sich sieben Teams aus Deutschland, Österreich, Indien und Ungarn beworben. Alle sieben Bewerber wurden für die Teilnahme zugelassen und werden ihre Herangehensweise am Dienstag den 07.03.17 im Rahmen des Workshopprogramms vorstellen. Die Gewinner werden im Rahmen der Abendveranstaltung am Donnerstag den 09.03.17 bekanntgegeben und haben dort die Gelegenheit ihren Ansatz erneut vor einem größeren Publikum zu präsentieren.

Die Gewinner, sowie ausgewählte Nächstplatzierte, werden nach der BTW2017 eingeladen an einem Sonderbeitrag für das Datenbankspektrum mitzuwirken. Dort werden die Challenge-Aufgabe, sowie die unterschiedlichen Herangehensweisen im Detail vorgestellt.

5 Organisatoren

M. Behrendt, IBM
P. Hirmer, Univ. Stuttgart
M. Mähler, IBM
K.-U. Sattler, TU Ilmenau
T. Waizenegger, Univ. Stuttgart

Autorenverzeichnis

A
Algergawy, Alsayed, 79
Amann, Wolfgang, 385
Amara, Jihen, 79
Askinadze, Alexander, 345
Authmann, Christian, 117

B
Bader, Andreas, 249
Baum, Marcus, 291
Beilschmidt, Christian, 117
Bergen, Eduard, 65
Bieth, Tina, 137
Birnbaum, Frederick, 139
Böhmer, Kristof, 25
Bouaziz, Bassem, 79
Burkhart, Sebastian, 25

C
Carnein, Matthias, 33
Cornelius, Joseph, 375
Csar, Theresa, 163
Curio, Cristóbal, 137
Czora, Sebastian, 57

D
Diller, Martin, 169
Dix, Marcel, 57
Dreissig, Felix, 281
Drönner, Johannes, 117

E
Edlich, Stefan, 65
Eibl, Günther, 25
Eichelberger, Holger, 49
Elmamooz, Golnaz, 127
Endres, Markus, 155, 181
Engel, Dominik, 25
Erdélyi, Gábor, 185

F
Falkenthal, Michael, 249
Ferner, Cornelia, 25
Finzel, Bettina, 127
Friedrich, Steffen, 215, 269
Fromm, Hansjörg, 57

G
Gessert, Felix, 211, 269, 399
Giebler, Corinna, 311
Grimmer, Martin, 227
Groß, Anika, 75

H
Hickler, Thomas, 117
Hildebrandt, Tobias, 25
Hirmer, Pascal, 111
Homann, Leschek, 33
Hunter, Anthony, 169

J
Janusz, Daniel, 237
Junghanns, Martin, 105

K
Kahlert, Roland, 375
Kaltenthaler, Daniel, 89
Kemper, Stephan, 105
Kiefer, Cornelia, 99
Klöpper, Benjamin, 23, 57
König-Ries, Birgitta, 75
Kopp, Oliver, 249
Kraume, Karsten, 33
Kricke, Matthias, 227
Kröger, Peer, 89

L
Lackner, Martin, 163
Liebeck, Matthias, 375
Lohrer, Johannes-Y., 89
Ludmann, Cornelius A., 41

M
Maly, Jan, 193
Mattig, Michael, 117
Moewes, Christian, 139

N
Niamir, Aidin, 117
Nicklas, Daniela, 127, 139
Noll, Stefan, 335

O
Obermaier, Henriette, 89

P
Paar, Alexander, 137
Peska, Ladislav, 203
Petermann, André, 105
Pfandler, Andreas, 155
Pichler, Reinhard, 163
Pollner, Niko, 281
Pretzsch, Florian, 321

Q
Qin, Cui, 49

R
Rakow, Thomas C., 355
Ramadani, Jasmin, 143
Reger, Christian, 185
Reimann, Peter, 75
Rinderle-Ma, Stefanie, 25
Ritter, Norbert, 211, 215, 269, 399
Romberg, Julia, 301
Romero, Javier, 159
Rudenko, Lena, 181

S
Salgert, Björn, 355
Sallinger, Emanuel, 163
Schmeißer, Michael, 227
Schmid, Klaus, 49
Schmid, Ute, 139
Schmidt, Marco, 117
Schmitz, Björn, 57
Seeger, Bernhard, 75, 117
Seidl, Thomas, 397
Speidel, Stefanie, 137
Stach, Christoph, 311
Stertz, Florian, 25
Swoboda, Oliver, 365

T
Taeschner, Jochen, 237
Tashkandi, Araek, 291
Trautmann, Heike, 33

V
Vojtas, Peter, 203
Vossen, Gottfried, 33

W
Wagner, Stefan, 143
Waizenegger, Tim, 405
Wiese, Lena, 23, 291
Wingerath, Wolfram, 215, 269, 399
Witt, Erik, 269
Woltran, Stefan, 193