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: $\text{remove}(s,e) = s-1$.

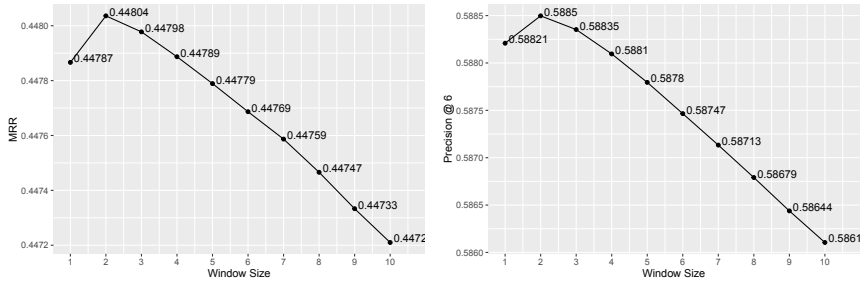
Nach jeder Änderung des Zustands ruft der Aggregationsoperator eine Funktion $\text{eval}(s) \mapsto r$ aus und gibt das Ergebnis r aus. Für COUNT ist $\text{eval}(s) = s$. Der Operator gibt somit einfach den Zustand aus. Andere Funktionen, wie z. B. AVG mit dem Zustand bestehend aus Summe und Anzahl,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