Towards Learned Metadata Extraction for Data Lakes

Sven Langenecker[1] Christoph Sturm[2] Christian Schalles[3] Carsten Binnig[4]

Abstract: An important task for enabling the efficient exploration of available data in a data lake is to annotate semantic type information to the available data sources. In order to reduce the manual overhead of annotation, learned approaches for automatic metadata extraction on structured data sources have been proposed recently. While initial results of these learned approaches seem promising, it is still not clear how well these approaches can generalize to new unseen data in real-world data lakes. In this paper, we aim to tackle this question and as a first contribution show the result of a study when applying Sato — a recent approach based on deep learning — to a real-world data set. In our study we show that Sato without re-training is only able to extract semantic data types for about 10% of the columns of the real-world data set. These results show the general limitation of deep learning approaches which often provide near-perfect performance on available training and testing data but fail in real settings since training data and real data often strongly vary. Hence, as a second contribution we propose a new direction of using weak supervision and present results of an initial prototype we built to generate labeled training data with low manual efforts to improve the performance of learned semantic type extraction approaches on new unseen data sets.

Keywords: data lakes; dataset discovery and search; semantic type detection

1 Introduction

Motivation: Data lakes are today widely being used to manage the vast amounts of heterogeneous data sources in enterprises. Different from classical data warehouses, the idea of data lakes is that data does not need to be organized and cleaned upfront when data is loaded into the warehouse [Di14]. Instead, data lakes follow a more "lazy" approach that allows enterprises to store any available data in its raw form. This raw data is organized and cleaned once it is needed for a down stream task such as data mining or building machine learning models. However, due to the sheer size of data in data lakes and the absence (or incompleteness) of a comprehensive schema, data discovery in a data lake has become an important problem [Ma17; Na20; RZ19].

One way to address the data discovery problem, is to build data catalogs that allow users to browse the available data sources [Na19]. However, building such a data catalog manually would again pose high effort since metadata needs to be annotated on data sources. An important task for cataloging structured (table-like) data in a data lake (e. g., originating from CSV files) is to derive semantic type information for the different columns of a data

[1] DHBW Mosbach, Germany sven.langenecker@mosbach.dhbw.de
[2] DHBW Mosbach, Germany christoph.sturm@mosbach.dhbw.de
[3] DHBW Mosbach, Germany christian.schalles@mosbach.dhbw.de
[4] TU Darmstadt, Germany carsten.binnig@tu-darmstadt.de

doi:10.18420/btw2021-17

set. The reason is that this information is often missing in many data sources or the column labels available in data sources are not really helpful for data discovery since they use technical names or have been annotated from users with a different background.

In order to tackle the problem of extracting semantic data types from structured data sources in data lakes, recently learned approaches for metadata extraction have been proposed [Ca18b; Hu19; Zh20]. The main idea of these learned approaches is that they use a deep learning model for semantic type detection where the models are trained on massive table corpora with already annotated columns. While initial results of these learned approaches seem promising, it is still not clear how well these approaches can deal with the variety of data in real data lakes.

Contributions: In this paper, we aim to tackle this question and report on our initial results of analyzing the quality of the state-of-the-art learned approaches for metadata extraction on real-world data. Moreover, we also show initial results of a new direction of tackling the open problems of the learned approaches that we discovered in our analyses. In the following, we discuss the two main contributions of this paper.

As a first contribution, we show the result of a study when applying Sato [Zh20] - a recent approach based on deep learning to extract semantic types - to a real-world data set. A inherent problem of deep learning-based approaches for semantic type extraction is that they rely on a representative training data set; i.e., a set of columns with labeled semantic types. Otherwise, if the training data set does not cover the broad spectrum of data characteristics and types, the performance of the learned models quickly degrades when applied to a new data set. In fact, we show that Sato without re-training was only able to extract semantic data types for about 10% of the columns on the data sets used in our study.

As a second contribution, we thus suggest to take a new direction for learned metadata extraction to tackle the shortcomings of the existing deep learning approaches. As mentioned before, the root cause of why existing approaches often fail to extract semantic types is that the training data of the learned approaches is too narrow and thus the performance on new data sets is often poor. Hence, in this paper we propose a new direction of using weak supervision to generate a much broader set of labeled training data for semantic type detection on the new data set. Our initial results show that our approach can significantly boost the performance of deep learning-based approaches such as Sato when re-training these approaches on the additional synthesized training data.

Outline: In Section 2, we first provide an overview of approaches for metadata extraction from structured data in data lakes. Afterwards, we discuss the results of our study of using Sato as a recent learned approach on a real-world data set in Section 3. Moreover, we then discuss our new approach based on weak supervision in Section 4. Finally, we present the initial results of using our current prototype in Section 5 before we conclude in Section 6.

2 Overview of Existing Approaches

In the following, we give a short overview of selected existing approaches for metadata extraction. We first discuss approaches for semantic type extraction before we briefly summarize recent approaches for the extraction of relationships.

2.1 Extraction of Semantic Types

Approaches that automatically extract types from metadata of data sources are already well established in industry. Prominent examples are Azure Data Catalog [Az20], AWS Glue [AW20] and GOODS [Ha16]. In addition, many other research efforts exist for developing generic metadata models and special algorithms for metadata extraction (e. g. [QHV16]).

All these approaches rely on the fact that basic metadata information is already annotated in the data source (e. g., as a header row in a CSV file) such as column and table names. However, header rows exist only in few cases and even when they do, the attribute names are not always useful as a semantic type. In this case, existing systems opt for manual metadata annotation.

Considering the huge amount of heterogeneous, independent, quickly changing data sources of real-world data lakes these approaches reach their limit. Therefore, some systems aim to detect semantic types from the columns content instead of relying on already existing labels in the sources. For this purpose, there exist two main research directions for automatic semantic type detection: search-based approaches and learning-based approaches.

Search-based Approaches: The main idea of search-based approaches is to use external information to annotate semantic information to data sets. One approach in this direction is AUTOTYPE [YH18] which searches for existing custom extraction code to handle more specific (domain dependent) semantic data types. By helping developers to find and extract existing type detection code the supported semantic types of AUTOTYPE can be extended semi-automatically.

Learning-based Approaches: In contrast to search-based approaches, learning-based approaches aim to build a machine learning model that can derive semantic types of columns from example data and not from extraction code. An early approach in this class is [LSC10] which uses machine learning techniques to annotate web tables and their columns with types. While this approach relies on graphical models for extracting semantic labels for columns, more recent approaches such as [Hu19; Zh20] are based on a deep neural network.

Sato [Zh20] which is the successor of Sherlock [Hu19] thus requires training data with labeled semantic types. While Sherlock only uses the individual column values as features for predicting the semantic type, Sato also uses context signals from other columns in the table to predict the semantic type of a given column.

2.2 Extraction of Relationships

While extraction of semantic types is one important direction for metadata extraction, there also exist other approaches that are able to derive relationships between datasets (e. g., an author *writes* a book) from data automatically. One prominent example for such an approach is AURUM [Ca18a]. While AURUM represents an overall system for building, maintaining and querying an enterprise knowledge graph for available data sources, SEMPROP [Ca18b] is the subsystem of AURUM, which automatically derives links (i.e., relationships) between the data sources using word embeddings.

As mentioned before, different from this work and similar to Sato in this paper we focus on the extraction of semantic types for structured data sets. Hence, in the following sections we will limit the analysis of learned approaches to this direction. However, extending our approach towards relationship extraction is an interesting avenue of future work.

3 Study of Using Learned Approaches

In the following, we present the results of our study of using learned semantic type extraction approaches on real-world data. For our study, we selected Sato [Zh20] as a recent approach based on deep learning.

3.1 Data Sets and Methodology

Data Sets: As a data set in this study, we use the Public BI Benchmark[5] data corpus. The data corpus contains real-world data, extracted from the 46 biggest public workbooks in Tableau Public[6]. In this corpus there are 206 tables each with 13 to 401 columns. The main reason for choosing this corpus for our study was that it contains labeled structured data from different real-world sources across various domains (e. g. geographic, baseball, health, railway, taxes, social media, real estate). Hence, the benchmark comes with a high diversity and heterogeneity of data sources that can typically also be found in data lakes of enterprises today.

Methodology: As mentioned before, the inherent problem of deep learning-based approaches for semantic type extraction is that they rely on a representative training data set. To put it differently, if the training data set does not cover the variety of cases that are also seen in the real-world data, the performance of the learned models quickly degrades. As part of our analysis, we wanted to see to which extent this inherent limitation influences the overall quality of a learned approach such as Sato.

For the study, we thus annotated the data in the Public BI Benchmark manually with the correct semantic types of Sato. For the annotation, we first preprocessed the data automatically and searched for string matches between the column headers of the tables in the Public BI Benchmark and the semantic types supported by Sato. To guarantee the

[5] https://github.com/bogdanghita/public_bi_benchmark-master_project
[6] https://public.tableau.com

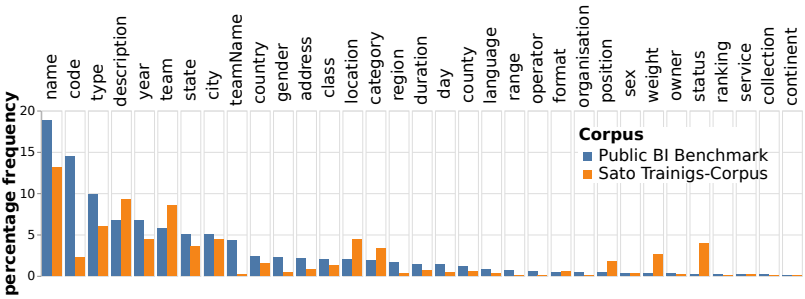

Fig. 1: Distribution of semantic types in training data of Sato and the Public BI Benchmark

correctness of labels every column was additionally inspected and missing types where added manually.

3.2 Results of the Study

As a first question, we analyzed the coverage rate of the 78 semantic types supported by Sato in the Public BI Benchmark to see to which extend a pre-trained model can support real-world data if no new training data is used for re-training. For this question, we analyzed what fraction of columns in the Public BI Benchmark had a type that was covered by the training data set of Sato. The main result of this analysis was that only 10.6% of the columns are assignable to one of the semantic types.

As a second question, for the columns of the Public BI Benchmark that have types which are supported by Sato, we then wanted to see how the distribution of the 78 semantic types in the training data used for Sato and the Public BI Benchmark look like. The reason is that different distribution of labels in the training and testing data can have a negative impact on the overall quality of a learned approach. As can be seen in Fig. 1, the frequency for many semantic types in both data sets (i.e., original training data of Sato and the Public BI Benchmark), however, is almost identical.

As a final question, we thus aimed to analyze the 10.6% of the columns in the Public BI Benchmark that are in principle covered by the training data of Sato. For this, we used the pre-trained Sato model and applied it to only this fraction of the data of the Public BI Benchmark. For this subset, Sato achieves an F_1 score (macro average and weighted[7]) of 0.090 and 0.300 respectively, which is also shown in Tab. 1 in our evaluation in Section 5. The original paper [Zh20] reports an F_1 score of 0.735 and 0.925 on the VizNet[8] data corpus. This indicates that the data characteristics of the supported data types of the Public BI Benchmark is different from the data characteristics of the training data of VizNet and thus Sato can not infer types in a robust manner (even if they should be supported in principle).

[7] **F_1 score macro average**: averaging the unweighted mean F_1 score per label
F_1 score weighted average: averaging the support-weighted mean F_1 score per label

[8] `https://github.com/mitmedialab/viznet`

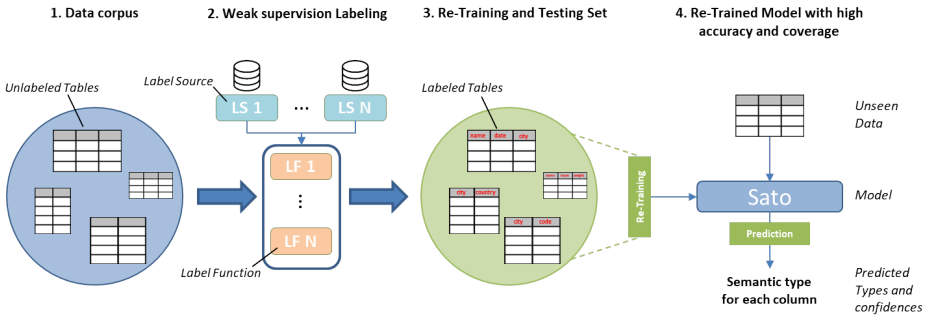

Fig. 2: Concept and step-by-step procedure of our weak supervision approach

Main Insights: As suspected, our study has shown that a deep model such as Sato trained on one data set can only cover a fraction of data types of a new data set. Moreover, for the overlapping data types, the accuracy is still pretty low due to different data characteristics of the training data and the new data set. While the results of our study are specific to Sato, we believe that our findings are much more general and translatable to any learned approach that relies on manually curated training data (which is inherently limited as discussed before). Hence, a new approach is required where one can easily adapt learning-based models for type extractors to new data sets that covers types and data characteristics not covered in the available manually labeled training data. As a solution for this requirement, we next present our new weak supervision approach in the next section.

4 Weak Supervision for Semantic Type Extraction

The root cause of why deep learning-based approaches such as Sato often fail to extract semantic types on a new data set is that the training data lacks generality as discussed before. The main idea of using weak supervision is to generate a broad set of labeled training data with only minimal manual effort and thus increase the robustness when applying a learned approach such as Sato to a new data set. In the following, we discuss our initial ideas for such an approach and present the first results of our prototype to showcase its potential.

4.1 Overview of Our Approach

Fig. 2 shows an overview of our approach. The main idea is that based on a set of simple labeling functions, we generate new (potentially noisy) training data that is then used to re-train a model such as Sato to increase the coverage of data types and data characteristics of the learned model. In other words, we apply the ideas of data programming discussed in [Ra17] for the domain of semantic type extraction.

For generating new training data in our approach, we differentiate between two different classes of labeling functions: (1) The first class are labeling functions that can generate labels (i.e., semantic types) for completely new semantic types in a data lake that are not yet covered by a manually labeled training data set. Labeling functions of this class can

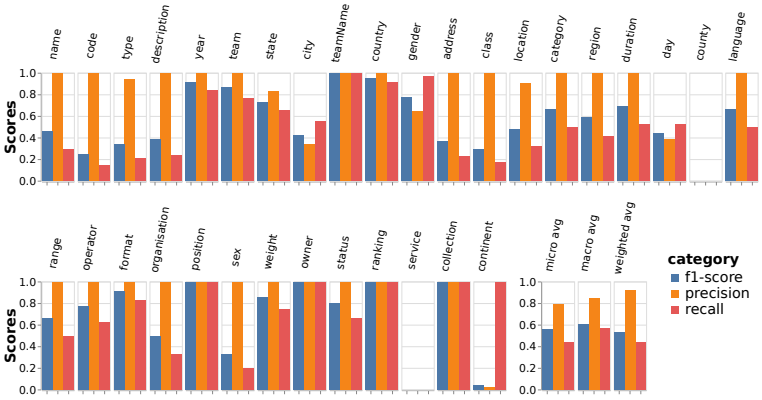

Fig. 3: Performance of clustering semantically similar columns

be, for example, regular expressions, dictionary lookups, or other techniques such as using alignment with existing ontologies. (2) Second, as we have seen in our study, another problem of learned approaches such as Sato is that they often fail to predict semantic types even if in principle the semantic type is already covered by the training data. The main reason for this case is that the training data does not cover the wide spectrum of data characteristics that might appear in a new data set. Hence, as a second class of labeling functions we support functions that can generate new labeled columns that cover more data characteristics (e.g., new values) for data types that are already available in a training data set. One idea for a labeling function of this class is the use of word embeddings [Mi13] to cluster new unlabeled with already labeled columns and thus generate new labeled columns for existing semantic types. A more detailed description of such a labeling function is given below.

4.2 Label Generation using Clustering

For generating more labeled training data for an existing semantic type, we implemented a method based on clustering in our prototype system that we briefly introduced before. As mentioned, the main idea is that we can start with a small training corpus of labeled columns and by clustering new non-labeled to the labeled columns, we can derive new labeled training data.

To implement this labeling approach, we first compute column embeddings for labeled and unlabeled columns based on word embeddings of individual values. As word embeddings, we currently use *Google USE*[9] that was trained on 16 different languages and showed good results. But in principle we could also use other word embeddings. Based on the embeddings of individual values, we compute an embedding for all values of a column by calculating the average across the embeddings of all values which is the dominant approach for building representations of multi-words also mentioned in other papers [So12]. This approach is

[9] https://tfhub.dev/google/universal-sentence-encoder-multilingual-large/3

reasonable also for us, since string-typed column values in the Public BI Benchmark are only composed of single values. In general, in the case the column values themselves consist of a sequence of words, we could also consider word embedding combining techniques as represented in [LM14] or [Ca18b].

Once we computed an embedding for all values of a column, we next carry out the clustering of labeled and unlabeled columns based on these embeddings. For this step, we use a agglomerative clustering algorithm[10]. In our prototype, we use this clustering method to not generate a fixed number of clusters, but to form groups based on the cosine similarity of vectors (i.e., our embeddings) and a distance threshold that we discuss below. Once clustered, we than compute a semantic type per cluster based on the majority vote of columns with the same label. [Ma19] represents a system called Raha, which relies on a similar idea for generating training data but for error detection and not for semantic type extraction.

A key parameter to be set in our clustering approach is the distance threshold which can vary between 0.0 and 1.0 (i.e., a lower value means that we produce more clusters). In our experiments, we used a threshold of 0.1 based on a hyper-parameter search on the already labeled columns. This threshold provided high accuracy on the broad spectrum of data sets in the Public BI Benchmark.

Initial Results: To analyze if the basic idea of clustering is working, we conducted a small experiment where we measure how well the clustering approach works on the Public BI Benchmark using our annotations of the 78 Sato types. By clustering, we wanted to see whether columns with the same type would be assigned to the same cluster. As we see in Fig. 3, with a few exceptions, the clustering algorithm achieves high precision. This means that there is a very high probability that all elements in one cluster belong to the semantic type representing the specific cluster. For many types, we achieve an F_1 score of 1.0 such as for the semantic types *teamName*, *position*, *owner*, *ranking* and *collection*.

Moreover, in a second experiment, we wanted to show the robustness of our clustering approach to different data characteristics. For showing this, we analyzed the entropy and the jaccard-coefficient for all columns with the same semantic type in the Public BI Benchmark. The intuition is that columns with a high entropy (i.e., a high degree of divergence) or pairs of columns which have a low jaccard-coefficient (i.e., where columns values are not overlapping) are harder to cluster. Overall, our approach assigns column pairs with the same semantic type to the very same cluster even if they strongly vary in the entropy or have a low overlap (i.e., a low jaccard-coefficient). Unfortunately, due to space limitations we could not add further details about this experiment to the paper.

[10] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering`

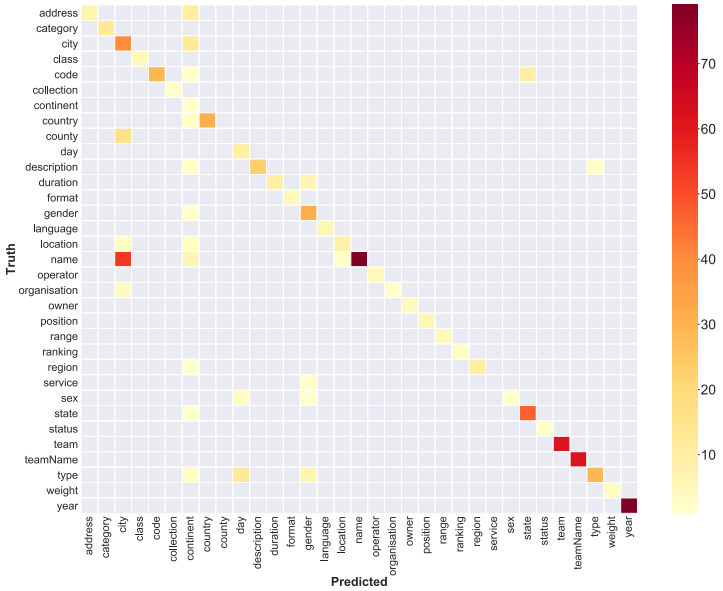

Fig. 4: Confusion matrix of the clustering method

4.3 Future Directions

As mentioned before, in this paper we showed only a very first prototype where we apply the idea of weak supervision for synthesizing labeled training data for semantic type extraction. In our first prototype, we only covered the labeling approach based on clustering as discussed before. Hence, the main avenue of future work is to extend this prototype and add a much broader set of labeling functions.

Furthermore, another direction is to study alternatives for the training data generation process. Currently, we directly use the potentially noisy training data generated by the labeling functions for re-training. Another possible direction as shown in [Ra17], would be to first train a generative model that can learn how to generalize from the additional training data and thus mitigate the negative effects such as noisy data to a certain extent.

Finally, in the current state, we only consider semantic types whose data values are strings or types that provide a semantic meaning when converted to a string (such as *weights* and *dates*). The semantic type detection of numeric types such as *temperature* require additional labeling functions and therefore represent future research.

5 End-to-End Evaluation

In the section before, we have already shown that the basic idea of weak supervision can help to generate training data by clustering to improve the robustness w.r.t different data characteristics. In the following, we report on the initial results of using this approach in an

	Macro average F_1	Precision	Recall	Support-weighted F_1
SHERLOCK (not re-trained)	0.114	0.375	0.309	0.322
SHERLOCK (re-trained)	0.806	0.879	0.859	0.860
SATO (not re-trained)	0.090	0.322	0.304	0.300
SATO (re-trained)	**0.811**	**0.912**	**0.894**	**0.894**

Tab. 1: Performance comparison of the models on Public BI Benchmark

end-to-end evaluation to show how this can boost the performance of learned type extraction approaches such as Sato.

Setup and Data Preparation: We implemented our approach for automatic labeling in Python using the Google USE embeddings as mentioned before. Moreover, for training and evaluation, we used the source code provided by Sato[11]. However, Sato is designed to be built and trained from scratch. Hence, we extended Sato with the appropriate functionality for incremental re-training.

End-to-End Results: For showing the end-to-end performance of our approach, we restricted ourselves to the 10% of the Public BI Benchmark data that is supported by Sato and its semantic types. For this, we first generated additional training data and then re-trained the pre-trained Sato model with our additionally labeled data. For generating additional training data, we used the clustering approach discussed before for the Public BI Benchmark. For this purpose, we split the Public BI Benchmark into a training and testing set.

As we see in Tab. 1, after re-training the Sato model with the synthesized training data of our approach, Sato achieves F_1 scores (macro average and weighted) of 0.811 and 0.89 respectively. This is a significant improvement of almost +0.60 compared to the performance of Sato without re-training. In addition to show that our approach also generalizes to other learned approaches, we furthermore used Sherlock [Hu19] (without and with re-training). As shown in Tab. 1, this leads to a similar performance gain. In summary, these results show that our approach is in principle able to boost the performance of learning-based approaches that have been pre-trained on only a small training data set not covering all data characteristics found in a new unlabeled data set.

6 Conclusions

Detecting semantic types for columns of data sets stored in data lakes results in an enormous benefit building a data catalog to address the data discovery problem. While recent papers have shown initial results for learned approaches that can be used for extracting semantic types, they cannot support many real-world data sets since they only support a limited set of semantic data types as we have shown in our study. To tackle this problem, we suggested a new direction of using weak supervision for generating additional labeled training data and use this for re-training the existing learned model. An initial evaluation of our new direction using our current prototype shows that this approach can lead to huge performance gains.

[11] `https://github.com/megagonlabs/sato/tree/master`

References

[AW20] AWS, A.: AWS Glue Concepts - AWS Glue, https://docs.aws.amazon.com/glue/latest/dg/components-key-concepts.html, 2020.

[Az20] Azure, M.: Common Data Model and Azure Data Lake Storage Gen2 - Common Data Model | Microsoft Docs, 2020.

[Ca18a] Castro Fernandez, R. et al.: Aurum: A Data Discovery System. In: 2018 IEEE 34th International Conference on Data Engineering (ICDE). Pp. 1001–1012, 2018.

[Ca18b] Castro Fernandez, R. et al.: Seeping Semantics: Linking Datasets Using Word Embeddings for Data Discovery. In: ICDE '18. Pp. 989–1000, 2018.

[Di14] Dixon, J.: Data Lakes Revisited, https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited/, 2014.

[Ha16] Halevy, A. et al.: Goods: Organizing Google's Datasets. In: SIGMOD '16. ACM, pp. 795–806, 2016.

[Hu19] Hulsebos, M. et al.: Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In: KDD '19. ACM, pp. 1500–1508, 2019.

[LM14] Le, Q.; Mikolov, T.: Distributed Representations of Sentences and Documents. In: ICML. 2014.

[LSC10] Limaye, G. et al.: Annotating and Searching Web Tables Using Entities, Types and Relationships. Proc. VLDB Endow. 3/1–2, pp. 1338–1347, Sept. 2010.

[Ma17] Mathis, C.: Data Lakes. Datenbank-Spektrum 17/3, pp. 289–293, Nov. 2017.

[Ma19] Mahdavi, M. et al.: Raha: A Configuration-Free Error Detection System. In: SIGMOD '19. ACM, 2019.

[Mi13] Mikolov, T. et al.: Distributed Representations of Words and Phrases and Their Compositionality. In: NIPS'13. Pp. 3111–3119, 2013.

[Na19] Nargesian, F. et al.: Data Lake Management: Challenges and Opportunities. Proc. VLDB Endow. 12/12, pp. 1986–1989, 2019.

[Na20] Nargesian, F. et al.: Organizing Data Lakes for Navigation. In: SIGMOD '20. ACM, pp. 1939–1950, 2020.

[QHV16] Quix, C. et al.: Metadata Extraction and Management in Data Lakes With GEMMS. Complex Syst. Informatics Model. Q. 9/, pp. 67–83, 2016.

[Ra17] Ratner, A. et al.: Snorkel: Rapid Training Data Creation with Weak Supervision. Proc. VLDB Endow. 11/3, pp. 269–282, Nov. 2017.

[RZ19] Ravat, F.; Zhao, Y.: Metadata Management for Data Lakes. In: New Trends in Databases and Information Systems. Springer, pp. 37–44, 2019.

[So12] Socher, R. et al.: Semantic compositionality through recursive matrix-vector spaces. In: Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning. Association for Computational Linguistics, pp. 1201–1211, 2012.

[YH18] Yan, C.; He, Y.: Synthesizing Type-Detection Logic for Rich Semantic Data Types Using Open-Source Code. In: SIGMOD '18. ACM, pp. 35–50, 2018.

[Zh20] Zhang, D. et al.: Sato: Contextual Semantic Type Detection in Tables. Proc. VLDB Endow. 13/12, pp. 1835–1848, July 2020.