Table Discovery in Data Lakes

Grace Fan
Supervised by Professor Renée J. Miller
Northeastern University
Boston, United States
fan.gr@northeastern.edu

ABSTRACT

Data scientists use table discovery systems to search for relevant tables in a large data lake. With tasks like compiling training data for machine learning models, for example, data scientists need to find tables that are relevant to a table at hand, or a query table. To address this challenge, we present automated, effective, and efficient methods for data discovery problems in the large scale of data lakes. Our approaches aim to solve a general data discovery problem called table union search and a data discovery problem for a specific downstream task called table reclamation. Existing solutions for table union search focus on aligning column semantics with the query table to find unionable tables, and return tables that are unrelated to the table context of the query table. We present approaches that take columns' relationships and the entire table context into account when retrieving unionable tables. For the problem of table reclamation, we present a system that verifies if a query table can be reproduced from a retrieved set of tables from a data lake. This validation process allows users to assess the origins of their query table, as well as the completeness, validity, and currency of the query table.

VLDB Workshop Reference Format:
Grace Fan. Table Discovery in Data Lakes. VLDB 2024 Workshop: VLDB Ph.D. Workshop.

1 INTRODUCTION

The number of datasets stored in data lakes has grown drastically in the past decade, and continues to grow as more datasets are created. These datasets come from a wide range of sources, including governments, companies, and academic institutions. To search and analyze these datasets, researchers have built systems for table discovery [12], with techniques for keyword search [2, 6, 7, 21, 30, 34] and data-driven search [15, 27, 29, 37, 38] that support joinable table search and unionable table search.

Tabular data in data lakes often have limited or indecipherable metadata, with missing or inconsistent column names [2, 14, 26, 27]. Thus, we rely solely on table values when finding relevant tables to a query table. This research focuses on two types of table discovery tasks – table union search and table reclamation. In table union search, we search over a data lake for tables that can be unioned with a given query table and extend it with tuples. We explore different ways of preserving the table context of the query table when retrieving unionable tables from a data lake. In a new problem called table reclamation, we perform targeted table discovery and integration by efficiently searching for data lake tables that, when integrated, can reproduce a given query table.

In the following sections, Section 2 discusses two solutions for table union search. Section 3 describes a new problem called Table Reclamation and discusses our proposed solution. Finally, we conclude with future work (Section 4).

Proceedings of the VLDB Endowment. ISSN 2150-8097.

2 TABLE UNION SEARCH

When data scientists have tables as queries, they often want to find relevant tables from a table repository, such as a data lake, to their query tables. One notion of relevance we consider is "unionability", where we find data lake tables that can be unioned with a given query table. Then, data scientists can augment these unionable tables with their current tables to use as training or testing data for their machine learning models. In addition, they can use unionable tables to generalize tuples in their current tables along different dimensions, such as space and time.

In table union search, the user (e.g., data scientist) provides a query table and a data lake. From the data lake, we return tables that contain data that is semantically similar to the data in the query table, and can thus extend the query table with new tuples. Formally, given a query table Q and a data lake D, we return the top-k unionable tables that, when unioned with the query table, preserve the query table's semantics.

2.1 Previous work on Table Union Search

Early work on table union search (e.g., Sarma et al. [29]) first define unionable tables as *entity complements*. These are tables that share similar column headers (schemas) and a subject column, or a column that contains entities that the table is about, with the query table. Relaxing the strong assumption that unionable tables share schemas, Nargesian et al. [27] formally define unionable tables as those that share attributes from the same domain as the query table. They propose a data-driven approach that uses probabilistic models to measure the likelihood that attributes originate from the same domain. More recently, Bogatu et al. [4], the state-of-the-art method, also propose an attribute unionability methodology. Their method extends Nargesian et al. by considering additional similarity metrics that use the schemas and attribute distributions.

2.2 Relationship-based Table Union Search

We propose a solution called SANTOS [19] that extends the definition of table union search to finding tables that not only share

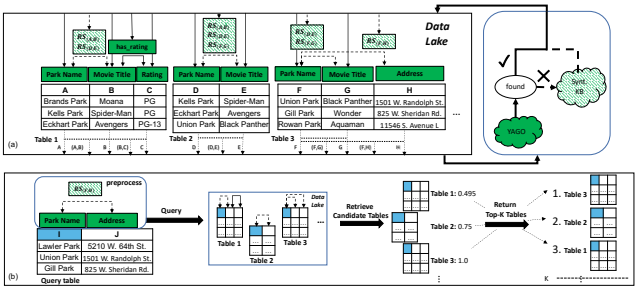

Figure 1: Pipeline of SANTOS. (a) Data lake tables are labeled with YAGO types, if found, and from the synthesized knowledge base otherwise. (b) Given a query table that we also annotate with knowledge bases, we query the data lake for unionable tables.

columns from the same domain, but also share relationships between columns. Consider an example in which a possible unionable table from the data lake shares columns "Year" and "City" with the query table. However, the query table has columns about city parks and when they were founded, whereas the data lake table has columns about people, their birthplaces and birthdates. If we only look at shared columns with the query table, we will falsely return this table as a unionable table. Instead, we can also look at relationships between columns, in addition to shared columns. This way, we can better preserve the query table's semantics when finding unionable tables.

To align semantics of the query table with data lake tables, we need to find semantics of columns and relationships between columns. First, we label table semantics using a knowledge base. To do so, we query a knowledge base (e.g., YAGO [32]) by mapping columns to classes and column relationships to properties. Since knowledge bases only consider relationships between two columns, we find semantics of binary relationships between two columns. However, open knowledge bases like YAGO tend to have limited coverage of data lake tables [19]. Hence, we create a novel, data-driven "synthesized" knowledge base that is made up of table values from the data lake. This way, we can assess if columns and their relationships in the query table and data lake tables have similar semantics. Putting it all together (shown in Figure 1), we first take data lake tables and find column and relationship semantics using an existing knowledge base (YAGO). If semantics are not found, then we find semantics using our synthesized knowledge base (Synt. KB). When we are given a query table, we find column and relationship semantics of the query table in the same way, and find the top-k unionable tables that have similar semantics.

In our evaluations, we experiment on the TUS benchmark, an existing benchmark from Nargesian et al. [27] that consists of 1,530 tables derived from 10 tables from Canada open data. In addition, we create two new, publicly available benchmarks, SMALL and LARGE, with 550 and 11K tables, respectively, from Canada, UK, US, and Australian Open data [1]. We use SMALL for effectiveness experiments, using the groundtruth that we manually labeled, and LARGE for efficiency experiments. SANTOS is highly effective in the experiments, outperforming the state-of-the-art method (Bogatu et al. [4]) by 25-165% in Mean Average Precision@k (MAP@k) across all benchmarks. To analyze how well the existing and synthesized knowledge bases work independently and together, we perform an ablation study in which we measure MAP@k. The synthesized knowledge base improves MAP@k by 8% on the TUS benchmark and by 43% on the SMALL benchmark. The best performance comes from using both knowledge bases, having a MAP@k of 80% on the TUS benchmark and 93% on the SMALL benchmark. This shows that the synthesized knowledge base, which stores semantics that are not found in the existing knowledge base, is needed along with an existing knowledge base. Finally, in efficiency experiments, we see that SANTOS's average query time on the LARGE benchmark is ~36 sec, which is 5X faster than the state-of-the-art method [4].

2.3 Context-based Table Union Search

From SANTOS, we see that using the semantics of relationships between columns, in addition to column semantics, improves the effectiveness of finding unionable tables. However, SANTOS can still falsely return unionable tables that have different table semantics from the query table. For example, suppose we have a query table listing "Years" when business officials took business trips and "Cities" that they traveled to. A possible unionable table from the data lake has attributes "Years" and "Cities" that species of birds were first seen. SANTOS would return this table as unionable since it finds similar semantics between "Year" and "City" in the two tables. Thus, we need a solution for table union search that uses the entire table context to better preserve the semantics of the query table when finding unionable tables. In a newer work on table union search called Starmie [13], we see the effectiveness of using the *entire* table context when finding unionable tables.

In Starmie, we take a natural language approach in which we use a language model to capture the table context when encoding columns. First, we run a pre-trained language model (e.g., BERT, RoBERTa) over the data lake tables to get column embeddings. Considering the size and noise of data lake tables, it is difficult to manually label training data that is needed to run language models in a supervised manner. Hence, we use a self-supervised technique from computer vision called Contrastive Learning [8] that allows us to learn representations without any external labels (shown in Figure 2). The goal of Contrastive Learning is to *connect* representations of unionable columns in their representation space while *separating* representations of distinct columns. To create unionable columns (positive samples), we apply an augmentation operator that uniformly samples values from a column X to generate a semantically preserving view X_{aug}. Possible augmentation operators include dropping cells and taking a sample of rows. All distinct columns (e.g., X and Y) make up negative samples. To encode table context into column representations, we use multi-column table models. We input serialized table values into a pre-trained language model, which learns contextualized column embeddings.

When we are given a query table, we run model inference to get column embeddings for the query table. We then use cosine similarity between column embeddings to find the column unionability score, and create a bipartite matching to find the table unionability score. Finally, we return the top-k tables with the highest table unionability score with the query table. In our effectiveness experiments, we see that Starmie outperforms SANTOS, the current

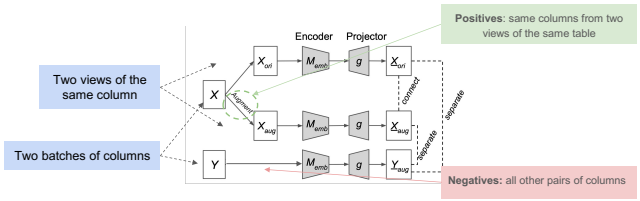

Figure 2: Contrastive Learning allows us to learn column representations in a self-supervised manner.

state-of-the-art method, by 6.8% in both MAP and Recall, with a MAP of 99.3%. Thus, by encoding the entire table context when learning column representations, Starmie proves to be very effective in returning semantically unionable tables.

To improve the efficiency of Starmie, we use indexing over column (attribute) embeddings to expedite the attribute retrieval from a data lake. A popular indexing approach that allows for approximate similarity search over high-dimensional vectors is LSH index [16], a hash-based indexing approach [4, 27, 38]. We also explore a graph-based indexing approach called HNSW index [23]. When we experiment with different data lake sizes, Starmie with LSH index times out after the data lake exceeds 10M tables. Using HNSW, Starmie can scale up to 50M tables with query times of 60ms.

Thus, SANTOS and Starmie both show that contextual information of table values can be used to improve the effectiveness of finding unionable tables.

3 TABLE RECLAMATION

From Section 2, we developed solutions to find unionable tables to a given query table. In addition to developing solutions for table union search, we define a new, but related, problem of **Table Reclamation**, for which we propose a method called Gen-T [11].

In Table Reclamation, we are given a query table and a data lake. From the data lake, we see if we can find a set of *originating* data lake tables that, when integrated, can reproduce or *reclaim* a query table as close as possible. We return to the user the reclaimed query table and a discovered set of (originating) data lake tables that can be combined to reproduce the query table as best as possible. This data discovery problem allows users to verify if their query table can be reproduced from a current data lake. More importantly, it allows users to confirm whether all values in their query table can be reproduced from a data lake. If there are any discrepancies between the query table and the output reclaimed table, a user can analyze the origins of these discrepancies.

Suppose a user has a tabular report from a news article that lists employee demographics in Top US tech companies from 2021. However, after accessing a similar report from one of the companies, she finds contradicting numbers. She discovers that the company's report includes US statistics, whereas the news article reports international numbers. Using table reclamation and her data lake, she can see if there is a set of originating tables that, when integrated, recreate the data in the news article. This way, she can verify all the data in the article with tables in the data lake before using it further in downstream tasks.

In addition to verifying news articles, a method for Table Reclamation can also verify tabular outputs of large language models.

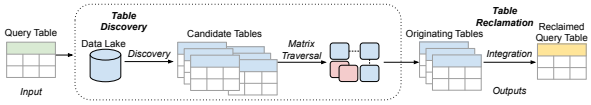

Figure 3: Gen-T Architecture: Given a query table and a data lake, Gen-T returns a set of originating tables and a reclaimed query table.

To do so, the method can verify that tabular data produced by a model can be reclaimed (reproduced) using a query over tables in the training date.

3.1 Related Work

Table Reclamation is related to the well-known problem of Data Provenance [9] that explains where tuples in a query table come from, along with why and how they were produced. However, in our problem setting, we have no knowledge of the original query or set of originating tables that first generated the query table. In fact, we do not know if the query table can even be reproduced. Query-by-example methods [3, 17, 33] also aim to reproduce a query table that is assumed to be a partial table. However, we do not assume that the query table is partial and aim to reproduce the query table as closely as possible. Query-by-target method [35] synthesizes a pipeline used to create a given target table from a given set of tables. Unlike these methods, we consider more integration operators when integrating tables to reproduce the query table, namely Select-Project-Join-Union queries. We also aim to reclaim large tables with thousands of tuples unlike most Query-by-example approaches.

3.2 Proposed solution for Table Reclamation

We propose a method for the problem of Table Reclamation called Gen-T (shown in Figure 3). First, given a query table that we want to reclaim and a data lake, we run existing table discovery methods (e.g., Starmie) to find related tables, which we call *candidate* tables. Next, we prune this set of candidate tables to only contain tables that, when integrated, produce the query table. Instead of performing expensive integration operators to perform this pruning, we simulate table integration by representing them as matrices and combining matrices. To encode each candidate table as a matrix, we encode a one if we find a value from the query table. By combining different sets of matrices, we keep track of how many values from the query table have been reclaimed. We then find the set of matrices that, after integration, produce a matrix made up of the greatest number of ones. The tables with these matrix representations become our *originating* tables. Finally, we integrate the set of originating tables to produce a reclaimed table whose values are as close as possible to those in the query table.

In our experiments, we compare against the state-of-the-art methods for query-by-target problem (Auto-Pipeline [35]), query-by-example problem (Ver [17]), and table integration (ALITE [20]). We also compare against a variation of ALITE, ALITE-PS, that performs the same table operators as Gen-T.

In our effectiveness experiments, Gen-T outperforms the best-performing baseline ALITE-PS by 16% in recall and by 56% in precision on a large, real data lake with 11K tables. Gen-T is effective even when we scale up to a large, real data lake that contains up to

15K tables. Gen-T performs 5X faster than the next-fastest baseline, ALITE-PS, on a data lake that has an average of 1M rows, achieving an average query time of 76 sec.

4 FUTURE RESEARCH

We have explored some interesting table discovery problems and presented solutions for table union search and table reclamation. In future work, we would like to help users make better use of table discovery methods by incorporating users' tasks and intent into the table retrieval process. In addition, with the rise of generative AI and large language models, there is a need to verify their outputs before users use them in downstream tasks [31]. Using our method for table reclamation, we can verify their outputs and measure the fairness of data during the validation process.

4.1 Incorporate user intent

While there has been work on incorporating user intent in table discovery methods [24, 28], an active exploration of tables returned by table discovery techniques is still understudied. For example, after SANTOS or Starmie returns the top-k unionable tables, users can use the results to refine their query table, or annotate the query columns that they want the returned unionable tables to also have. Users can also provide feedback on the results, indicating what kind of unionable tables they are looking for and why certain results are more desirable than others. Thus, users can benefit from a system that allows them to actively explore their results and recursively returns tabular results until users get the data that they need for their downstream tasks.

4.2 Verifying Generative AI Tabular Results

Gen-T is effective in solving the problem of Table Reclamation and verifying if tables in a data lake can support the facts in a query table. In future work, we would like to see how well Gen-T performs on downstream tasks, such as verifying tabular outputs of large language models. We can use Gen-T to find the training data that the model may have used to create their tabular result, and see what data in the tabular result cannot be reclaimed. For data that is not reproducible, we can use the training data to correct potential errors in the data.

Unlike existing work on using large language models for data preprocessing tasks like error correction and data imputation [36], we want to see if tabular outputs of large language models can be verified with the model's training data. This way, we can improve the model's interpretability by explaining how the model derived its output. Adopting ideas from data provenance [9], we can see what tuples from training data witness the model's output, how the model produced its output, and where in the training data the model drew from to produce its output.

4.3 Data Cleaning using Data lakes

In future work, I would like to incorporate some measures of accuracy into dataset discovery methods. For example, we can take our method for table reclamation, Gen-T, and assess if a query table and its originating tables are incomplete or out-of-date. While there has been work on data cleaning [18], methods for error correction [10, 22, 25] and data imputation [5] have mainly relied on external sources such as knowledge bases and models. To the best of my knowledge, there has yet to be work on using data from data lakes to fill in incomplete data or correct erroneous data. In addition, future research on discovering *versions* of tables can help update out-of-date data by discovering the most up-to-date version of table values.

We can also measure fairness of data in a query table and originating tables. For example, we can examine if any column from a query table exhibits social biases like gender or racial biases, such as in salary reporting. If a query table has any social bias, we can find the data from originating tables that contributed to the bias in the query table. If no originating table exhibits the same bias, we can debias the query table using originating tables. We can also use originating tables to uncover confounding variables that potentially explain the query table's biases, or look for different versions of the originating tables that do not exhibit this bias. In addition, we can see if tables have biases due to unfair data availabilities, in which case we can search the data lake to fill in missing data for underrepresented groups. If data for underrepresented groups are inaccurate, we can verify unfair information with trusted sources.

ACKNOWLEDGMENTS

This work was supported in part by NSF award numbers IIS-2107248, IIS-1956096, and IIS-2325632.

REFERENCES

[1] 2023. SANTOS Benchmarks. https://github.com/northeastern-datalab/santos/tree/main/benchmark.
[2] Marco D. Adelfio and Hanan Samet. 2013. Schema Extraction for Tabular Data on the Web. *Proc. VLDB Endow.* 6, 6 (2013), 421–432.
[3] Rohan Bavishi, Caroline Lemieux, Roy Fox, Koushik Sen, and Ion Stoica. 2019. AutoPandas: neural-backed generators for program synthesis. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 168:1–168:27.
[4] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *ICDE*. 709–720.
[5] Bernardo Breve, Loredana Caruccio, Vincenzo Deufemia, Giuseppe Polese, et al. 2022. RENUVER: A Missing Value Imputation Algorithm based on Relaxed Functional Dependencies.. In *EDBT*. 1–52.
[6] Dan Brickley, Matthew Burgess, and Natasha F. Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *WWW*. 1365–1375.
[7] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. 2009. Data Integration for the Relational Web. *Proc. VLDB Endow.* 2, 1 (2009), 1090–1101.
[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, Vol. 119. 1597–1607.
[9] James Cheney, Laura Chiticariu, and Wang Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Found. Trends Databases* 1, 4 (2009), 379–474.
[10] Xu Chu, John Morcos, Ihab F Ilyas, Mourad Ouzzani, Paolo Papotti, Nan Tang, and Yin Ye. 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In *SIGMOD*. 1247–1261.
[11] Grace Fan, Roee Shraga, and Renée J. Miller. 2024. Gen-T: Table Reclamation on Data Lakes. In *ICDE*. 3532–3545.
[12] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *SIGMOD Companion*. 69–75.
[13] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.* 16, 7 (2023), 1726–1739.
[14] Mina H. Farid, Alexandra Roatis, Ihab F. Ilyas, Hella-Franziska Hoffmann, and Xu Chu. 2016. CLAMS: Bringing Quality to Data Lakes. In *SIGMOD*. 2089–2092.
[15] Raul Castro Fernandez, Essam Mansour, Abdulhakim Ali Qahtan, Ahmed K. Elmagarmid, Ihab F. Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018. Seeping Semantics: Linking Datasets Using Word Embeddings for Data Discovery. In *ICDE*. 989–1000.
[16] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proc. VLDB Endow.* 518–529.

[17] Yue Gong, Zhiru Zhu, Sainyam Galhotra, and Raul Castro Fernandez. 2023. Ver: View Discovery in the Wild. In *ICDE*. 503–516.

[18] Ihab F Ilyas and Xu Chu. 2019. *Data cleaning*. Morgan & Claypool.

[19] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. In *SIGMOD*.

[20] Aamod Khatiwada, Roee Shraga, Wolfgang Gatterbauer, and Renée J. Miller. 2022. Integrating Data Lake Tables. *Proc. VLDB Endow.* 16 (2022), 932–945.

[21] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and Searching Web Tables Using Entities, Types and Relationships. *Proc. VLDB Endow.* 3, 1 (2010), 1338–1347.

[22] Mohammad Mahdavi and Ziawasch Abedjan. 2020. Baran: Effective Error Correction via a Unified Context Representation and Transfer Learning. *Proc. VLDB Endow.* 13, 11 (2020), 1948–1961.

[23] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[24] Hamed Mirzaei and Davood Rafiei. 2023. Table Union Search with Preferences. In *Joint Proceedings of Workshops at the 49th International Conference on Very Large Data Bases (VLDB)*.

[25] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proc. VLDB Endow.* 16, 4 (2022), 738–746.

[26] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. 2019. Data Lake Management: Challenges and Opportunities. *Proc. VLDB Endow.* 12, 12 (2019), 1986–1989.

[27] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proc. VLDB Endow.* 11, 7 (2018), 813–825.

[28] El Kindi Rezig, Anshul Bhandari, Anna Fariha, Benjamin Price, Allan Vanterpool, Vijay Gadepally, and Michael Stonebraker. 2021. DICE: Data Discovery by Example. *Proc. VLDB Endow.* 14, 12 (2021), 2819–2822.

[29] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y. Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *SIGMOD*. 817–828.

[30] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. 2020. Ad Hoc Table Retrieval using Intrinsic and Extrinsic Similarities. In *WWW*. 2479–2485.

[31] Nan Tang, Chenyu Yang, Ju Fan, Lei Cao, and Alon Halevy. 2024. VerifAI: Verified Generative AI. In *CIDR*.

[32] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian M. Suchanek. 2020. YAGO 4: A Reasonable Knowledge Base. In *ESWC (Lecture Notes in Computer Science)*, Vol. 12123. Springer, 583–596.

[33] Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. 2009. Query by output. In *SIGMOD*. 535–548.

[34] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. 2020. Web Table Retrieval using Multimodal Deep Learning. In *SIGIR*. 1399–1408.

[35] Junwen Yang, Yeye He, and Surajit Chaudhuri. 2021. Auto-Pipeline: Synthesize Data Pipelines By-Target Using Reinforcement Learning and Search. *Proc. VLDB Endow.* 14, 11 (2021), 2563–2575.

[36] Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2023. Jellyfish: A Large Language Model for Data Preprocessing. *CoRR* abs/2312.01678 (2023).

[37] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *SIGMOD*. 847–864.

[38] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *Proc. VLDB Endow.* 9, 12 (2016), 1185–1196.