

Making Data More Findable and Reusable

by

Tianji Cong

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2025

Doctoral Committee:

Professor H. V. Jagadish, Chair

Dr. Madelon Hulsebos, Centrum Wiskunde & Informatica

Professor Margaret Levenstein

Professor Fatemeh Nargesian, University of Rochester

Professor Lin Ma



Tianji Cong

congtj@umich.edu

ORCID iD: 0000-0002-7793-7059

© Tianji Cong 2025

To my parents

ACKNOWLEDGEMENTS

If I had known how much support I would need to complete my PhD, I might have thought twice about starting. I can not be more grateful to the people who have supported me during this journey. Foremost, I want to thank my advisor H. V. Jagadish. Throughout my time with him, Jag has always put my interests first. He gives me the freedom to explore my own path while patiently ensuring I stay on the right track. I truly feel I have become the independent researcher he hoped I would be at the beginning.

I was also lucky to collaborate with every committee member: Margaret Levenstein (Maggie), Fatemeh Nargesian, Lin Ma, and Madelon Hulsebos. Maggie funded my research and gave me the opportunity to interact with the lovely folks at ICPSR (Bing She, Kelly Doonan-Reed, Jeannette Jackson, to name just a few) and learn about their data management practices in maintaining the world’s largest social and behavioral science data archive. This inspired my research and allowed me to develop something with real-world assumptions that could, I hope, help them in their work. Fatemeh and I worked together on two projects. Without her discussions and support, I could not have finished them. Lin has rich experience in data systems (both in academia and industry). My many conversations with him have deepened my desire to work more on the system side after graduation. Lastly, I want to express my deepest gratitude to Madelon Hulsebos. We had a lovely and fruitful collaboration. More importantly, Madelon connected me to many opportunities: my first internship, my first trip to the Dagstuhl seminar, and many others. Without her, my PhD would have been much harder and less fun. Beyond the committee, I want to thank Prof. Atul Prakash and Prof. Michael Cafarella for their advising and guidance during my undergrad, which led me to pursue a PhD.

Throughout my PhD, I also very much enjoyed my internship experiences in industry. Madelon introduced me to Cagatay Demiralp’s research group at Sigma Computing, a unicorn startup in the Business Intelligence market. It was eyes-opening to work in an established startup in San Francisco. The company was on an upward trend (they still are), and it was exciting to see how many deals and how much money the sales team closed in a Slack channel. We had the luxury of eating really good food at great restaurants in the city (fun fact: Madelon was too full to have her dessert, so I helped her use the budget), and

Cagatay invited many big names in academia to give talks at the company. It was a really memorable experience. In 2024, I received a part-time offer from a Miami-based startup and had the luxury to spend one week onsite with Wes Madigal and John Corrigan. Although I only there for less than three months before joining Microsoft Gray Systems Lab for another summer internship, it was a unique experience to be the third employee, alongside the two founders, for that short period. At Microsoft, I met with my lovely manager, Yuanyuan Tian, who was very nice and supportive, and we collaborated on a very interesting project. One cannot really expect to have a manager as patient and thoughtful as Yuanyuan. I also still remember how good the food was on Microsoft's Mountain View campus. Lastly, in 2025, I joined Google AlloyDB for another internship. This was the first time I worked on database kernels, and I found I liked it so much that about two and a half months after the internship, I joined the same team full time. I really admire my manager, Viral Shah, who is tech-driven and was unbelievably supportive throughout my internship. I remember explaining something to Viral in the hallway, and he told me to grab my laptop and "let's debug together," despite having many other priorities. I learned a lot that summer and enjoyed every interaction at Google.

I also want to extend my thanks to my peers in the department. I greatly enjoyed my conversations with Neal Mangaokar, Jae-Won Chung, the folks in my office and my partner's office, and Zhenjie Sun, the only undergraduate I mentored throughout my PhD. Outside of work, I very much enjoyed playing tennis with Rui Chen, Adan Newman, Changyu Deng, Zhuodong Tang, and many others I met on public courts and in the Chippewa Tennis Club. Tennis was the one sport I played consistently, regardless of the season.

Finally, I want to thank my parents, Guimao Cong and Chunyun Xu, for their unconditional love and support since day one. They always let me make my own decision and back me up, even when there seemed to be huge uncertainties and little chance of success (e.g., when I wanted to transfer to a U.S. university for my undergrad, while we had little idea what we needed to go through and how likely I was to make it). My parents' hard working and love for their family have deeply shaped who I am and continue to inspire me to keep exploring. I also want to thank my partner, Juechu Dong, for her companionship since the middle of my PhD. Our love of nature has led us to visit many national parks in the U.S., and those fantastic scenes and lovely memories have colored my graduate student life to a great extent. Juechu's cooking also comforted me from day to day.

TABLE OF CONTENTS

DEDICATION	ii
ACKNOWLEDGEMENTS	iii
LIST OF FIGURES	ix
LIST OF TABLES	xii
LIST OF ACRONYMS	xiv
ABSTRACT	xv
CHAPTER	
1 Introduction	1
1.1 Technical Challenges	3
1.1.1 Discoverability Issues	3
1.1.2 Data Heterogeneity and Lack of Standardization	3
1.2 Contributions Overview	4
2 Background	7
2.1 Data Discovery	7
2.2 Metadata	9
2.3 Representation Learning	10
2.4 Large Language Models	11
3 Pylon: Semantic Table Union Search in Data Lakes	13
3.1 Introduction	13
3.2 Problem Definition & Background	16
3.2.1 Table Union Search	16
3.2.2 How Table Union Search Differs from Schema Matching and Join Discovery?	18
3.2.3 General Challenges	18
3.3 Pylon: A Self-Supervised Contrastive Learning Framework for Tabular Data	19
3.3.1 Contrastive Learning	20
3.3.2 PYLON Workflow	20
3.3.3 Choices of the Base Encoder	22

3.3.4	Embedding Indexing and Search	24
3.3.5	Integrating Syntactic Methods	26
3.4	Experiments	26
3.4.1	Datasets and Metrics	26
3.4.2	Baselines	27
3.4.3	Comparisons of Interest	28
3.4.4	Experiment Details	29
3.4.5	Results	30
3.4.6	Discussion	35
3.5	Related Work	36
3.6	Conclusion	37
4	WarpGate: A Semantic Join Discovery System for Cloud Data Warehouses	38
4.1	Introduction	38
4.2	Problem Definition	40
4.2.1	Data Discovery Need in Sigma	40
4.2.2	Semantic Join Discovery	41
4.3	WarpGate Overview	41
4.3.1	Embedding-Based Solution	42
4.3.2	WarpGate Interface	44
4.4	Experiments	45
4.4.1	Datasets	45
4.4.2	Baselines & Metrics	46
4.4.3	Experiment Results	47
4.4.4	Sample Efficiency	49
4.5	Discussion	49
4.5.1	Sigma Customer Data Scale	49
4.5.2	Optimization Opportunities	50
4.6	Related Work	51
4.7	Conclusion	52
5	Nexus: Inferring Join Graphs from Metadata Alone via Iterative Low-Rank Matrix Completion	53
5.1	Introduction	53
5.2	Preliminaries and Observations	57
5.2.1	Join Graphs	57
5.2.2	Problem Statement	58
5.2.3	Empirical Analysis and Observations	59
5.3	NEXUS Design	60
5.3.1	Overview	61
5.3.2	Metadata and Join Candidates Pruning	61
5.3.3	Scoring Join Candidates	62
5.3.4	Join Graph Inference through Low-Rank Matrix Completion	63
5.3.5	Iterative Join Graph Inference via Expectation-Maximization Algorithm	65

5.3.6	Optimization via Core Submatrix Reduction	69
5.3.7	Extension to Multi-Column Joins	70
5.4	Experiments	70
5.4.1	Experiment Setup	71
5.4.2	Quality Comparison	72
5.4.3	When Query Logs are Available	74
5.4.4	Fast Mode Inference	74
5.4.5	Efficiency Comparison	76
5.4.6	When Data Values are Available	76
5.5	Related Work	77
5.6	Conclusion	79
6	OpenForge: Probabilistic Metadata Integration	80
6.1	Introduction	80
6.2	Metadata Integration Problem	83
6.2.1	Problem Statement	83
6.2.2	Probabilistic Formulation	84
6.3	OPENFORGE Framework	87
6.3.1	Obtaining Prior Beliefs	87
6.3.2	MRF Modeling of Metadata Integration	88
6.4	Scalable Metadata Integration	92
6.4.1	Inference in Markov Random Field	92
6.4.2	Scaling MRF Modeling	95
6.5	Implementation	96
6.5.1	Prior Models	96
6.5.2	MRF Inference and Parameter Tuning	97
6.6	Experiments	98
6.6.1	Experimental Setup	98
6.6.2	Quality Comparison	100
6.6.3	Prior Models	103
6.6.4	Efficiency of MRF Modeling	103
6.6.5	Scalability	105
6.7	Discussion	106
6.8	Related Work	106
6.9	Conclusion	108
7	Observatory: Characterizing Embeddings of Relational Tables	110
7.1	Introduction	110
7.2	Related Work	112
7.2.1	Language and Table Embedding Models	112
7.2.2	Analysis of Embedding Models	112
7.3	Observatory	113
7.3.1	Problem Statement	113
7.3.2	Relational Properties	114
7.3.3	Data Distribution Properties	119

7.4	Experiment Setup	122
7.4.1	Embedding Models	122
7.4.2	Datasets	123
7.4.3	Implementation	125
7.5	Results	126
7.5.1	Row Order Insignificance	126
7.5.2	Column Order Insignificance	129
7.5.3	Join Relationship	129
7.5.4	Functional Dependencies	131
7.5.5	Sample Fidelity	132
7.5.6	Entity Stability	133
7.5.7	Perturbation Robustness	134
7.5.8	Heterogeneous Context	136
7.6	Connection to Downstream Tasks	137
7.7	Discussion	138
7.8	Conclusion	139
8	Conclusion and Future Work	140
8.1	Future Work	140
8.1.1	Towards a Unified System for Hybrid Search	140
8.1.2	Multi-Modality Data Processing Systems	141
8.1.3	Lakehouse Movement	142
8.2	Final Words	143
	BIBLIOGRAPHY	144

LIST OF FIGURES

FIGURE

2.1	Examples of dataset metadata from two dataset hosting platforms: data.gov (left) and ICPSR (right).	8
3.1	An example of two tables unionable over four pairs of attributes: title - Title, authors - Authors, venue - Platform, and year - Year.	14
3.2	Training workflow of PYLON (with online training data construction).	20
3.3	Top-k precision and recall of embedding measures on the PYLON dataset.	28
3.4	Indexing time and query response time on the PYLON dataset.	28
3.5	Precision and recall (with respect to varying k) of the ensemble of PYLON embedding models and syntactic measures.	29
3.6	Pairwise cosine similarity of column embeddings: (a) <i>WTE</i> embeddings; (b) PYLON- <i>WTE</i> embeddings.	32
3.7	Comparison of query response time between including and excluding the format measure.	33
3.8	Comparison of precision and recall between D^3L instances and our best model PYLON- <i>WTE-NV</i> .	34
3.9	Comparison of query response time between D^3L instances and PYLON- <i>WTE-NV</i> .	34
3.10	Top-k precision and recall of 6 embedding measures on the PYLON dataset.	35
3.11	Indexing time and query response time on the PYLON dataset.	35
4.1	The interface of Sigma Workbooks connecting to a CDW. Consider a business user Joey who is interested in launching a sales campaign and identifying ideal customers. She first opens up the ACCOUNT table in their SALESFORCE database, which lists all customer companies. Next, she wants to augment the table with relevant columns to help make customer selections.	39
4.2	WARPGATE pipeline.	42
4.3	WARPGATE is integrated into a functionality called Add column via lookup in Sigma Workbooks.	44
4.4	Top- k precision and recall.	47
5.1	CDF of density and normalized rank of the curated real-world schemas.	55
5.2	Overview of NEXUS pipeline that infers join graphs from only metadata.	61
5.3	LLM prompts for column entity type inference and soft type matching.	68
5.4	F1 score of NEXUS-EM w.r.t the percentage of ground truth joins available (assuming the availability of query logs) on TPC-DS, TPC-H, and REAL datasets.	73

5.5	F1 score comparison of base models, NEXUS-Fast, and NEXUS-EM on TPC-DS, TPC-H, and REAL datasets.	74
5.6	Efficiency comparison on TPC-DS, TPC-H, and REAL datasets. Auto-BI shares Nexus’s implementations of metadata preparation, join candidates pruning and join candidates scoring.	75
5.7	Comparison of F1 score on TPC-DS and TPC-H when data values are available.	77
6.1	Illustration of metadata integration problem.	81
6.2	Illustration of the relationship transitivity (left) and inconsistent relationship assignments that violate the transitivity (right). Green solid edges indicate correct predictions and red dashed edges indicate conflicting predictions.	84
6.3	Overview of the proposed two-stage prior-posterior framework for integrating metadata concepts.	86
6.4	The plot on the left demonstrates an instance of our proposed MRF model containing six nodes/random variables and their dependencies; the plot on the right, known as a factor graph, visualizes the correspondence between factors and random variables in the MRF.	89
6.5	Creating independent MRFs for concept pairs in large datasets with sparse relationships. Ordered pairs are first grouped by the left concept and top- k neighbors (represented by the purple semi-circles) are retrieved for the left concept to construct a local MRF of random variables. Inference over independent MRFs is parallelized on available CPUs and posterior predictions of test pairs are collected.	95
6.6	Comparison of F1 score between prior models and OPENFORGE on (a) SOTAB, (b) Walmart-Amazon, (c) ICPSR-Detection and (d) ICPSR-Direction datasets. We omit gemma-2-lora and qwen2.5-lora on SOTAB due to insufficient training data.	102
6.7	MRF construction and inference time of three inference algorithms using a single CPU or GPU. WA, I-Det, and I-Dir are shorthands for Walmart-Amazon, ICPSR-Detection, and ICPSR-Direction datasets, respectively.	104
6.8	Scalability of MRF inference w.r.t. the number of concepts and connectivity k . $k = 12$ scales up to 4000 concepts (nearly 8 millions of random variables) and $k = 16$ scales up to 3000 concepts (about 4.5 millions of random variables). . . .	104
7.1	Overview of OBSERVATORY and how it solicits understanding of opaque table embedding models by measuring properties motivated by the relational data model and data distributions. We illustrate the framework for two out of eight properties: 1) row order insignificance, and 2) sample fidelity.	111
7.2	Illustration of row permutations.	115
7.3	Table with a functional dependency <code>country</code> $\rightarrow$ <code>continent</code> . The colors illustrate different FD groups determined by the unique values in the <code>country</code> column. .	118
7.4	A table (without header) comprising textual and non-textual data columns. . .	122
7.5	Cosine similarity and MCV distributions of column (top), row (middle), and table (bottom) embeddings from row shuffling. Across three levels of embeddings, table embedding models exhibit comparably lower cosine similarity while both language and table embedding models may exhibit high MCV.	127

7.6	PCA visualization of high-dimensional column embeddings from a table of six columns, for BERT and T5 . Each subplot draws $6!=720$ row-wise permutation variants of a column. While BERT embeddings are centered around the origin with some variation, the T5 embeddings are more stretched along the horizontal axis, resulting in the relatively high cosine similarity as well as high MCV value.	128
7.7	Cosine similarity and MCV distributions of column (top) and row (bottom) embeddings from column shuffling. Both column and row embeddings manifest similar patterns as in row shuffling.	130
7.8	PCA visualization of high-dimensional column embeddings from the same table as used in Figure 7.6. Each subplot draws $6!=720$ variants of a column from column order shuffling. The embeddings exhibit similar patterns as in row order shuffling but show larger spread across all columns.	131
7.9	Scatter plots of embedding cosine similarity vs. multiset Jaccard similarity derived from pairs of joinable columns in the NextiaJD XS dataset, which illustrate a positive correlation between the two measures.	133
7.10	Distributions of the group-wise variances over embedding translations across column pairs with and without the relationship of functional dependencies. None of the models show clear separation between the two variance distributions. . .	134
7.11	Distributions of sample fidelity of column embeddings under three sample ratios. Overall, vanilla LMs exhibit higher sample fidelity compared to table embedding models.	134
7.12	Pairwise top-10 entity stability with query entities from three distinct domains. Different pairs of models show high entity stability for different domains. . . .	135
7.13	Distributions of embedding cosine similarities between original columns and perturbed columns. Perturbations only to schemas cause relatively small changes in cosine similarity (except for TaBERT).	135

LIST OF TABLES

TABLE

3.1	Basic statistics of evaluation datasets.	26
3.2	Model training time (min / epoch) where each model is defined by the encoder choice and the training data construction strategy.	30
4.1	Basic statistics of evaluation datasets. XS is shorthand for testbedXS in the NextiaJD repository, similarly for S, M, and L.	46
4.2	End-to-end query response time in seconds/query (for $k = 10$) along with index lookup time reported in parentheses for WARP GATE.	48
5.1	Statistical summary of the curated schemas.	59
5.2	Summary of experimental dataset statistics.	71
5.3	Quality comparison of all approaches on TPC-DS, TPC-H and REAL datasets, using metadata only.	72
6.1	Example of the parameterized potential function ϕ for ternary cliques of binary random variables r_{ij} , r_{jk} and r_{ik} where 0 represents no relationship and 1 represents an equivalence relationship. The potential values $\{\theta_i\}_{i=1}^5$ are learnable parameters.	92
6.2	Summary of dataset statistics in the training, validation and test splits. All four datasets exhibit class imbalance.	98
6.3	Comparison of F1 score across methods and datasets.	101
7.1	Overview of table embedding models and their design specifications (Column is abbreviated to Col.).	123
7.2	Overview of datasets and models for each property.	125
7.3	Spearman coefficients between a value overlap measure and embedding cosine similarity on the NextiaJD-XS dataset. Multiset Jaccard is most positively correlated to embedding cosine similarity across all models. All coefficient numbers are statistically significant (p-value < 0.01).	132
7.4	Average group-wise variances of embedding translations over columns with and without functional dependencies across five models. Only TAPAS yields $\overline{S^2}_{FD} < \overline{S^2}_{-FD}$ with $\overline{S^2}_{FD}$ close to 0, while language models and other table embedding models do not follow this pattern.	132

7.5	Summary statistics (min, median, and max) of cosine similarities between single column embeddings and contextual embeddings for non-textual and textual data types, on the first and second row, respectively. Incorporating context, especially the entire table, can change column embeddings significantly w.r.t cosine similarity (highlighted in bold).	136
-----	--	-----

LIST OF ACRONYMS

AI Artificial Intelligence

FAIR Findable, Accessible, Interoperable, Reusable

ABSTRACT

The FAIR Data Principles, which emphasize **F**indability, **A**ccessibility, **I**nteroperability, and **R**eusability of scholarly data, have garnered substantial attention for their pivotal role in advancing open science. We consider these principles not only applicable but also critical for managing modern data stores such as data lakes. While the principles provide a valuable guide, operationalizing them remains a significant challenge due to a lack of concrete designs, especially as data volume and complexity continue to grow.

This dissertation addresses this gap by focusing on enhancing the findability and reusability of relational tables. We contribute a suite of novel techniques that leverage Artificial Intelligence (AI)—spanning representation learning, probabilistic models, and Large Language Models—to make data in large-scale stores more findable and reusable.

A primary barrier to data reuse is finding the right data. To address this findability challenge, we developed systems that discover semantically related tables for data integration and augmentation. Our framework began with PYLON, a system for semantic table union search using learned data representations, and its extension WARPGATE, which discovers joinable tables in cloud data warehouses. Addressing the need for discovery without requiring direct data access, a common constraint in enterprise environments, we then proposed NEXUS, a novel algorithm that infers complex join graphs using metadata alone. By leveraging low-rank matrix completion and Large Language Models, NEXUS automatically detects join relationships in data-restricted settings.

Beyond discovery, making data reusable also requires consistent metadata and a deep understanding of the AI models used in the process. To enhance metadata interoperability, a key component of reusing data, we introduce OPENFORGE, a probabilistic solution that automatically standardizes open-domain metadata. It induces a coherent taxonomy by resolving various relationships between metadata concepts, ensuring consistent usage over time. To facilitate the efficient reuse of the AI models themselves, we developed OBSERVATORY, a framework that systematically characterizes embedding representations of relational tables. OBSERVATORY provides crucial insights into the strengths and weaknesses of different table embedding models, enabling users to better anticipate model behaviors and select appropriate models for their downstream tasks.

CHAPTER 1

Introduction

The exponential proliferation of digital information has created a global datasphere of unprecedented scale. A report by International Data Corporation (IDC) [140] forecasts that the world will create and replicate 163 zettabytes of data by 2025. This is a trillion gigabytes and represents a staggering increase of more than ten-fold since 2016, when the total volume was 16.1 zettabytes. This growth is propelled by the expansion of the Internet of Things, the computational demands of artificial intelligence (AI) systems, and the ceaseless digital interactions of a globally connected populace. A defining characteristic of this data deluge is its composition; an estimated 80% to 90% of all data is unstructured (information that does not have a predefined format or organized model for direct machine processing and analysis), including text, images, videos, and sensor readings.

This explosion in data generation presents a profound paradox, which can be conceptualized as a vast data iceberg. The visible tip represents the small fraction of data that is actively managed, analyzed, and leveraged for decision-making. The immense, submerged bulk of this iceberg, however, constitutes “dark data”—the vast repositories of information that organizations collect and store but fail to utilize for analytical insight. Empirical studies quantify this disparity: a significant 68% of all data available to enterprises remains untapped, while a global survey by Splunk found that 55% of an organization’s data is “dark” while being considered very or extremely valuable to business success [153]. The problem is particularly acute in certain domains; for instance, IBM estimates that approximately 90% of data generated by sensors is never used [139]. An IDC 2012 study by highlighted this systemic underutilization, finding that while nearly a quarter of the global datasphere held potential analytical value, less than 1% was actually being analyzed [155].

This enormous, submerged mass of unused data underpins the contemporary data reusability challenge. It represents not only a significant unrealized asset but also a substantial cost in terms of storage and potential legal liability. Data remains “dark” not due to an intrinsic lack of value, but because of systemic barriers: it is frequently trapped in functional silos, is inadequately documented with metadata, exists in heterogeneous and

incompatible formats, or is simply unknown to potential users within an organization. To unlock the potential concealed within this digital iceberg, a fundamental paradigm shift in data stewardship is required—one that moves beyond mere collection and storage to a strategic focus on making data findable, accessible, interoperable, and, ultimately, reusable.

The FAIR data principles—a conceptual framework designed to make data **F**indable, **A**ccessible, **I**nteroperable, and **R**eusable—offer a strategic and globally recognized response to this crisis. Established in 2016, FAIR is not a rigid standard but a set of high-level principles that emphasize machine-actionability, aiming to create a data ecosystem where computational systems can autonomously discover, evaluate, and integrate data with minimal human intervention [171]. The principles are initially proposed to promote the use of persistent identifiers, rich metadata, standardized protocols, shared vocabularies, and clear usage licenses for sharing scholarly data and advancing open science. Within the open science ecosystem, FAIR is becoming a cornerstone of policy, driven by funder mandates from institutes like the NIH [125] and the European Commission. Such top-down mandates have become one of the most significant drivers of FAIR adoption in the academic world. A key mechanism for enforcing this is the requirement for researchers to submit detailed Data Management Plans as part of their grant proposals, which must explicitly outline the strategies for making their data FAIR.

While the FAIR principles originated in the academic research community, their applicability extends far beyond it. In the corporate world, FAIR is rapidly being recognized not as a matter of compliance or open-access ideology, but as a core component of modern data strategy and a critical enabler of digital transformation (Companies such as Splunk¹ and TileDB² are leaders on this front). The enterprise value proposition for FAIR is rooted in its ability to convert chaotic, siloed data into a coherent, reusable, and machine-actionable asset that drives efficiency, innovation, and competitive advantage. The business case is even more compelling in the AI era, especially due to the rise of Large Language Models (LLMs) that are trained on the whole internet of data while seeking more. FAIR data is “AI-ready,” capable of fueling AI models. The ultimate “reuser” of FAIR data is increasingly not a human analyst, but an automated process or an AI algorithm.

In this dissertation, we propose algorithms that make data, particularly tabular data, more findable and reusable in both open data stores and enterprise environments. We leverage AI techniques including representation learning, probabilistic models and LLMs, which are key enablers of our designs. Below we discuss specific challenges we address for reusing data in Section 1.1 and give an overview of our contributions in Section 1.2.

¹<https://www.splunk.com>

²<https://www.tiledb.com>

1.1 Technical Challenges

1.1.1 Discoverability Issues

A primary and foundational barrier to data reuse is that a significant volume of data is simply not discoverable by standard search mechanisms like keyword search. Before any assessment of quality or usability can occur, a potential reuser must first be aware that a dataset exists. However, a substantial portion of data remains isolated on individuals' hard drives, local servers, or within institutional file systems that are not indexed for search. Even when data is deposited into a repository, discoverability is frequently compromised by poor data quality and documentation.

Search engines, whether for general web content or specialized data catalogs, rely on rich, machine-readable metadata to index resources and determine their relevance to a query. When metadata is not detailed, incomplete, or uses non-standard terminology, it severely limits the ability of both human users and automated systems to find the data and assess its utility. Furthermore, the requirements of modern scientific inquiry often transcend the capabilities of simple keyword-based searches. Researchers increasingly need to discover data based on complex conceptual relationships, a process known as semantic search. This requires a more sophisticated approach to metadata, one that utilizes standardized, controlled vocabularies and ontologies that are shared within and across disciplines. Such semantically enriched metadata allows a search engine to understand the meaning behind the terms, enabling it to find datasets based on related concepts rather than exact keyword matches.

1.1.2 Data Heterogeneity and Lack of Standardization

Data is rarely born interoperable and reusable. It is created using a vast array of instruments, software, and methodologies, resulting in a chaotic landscape of heterogeneous data formats, inconsistent terminologies, and non-standardized data structures. Many of these formats are proprietary, locking data into specific software ecosystems and creating immediate barriers to access for those without the requisite tools. This lack of a common language makes integrating data from different sources a labor-intensive and error-prone process of data wrangling and transformation.

Beyond the technical issues, it is worth noting that a powerful set of social and cultural norms in both academia and enterprise actively discourages data sharing and reuse including but not limited to sociocultural inertia, legal and ethical complexity, and economic and resource constraints. These barriers complicate the data reuse landscape while they are outside the scope of this dissertation.

1.2 Contributions Overview

The overarching contribution of this dissertation is a suite of AI-driven techniques that automate critical “sense-making” tasks, such as data discovery and metadata standardization, which are essential for data usability but currently require manual, non-scalable human effort. By automating these bottlenecks, this work significantly enhances the machine actionability of data, paving the way for systems that can autonomously discover, integrate, and utilize data with minimal human intervention—a core goal of the FAIR principles.

To support reproducibility and encourage future research, the non-proprietary research artifacts from this dissertation have been made open-source.

Pylon (Chapter 3) The large size and fast growth of data repositories, such as data lakes, has spurred the need for data discovery to help analysts find related data. The problem has become challenging as (i) a user typically does not know what datasets exist in an enormous data repository; and (ii) there is usually a lack of a unified data model to capture the interrelationships between heterogeneous datasets from disparate sources. In this work, we address one important class of discovery needs: finding union-able tables.

The task is to find tables in a data lake that can be unioned with a given query table. The challenge is to recognize union-able columns even if they are represented differently. In this paper, we propose a data-driven learning approach: specifically, an unsupervised representation learning and embedding retrieval task. Our key idea is to exploit self-supervised contrastive learning to learn an embedding model that takes into account the indexing/search data structure and produces embeddings close by for columns with semantically similar values while pushing apart columns with semantically dissimilar values. We then find union-able tables based on similarities between their constituent columns in embedding space.

WarpGate (Chapter 4) Data discovery is a major challenge in enterprise data analysis: users often struggle to find data relevant to their analysis goals or even to navigate through data across data sources, each of which may easily contain thousands of tables. One common user need is to discover tables joinable with a given table. This need is particularly critical because join is a ubiquitous operation in data analysis, and join paths are mostly obscure to users, especially across databases. Furthermore, users are typically interested in finding “semantically” joinable tables: with columns that can be transformed to become joinable even if they are not joinable as currently represented in the data store.

We present WARP GATE, a system prototype for data discovery over cloud data warehouses. WARP GATE implements an embedding-based solution to semantic join discovery, which encodes columns into high-dimensional vector space such that joinable columns map to points that are near each other. Through experiments on several table corpora, we show

that WARP GATE (i) captures semantic relationships between tables, especially those across databases, and (ii) is sample efficient and thus scalable to very large tables of millions of rows. We also showcase an application of WARP GATE within an enterprise product for cloud data analytics.

Nexus (Chapter 5) Automatically inferring join relationships is a critical task for effective data discovery, integration, querying and reuse. However, accurately and efficiently identifying these relationships in large and complex schemas can be challenging, especially in enterprise settings where access to data values is constrained. In this paper, we introduce the problem of join graph inference when only metadata is available. We conduct an empirical study on a large number of real-world schemas and observe that join graphs when represented as adjacency matrices exhibit two key properties: high sparsity and low-rank structure. Based on these novel observations, we formulate join graph inference as a low-rank matrix completion problem and propose NEXUS, an end-to-end solution using only metadata. To further enhance accuracy, we propose a novel Expectation-Maximization algorithm that alternates between low-rank matrix completion and refining join candidate probabilities by leveraging Large Language Models.

OpenForge (Chapter 6) Modern data stores increasingly rely on metadata to enable diverse activities such as data cataloging and search. However, metadata curation remains a labor-intensive task, and the broader challenge of metadata maintenance—ensuring its consistency and usefulness—has been largely overlooked. In this work, we tackle the problem of resolving relationships among metadata concepts from disparate sources. Inferring these relationships are critical for creating clean and consistent metadata repositories, and a central challenge for metadata integration.

We propose OPENFORGE, a two-stage prior-posterior framework for metadata integration. In the first stage, OPENFORGE exploits multiple methods including fine-tuned large language models to obtain prior beliefs about concept relationships. In the second stage, OPENFORGE refines these predictions using the Markov Random Field, a probabilistic graphical model. We formalize metadata integration as an optimization problem, where the objective is to identify the relationship assignments that maximize the joint probability of assignments. The MRF formulation allows OPENFORGE to capture prior beliefs while encoding critical relationship properties, such as transitivity, in probabilistic inference.

Observatory (Chapter 7) Language models and specialized table embedding models have recently demonstrated strong performance on many tasks over tabular data. Researchers and practitioners are keen to leverage these models in many new application contexts; but limited understanding of the strengths and weaknesses of these models, and the table representations they generate, makes the process of finding a suitable model for a given

task reliant on trial and error. There is an urgent need to gain a comprehensive understanding of these models to minimize inefficiency and failures in downstream usage.

To address this need, we propose OBSERVATORY, a formal framework to systematically analyze embedding representations of relational tables. Motivated both by invariants of the relational data model and by statistical considerations regarding data distributions, we define eight primitive properties, and corresponding measures to quantitatively characterize table embeddings for these properties. Based on these properties, we define an extensible framework to evaluate language and table embedding models. We collect and synthesize a suite of datasets and use OBSERVATORY to analyze nine such models. Our analysis provides insights into the strengths and weaknesses of learned representations over tables. We find, for example, that some models are sensitive to table structure such as column order, that functional dependencies are rarely reflected in embeddings, and that specialized table embedding models have relatively lower sample fidelity. Such insights help researchers and practitioners better anticipate model behaviors and select appropriate models for their downstream tasks, while guiding researchers in the development of new models.

CHAPTER 2

Background

Data often becomes “dark” or underutilized for a myriad of reasons, including but not limited to being unaware of its existence, being trapped in organizational silos or suffering incomplete data integration [139]. Such issues, in combination with the rising demand for advanced analytics and machine learning on unstructured data, have promoted the adoption of data lakes. Built upon distributed file systems like HDFS [150] or cloud object stores like Amazon S3, data lakes offer a low-cost, highly scalable solution for storing vast quantities of structured, semi-structured, and unstructured data in its raw format. Ideally, data lakes are envisioned to provide a single, unified source of truth for diverse use cases.

Despite the flexibility of data lakes, many promises of data lakes have not been realized. Many organizations find their data lakes degrading into “data swamps” due to persistent issues with data quality, slow query performance, and inadequate data governance. Consequently, it remains challenging for users, from data scientists to analysts, to efficiently find, understand, and reuse their organization’s valuable data assets.

2.1 Data Discovery

Data lakes, viewed simply, are massive collections of files. Much of the academic effort to rescue these data lakes from becoming swamps falls under the umbrella of data discovery: the process of identifying and retrieving relevant datasets or files to satisfy a user’s information need. Due to the variety of these information needs and the ambiguity inherent in expressing them, researchers have investigated numerous tasks targeting different forms of user queries [116, 32, 185, 64], including keyword-based search [25, 74, 184], table-based search [144, 124, 60, 191, 187, 17, 43, 91, 54, 42], and natural language queries [177, 76, 167, 86].

Given the ubiquitous existence of tabular data, table-based search has been a long-standing problem attracting significant interest from the data management, information

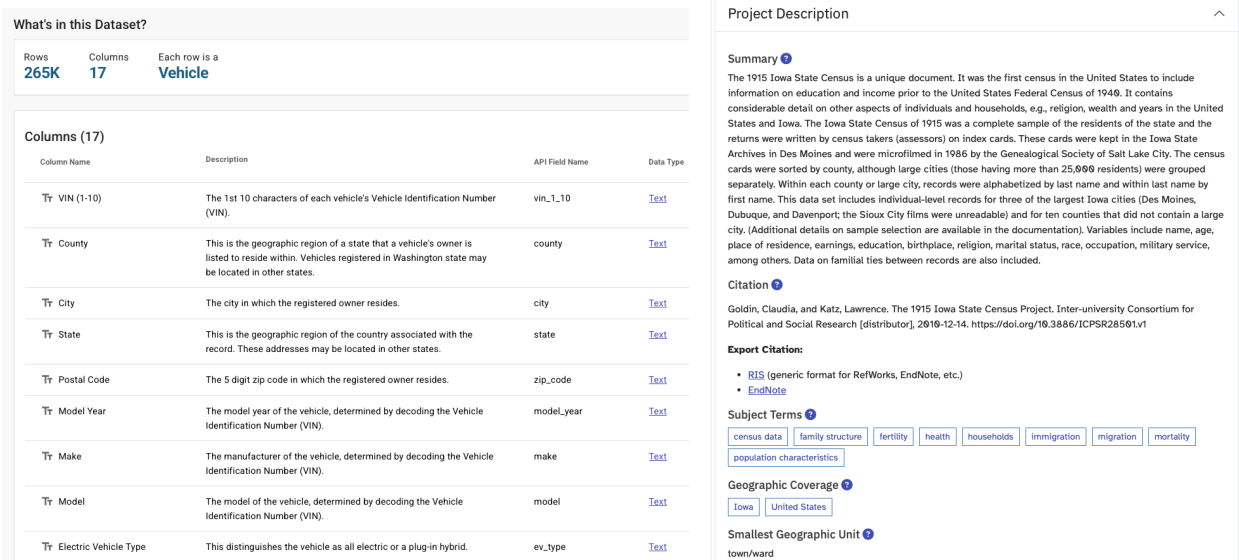


Figure 2.1: Examples of dataset metadata from two dataset hosting platforms: data.gov (left) and ICPSR (right).

retrieval, and natural language processing communities. Rooted from the semantic web era, which focused on direct matching of schema or table content [25, 174, 144]. table-based search has evolved considerably for table retrieval, data augmentation and repository exploratory purposes with more semantic and scalability requirements. In Chapter 3 and Chapter 4 of this dissertation, we contribute to enhancing the findability and reusability of tabular data in data lakes. We focus on two primary tasks of table-based search: unionable table search and joinable table search. The goal of unionable table search is to find relevant tables that can be unioned (vertically combined) with a query table, thereby augmenting it with more tuples. Similarly, the goal of joinable table search aims to find tables that can be joined (horizontally combined) with a query table to augment it with more attributes. Such tasks are critical for data integration pipelines and as a preparation step for machine learning applications. To achieve relevant and scalable search, datasets or tables are commonly transformed into one or multiple types of more compact representations, such as statistical profiles, hashes, and embeddings and indexed in a data structure for fast retrieval. Readers can refer to Section 3.5 and Section 4.6 for a detailed overview of related work.

It is also worth noting that data discovery is not a purely technical problem as many have pointed out [69, 80, 59]. Social factors—including user practices, how individuals make sense of data, undocumented domain knowledge—profoundly influence the ultimate success and adoption of any data discovery system.

2.2 Metadata

Major dataset hosting platforms, such as Google Dataset Search [74], the United States Government’s open data site [162] and ICPSR (maintaining the world’s largest social and political data archive) [82], employ a keyword-based search model and rely heavily on metadata to help users find and reuse datasets. As shown in Figure 2.1, common metadata includes dataset identifiers, descriptions, and annotations at various granularities, from the dataset level (e.g., subject terms, geographic coverage) to the column level (e.g, column descriptions, data types). When a user submits a query, these systems attempt to syntactically match query terms against the metadata fields.

This query model, despite simple and straightforward, suffers from three significant limitations. First, keywords cannot fully express the user’s intent. Second, the model’s effectiveness is entirely dependent on the existence of comprehensive, high-quality metadata catalogs. Third, the lack of semantic matching means a semantically relevant dataset can be easily missed if its metadata does not contain the exact query keywords. Additionally, curating high-quality metadata requires thorough examination and processing of human experts with domain knowledge [163]. Given the sheer volume of deposited datasets, it is infeasible for curators to go through each dataset and annotate every metadata field. Consequently, many datasets are deposited “as-is” with only a few required metadata fields filled out by dataset depositors, severely limiting their findability and reusability.

As metadata may be missing, inconsistent, or incomplete, a large body of recent work in data discovery has proposed data-driven approaches. These methods extract and leverage signals (e.g., statistical profiles, hashes and embeddings) directly from the data content itself [124, 191, 187, 43, 54, 91]. While data-driven approaches can often capture richer signals and demonstrate promising result quality for specific tasks, we emphasize several factors that prevent their widespread adoption in real-world data search systems: (1) Data-driven approaches are often computational heavy, requiring at least one full data pass to compute their signals, which is costly to compute and maintain for large repositories. (2) They lack the mature system support that full-text search engines like Elasticsearch ¹ or traditional relational databases like PostgreSQL ² provide for keyword search. (3) Abandoning existing catalogs is impractical, as metadata-based search remains an intuitive and useful paradigm for general-purpose use. (4) Many scenarios, particularly in enterprise settings, have strict governance and privacy policies that constrain or forbid direct access to raw data values.

While the development and promotion of novel, data-driven discovery systems continues,

¹<https://www.elastic.co/elasticsearch>

²<https://www.postgresql.org/docs/current/textsearch.html>

comparatively little work has investigated how to mitigate metadata “pain points” (e.g., inconsistency) for existing systems, or how much value can be unlocked by leveraging existing metadata. This dissertation attempts to fill that gap. In Chapter 6, we contribute an approach that integrates metadata and improve its consistency by resolving various relationships between metadata concepts from disparate sources. Ensuring metadata consistency is critical for interconnecting siloed datasets that are semantically relevant but annotated with different vocabularies or catalogs. In Chapter 5, we further investigate how to recover join graphs of relational tables in enterprise data lakes using only metadata. This problem of inferring join relationships between relational tables is typically approached with methods that assume direct access to the data. We demonstrate that by leveraging metadata commonly collected by data services for query optimization, along with the structural properties of these join graphs, we can achieve comparable or even better result quality compared to approaches that rely on accessing data values.

2.3 Representation Learning

Learning representations of raw data that make it easier to extract useful information when building classifiers or other predictors has been a long-standing and foundational topic in the machine learning community [14]. The advance of representation learning has profoundly impacted data management research, particularly in the data discovery domain that is central to this dissertation, from table understanding [79, 181, 157] to table-based search [60, 43, 54], from task-specific models to multi-task models [47, 161, 164].

The primary artifacts of representation learning are embeddings—dense vector representations of raw data. In the learned vector space, proximity between embeddings should ideally approximate some similarity notion determined by task semantics, enabling efficient pairwise comparisons or cluster-based analysis of the original objects. For tabular data, embeddings can be generated at multiple granularities, representing an individual cell, a row, a column, or even an entire table. These embeddings, once trained or fine-tuned, facilitate a wide range of downstream tasks. These include classification tasks like entity matching [117, 103] and column type annotation [79, 181, 157]; table question answering [177, 76, 106]; and retrieval tasks such as unionable table search [124, 43] and joinable table search [60, 42, 54].

Depending on the availability of labeled data, the choice of learning techniques falls broadly into supervised or unsupervised learning. A particularly relevant sub-category of unsupervised learning is self-supervised learning, which has received significant interest. The learning models are trained on unlabeled data by creating supervisory signals from the data itself [49, 34, 12], thereby bypassing the need for expensive, manual labeling. This approach is

exceptionally valuable in the data lake context, which is often characterized by vast quantities of unlabeled data.

This dissertation engages with representation learning on multiple aspects. In Chapter 3, we introduce a novel self-supervised learning technique to learn column representations specifically for unionable table search, a method that is also applicable to joinable table search. In Chapter 6, we leverage supervised learning by fine-tuning billion-parameter language models with labeled data for metadata relationship classification. In Chapter 7, we contribute a critical analysis of relational table embeddings, providing new insights into the strengths and weaknesses of learned representations for tabular data.

2.4 Large Language Models

Representing a new apex in representation learning, Large Language Models (LLMs) are AI models trained on a vast amount of text data for natural language processing tasks, e.g., understanding, generating and translating human language. Recent LLMs, consisting of billions to trillions of parameters, exhibit emergent capabilities for generating, summarizing, translating, and reasoning over text and other modalities [126, 6]. The rise of LLMs, driven by their out-of-the-box general-purpose capabilities, has begun to reshape numerous research fields, including data management.

Historically, finding and reusing data has been impeded by significant challenges, primarily context ambiguity and the lack of a true general-purpose model. Data values are often ambiguous, especially when crucial metadata is missing. This issue is pervasive in data integration tasks like entity matching—a long-standing problem that seeks to determine if two entity mentions refer to the same real-world entity. Prior efforts to capture semantics and mitigate ambiguity have relied on training task-specific models with costly, manually-labeled data [117, 103, 47, 157, 54, 164]. A fundamental limitation of this approach is that these specialized models lack the broad transferability and general-world knowledge inherent to LLMs.

In this dissertation, we made early-stage attempts of applying LLMs to tackle these challenges. In Chapter 6, we use LLMs—through both prompting and efficient fine-tuning with small amounts of labeled data—to provide prior information within a probabilistic framework for resolving relationships between metadata concepts. Similarly, in Chapter 5, we employ LLMs to generate prior predictions for recovering relational join graphs. Furthermore, we utilize LLMs to perform open-domain semantic column type annotation and equivalence checking. This constitutes a critical departure from previous work, which typically relies on fixed vocabularies to annotate the semantic types of columns [79, 181, 157]. Prior ap-

proaches cannot generalize to the diverse and evolving semantics of data lakes, as any fixed vocabulary lacks the comprehensiveness required for such an environment. LLMs, due to their strong semantic understanding capacity, can effectively mitigate this limitation, acting as true open-domain predictors.

CHAPTER 3

Pylon: Semantic Table Union Search in Data Lakes

3.1 Introduction

Recent years have witnessed a vast growth in the amount of data available to the public, particularly from data markets, open data portals, and data communities (e.g., Wikidata and Kaggle) [32]. To benefit from the many new opportunities for data analytics and data science, the user first usually has to find related datasets in a large repository (e.g., data lake). Common practice in production is to provide a keyword search interface over the metadata of datasets [23] but users often have discovery needs that cannot be precisely expressed by keywords. The challenge for a system is to support users with varying discovery needs, without the help of a unified data model capturing the interrelationships between datasets.

In response to the challenge, there are many ongoing efforts under the umbrella of data discovery. One task of interest in data discovery is to find unionable tables [26, 144, 124, 17] with the aim of adding additional relevant rows to a user-provided table. Figure 3.1 shows an example of two tables unionable over four pairs of attributes. In general, the literature considers two tables unionable if they share attributes from the same domain and assumes the unionability of two attributes can be implied by some notion of similarity. We refer to the problem of finding unionable tables as table union search (as termed in [124]) in the rest of the paper.

The typical solution path is to first identify unionable attributes (or columns in the tables) with the aid of an indexing/search data structure (e.g., locality sensitive hashing) and then aggregate column-level results to obtain candidate unionable tables. To uncover the unionability of attributes, both syntactic and semantic methods have been exploited. Syntactic methods are the easiest, and have been used the longest. While they are robust at catching small changes, such as capitalization or the use of a hyphen, they are unable to address even the use of common synonyms. Semantic methods offer the possibility of finding

id	title	authors	venue	year
671167	A Database System for Real-Time Event Aggregat...	Jerry Baulier, Stephen Blott, Henry F. Korth, ...	Very Large Data Bases	1998
672964	Integrating a Structured-Text Retrieval System...	Tak W. Yan, Jurgen Annevelink	Very Large Data Bases	1994
872823	Evaluating probabilistic queries over imprecis...	Reynold Cheng, Dmitri V. Kalashnikov, Sunil Pr...	International Conference on Management of Data	2003
...	...	...	...	...

Title	Authors	Platform	Cited_url	Cited_count	Year
Fg-index: towards verification-free query proc...	J Cheng, Y Ke, W Ng, A Lu	Proceedings of the 2007 ACM SIGMOD internat...	https://scholar.google.com/scholar?oi=bibs&hl=...	286	2007
Efficient query processing on graph databases	J Cheng, Y Ke, W Ng	ACM Transactions on Database Systems (TODS) 34...	https://scholar.google.com/scholar?oi=bibs&hl=...	83	2009
Context-aware object connection discovery in L...	J Cheng, Y Ke, W Ng, JX Yu	2009 IEEE 25th International Conference on Dat...	https://scholar.google.com/scholar?oi=bibs&hl=...	66	2009
...	...	...	...	...	...

Figure 3.1: An example of two tables unionable over four pairs of attributes: title - Title, authors - Authors, venue - Platform, and year - Year.

unionable columns of semantically similar values despite their syntactic dissimilarity (e.g., “venue” column and “platform” column in figure 3.1). Sarma et al. [144] and Nargesian et al. [124] link cell values to entity classes in an external ontology and compare similarity of entity sets. Nargesian et al. [124] and Bogatu et al. [17] use off-the-shelf word embeddings to measure semantics. Both methods have notable limitations. Nargesian et al. [124] observed that only 13% of attribute values of their collected Open Data tables can be mapped to entities in YAGO [156], one of the largest and publicly available ontologies. Although word embeddings can provide more semantic coverage of attributes, they are subject to the training text corpus and may not generalize well to textual data in tables [72, 98]. Recent advance in tabular embeddings [47, 161] may mitigate the issue, however, we argue that these models in training do not take into account the indexing/search data structure, which is a core component of any efficient solution to table union search. In particular, [124, 17] rely on the correlation between column unionability and cosine similarity of their embeddings. Nevertheless, the embeddings they use and recent more advanced tabular models are not optimized in training for approximate cosine similarity search so the performance of indexing/search data structure is suboptimal.

Instead of relying on low-coverage ontologies or pre-trained embeddings that are unaware of required data structure in table union search, we propose a data-driven learning approach to capture data semantics and model characteristics of the essential indexing/search data structure. We also argue that the popular classification formulation in the literature [47, 103, 161] is less feasible for table union search. On one hand, there is no large-scale labeled dataset for table union search. The only publicly available benchmark [124] with table- and column-level ground truth contains a limited number of tables synthesized from only 32 base tables, which is far from being enough for training purposes. On the other hand, even if the training data problem were resolved, we would only be able to determine column matches pairwise. Unlike tasks (e.g., semantic column type annotation and entity matching) that can be formulated as a classification problem, finding unionable tables is a search problem.

It would be very inefficient to exhaustively consider every query column and every column in the data lake pairwise to predict unionability. In short, the inherent search nature of the problem makes the supervised classification formulation infeasible and it is important to jointly consider representation learning and the indexing/search data structure.

In this work, we overcome the aforementioned difficulties by casting table union search as an unsupervised representation learning and embedding retrieval task. Our goal is to learn column-level embeddings into a high-dimensional feature space that also models characteristics of the indexing/search data structure. Locality search in this feature space can then directly be used for unionable table search. To achieve this goal, our key idea is to exploit self-supervised contrastive learning to learn an embedding model that produces embeddings with high similarity measure (which is used in indexing/search data structure) for columns with semantically similar values and pushes away columns with semantically dissimilar values. We propose PYLON, a novel self-supervised contrastive learning framework that learns column representations and serves table union search without relying on labeled data.

There are two main challenges in the development of PYLON:

1. How to learn embeddings that capture both data semantics and model characteristics of indexing/search data structure? Existing embedding models may capture semantics from a large training corpus but they are ignorant of additional data structure necessary in table union search. In other words, the potential of locality search is not fully realized with existing embeddings.
2. How to create training data without human labeling? The self-supervised contrastive learning technique requires constructing positive and negative examples from data themselves. In the field of computer vision where contrastive learning first took off, [34] applies a series of random data augmentation of crop, flip, color jitter, and grayscale to generate stochastic views of an image. These views preserve the semantic class label of the image and so make positive examples for training. They further consider any two views not from the same image as a negative example. However, the tabular data modality is dramatically different from images and it remains unclear how to create different views of tables while keeping the semantics.

In summary, we make the following contributions:

- We formulate semantic table union search as an unsupervised representation learning and embedding retrieval problem, and propose to use self-supervised contrastive learning to model both data semantics and indexing/search data structure.

- We present PYLON, a contrastive learning framework for learning semantic column representations from large collections of tables without using labels. In particular, we propose two simple yet effective strategies to construct training data in an unsupervised manner.
- We empirically show that our approach is both more effective and efficient than existing embedding methods on a real-world data lake of GitHub tables and a synthetic public benchmark of open data tables. On the GitHub data lake, two of our model variants outperform their corresponding baseline version by 14% and 6% respectively on both precision and recall. We also observe that they speed up the query response time by 2.7x and 9x respectively. We (plan to) open-source our new benchmark of GitHub tables for future research study.
- We demonstrate that our embedding approach can be further augmented by syntactic measures and that our best ensemble model has significant advantages over the state-of-the-art (namely, D^3L [17]), more than 15% improvement in precision and recall, and 7x faster in query response time.

We give a formal problem setup and background about embedding models in Section 3.2. We describe our framework PYLON including embedding training and search in Section 3.3. Section 3.4 reports experiments that validate our approach. We discuss related work in Section 3.5 and conclude in Section 3.6.

3.2 Problem Definition & Background

In this section, we start by describing the formal problem setup in 3.2.1, and then provide an overview of how table union search is different from a well-established data management problem, namely schema matching, and a related problem of join discovery in 3.2.2. We finally elaborate on the challenges of applying representation learning for table union search in 3.2.3.

3.2.1 Table Union Search

The table union search problem [124] is motivated by the need to augment a (target) table at hand with additional data from other tables containing similar information. For example, starting with a table about traffic accidents in one state for a particular year, an analyst may wish to find similar traffic accident data for other states and years. Ideally, these tables would have the same schema (e.g., data from the same state agency for two different years)

so that we could simply union the row-sets. However, this is typically not the case for data recorded independently (e.g., data from different states). We consider two tables unionable if they share attributes from the same domain. Also, as in prior work on this topic, we assume the unionability of attributes can be quantified by some notion of similarity.

Definition 1 (Attribute Unionability). Given two attributes A and B , the attribute unionability $\mathcal{U}_{attr}(A, B)$ is defined as

$$\mathcal{U}_{attr}(A, B) = \mathcal{M}(\mathcal{T}(A), \mathcal{T}(B))$$

where $\mathcal{T}(\cdot)$ is a feature extraction technique that transforms raw columns (attribute names, attribute values, or both) to a feature space and $\mathcal{M}(\cdot, \cdot)$ is a similarity measure between two instances in the feature space.

With the definition of attribute unionability, we can define table unionability as a bipartite graph matching problem where the disjoint sets of vertices are attributes of the target table and the source table respectively, and edges can be defined by attribute unionability. In this paper, we restrict ourselves to the class of greedy solutions. Therefore, we formalize the definition table unionability as a greedy matching problem as follows:

Definition 2 (Unionable Tables). A source table S with attributes $\mathcal{B} = \{B_j\}_{j=1}^n$ is unionable to a target table T with attributes $\mathcal{A} = \{A_i\}_{i=1}^m$ if there exists a one-to-one mapping $g : \mathcal{A}' (\neq \emptyset) \subseteq \mathcal{A} \rightarrow \mathcal{B}' \subseteq \mathcal{B}$ such that

1. $|\mathcal{A}'| = |\mathcal{B}'|$;
2. $\forall A_i \in \mathcal{A}', \mathcal{U}_{attr}(A_i, g(A_i)) \geq \tau$ where

$$g(A_i) = \arg \max_{B_j} \{\mathcal{U}_{attr}(A_i, B_j) : 1 \leq j \leq n\}$$

and τ is a pre-defined similarity threshold.

Definition 3 (Table unionability). Following notations in Definition 2, the table unionability $\mathcal{U}(S, T)$ is defined as

$$\mathcal{U}(S, T) = \frac{\sum_{i=1}^l w_i \cdot \mathcal{U}_{attr}(A_i, g(A_i))}{\sum_{i=1}^l w_i}$$

where l is the number of unionable attribute pairs between the target table T and a source table S , and w_i weights the contribution of the attribute pair $(A_i, g(A_i))$ to the table unionability.

Considering the scale of the dataset repository, we also follow the common practice [144, 124, 17] of performing top- k search. The table union search problem is formally defined as below.

Definition 4 (Top- k Table Union Search). Given a table corpus $\mathcal{S}$, a target table T , and a constant k , find up to k candidate tables $S_1, S_2, \dots, S_k \in \mathcal{S}$ in descending order of table unionability with respect to the query table T such that $S_1, S_2, \dots, S_k$ are most likely to be unionable with T .

3.2.2 How Table Union Search Differs from Schema Matching and Join Discovery?

Despite of overlapping elements, we emphasize the complexity of table union search over related problems of schema matching and join discovery. As a long-standing and well-studied problem in data integration, schema matching takes a pair of schemas as input and returns a mapping between columns of two schemas that semantically correspond to each other [138]. Conceptually, table union search can be viewed as an extreme extension of schema matching. Instead of having two schemas as inputs, table union search only has one while having to search another (partially) matching schema in a large corpus (e.g., data lakes) in the first place. Essentially, aside from the matching component, table union search needs to address the additional problem of identifying matching candidates among many non-matching ones, which is a significantly more challenging setup. Similarly, fuzzy join [36, 160] assumes a restrictive setup with a pair of input datasets. In their experiments, the second dataset in the pair is usually a syntactically perturbed variant of the first dataset and thus cannot mimic the complexity of data lakes consisting of heterogeneous datasets across domains.

A problem more related to table union search is join discovery [60, 191, 17], which targets at finding joinable tables in data lakes. Both are search problems, nevertheless, table union search needs to ideally identify a matching between blocks of columns in two tables as opposed to identifying a pair of join keys between two tables in join discovery. This difference poses higher demand for embedding quality in table union search because moderately high similarity between every pair of columns makes it harder to match unionable pairs.

3.2.3 General Challenges

Representation learning for tables has achieved excellent results for many table-centric tasks. We hypothesize that the table union search problem can also benefit from advances in table modeling. However, several challenges remain to be addressed.

1. To the best of our knowledge, no prior work has taken the learning approach for table union search. We argue that this is mainly because neither the supervised learning setting nor the popular pre-training and fine-tuning paradigm is directly applicable for the problem. It is inefficient to formulate the underlying search of unionable columns as a classification problem. In a supervised learning setting, one can attempt to train a classifier to predict whether two columns are unionable, but it will quickly become computationally prohibitive in the search phase to classify every pair of target column in a query table with every column in a large corpus.
2. The scarcity of table union search datasets is another severe bottleneck of applying a learning approach and studying the problem in general. The only publicly available benchmark [124] with table- and column-level ground truth is synthesized from only 32 base tables, which is barely enough for evaluation. It is also very laborious and time consuming to label such datasets, as curators need to examine every pair of columns for every pair of tables in a collection.
3. Efficient solutions to table union search involve two stages: profiling (e.g., embed columns into a feature space) and index-based search. Taking off-the-shelf embedding models or training a new model without considering the indexing/search data structure indispensable in table union search is suboptimal. We argue that aligning representation learning with indexing/search data structure can further improve effectiveness and efficiency of a solution to table union search.

In the next section, we present our design that contributes a representation learning approach to table union search while effectively addressing the challenges we point out here.

3.3 Pylon: A Self-Supervised Contrastive Learning Framework for Tabular Data

Our key idea is to exploit self-supervised contrastive learning that can provide a feasible training objective for learning effective column representations for the table union search problem while not requiring any labeled data and taking into account the indexing/search data structure (corresponding to challenge 1 and 3). Within the framework of contrastive learning, we propose two strategies that arithmetically construct training data from unlabeled data to tackle challenge 2.

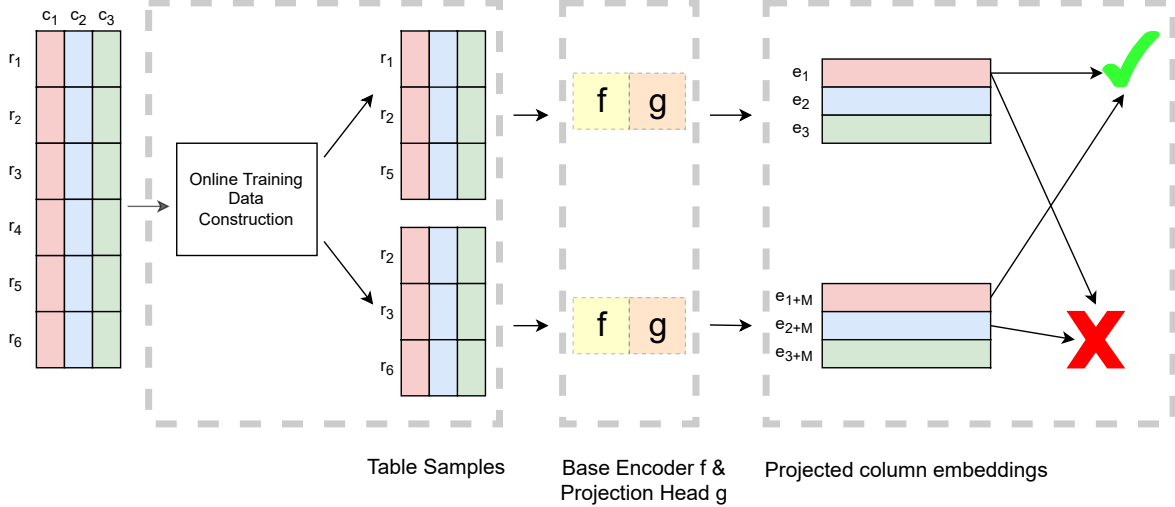


Figure 3.2: Training workflow of PYLON (with online training data construction).

3.3.1 Contrastive Learning

The high-level goal of contrastive learning is to learn to distinguish (so called "contrast") between pairs of similar and dissimilar instances. Ideally, in the learned representation space, similar instances stay close to each other whereas dissimilar ones are pushed far away. A pair of instances is considered similar and labeled a positive example in training if it comprises different views of the same object; otherwise, they are considered dissimilar and make a negative example. Contrastive learning has been used extensively in computer vision [34], where a positive example consists of a pair of augmented images transformed from the same image (e.g., by applying cropping or color distortion).

We introduce, PYLON, our self-supervised contrastive learning framework for table union search. As table union search begins by finding unionable columns, PYLON is designed to generate a vector representation for each column of input tables where columns containing semantically similar values have embeddings closer to one another.

3.3.2 Pylon Workflow

Figure 3.2 shows the training workflow of the framework that consists of the following major components.

Training Data Construction. Without labeled data, the success of contrastive learning hinges on the construction of positive and negative examples from data themselves. To make positive examples, it requires an operation to transform a data instance in a way that introduces variations while preserving the semantics. As table union search builds

on unionable column search, we propose two strategies to construct positive and negative examples at the column level.

1. *Online sampling strategy.* Consider a training batch of N tables $\{T_i\}_{i=1}^N$ where each table T_i has m_i columns $\{C_j^i\}_{j=1}^{m_i}$, giving $M = \sum_{i=1}^N m_i$ columns in total. We obtain a positive example of column pairs (x_k, x_{k+M}) ($1 \leq k \leq M$) by randomly sampling values from each column C_j^i of each table T_i . Since both x_k and x_{k+M} are samples from the same column, we consider they share semantics and make a positive example. The sampling process yields $2M$ column instances, and we treat the other $2(M-1)$ samples as negatives with respect to x_k . In other words, considering (x_k, x_{k+M}) and (x_{k+M}, x_k) as distinct positive examples, we construct $2M$ positive examples and $2M(M-1)$ negative examples from each training batch.
2. *Offline approximate matching strategy.* An alternative is to construct positive examples ahead of training. Instead of relying on ad-hoc sampling, we can leverage existing approaches to find a unionable candidate for each column, which in turn makes positive examples in training. Based on the observation that top- k unionable column search of existing techniques is reasonably precise when $k = 1$, we are able to use this approximate matching without human involvement. We find it produces valid results and models do not deteriorate due to potential false positives in training data.

Base Encoder & Projection Head. We pass column instances $\{x_k\}_{k=1}^{2M}$ through a base encoder $f(\cdot)$ to get initial column embeddings $\{e_k\}_{k=1}^{2M}$. Note that our contrastive learning framework is flexible about the choice of the base encoder. The encoder can give embeddings at token/cell/column level, and if necessary, we can apply aggregation (e.g, average) to obtain column-level embeddings. Our framework has the flexibility to benefit from the advance of modeling techniques over time without being tied to a specific model. We describe the choices of $f(\cdot)$ we experiment with in subsection 3.3.3.

Following the encoder, a small multi-layer neural network $g(\cdot)$, called projection head, maps the representations from the encoder to a latent space through linear transformations and non-linear activation in between. Note that unlike the practice in CV which discards projection head in inference and uses encoder outputs for downstream tasks, we preserve projection head and use projected embeddings for table union search. This is because we found projected embeddings yield better performance in initial experiments, and for encoders like word embedding models, only projection head is trainable and has to be preserved for inference. For simplicity, we keep using the notations $\{e_k\}_{k=1}^{2M}$ for projection outputs.

Contrastive Loss. The training objective is a core component in the framework, which drives the learned representations towards the characteristics of the indexing/search data

structure in table union search. For example, considering an indexing/search data structure that approximates cosine similarity of vectors, the model should ideally learn to produce embeddings with high cosine similarity for unionable columns in our case.

One common setting of contrastive learning defines a prediction task of identifying positive examples from the training batch. Given embedded columns $\{e_k\}_{k=1}^{2M}$, the model learns to predict e_{k+M} as the most similar one to e_k and vice versa for each $e_k (1 \leq k \leq M)$. Here, one can choose a similarity measure that aligns with locality search. In other words, depending on what similarity measure for which the indexing/search data structure is designed, we can push that similarity measure into model learning. As an illustration, the similarity between any two instances e_i and e_j can be measured by their cosine similarity as

$$\text{sim}(i, j) = \frac{e_i^T e_j}{\|e_i\| \|e_j\|}$$

and the loss is calculated as

$$l(k, k + M) = -\log \frac{\exp(\text{sim}(k, k + M) / \tau)}{\sum_{l=1, l \neq k}^{2M} \exp(\text{sim}(k, l) / \tau)}$$

where $\tau > 0$ is a scaling hyper-parameter called temperature. Minimizing $l(k, k + M)$ is equivalent to maximizing the probability of e_{k+M} being the most similar to e_k in terms of cosine similarity among all the embedded columns except e_k itself. In this way, the learned column representations are more feasible to be used with the indexing/search data structure that approximates cosine similarity compared to embeddings given by any other models that are not trained to optimize for this purpose (e.g., tabular language models that are trained with the objective to recover masked tokens or entities [47]).

Finally, the loss over all the $2M$ positive column pairs in a training batch is computed as

$$L = \frac{1}{2M} \sum_{k=1}^M [l(k, k + M), l(k + M, k)].$$

Algorithm 1 summarizes the training process of contrastive learning in PYLON.

3.3.3 Choices of the Base Encoder

Although we expect the input to the contrastive loss function to be column embeddings, the base encoder does not necessarily need to give column embeddings directly. It is possible for the encoder model to generate embeddings at different granularity (i.e., token/cell/column) because we can apply aggregation if necessary. We describe the basic encoding process of

embedding models we experimented with in section 3.4.

Word Embedding Models (WEM). As a WEM assigns a fix representation to a token, WEM-based encoders treat each column independently as a document where a standard text parser tokenizes data values in a column. With a fastText embedding model, we first get cell embeddings by averaging token embeddings in each cell and then aggregate cell embeddings to get a column embedding. More interestingly, web table embedding models [72] consider each cell as a single token (they concatenate tokens in a cell with underscores) and output embeddings at cell level. Nevertheless, we aggregate cell embeddings to derive the column embedding.

Language Models (LM). Since a table is a cohesive structure for storing data, considering values in neighboring columns could integrate context into the embeddings and help mitigate ambiguity in unionable column search. For example, encoding column "year" in figure 3.1 individually loses the context that this column refers to the publication year of research papers. In this case, the embeddings of "Year" columns in the corpus are less distinguishable (in terms of cosine similarity) even though they may refer to different concepts of year such as the birth year of people or the release year of movies. With context provided by other columns like "Title" and "Venue", it is more likely that "Year" columns appearing in tables about papers are more close to each other than "Year" columns in tables about other topics, which helps find more related tables.

We leverage LMs to derive contextual column embeddings. We first serialize each row in T_i as a sequence by concatenating tokenized cell values. For example, the first row of the table at the top in Figure 3.1 will be linearized as follows

[CLS] title | A Database ... [SEP] authors | Jerry ... [SEP] ... [END]

The sequence is annotated with special tokens in the LM where [CLS] token indicates the beginning of the sequence, [END] token indicates the end, and [SEP] tokens separate cell values in different columns. Then the LM takes in each sequence and generates a contextual representation for each token in the sequence (essentially taking into account the relation between values in the same row). We apply mean pooling to tokens in the same cell and get cell embeddings. To consider the relation of values in the same column, we adopt the vertical attention mechanism in [177] to have weighted column embeddings by attending to all of the sampled cells in the same column.

Word embedding models have previously been used to find unionable tables. Two state-of-the-art choices are **fastText** and **WTE** (web table embeddings [72]). Language models have not thus far been used for the unionability problem. **BERT** [49] is a leading language

Algorithm 1: PYLON Contrastive Learning

Input : $\mathcal{S}$, a corpus of tables;
 N , batch size;
 p , training hyper-parameters.
Output: $g \circ f$, a PYLON model.

```
1  $g \circ f \leftarrow \text{initialize\_model}();$   
2 for mini-batch of  $N$  tables  $\{C_j^i\}_{i=1}^N$  from  $\mathcal{S}$  do  
3    $\{x_k, x_{k+M}\}_{k=1}^M \leftarrow \text{construct\_training\_data}(\{C_j^i\}_{i=1}^N);$   
4    $\{e_k\}_{k=1}^{2M} \leftarrow g \circ f.\text{encode\_and\_embed}(\{x_k, x_{k+M}\}_{k=1}^M);$   
5   /* Define  $l(k, k+M) = -\log \frac{\exp(\text{sim}(k, k+M) / \tau)}{\sum_{l=1, l \neq k}^{2M} \exp(\text{sim}(k, l) / \tau)}$  and  $\text{sim}(i, j) = \frac{e_i^T e_j}{\|e_i\| \|e_j\|} *$  /  
6    $\mathcal{L} \leftarrow \frac{1}{2M} \sum_{k=1}^M [l(k, k+M), l(k+M, k)];$   
7    $g \circ f \leftarrow \mathcal{L}.\text{backpropagate}(p);$   
8 end  
9 return  $g \circ f;$ 
```

model used for many purposes today. We develop three versions of PYLON, one for each of these three encoder choices: **fastText**, WTE, and a BERT-based language model, and refer to the derived models as **PYLON-fastText**, **PYLON-WTE**, **PYLON-LM** respectively. We evaluate the effect of encoder choices in subsection 3.4.5.

3.3.4 Embedding Indexing and Search

To avoid exhaustive comparisons of column embeddings over a large corpus at query time, we use locality-sensitive hashing (LSH) [83] for approximate nearest neighbor search and treat unionable column search as an LSH-index lookup task [124, 17]. Note that the specific choice of indexing/search data structure is flexible (one can use a more advanced technique like Hierarchical Navigable Small World [109]). The key is that the similarity measure approximated by the indexing/search data structure should align with the similarity measure employed in the training objective of contrastive learning so that the learned embeddings are optimized for the indexing/search data structure.

LSH utilizes a family of hash functions that maximize collisions for similar inputs. The result of LSH indexing is that similar inputs produce the same hash value and are bucketed together whereas dissimilar inputs are ideally placed in different buckets. For approximate search relative to the cosine similarity, we index all column embeddings in a random projection LSH index [33]. The idea of random projection is to separate data points in a high-dimensional vector space by inserting hyper-planes. Embeddings with high cosine similarity

Algorithm 2: Top- k Table Union Search

Input : $\mathcal{I}$, a populated LSH index;
 Q , a query table;
 k , a constant.

Output: top- k unionable tables.

```
1  $column\_candidates \leftarrow \{\}$ ;
2  $column\_scores \leftarrow \{\}$ ;
3 for  $c \in Q.columns$  do
4    $c\_candidates, c\_scores \leftarrow \mathcal{I}.lookup(c)$ ;
5    $column\_candidates[c].add(c\_candidates)$ ;
6    $column\_scores[c].add(c\_scores)$ ;
7 end
8  $table\_candidates \leftarrow group\_by(column\_candidates)$ ;
9  $ranked\_table\_candidates \leftarrow cmpt\_table\_unionability(table\_candidates, column\_scores)$ ;
10 return  $ranked\_table\_candidates[:k]$ ;
```

tend to lie on the same side of many hyper-planes.

Algorithm 2 summarizes the top- k table union search. Following Definition 1, we instantiate the unionability of two attributes as the cosine similarity of their embeddings (c_scores in line 4 of Algorithm 2). Line 8 groups retrieved column candidates across query columns by their table sources. To decide on the table unionability from Definition 3 ($cmpt_table_unionability$ in line 9), we use the same weighting strategy as [17] over query attributes and corresponding matching attributes in candidate tables. For a target attribute A , let R_A denote the distribution of all similarity (unionability) scores between A and any attribute B returned by the LSH index. The weight w of a similarity score $\mathcal{U}_{attr}(A, B)$ is given by the cumulative distribution function of R_A evaluated at $\mathcal{U}_{attr}(A, B)$:

$$w = \Pr(\mathcal{U}_{attr}(A, B') \leq \mathcal{U}_{attr}(A, B)), \forall \mathcal{U}_{attr}(A, B') \in R_A$$

In other words, a similarity score is weighted by its percentile in the distribution. This weighting scheme helps balance between a candidate table with a few unionable attributes of high similarity scores and another candidate table with more unionable attributes but of lower similarity scores.

Using the same index and search structure as previous works makes it transparent to compare our embedding approach with theirs in effectiveness and efficiency.

Table 3.1: Basic statistics of evaluation datasets.

	# Tables	# Base Tables	Avg. # Rows	Avg. # Columns	# Queries	Avg. # Answers
Pylon	1,746	1,746	115	10	124	42
TUS-Small	1,530	10	4,466	10	1,327	174
TUS-Large	5,043	32	1,915	11	4,296	280

3.3.5 Integrating Syntactic Methods

Thus far, we have focused purely on semantic methods to unify similar attributes. It makes sense to prefer semantic methods to syntactic ones because of their potential robustness to many different types of variation. However, we note that syntactic methods are based on measures of similarity very different from semantic methods. Intuitively, one should expect to be able to do better if we can integrate the two.

Indeed, some previous work [124, 17] has made this observation as well, and shown that an ensemble of semantic and syntactic methods can do better than either on its own. The PYLON semantic method permits the use of an additional complementary syntactic method. As in [17], we independently obtain scores from the two methods and then use the average of the two as our final score.

3.4 Experiments

We first evaluate the effectiveness and efficiency of three model variants from our contrastive learning framework and compare them with their corresponding base encoders. We then demonstrate that our embedding approach is orthogonal to existing syntactic measures, which can further improve the results. We finally compare our best model with the state-of-the-art D^3L [17].

3.4.1 Datasets and Metrics

TUS Benchmark. Nargesian et al. [124] compiles the first benchmark with ground truth out of Canadian and UK open data lakes. They synthesize around 5,000 tables from 32 base tables by performing random projection and selection. They also generate a smaller benchmark consisting of around 1,500 tables from 10 base tables in the same manner. We refer to them as TUS-Large and TUS-Small respectively.

Pylon Benchmark. We create a new dataset from GitTables [78], a data lake of 1.7M

tables extracted from CSV files on GitHub. The benchmark comprises 1,746 tables including unionable table subsets under topics selected from Schema.org [71]: scholarly article, job posting, and music playlist. We end up with these three topics since we can find a fair number of unionable tables of them from diverse sources in the corpus (we can easily find unionable tables from a single source but they are less interesting for table union search as simple syntactic methods can identify all of them because of the same schema and consistent value representations).

Cleaning and Construction. We download three largest subsets of GitTables (“object”, “thing”, and “whole”) and preprocess them by removing HTML files, tables without headers, rows with foreign languages, and finally small tables with fewer than four rows or four columns. We cluster the resulting tables by their schema and perform a keyword search over schema with keywords related to three topics. We manually select 35 unionable tables of topic scholarly article, 41 tables of topic job posting, and 48 tables of topic music playlist. We then randomly sample 100,000 tables for training, 5,000 tables for validation, and put the rest of tables as negatives¹ in a pool with selected unionable table subsets for the search evaluation.

Table 3.1 provides an overview of basic table statistics of each evaluation benchmark.

Metrics. For effectiveness, we report both precision and recall of top- k search with varying k . At each value of k , we average the precision and recall numbers over all the queries. We consider a table candidate a true positive with respect to the target table if it is in labeled ground truth. We do not require perfect attribute pair matching as it is a more challenging setting and requires laborious column-level labeling.

As to efficiency, we report indexing time (i.e., total amount of time in minutes to index all columns in a dataset) and query response time (i.e., average amount of time in seconds for the LSH index to return results over all queries in a dataset²).

In evaluation, we randomly sample 1000 queries from TUS-Large for efficient experiment purposes. The query subset has an average answer size of 277, which is very close to that of the full query set (i.e., 280). We use all the queries in PYLON and TUS-Small datasets.

3.4.2 Baselines

We consider two embedding methods and one full approach as baselines for comparison.

fastText. Many data management tasks not limited to table union search [62, 124, 28] have adopted **fastText** in their approach, which is a popular word embedding model trained

¹we filtered these tables using their schema to reduce the chance of them being unionable to selected tables in the unionable subsets.

²Current implementations include data loading and logging time as well.

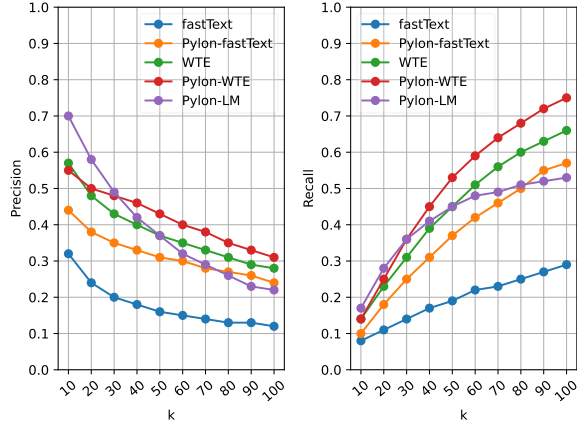


Figure 3.3: Top-k precision and recall of embedding measures on the PYLON dataset.

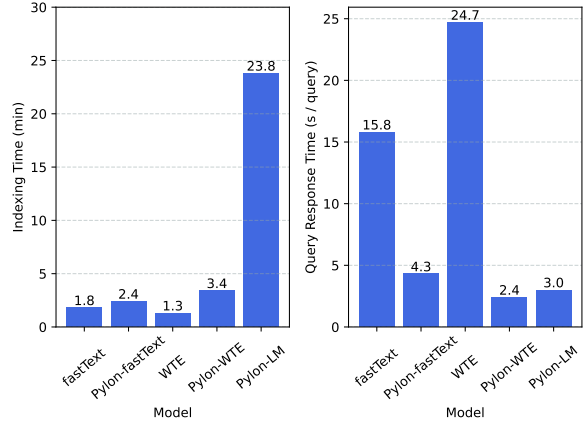


Figure 3.4: Indexing time and query response time on the PYLON dataset.

on Wikipedia documents.

WTE. Gunther et al. [72] devises a word embedding-based technique to represent text values in Web tables. They generate text sequences from tables for training by serializing tables in two different ways that capture row-wise relations and relations between schema and data values respectively. It is reported that the model using both serialization obtained the best performance in a task of ranking unionable columns.

D^3L . Bogatu et al. [17] proposed a distance-based framework D^3L that uses five types of evidence to decide on column unionability: (i) attribute name similarity; (ii) attribute extent overlap; (iii) word-embedding similarity; (iv) format representation similarity; (v) domain distribution similarity for numerical attributes. Their aggregated approach is shown to be more effective and efficient than previous work [124, 60] on the TUS benchmark and another self-curated dataset of open data tables.

3.4.3 Comparisons of Interest

We have 5 variants of PYLON to compare against baseline systems for both effectiveness and efficiency in identifying unionable tables using semantic similarity methods: 3 variants from the online training data construction strategy and 2 variants from the offline data construction strategy. In addition, we have 3 syntactic similarity measures that could be used to augment each of these 5 variants. Finally, we have 3 baselines, two of which are semantic word embedding based, and hence could also be augmented with the syntactic similarity measures. The third baseline (D^3L) already integrates both syntactic and semantic similarity, and hence does not benefit from additional augmentation with syntactic techniques.

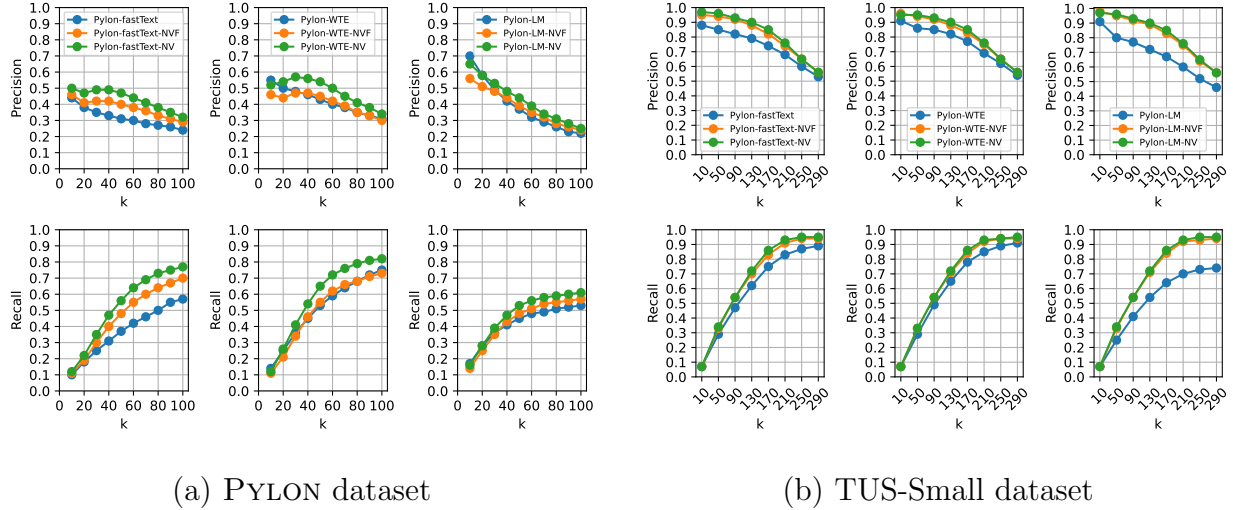


Figure 3.5: Precision and recall (with respect to varying k) of the ensemble of PYLON embedding models and syntactic measures.

Since there are a very large number of alternatives to compare, we break up the comparisons into four sets, as follows, and present the results for each set separately. For the first three sets, we restrict ourselves to the online training data construction strategy for PYLON. We refer to the derived models as **PYLON-fastText**, **PYLON-WTE**, **PYLON-LM** respectively based on the corresponding encoder choice. Results for the offline data construction strategy show generally similar trends, and the most interesting are shown in the fourth set.

The first set of comparisons look purely at semantic methods, considering the 3 variants of PYLON and comparing them to the first two baselines. We leave out D^3L because it already incorporates syntactic methods as well. The second set of comparisons look purely at the benefit obtained when semantic methods are enhanced with syntactic measures. We do so for all methods evaluated in the first set. Finally, we bring everything together by comparing the best methods of the second set with the best integrated baseline, D^3L . This is the final top line “take away” from the experiments, eliding details from the first two sets of comparisons.

3.4.4 Experiment Details

As to model training, we train **PYLON-fastText** for 50 epochs with a batch size of 16 on 2 NVIDIA GeForce RTX 2080 Ti GPUs; **PYLON-WTE** for 20 epochs with a batch size of 32 on a single NVIDIA Tesla P100 GPU; **PYLON-LM** for 20 epochs with a batch size of 8 on 4 NVIDIA Tesla P100 GPUs from Google Cloud Platform. As seen in table 3.2, the

Table 3.2: Model training time (min / epoch) where each model is defined by the encoder choice and the training data construction strategy.

	Online Sampling	Offline Approximate Matching
<i>Pylon-fastText</i>	6.5	0.42
<i>Pylon-WTE</i>	0.99	0.13
<i>Pylon-LM</i>	33	-

training is especially efficient for simple word embedding encoders (as only parameters in projection head are updated) and the offline data construction strategy (as embeddings are pre-computed before training). We save the models with the smallest validation loss. The model training is implemented in PyTorch [131] and PyTorch Lightning.

For evaluation of table union search, we set the similarity threshold of LSH index to 0.7 in all experiments and use the same hash size as D^3L . We run all evaluation on a Ubuntu 20.04.4 LTS machine with 128 GiB RAM and Intel(R) Xeon(R) Bronze 3106 CPU @ 1.70GHz.

3.4.5 Results

As PYLON is an embedding-based approach, we first evaluate PYLON model variants against embedding baselines `fastText` and `WTE`, and inspect the effects of contrastive learning on them.

Experiment 1(a): Comparison of effectiveness between Pylon model variants and their corresponding base encoders. Figure 3.3 shows the precision and recall of each embedding measure on the PYLON benchmark. Both `PYLON-WTE` and `PYLON-fastText` outperform their corresponding base models with a notable margin. When $k = 40$, around the average answer size, `PYLON-WTE` is 6% better than `WTE` on both metrics, and `PYLON-fastText` performs better than `fastText` by 15% on precision and 14% on recall.

Overall, our `PYLON-WTE` model consistently achieves the highest precision and recall as k increases. We also note that `PYLON-LM` has strong performance up until $k = 30$ but degrades after that. This is because `PYLON-LM` only samples 10 rows from each table to construct embeddings (for indexing and query efficiency) while other word-embedding methods can afford to encode the entire table at low indexing time, which we demonstrate in experiment 1(b).

Experiment 1(b): Comparison of efficiency between Pylon model variants and

their corresponding base encoders. In figure 3.4, we see both embedding baselines are very efficient in index construction and it takes less than 2 minutes to index the entire PYLON dataset. Unlike fixed embeddings, our models need to infer embeddings at runtime. For PYLON-fastText and PYLON-WTE, since the encoder is fixed, the inference cost is exclusively from projection head. It takes both less than 3.5 minutes to build the index. In contrast, the runtime inference cost of PYLON-LM is more expensive as the language model has much more complex architecture and has 130M parameters versus 35.8K parameters in projection head. We also acknowledge the less efficient implementation of embedding inference at this point (e.g., run inference for each column without batch predictions). Nevertheless, indexing time, as a one-time overhead, can be amortized among queries.

On the other hand, all of our models are considerably more efficient in query response time. PYLON-fastText is 2.7x faster than fastText and PYLON-WTE is 9x faster than WTE. The significant speedup of query response time is attributed to contrastive learning where embeddings of attribute values occurring in the same context are pushed close to each other and to have high cosine similarity whereas embeddings of two random columns are pushed apart and to have low cosine similarity. As the embedding similarity between two random columns is suppressed, this dramatically reduces the chance of two random columns sharing many LSH buckets. In other words, our column embeddings are trained in a specific way that aligns with the similarity measure LSH index approximates and LSH index can process much fewer candidates at the configured similarity threshold.

To illustrate the suppression effect of contrastive learning, we compare heatmaps of pairwise cosine similarity of column embeddings encoded by WTE and PYLON-WTE respectively. Consider the three text columns of the first table in Figure 3.1. As shown in Figure 3.6(a), the pairwise cosine similarity of WTE embeddings is mostly above 0.5. There is a very high similarity (0.87) between the “title” column and the “venue” column and they will be mistakenly viewed as unionable. But this is not an issue for PYLON-WTE embeddings as shown in Figure 3.6(b) where the pairwise similarity between different columns are much lower (below 0.51) and the LSH index will not return the “venue” column as a unionable candidate of the “title” column.

In the next set of experiments, we consider three syntactic measures used by D^3L and evaluate how much they can augment our embedding measures.

1. Name (N): Jaccard similarity between q-gram sets of attribute names.
2. Value (V): Jaccard similarity between the TF-IDF sets of attribute values.
3. Format (F): Jaccard similarity between regular expression sets of attribute values.

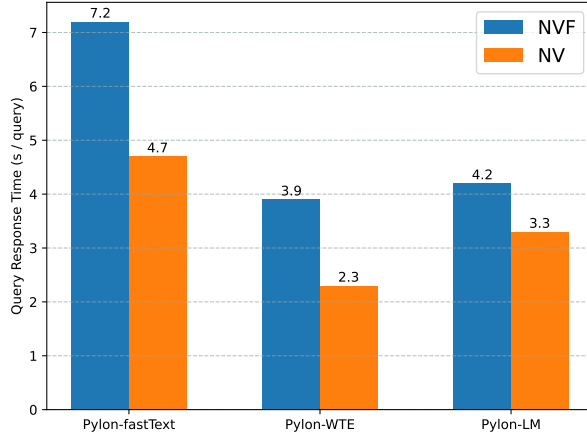


Figure 3.6: Pairwise cosine similarity of column embeddings: (a) *WTE* embeddings; (b) *PYLON-WTE* embeddings.

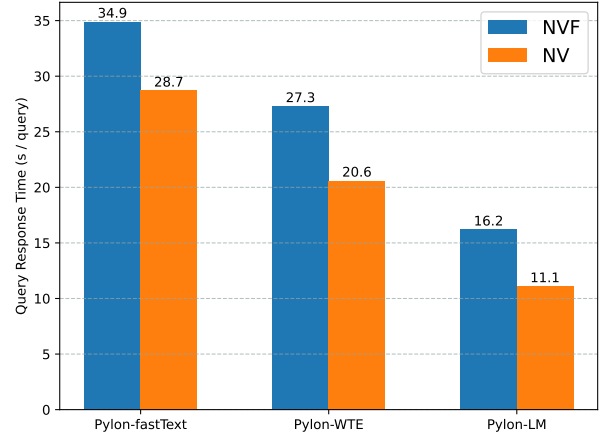
Experiment 2: Effectiveness of the ensemble of Pylon model variants and syntactic measures. Figure 3.5(a) and (b) show the precision and recall of the ensemble of *PYLON* embedding models and syntactic measures on *PYLON* and TUS-Small datasets respectively. We consistently observe from both datasets that adding syntactic measures can further enhance the performance. In particular, name (N) and value (V) similarity are most effective syntactic measures. Around the average answer size of the *PYLON* dataset ($k = 40$), N and V together raise up the precision and recall of *PYLON-fastText* by nearly 20%, of *PYLON-WTE* by 10%, and of *PYLON-LM* by over 5%. Similarly, around the average answer size of the TUS-Small dataset ($k = 170$), there is an increase of about 10% in both precision and recall for *PYLON-fastText*, about 5% for *PYLON-WTE*, and more than 10% for *PYLON-LM*.

We also observe that adding additional format measure (F) hurts the performance (notably on the *PYLON* dataset and slightly on TUS-small). This is because tables in the *PYLON* dataset are mostly from disparate sources and so the value format tends to be inconsistent across tables whereas tables in TUS-Small are synthesized from only 8 base tables and it is much more likely for many tables to share format similarity. Even worse, including format index imposes non-trivial runtime cost (see figure 3.7). For example, compared to model *PYLON-WTE-NV*, the query response time of *PYLON-WTE-NVF* (with the extra format measure) surges by 66.7% on the *PYLON* dataset and by 32.2% on TUS-Small.

Finally, we compare our best-performing model *PYLON-WTE-NV* with the state-of-the-art D^3L . As *PYLON-WTE-NV* does not use format and domain measures in D^3L , for fair comparison, we consider three versions of D^3L . We refer to the full version of D^3L as D^3L -



(a) PYLON dataset



(b) TUS-Small dataset

Figure 3.7: Comparison of query response time between including and excluding the format measure.

5, the one without the format measure as D^3L -4, and the one without format and domain measures as D^3L -3.

Experiment 3: Comparison of effectiveness and efficiency between our best model and D^3L . Figure 3.8 shows the performance of PYLON-WTE-NV and three D^3L variants on PYLON, TUS-Small, TUS-Large datasets respectively. Around the average answer size ($k = 40$) of the PYLON dataset, PYLON-WTE-NV is around 15% better than the strongest D^3L instance (i.e., D^3L -3) in both precision and recall. PYLON-WTE-NV performs much better than D^3L in this case because our embedding model using contrastive learning is optimized for indexing/search data structure and is trained on a dataset of a distribution similar to the test set and thus can capture more semantics than the off-the-shelf fastText embedding model used in D^3L .

On TUS-Small and TUS-Large, we observe all instances have relatively competitive performance while PYLON-WTE-NV performs marginally better compared to all D^3L variants. On TUS-Small, around the average answer size ($k = 170$), PYLON-WTE-NV is 2% better than D^3L -3 and 5% better than D^3L -5 in both precision and recall. On TUS-Large, around the average answer size ($k = 290$), PYLON-WTE-NV is more than 2% better than D^3L variants in both metrics. The smaller performance gap is due to the synthetic nature of TUS benchmark where most of unionable tables are generated from the same base table and share common attribute names and many attribute values. So syntactic measures (N and V) can capture most of similarity signals and obtain high precision and recall even without support of semantic evidence.

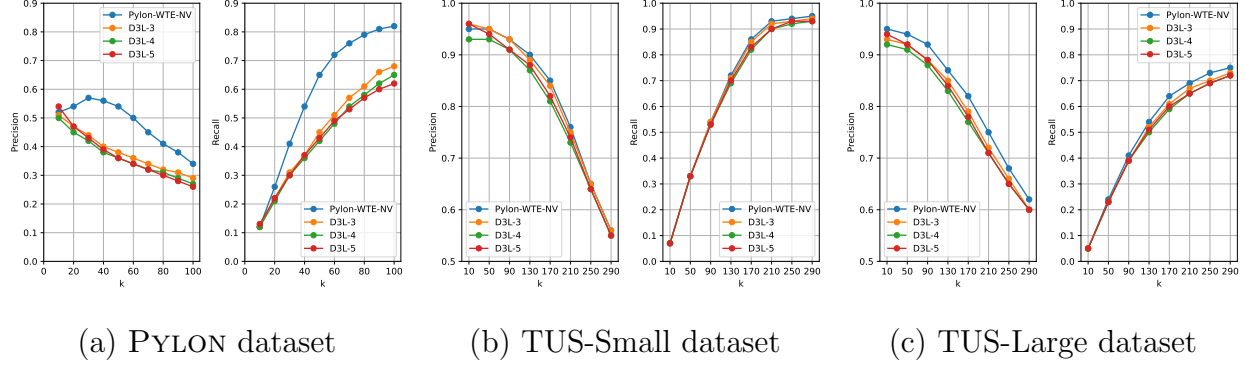


Figure 3.8: Comparison of precision and recall between D^3L instances and our best model *PYLON-WTE-NV*.

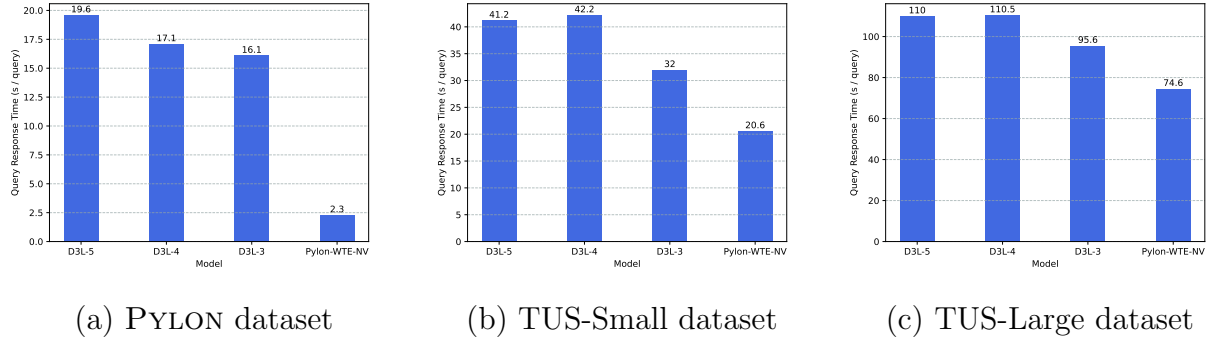


Figure 3.9: Comparison of query response time between D^3L instances and *PYLON-WTE-NV*.

Additional to the performance gain, the biggest advantage of *PYLON-WTE-NV* is the fast query response time. On the PYLON dataset, our model is nearly 9x faster than the full version D^3L-5 and 7x faster than D^3L-3 . Even on TUS-Small and TUS-Large, which are datasets of a different data distribution (open data tables), we still save runtime by 44% and 32% respectively compared to D^3L-5 , and by 35.5% and 21.9% respectively compared to D^3L-3 .

Experiment 4: Effectiveness and efficiency of Pylon model variants from the offline training data construction strategy. Figure 3.10 shows the precision and recall of 4 PYLON variants from two training data construction strategies and their baselines. On the PYLON dataset, around the average answer size ($k = 40$), two PYLON models from the alternative data construction strategy, *PYLON-WTE-offline* and *PYLON-fastText-offline*, retain strong performance and outperform the corresponding baseline by 3% and 9% respectively. Note that PYLON models derived from the sampling data construction strategy have

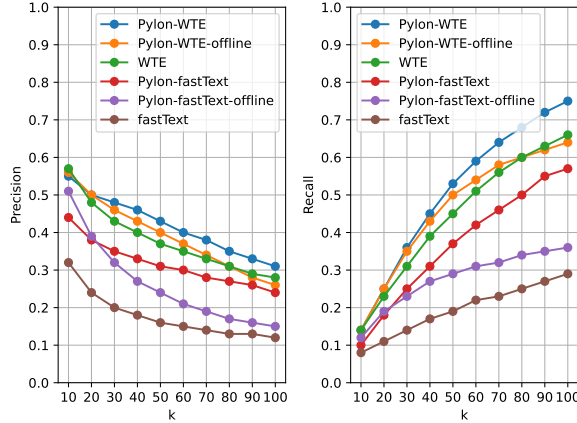


Figure 3.10: Top-k precision and recall of 6 embedding measures on the PYLON dataset.

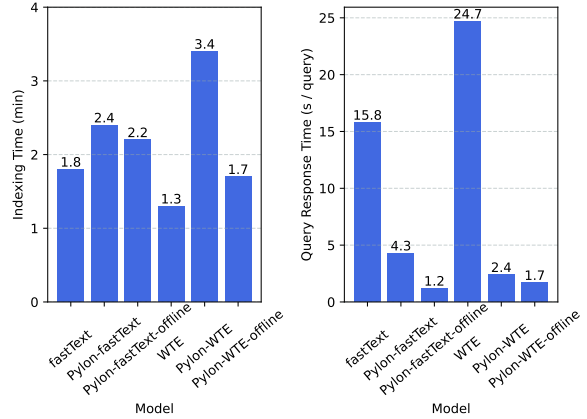


Figure 3.11: Indexing time and query response time on the PYLON dataset.

consistently better performance as k increases. We also observe a similar trend on the TUS benchmark while the performance gap of all instances is smaller.

As shown in figure 3.11, both new models are efficient in indexing time and query response time. Compared to the corresponding baseline, *PYLON-WTE-offline* is 12x faster and *PYLON-fastText-offline* is 14.5x faster in query response time. Again, this significant speedup demonstrates the value of considering characteristics of indexing/search data structure in the training of embedding models and the distinguishing power of contrastive learning, which enables the LSH index to work more effectively with embeddings.

3.4.6 Discussion

We also leave and discuss a few clues for future extensions.

Alternative Contrastive Loss. While the loss function used in this project is an effective one, it is not the only feasible training objective for self-supervised contrastive learning. For example, triplet loss [145] considers a triplet (x, x^+, x^-) as a training example where x is an input, x^+ is a positive sample (belonging to the same class as x or semantically similar to x) and x^- is a negative sample. Additionally, what considers as negative examples and “hardness” of negative examples are also interesting aspects to explore.

Verification of Column unionability. Besides quantitative evaluation, we also manually inspect results of a few queries for each dataset. We observe that in some correct table matches, there are misalignment of unionable columns. To mitigate this issue, we consider that progress in semantic column type prediction [181, 157] can be beneficial for verifying the unionability of columns as a post-processing step.

3.5 Related Work

Our work is most related to data integration on the web and data discovery in enterprise and open data lakes.

Web Table Search. Cafarella et al. [25] presents OCTOPUS that integrate relevant data tables from relational sources on the web. OCTOPUS includes operators that perform a search-style keyword query over extracted relations and their context, and cluster results into groups of unionable tables using multiple measures like TF-IDF cosine similarity. [174] defines three information gathering tasks on Web tables. The task of augmentation by example essentially involves finding unionable tables that can be used to fill in the missing values in a given table. Their Infogather system leverages indirectly matching tables in addition to directly matching ones to augment a user input. Sarma et al. [144] formalizes the problem of detecting related Web tables. At the logical level, the work considers two tables related to each other if they can be viewed as results to queries over the same (possibly hypothetical) original table. In particular, one type of relatedness they define is *Entity Complement* where two tables with coherent and complementary subject entities can be unioned over the common attributes. This definition requires each table to have a subject column of entities indicating what the table is about and that the subject column can be detected. Following the definition, the work captures entity consistency and expansion by measuring the relatedness of detected sets of entities with signals mined from external ontology sources. Finally, they perform schema mapping of two complement tables by computing a schema consistency score made up of the similarity in attribute names, data types, and values.

Data Discovery in the Enterprise. Fernandez et al. [61] identifies data discovery challenges in the enterprise environment. The position paper describes a data discovery system including enrichment primitives that allow a user to perform entity and schema complement operations. Building on top of this vision, Fernandez et al. presents AURUM [60], a system that models syntactic relationships between datasets in a graph data structure. With a two-step process of profiling and indexing data, AURUM constructs a graph with nodes representing column signatures and weighted edges indicating the similarity between two nodes (e.g., content and schema similarity). By framing queries as graph traverse problems, AURUM can support varied discovery needs of a user such as keyword search and similar content search (which can be used for finding unionable columns and tables). Fernandez et al. [62] further employs word embeddings in AURUM to identify semantically related objects in the graph.

Data Discovery in Open Data Lakes. Nargesian et al. [124] defines the table union search problem on open data and decomposes it as finding unionable attributes. They

propose three statistical tests to determine the attribute unionability: (1) set unionability measure based on value overlap; (2) semantic unionability measure based on ontology class overlap; and (3) natural language unionability measure based on word embeddings. A synthesized benchmark consisting of tables from Canadian and UK open data shows that natural language unionability works best for larger k in top- k search. In the meantime, set unionability is decent when $k = 1$ for each query but vulnerable to value overlap in attributes of non-unionable tables, and semantic unionability stays competitive to find some unionable tables for most queries despite incomplete coverage of external ontologies. The ensemble of three measures further improves the evaluation metrics. [17] adopts more types of similarity measures based on schema- and instance-level fine-grained features. Without relying on any external sources, their D^3L framework is shown effective and efficient on open data lakes. EMBDI [28] proposes a graph model to capture relationships across relational tables and derives training sequences from random walks over the graph. They further take advantage of embedding training algorithms like *fastText* to construct embedding models. Their relational embeddings demonstrate promising results for data integration tasks such as schema matching and entity resolution. SANTOS [91], a very recent work, leverages the relationship of pairs of attributes for table union search. However, it relies on the existence of a relationship and the coverage of the relationships in a knowledge base. Although SANTOS also draws on the data lake itself to discover new relationships using the co-occurrence frequency of attribute pairs, it particularly misses rare yet important relationships.

For a broader overview of the literature, we refer readers to the survey of dataset search [32].

3.6 Conclusion

In this work, we formulate the table union search problem as an unsupervised representation learning and embedding retrieval task. We present PYLON, a self-supervised contrastive learning framework that models both data semantics and characteristics of indexing/search data structure. We also demonstrate that our approach is both more effective and efficient compared to the state-of-the-art.

CHAPTER 4

WarpGate: A Semantic Join Discovery System for Cloud Data Warehouses

4.1 Introduction

Three major correlated forces act on enterprise data analysis today. First, more and more enterprise data are stored on the cloud. Second, the size and the diversity of data available to enterprises increase faster than ever as a growing number of data sources feed into enterprises’ cloud data warehouses (CDWs). Third, the number of users in enterprises who would like direct access to cloud data for decision-making is also rapidly growing, boosting the demand for easy-to-use cloud data analysis systems such as Sigma Workbooks [65]. Hence enabling easy and effective data discovery is increasingly more valuable. However, existing commercial solutions such as data catalogs are typically inaccessible from business intelligence tools, inadequate for surfacing relationships unknown to users, and inflexible in capturing data semantics.

Not surprisingly, recent years have seen a rising interest in data discovery, especially for Open Data [116, 124, 191, 17]. This is a challenging problem due to the unstructured/semi-structured nature of Open Data and the lack of high-quality metadata. However, we observe that modern organizations also face data discovery challenges when dealing with structured data (i.e., databases in CDWs) for similar reasons: (i) in large CDWs, few people have a good understanding of data and their relationships; (ii) data is added in a much less modeled state as cloud computing power and storage become easily accessible; and (iii) business users who do not have comprehensive knowledge of underlying schema designs, are usually unaware of relationships between datasets (e.g., PK/FK relationship between two tables). There is a need for surfacing such relationships to help users connect related datasets within and across databases. We also observe that CDW data tends to have more structure than Open Data, sometimes has high update rates, and usually has far more stringent requirements for completeness of discovery results.

Name	Billing Street	Billing City	Billing State	Billing Postal Code	Industry
Affiliated Managers Group, Inc.	183 Carey Park	Pueblo	Colorado	81015	Investment Managers
J P Morgan Chase & Co	968 Forster Center	Chicago	Illinois	60686	n/a
Blackrock MuniYield Quality Fund, Inc.	8 Northridge Avenue	Evansville	Indiana	47705	n/a
Atlantica Yield plc	7 Arapahoe Pass	Rochester	New York	14646	Electric Utilities: Central
Stantec Inc	9857 Tennessee Pass	Washington	District of Columbia	20029	Military/Government/Technical
Quaker Chemical Corporation	745 Leroy Point	Chicago	Illinois	60691	Major Chemicals
VictoryShares US 500 Enhanced Volatility Wtd ETF	17 Sunfield Pass	Hamilton	Ohio	45020	n/a
NAPCO Security Technologies, Inc.	0841 Annamark Pass	Fresno	California	93726	Telecommunications Equipment
Customers Bancorp, Inc	9 Basil Park	Washington	District of Columbia	20420	n/a
The Rubicon Project, Inc.	5 Miller Avenue	Fresno	California	93773	Computer Software: Programming, D
Dicerna Pharmaceuticals, Inc.	30 Mendota Pass	Louisville	Kentucky	40233	Major Pharmaceuticals
Olin Corporation	9 Prairieview Way	Kansas City	Missouri	64193	Major Chemicals
Cubic Corporation	3 Loepich Court	Fort Myers	Florida	33906	Industrial Machinery/Components
PartnerRe Ltd.	187 Dovetail Pass	Sacramento	California	94263	n/a
Autobyte Inc.	0913 Kim Crossing	Johnstown	Pennsylvania	15906	Computer Software: Programming, D
K2M Group Holdings, Inc.	5 Meadow Vale Court	West Hartford	Connecticut	06127	Medical/Dental Instruments
Movado Group Inc.	91 Vernon Junction	Minneapolis	Minnesota	55458	Consumer Specialties
Ascendis Pharma A/S	713 Eagan Court	Washington	District of Columbia	20022	Major Pharmaceuticals
BB&T Corporation	17 Sommers Terrace	Boise	Idaho	83757	n/a
National CineMedia, Inc.	9 Hoard Hill	El Paso	Texas	88569	Advertising
Knight Transportation, Inc.	97459 Beilfuss Crossing	Pensacola	Florida	32505	Trucking Freight/Courier Services
Gladstone Investment Corporation	14556 Dunning Plaza	Washington	District of Columbia	20057	n/a
J P Morgan Chase & Co	7 Colorado Trail	San Francisco	California	94159	n/a
Provident Financial Services, Inc	0 Summerview Hill	Memphis	Tennessee	38197	Savings Institutions
Shinhan Financial Group Co Ltd	50477 Jackson Place	Albuquerque	New Mexico	87201	Major Banks

Figure 4.1: The interface of Sigma Workbooks connecting to a CDW. Consider a business user Joey who is interested in launching a sales campaign and identifying ideal customers. She first opens up the **ACCOUNT** table in their **SALESFORCE** database, which lists all customer companies. Next, she wants to augment the table with relevant columns to help make customer selections.

Consider a business user Joey who has access to a CDW that comprises hundreds of databases and thousands of tables (Figure 4.1). In the interest of launching a sales campaign and finding potential customers, she starts with the **ACCOUNT** table in their **SALESFORCE** database, which contains member and billing data of their customer companies. To select ideal customers for the campaign, she wants to augment the **ACCOUNT** table with additional information such as sectors in which companies are engaged in their business activities. Now she has two options to find that complementary information: either she manually browses tables in the same database (even wanders through the entire data warehouse), or she can ask internal data teams for support. Either way, her workflow is significantly delayed. There is also a fair chance that she cannot go through all databases as it is a tedious and laborious process, and the data team does not have that particular piece of knowledge.

Despite efforts in prototyping data discovery for real-world applications [60, 16, 18], aspects of data discovery pertaining to enterprise settings are overlooked. For example, previous works assume a full pass of underlying data in the indexing/profiling phase. However, it is computationally and monetarily expensive to pull all data out of CDWs. In addition, existing solutions, including those for Open Data, focus on improving index lookup time for the scale of Open Data lakes or Web table corpora, which usually consist of a very large number (million scale) of small tables. In contrast, we observe that Sigma customers' CDWs

have several thousand tables on average but can have billions of rows. Due to the relatively smaller number of tables but larger table sizes, index lookup time becomes less dominant compared to data loading time and profiling time, both of which can now bottleneck end-to-end query response time in the pipeline. So, the main challenge in the Sigma scenario shifts in part from searching over a massive number of small tables to optimizing the entire pipeline for a fair number of very large tables (e.g., reducing data loading time and using efficient profiling methods).

In this work, we introduce a data discovery system WARPGate and share our progress. WARPGate prototypes semantic join discovery in Sigma, an enterprise platform for cloud data analytics. Specifically, we make the following contributions:

- We present WARPGate, a semantic join discovery system, which demonstrates a real-world usage scenario of data discovery within an enterprise product.
- We identify unique challenges in Sigma and propose an embedding-based solution to capture semantic relationships between tables, especially those across databases.
- We evaluate WARPGate on three cross-domain table corpora, including one collected from a CDW, and show our embedding-based approach is effective and robust to sampling. We also discuss further optimization opportunities for deployment and larger-scale discovery.

4.2 Problem Definition

In this section, we first briefly introduce Sigma and describe the data discovery needs of Sigma users. Then we formally define our problem.

4.2.1 Data Discovery Need in Sigma

Sigma is a web-based SaaS platform for cloud data analytics and business intelligence. It features Sigma Workbooks, which provides a spreadsheet-like interface for business users to perform interactive visual analysis of data in CDWs.

Among many features familiar to business users who are more skilled in spreadsheet functions than SQL, **Lookup** enables users to write a spreadsheet formula and add a column from another data source (i.e., another table) to enrich the dataset at hand. Under the hood, Workbooks translates spreadsheet formulas into SQL queries required to fetch data from cloud databases and performs cardinality-preserving joins on related tables.

A major limitation of **Lookup** is that users have to specify the join path in formulas. In other words, it requires users to have prior knowledge about relationships between datasets. However, users often lack such knowledge and have to rely on their data teams for help or manually browse tables in databases to find a clue. This limitation has become a bottleneck in fully utilizing the **Lookup** functionality, significantly delaying their data exploration and analysis workflows. We believe this sort of data discovery will be a challenge for all cloud analytics products and hence has to be addressed.

4.2.2 Semantic Join Discovery

Problem Statement (Top- k Semantic Join Discovery). Given a corpus of tables $\mathcal{S}$, a query column $\mathbf{c}_q$ from a table $\mathbf{Q}$, and a constant k , find up to k candidate columns from $\mathcal{S}$ such that they are most likely to be joinable with $\mathbf{c}_q$.

Traditionally, joinable tables have been discovered through syntactic analysis, which can miss many join opportunities, particularly when data are independently sourced. High dimensional embeddings have recently been shown to capture semantics effectively in many domains [47, 181]. We expect that the semantic similarity of two columns can be quantified by some similarity measure between their embeddings, which we expect can serve as a proxy for their join-ability as well.

Definition 5 (Semantic Column Join-ability). Given two columns A and B , we define semantic column join-ability $\mathcal{J}(A, B)$ as

$$\mathcal{J}(A, B) = \mathcal{M}(\mathcal{T}(A), \mathcal{T}(B))$$

where $\mathcal{T}(\cdot)$ is an embedding function that projects a given column into a vector space and $\mathcal{M}(\cdot, \cdot)$ is a similarity measure between two embeddings in the vector space.

Based on this metric, we can attempt to solve the semantic join discovery problem by returning candidate tables in descending order of semantic column join-ability $\mathcal{J}$ relative to the query column $\mathbf{c}_q$. How to do this effectively is the central challenge addressed in this paper.

4.3 WarpGate Overview

We discuss key challenges to implementing an embedding-based approach to the semantic join discovery problem, along with our solutions to these challenges in Section 4.3.1. Then we describe our system prototype **WARP**GATE incorporating these ideas in Section 4.3.2.

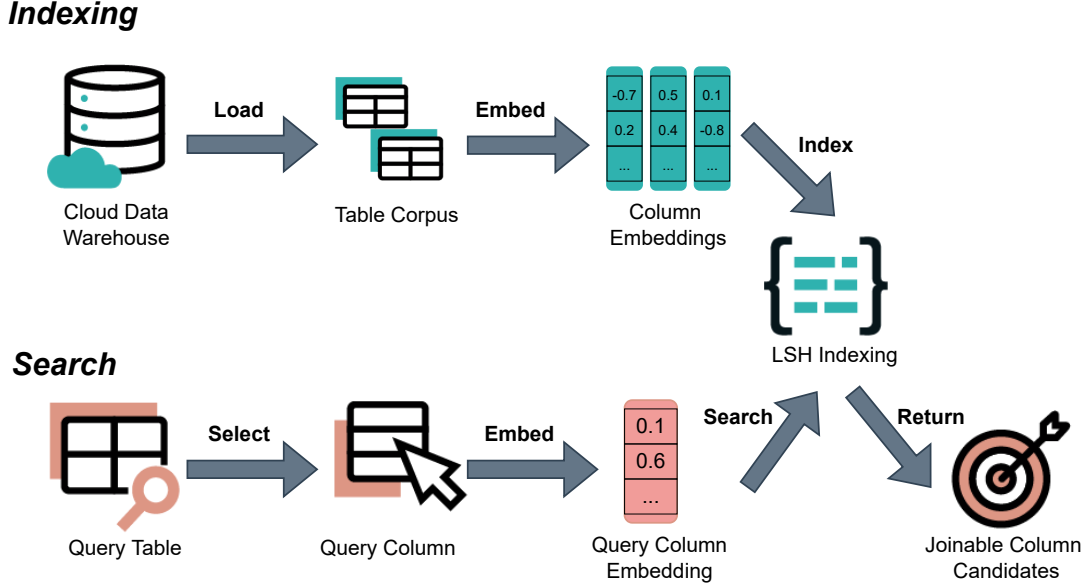


Figure 4.2: WARP GATE pipeline.

4.3.1 Embedding-Based Solution

Figure 4.2 gives an overview of the WARP GATE architecture, which consists of two main pipelines: indexing and search, which we consider in turn below. But first, we discuss the embedding step common to both pipelines.

4.3.1.1 Column Embeddings

We consider embeddings at the column level an effective encapsulation of semantics for join discovery. Intuitively, embeddings can capture semantic relationships between columns across databases where value formatting may vary widely despite semantic proximity.

However, the effectiveness of embeddings depends on the embedding model. We consider several factors in choosing an embedding model:

- (i) Whether the model is (pre-)trained over tabular data;
- (ii) The size of the training corpus;
- (iii) The efficiency of model inference.

We observe two primary paradigms in applying embeddings for data management tasks. Early work [20] leverages off-the-shelf embedding models from the natural language processing (NLP) field. Recent work [47, 72] adapts NLP model architectures and training algorithms for tabular data. As demonstrated in their work, embeddings from models trained

on tabular data manifest better performance in downstream tasks. For instance, using the same training algorithm, [72] shows that models trained on sequences extracted from tables outperform the model trained on an unstructured text corpus. The performance uplift is attributed to the serialization of tabular data, which considers the table structure in the embedding process. With this observation, we prefer embedding models designed for tabular data.

Empirically, more training data usually lead to better model performance and generalizability, given enough model capacity. In the database community, Web table corpora [57, 102] extracted from the Common Crawl contribute the largest publicly accessible relational table datasets. Efforts exploring representation learning for relational tables [47, 72] have trained embeddings on various Web table corpus, which also exhibits the impact of Web tables beyond Web applications. Thus, we favor embedding models derived from those large table corpora.

The embedding process for tabular data generally involves table serialization, input encoding (e.g., tokenization and transformation), and model inference. It is worth noting that although more complex models (in terms of the number of parameters) give embeddings that tend to yield better results in the downstream tasks, they also carry a larger inference cost (i.e., longer inference time, which directly affects query response time) compared to simple models. For interactive applications such as join discovery and recommendation in Sigma, we need to choose models that balance the embedding quality and model inference time.

4.3.1.2 Indexing

We employ locality-sensitive hashing (LSH) [67, 33] to turn the high-dimensional embedding similarity search problem into nearest neighbor search in a low-dimensional space. The basic idea of LSH is to maximize hash collisions for similar inputs. With respect to a similarity measure, LSH applies a specific family of hash functions that assign two similar inputs to the same “buckets” with high probability that is equal to the similarity score between them.

In our case, we consider the semantic join-ability of two columns to be the cosine similarity of their embeddings. To approximate the cosine similarity of two vectors, we rely on SimHash (also known as random projection) [33] that essentially uses hyperplanes in the vector space as hash functions. We hash all column embeddings from the table corpus into a SimHash LSH index. When a query comes in, the LSH index hashes the query embedding and only searches within the sub-universe of embedding vectors (much smaller than the entire universe) that share the same hash as the query, hence reducing the search time.

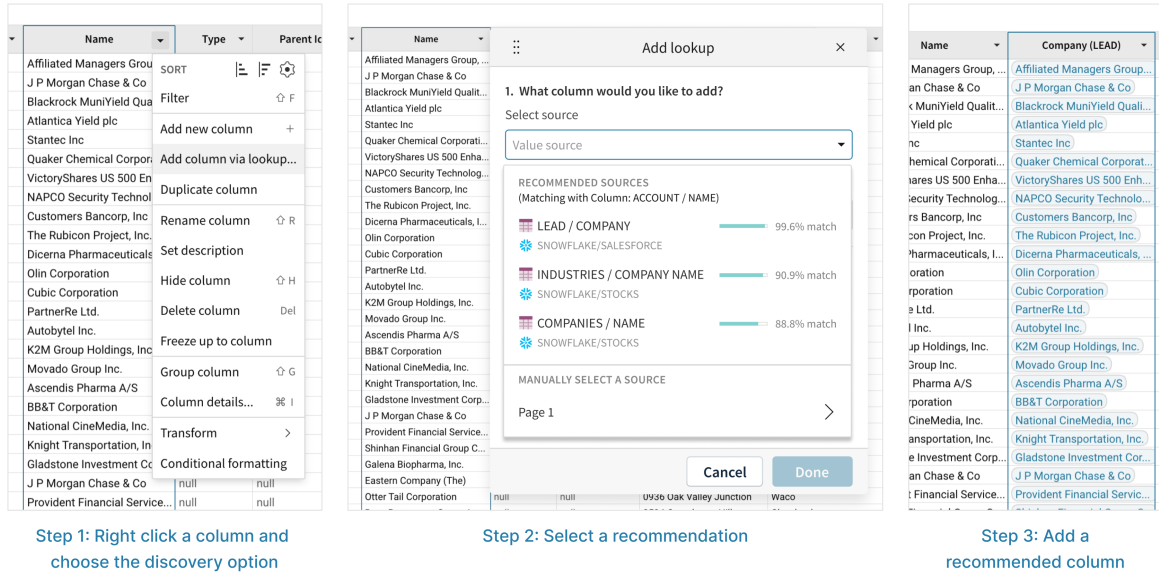


Figure 4.3: WARP GATE is integrated into a functionality called **Add column via lookup** in Sigma Workbooks.

4.3.1.3 Sampling

Existing data discovery systems [60, 17, 18] run a one-pass data scanning step to read all datasets once and retrieve profiling information (e.g., MinHash for each column). Reading full tables of millions of rows or even billions of rows from CDWs is expensive in terms of both computation time and monetary cost because CDW vendors that adopt a usage-based pricing model (i.e., pay-as-you-go) charge per GB of data scanned. It is then intuitive to use sampling to reduce the cost. But one side effect of sampling is potential disruptions to column profiles. For instance, profiling methods like MinHash are shown to be sensitive to the sample size [35]. It also remains unclear what a good sample size is for embeddings to preserve the similarity for data discovery.

4.3.2 WarpGate Interface

We have implemented the ideas described above in a prototype system we call WARP GATE. As shown in Figure 4.3, we integrate WARP GATE into Sigma Workbooks as a functionality called **Add column via lookup**.

Continuing the example of a business user Joey who wants to identify customers for a sales campaign, she opens up the **Account** table and sees a full list of companies (the **Name** column highlighted in Step 1 of Figure 4.3). To help make the decision, she wants to augment the table with additional information so that she can know which companies fall into the

scope of the sale campaign. In the current interface, she right-clicks on the **Name** column and chooses the **Add column via lookup** option in the menu. A window shows up (Step 2 in Figure 4.3) and displays top- k join path recommendations. Each recommendation gives the candidate join column, the table, and the database from which the candidate column comes, as well as a similarity score relative to the **Name** column (i.e., the query column). Although recommendations are ranked in descending order of the similarity score, Joey can pick one that fits her interest most. Once she chooses a candidate, the window will display a list of all the columns from the candidate table. Joey can browse and select the most relevant columns. For now, the interface will add user-selected columns to the **Account** table right at the side of the **Name** column for information complement and also load the candidate table into Sigma Workbooks for further inspection. In Step 3, we simply add and show the first recommended column, which is a foreign key in the same database.

The current interface design is based on the consideration that it is less desirable to simply join two wide tables with many columns and overwhelm users with a display of an even wider table that does not really help users with their information needs. In this case, Joey can first add the most relevant columns at first glance based on her domain knowledge while having the flexibility to see the entire candidate table for more careful examination.

4.4 Experiments

We evaluate WARP GATE on effectiveness and efficiency in comparison with two existing data discovery systems that support enterprise use cases. We obtain promising results of our embedding-based approach and show that it is sample-efficient and scales well to tables of millions of rows.

4.4.1 Datasets

We use three repositories for evaluation. Table 4.1 gives a summary of dataset characteristics.

Nextia_{JD}. Flores et al. [63] composes four testbeds of datasets from open repositories such as Kaggle and OpenML. Datasets are divided into testbeds according to their file size. For instance, Nextia_{JD}-XS contains datasets of size smaller than 1 MB while Nextia_{JD}-L consists of datasets of size larger than 1 GB. They also label the join quality of pairs of attributes based on a measure that considers both containment and cardinality proportion with empirically determined thresholds. In experiments, we use attribute pairs with quality labeled as Good and High by [63].

Spider. Released as a large-scale semantic parsing and text-to-SQL dataset, Spider [178]

Table 4.1: Basic statistics of evaluation datasets. XS is shorthand for testbedXS in the NextiaJD repository, similarly for S, M, and L.

	# Tables	# Columns	Avg. # Rows	# Queries	Avg. # Answers
<i>XS</i>	28	257	1,938	35	2.8
<i>S</i>	46	2,553	209,646	177	3.6
<i>M</i>	46	1,067	3,175,904	188	4.4
<i>L</i>	19	541	12,288,165	92	3.6
<i>Spider</i>	70	429	7632	60	1.1
<i>Sigma</i>	98	1,343	2,243,932	N/A	N/A

includes 5,693 SQL queries on 200 databases across domains. We parse schema SQL files and retrieve join paths between primary keys and foreign keys as ground truth. Spider has both the training set and the development set, and we currently use join paths from the development set for evaluation.

Sigma. The Sigma Sample Database is a collection of schema and tables that are accessible to all Sigma Computing accounts for exploration. It is stored in a Snowflake account managed by Sigma Computing and changes over time. The datasets in this database vary in origin; some are real, publicly accessible data, some are real data that has been obfuscated, and some are completely auto-generated data. The domain of the data ranges includes retail (transactions, products and stores), financial (daily attributes of many securities), demographic (census, restaurants and bikes) and usage (cloud application usage and metering, server logs).

4.4.2 Baselines & Metrics

For comparison, we consider as baselines two system prototypes Aurum [60] and D³L [17] that report on real-world data discovery (see Section 4.6 for more information). Note that Aurum stores detected relationships in a graph data structure and does not support top- k search.

When ground truth is available, we report top- k precision and recall with varying k . At each value of k , we average the precision and recall numbers over all the queries. We also limit k to a small range since the average number of answers to a query is small in evaluation datasets, and we don’t want to overwhelm users with too many recommendations.

Besides effectiveness, we consider index lookup time and end-to-end query response time

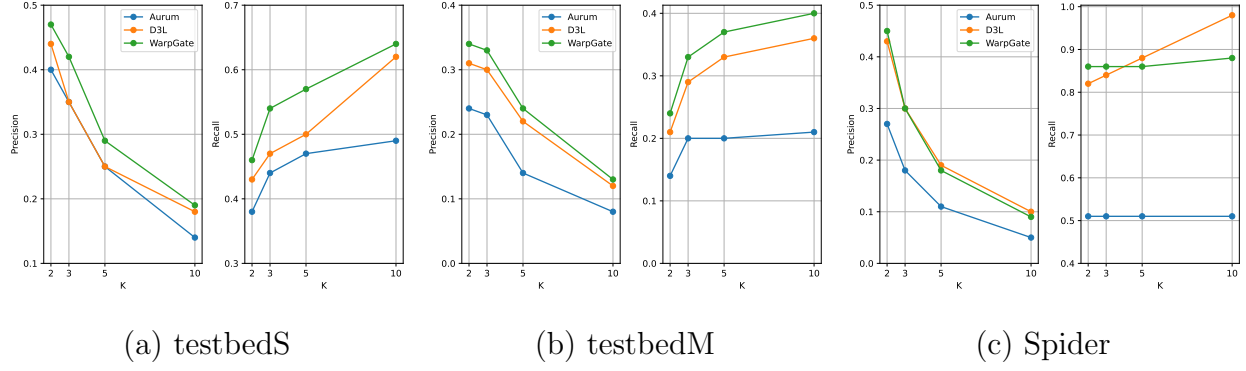


Figure 4.4: Top- k precision and recall.

for efficiency measurement. Index lookup time refers to the time for the index data structure to return top- k candidates, whereas end-to-end query response time is the total amount of time for a discovery system to respond to a query, which includes data loading and embedding inference time in our case. We report both metrics in seconds averaged over all queries in a dataset. It is crucial in our application scenario that the system responds to queries at interactive speed.

4.4.3 Experiment Results

Based on considerations in Section 4.3.1, we choose to use Web Table Embeddings [72] as our underlying embedding model. We set the similarity threshold of the SimHash LSH index to 0.7 and run baselines with their default setting. All experiments are conducted on an Amazon EC2 p3.8xlarge instance.

4.4.3.1 Join Discovery on nextiaJD Testbeds

We present discovery results on Nextia_{JD} testbedS and testbedM in Figure 4.4(a) and 4.4(b) respectively (results on four testbeds have similar trends while these two testbeds contain more tables and queries than the other two). As shown, WARP GATE consistently obtains higher precision and recall compared with two baselines as k increases.

Table 4.2 shows end-to-end query response time. Aurum is significantly faster because it indexes detected relationships in another graph data structure for result retrieval. In contrast, D³L is the slowest since it uses an ensemble approach and aggregating multiple types of signals takes longer. As we can also see here, given the equal number of tables in two testbeds, when the average number of rows increases by an order of magnitude, query response time increases linearly and becomes unacceptable for interactive purposes.

Table 4.2: End-to-end query response time in seconds/query (for $k = 10$) along with index lookup time reported in parentheses for WARPGate.

	Aurum	D ³ L	WARP Gate
<i>testbedS</i>	0.18	4.77	3.12 (1.04)
<i>testbedM</i>	0.03	57.69	38.73 (8.39)

Moreover, index lookup time accounts for less than 25% of query response time on testbedS and less than 13% on testbedM. This suggests that there are other bottlenecks in the pipeline and improving only index lookup time (as many earlier works focus on) is not enough to reduce overall query response time. Upon inspection, we found loading large datasets from disk to memory and embedding inference take most of the time besides index lookup. We show in Section 4.4.4 that sampling is an effective mitigation and our embedding approach is robust to sample size.

4.4.3.2 PK/FK Detection on Spider

Figure 4.4(c) shows the top- k precision and recall of three systems on the Spider dataset. We use this dataset mainly to demonstrate that for PK/FK detection within each independent database, the embedding measure alone (WARPGate) can compare favorably against the ensemble approach (D³L) and outperform the syntactic-only approach (Aurum) by a large margin. As to D³L, the jump of recall from $k = 5$ to $k = 10$ is attributed to the column name similarity measure in D³L, and we indeed observe in Spider that many PK/FKs share syntactically similar column names. Since Spider has a small number of tables that contain thousands of rows on average, the search time is fast for all three systems, which is no more than 2 seconds for running all the queries.

4.4.3.3 Ad-Hoc Discovery in Sigma

Since Sigma Sample Database is a corpus without ground truth, we ask four colleagues to pick columns of interest as queries. Here we describe one of the most interesting discoveries. In future work, we plan to conduct a user study to evaluate the value of WARPGate in a principled manner.

Continuing our running example of the business user Joey, who is interested in looking for additional information to help make the decision of customer selection. Figure 4.3 displays the top-3 recommendations from WARPGate. The first candidate is the **Company** column of the **LEAD** table from the same database as the query column, which contains information

on contact points in each company (e.g., name, title, and address of each contact point). Upon browsing columns in the **LEAD** table, Joey does not find auxiliary information for her need. She moves to inspect the second candidate, which is the **Company Name** column of the **INDUSTRIES** table from the **STOCKS** database. In the same table, the **Industry Group** column has sector information of US public companies, which can (partially) enrich the query table. Joey then acts on her customer selection by filtering companies with their sectors. Even more interestingly, she can add to the query table the **TICKER** column from the **INDUSTRIES** table, which can be further used as the join key to add columns of stock prices from the same **STOCKS** database. This way, Joey can track down high-performing companies in targeted sectors as her sales campaign customers.

4.4.4 Sample Efficiency

We run **WARP**GATE over **Nextia_{JD}-S** and **Nextia_{JD}-M** with sample size 10, 100, and 1000. For all sample sizes, the embedding approach remains as effective as using full values with $\pm 1\%/2\%$ variation at different k values. In the meanwhile, index lookup time is reduced by up to two orders of magnitude (e.g., from about 1 second to about 10 milliseconds on **Nextia_{JD}-S**), and end-to-end query response time also comes down to interactive speed (under 35 milliseconds per query on **Nextia_{JD}-S** and under 65 milliseconds per query on **Nextia_{JD}-M**). We discuss further optimization opportunities in Section 4.5.

To see if sample efficiency is unique to Web Table Embeddings, we also evaluate BERT [49] as our underlying embedding model. We observe that the effectiveness of BERT Embeddings is robust to sample size as well. Although BERT is a significantly more complex model, we do not see much improvement in effectiveness (mostly on par with Web Table Embeddings). Nevertheless, index lookup time and query response time are 10x slower without sampling due to more expensive embedding inference costs.

4.5 Discussion

4.5.1 Sigma Customer Data Scale

The scale of enterprise CDWs is a challenge for any deployment of join discovery. The median Sigma customer warehouse has 450 tables for analysis, but the average is over 12,700 tables with an average of 25.7 columns per table. The size of enterprise tables is also a challenge. The median table in these warehouses has just 7,700 rows, but the average is 1.7 billion rows. Actively sampling this large number of tables incurs usage costs in CDWs, so

a passive sampling of user queries is beneficial. Likewise, samples can be shared across ML applications in Sigma to amortize their cost.

4.5.2 Optimization Opportunities

4.5.2.1 Contextual Embeddings.

Existing approaches generally follow a two-step process: they first build profiles for each column independently and then determine join-ability of two columns based on some notion of similarity between their profiles (e.g., containment or Jaccard similarity). Typically, the context of a column is not taken into account when building those profiles. However, context (e.g., other columns in the same table, user activities, query logs) can potentially provide auxiliary information that is critical to find semantically related candidates. We plan to explore the option of incorporating context information into the underlying embedding model of WARP GATE.

4.5.2.2 Data Management for Data Discovery.

CSV is a common file format for tabular data in Open Data Lakes and table repositories on the Web. Despite the flexibility and human readability of CSV files, the flat-file format is not ideal for storage and query purposes. We note that (1) loading giant CSV files and running embedding inference can incur heavy memory usage and crash programs on small machines; (2) operations in join discovery like embedding inference are column-oriented; (3) what is unique in our scenario is that our data are pulled from CDWs, thus more structured and much better data quality than those in the wild. (1) and (2) make us think it is more memory and storage efficient to take advantage of (in-memory) column stores for data pulled out of CDWs, and (3) can make the process easier. This change will also have significant implications for deployment as we need to provision CPU and memory usage in Sigma’s Kubernetes cluster.

4.5.2.3 Efficient Search.

To further improve query response time for Sigma customers with a massive number of tables in their CDWs, we are considering two directions: (1) a block-and-verify strategy in [53], which employs pivot-based filtering to prune dissimilar vectors; (2) fine-tune off-the-shelf embedding models in a self-supervised way that pushes embeddings of joinable columns to have higher cosine similarity so that an index data structure like SimHash can be better utilized [43].

4.6 Related Work

WARPGATE builds on earlier work on data discovery for enterprise data systems and Open Data, which we summarize below. We distinguish WARPGATE from this earlier work mainly in two aspects: (i) we recognize and showcase the value of data discovery for structured data in enterprise CDWs; (ii) we do not assume a full pass of data in the indexing/profiling phase and show that the embedding-based approach is effective and robust to data samples.

Data Discovery Systems for Enterprise. Aurum [60] models syntactic relationships between datasets in a graph data structure to support data discovery. In a two-step process of profiling data and computing the relationship between data signatures, Aurum builds up a graph with nodes representing column profiles and weighted edges indicating syntactic relationships between two nodes (e.g., content similarity). With an in-memory index of the graph, Aurum can efficiently support various discovery needs, such as how similar two columns are, which can be used to find joinable datasets. SemProp [62] extends Aurum by using word embeddings to surface semantically related objects in the graph.

Voyager [18] built on top of D³L [17] formalizes and supports three tasks routinely performed by data scientists using their platform. It profiles data with a full scan and indexes columns with MinHash. Voyager also drops the embedding measure in D³L and relies only on syntactic measures.

Join Discovery over Open Data. JOSIE [191] considers the problem of finding joinable tables in modern data lakes. It formulates the problem as a top-k overlap set similarity search problem where columns are treated as sets and matching values as intersections between sets. By using a cost model for estimating the reading costs of data structures, JOSIE contributes an exact overlap set similarity search algorithm that can adapt to different data distributions.

Juneu [187] aims to help users find semantically related datasets in data lakes to augment data analytics tasks, including finding joinable tables in interactive data science environments (e.g., Jupyter Notebook). It combines multiple table-relatedness measures such as column and row overlap, provenance, and user context and intent for related-table search. In a similar vein, D³L [17] uses five types of evidence to determine column proximity: (i) column name similarity; (ii) column extent overlap; (iii) word-embedding similarity; (iv) format representation similarity; and (v) domain distribution similarity for numerical attributes. This ensemble approach is shown to be more effective and efficient than Aurum on table union search, which is a problem that has significant overlap with join path discovery.

In contrast to the search paradigm, Nargesian et al. [122] propose building a navigation graph over data lakes. By leveraging a probabilistic model, they show that navigation can help users find relevant tables that cannot be found by keyword search.

4.7 Conclusion

We present WARP_{GATE}, a system prototype for semantic join discovery in Sigma Workbooks. In future work, we expect to share our experience and lessons of deploying WARP_{GATE} in the production environment and conduct a user study to further evaluate its value for end users.

CHAPTER 5

Nexus: Inferring Join Graphs from Metadata Alone via Iterative Low-Rank Matrix Completion

5.1 Introduction

Detecting join relationships within a collection of tables is essential for tasks such as identifying primary and foreign keys (PK/FKs) in relational databases [143, 182, 35, 87], constructing Business Intelligence (BI) models for analytical workloads [16, 104], and finding related data in diverse sources like Web corpora [26, 25, 144, 135] and data lakes [192, 60, 191, 187, 53, 42, 54, 48].

This capability is particularly important for modern lakehouse platforms, which face unique challenges including minimal metadata (e.g., PK/FK relationships are not declared), data heterogeneity, and decentralized data ingestion and consumption. Uncovering all join relationships, collectively forming what we term the *join graph*, within a set of tables, is a foundational capability that enables key functionalities in these architectures, including automated data cataloging and governance, improved data search and exploration, machine learning (ML) feature engineering, knowledge graph construction, and natural language query support.

In this work, each node in a *join graph* represents a table column and each edge indicates a join relationship between two columns. Given a collection of tables, we address the problem of *join graph inference* where our objective is to infer the adjacency matrix of the corresponding join graph by uncovering all meaningful join relationships among the tables.

Existing approaches for join graph inference broadly fall into four categories. *Traditional non-ML methods* [114, 35] leverage heuristics like column name similarity, data type compatibility, and value-based overlap measures to detect joinable columns. However, these methods often miss semantically related columns with different names, and value overlap tests are

computationally expensive, especially for large tables. *Embedding-based approaches* [54, 42] enhance traditional methods by representing table names, column names, and column values as embeddings in a vector space. They then use approximate nearest neighbor search to identify joinable columns. Although these approaches can uncover joins beyond exact matches, embeddings provide fuzzy similarity rather than guaranteed correctness, yielding many false positives [110]. *ML-based methods* [143, 16] train models to predict join relationships using features such as value overlap, embedding similarity, and column statistics. However, feature computation and model training are costly, and pretrained models often generalize poorly to unseen datasets. For instance, in BI contexts, Lin et al. [104] observed that even ML models trained on over 100,000 real BI models with PK/FK definitions can still fall short in accuracy. Recognizing that BI models mostly follow arborescence (snowflake or multi-snowflake) structures, they proposed Auto-BI, which applies a graph-based global optimization (k-Min-Cost-Arborescence) on top of ML predictions, significantly improving accuracy over pure ML models. However, Auto-BI’s reliance on arborescence structures makes it too restrictive for non-BI contexts, particularly for lakehouses, where schemas are typically more loosely structured. More recently, *Large Language Models (LLMs)* have been explored for join prediction by prompting them with metadata and, optionally, sample data [51]. While promising, LLMs are prone to hallucination and incur high computational costs, especially compared to traditional methods.

Importantly, most existing approaches rely on both metadata and actual data values to detect join relationships, with very few methods exclusively using metadata. Utilizing data values—whether for direct value overlap detection, embedding generation, or feature computation for ML models—is computationally expensive, especially when dealing with large datasets containing numerous and sizable tables. Moreover, the assumption that join detection algorithms can freely access data values is often *unrealistic*. Data service providers, for example, operate under strict privacy and security constraints that limit access to customer data, even if such access could improve service usability. Therefore, *in this work, we only assume access to metadata*, excluding any column data values. Specifically, the metadata we consider includes only schema information and basic column statistics, which are typically available in standard catalog services for data platforms. Results presented in [16] show that ML models trained on metadata-based features can achieve adequate precision and recall for join detection in the context of Microsoft Cosmos, a large-scale analytics platform (although workloads on Cosmos are known to be highly recurrent [88]). These findings suggest that accurate join graph inference is feasible without accessing data values.

Additionally, without assuming that query logs are always available, they should be leveraged to improve join graph inference when accessible. Although the use of query logs for

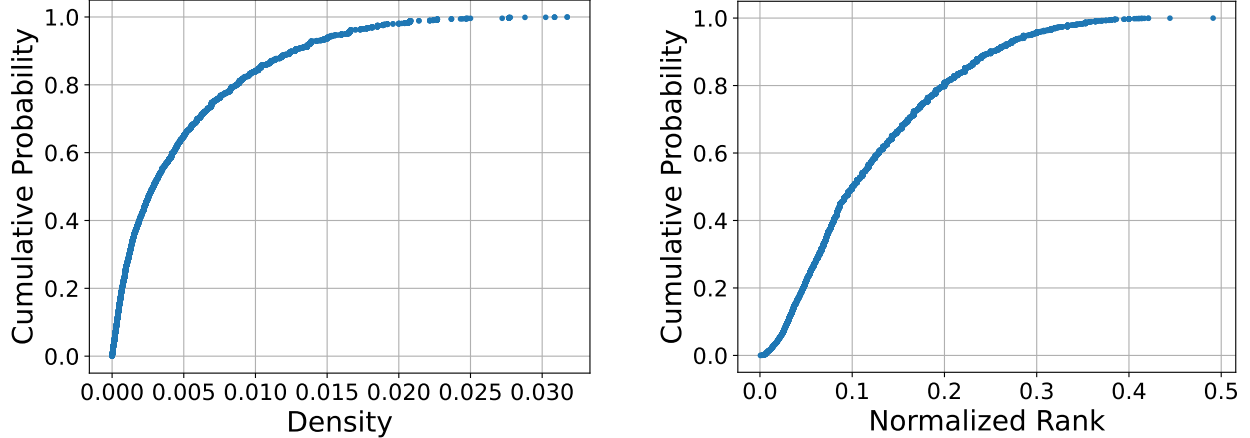


Figure 5.1: CDF of density and normalized rank of the curated real-world schemas.

discovering join relationships is rare in academic research, largely because such logs are proprietary, they are frequently available in industry settings, especially within cloud environments.

In this paper, we develop NEXUS, an *efficient* and *accurate* end-to-end solution for join graph inference. NEXUS leverages *only metadata information* and, when available, incorporates *query logs* to further improve its accuracy.

The design of NEXUS is guided by two key insights derived from analyzing 5,957 real-world schemas. When represented as adjacency matrices, *real-world join graphs* are (1) *highly sparse*, and (2) *exhibit a low-rank structure*. As shown by the empirical results in Figure 5.1, approximately 98% of these schemas have a density below 0.02, and over 95% have a normalized rank¹ less than 0.3. A more detailed analysis will be presented in Section 5.2.3. To the best of our knowledge, our discovery of high sparsity and low-rank structure in real-world join graphs is novel. High sparsity means that only a small fraction of all possible joins actually occur. Low-rankness suggests that much of the variability in the join graph matrix can be captured by a small number of latent factors. This redundancy means that a significant portion of the matrix can be approximated using a low-dimensional representation, revealing the inherent structure and dependencies among columns.

Based on the above insights, we formulate join graph inference as a *low-rank matrix completion problem*. Low-rank matrix completion aims to recover unknown or latent entries from a small number of observed entries, while ensuring that the recovered matrix has a low rank. In our context, the join graph is represented as an adjacency matrix. Before running the inference algorithm, entries that are pruned as invalid join candidates are set to 0 (known

¹The *normalized rank* of an $m \times n$ matrix A is defined as $\frac{\text{rank}(A)}{\min(m,n)}$.

zeros). If join relationships are observed in the query logs, the corresponding entries can be set to 1 (known ones). Although our approach does not require knowledge of existing joins from query logs, such information can be naturally incorporated to further improve matrix completion. For the remaining candidate entries, we leverage pretrained ML models or LLMs to score their likelihood of being true joins. We then augment the standard low-rank matrix completion objective with an additional loss term that encourages the recovered latent entries to align with these probability scores. This allows us to effectively leverage prior predictions from the pretrained models. Moreover, we introduce a regularization loss term to promote sparsity in the matrix, reflecting the observed sparsity of real-world join graphs.

In practice, pretrained models often struggle to generalize to new or unseen datasets, and forcing latent entries to conform to inaccurate predictions from these models can significantly degrade the quality of the inferred join graph. To address this challenge, we propose an Expectation-Maximization (EM) algorithm that iteratively refines the low-rank matrix completion by incorporating semantic guidance from LLMs. The algorithm alternates between two steps: (1) performing low-rank matrix completion, and (2) adjusting the prior probability estimates of join candidates using an LLM. A key aspect of our approach is leveraging an LLM to assess the semantic compatibility between column entity types, which is a prerequisite for a valid join. We define a *column entity type* as a free-text annotation that encapsulates the semantic meaning of a column’s values within the context of its table. For instance, an ID column in a Users table would be annotated as “user ID”.

Unlike traditional methods that depend on a fixed set of entity types, our approach does not require predefined vocabularies. This is because column semantics can vary widely across different tables and databases, and existing vocabularies fail to provide comprehensive coverage of column semantics. LLMs are particularly well-suited for this task due to their strong semantic understanding and ability to generalize across diverse contexts. We leverage the generative capabilities of LLMs to both annotate entity types on the fly and determine the semantic equivalence between column entity types. In our algorithm, when the LLM determines a strong semantic match between the entity types of two columns, we promote the corresponding latent entry by increasing its probability score. Conversely, if the LLM finds a semantic mismatch, we demote the entry by decreasing its probability score. This iterative refinement process allows NEXUS to effectively correct poor initial predictions from pretrained models and adapt to the unique semantics of unseen datasets.

To summarize, this paper makes the following contributions:

- We propose NEXUS, an efficient and accurate end-to-end solution for join graph inference, tailored for enterprise environments where data privacy is a high priority. By relying on metadata only, and optionally leveraging query logs when available, NEXUS

operates effectively without requiring direct access to data values.

- Our extensive empirical analysis of nearly 6K real-world schemas reveals a novel insight: join graphs, represented as adjacency matrices, are not only sparse but also consistently low-rank. This new discovery forms the foundation for our novel join graph inference algorithm.
- Building on our discovery, we formulate join graph inference as a low-rank matrix completion problem, leveraging pretrained models to score the likelihood of potential joins. We then introduce a novel optimization objective that incorporates both the observed sparsity and low-rank properties of real-world join graphs.
- To overcome the challenge of poor prior predictions from pretrained models, we propose a novel Expectation-Maximization algorithm that alternates between performing low-rank matrix completion and leveraging an LLM to iteratively refine the probability scores of join candidates based on their semantic compatibility.
- We demonstrate the effectiveness and efficiency of NEXUS on three datasets including a real production dataset. Our metadata-only approach significantly outperforms baselines and compares favorably to methods using data values.

The rest of the paper is organized as follows: Section 5.2 introduces preliminaries and provides a detailed analysis on the structures of real-world schemas. Section 5.3 describes the overall architecture and the various components of NEXUS. Section 5.4 presents our extensive experimental study, while Section 5.5 reviews the related work. And finally, Section 5.6 concludes this paper.

5.2 Preliminaries and Observations

This section establishes the foundation for our join graph inference solution. We will start by introducing key concepts and formally defining the problem. Following that, we will present findings from an empirical study on a large number of real-world database schemas, which directly motivate the design of our algorithm.

5.2.1 Join Graphs

We first give the definitions of join graphs, join graph matrices, and join graph probability matrices.

Definition 6 (Join Graph). Given a set of tables T , a join graph is an undirected graph $G(T) = (V, E)$ where:

- $V = \{c_1, c_2, \dots, c_n\}$ represents the set of all columns from the tables in T .
- E is the set of join relationships. An edge (c_i, c_j) exists between two columns c_i and c_j , if and only if a join relationship exists between them.

When the context is clear, we will refer to the join graph $G(T)$ simply as G .

Definition 7 (Join Graph Matrix). Let $G = (V, E)$ be the join graph for a set of tables T . The join graph matrix $A(T) \in \{0, 1\}^{n \times n}$, or simply denoted as A when there is no ambiguity, is the adjacency matrix of the join graph G :

$$A_{i,j} = \begin{cases} 1 & \text{if } (c_i, c_j) \in E \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

Since we consider the join graph undirected, the resulting matrix A is symmetric ($A_{i,j} = A_{j,i}$) with a zero diagonal ($A_{i,i} = 0$).

Definition 8 (Join Graph Probability Matrix). Let $G = (V, E)$ be the join graph for a set of tables T . The join graph probability matrix $S(T) \in [0, 1]^{n \times n}$, or simply denoted as S in case of no ambiguity, is an $n \times n$ matrix, where each entry $S_{i,j}$ denotes the probability that column c_i and column c_j are joinable.

S is also a symmetric matrix ($S_{i,j} = S_{j,i}$) with a zero diagonal ($S_{i,i} = 0$) due to the undirected nature of join graphs.

5.2.2 Problem Statement

In this work, the problem of *Join Graph Inference* is to reconstruct the join graph matrix A for a given set of tables T , *using only metadata*. Automatically inferring missing join relationships between relational tables is crucial for efficient data onboarding and reuse in enterprise environments. Users frequently rely on these relationships to effectively navigate and query their data. Join relationships are often missing or undeclared for various reasons, such as the intentional removal of primary/foreign key constraints to improve ingestion performance or data loss during migration processes. This challenge is amplified in enterprises adopting modern lakehouse architecture, where schemas are typically minimal and loosely structured. In such environments, the sheer volume of tables—often numbering in

Table 5.1: Statistical summary of the curated schemas.

Metric	Min	Max	Mean	Median
# of tables	4	977	15	9
# of columns	17	4572	108	56
Avg. columns / table	4	37	7	6
Density	1.6×10^{-7}	0.032	0.005	0.003
Normalized rank	5.6×10^{-4}	0.49	0.12	0.10

the hundreds or thousands—renders manual exploration for discovering these relationships both impractical and ineffective.

We focus on a metadata-only setup for two main reasons. First, assuming access to the underlying data is often untenable in enterprise settings due to privacy restrictions. Second, even when data access is granted, current approaches that calculate Jaccard similarity or containment between columns require full data scans. These operations become computationally prohibitive for tables with millions, let alone billions, of rows.

The metadata we consider in this work includes table names, column names, column data types, and simple column statistics (total number of values, number of distinct values, number of nulls, and minimum and maximum values). These basic statistics are commonly available in the catalog service of a lakehouse platform for the purpose of query optimization. We simply reuse them for join graph inference. Critically, and in line with enterprise privacy and security requirements, our approach assumes no access to the actual data values within the columns.

Our primary goal is to detect one-to-one (1:1) and many-to-one (N:1) relationships, which typically correspond to PK/FK joins. If needed, many-to-many (N:N) relationships can subsequently be derived in a post-processing step via the transitive closure of the initially detected joins. In addition, while this paper focuses on single-column joins, our solution can be extended to multi-column (composite key) joins, as discussed further in Section 5.3.7.

5.2.3 Empirical Analysis and Observations

To inform the design of our join graph inference algorithm, we conducted an extensive empirical study to understand the characteristics of real-world join graphs.

Our analysis used the SchemaPile-Perm dataset [51], which contains over 22,989 relational schemas from public GitHub repositories. However, this dataset required significant filtering

before analysis. We found that over 49% of the schemas lacked join relationship information. Many remaining schemas were also trivial; for example, nearly 29% had three or fewer tables, and over 30% averaged four or fewer columns per table. To focus on realistic, non-trivial databases, we filtered for schemas with more than three tables and an average of over four columns per table. Additionally, we excluded the 7.5% of schemas with multi-column joins to focus on single-column joins. This curation process resulted in a dataset of 5,957 schemas, which served as the basis for our empirical study.

Table 5.1 presents a statistical summary of our curated dataset. Besides basic statistics, our analysis focused on two critical properties of the join graph matrix for each schema: *sparsity* and *rank*, leading to the discovery of the following two key insights.

- *High Sparsity*: We measured sparsity using matrix density, defined as $\frac{\# \text{nonzeros}}{n^2}$ for an $n \times n$ matrix. The results confirm that real-world join graphs are extremely sparse, with an average density below 0.005 and a median density under 0.003. As shown in Figure 5.1, approximately 98% of schemas having a density less than 0.02.
- *Low-Rank Structure*: We assessed the low-rankness of join graphs using the normalized rank, which is defined as $\frac{\text{rank}(A)}{n}$ for an $n \times n$ matrix A . This metric indicates how close a matrix is to being full-rank. We found that the average and median normalized ranks were only 0.12 and 0.10, respectively. As Figure 5.1 illustrates, over 95% of the schemas have a normalized rank below 0.3.

These two key observations, high sparsity and low-rankness of join graphs, imply that most columns in a schema are not directly joinable and that the overall join structure contains significant redundancy, allowing it to be represented compactly.

These empirical observations are also intuitive. Columns in a schema that exhibit similar joinability patterns with a subset of other columns are likely governed by shared underlying semantics or schema characteristics. As a result, their joinability behavior tends to generalize across the matrix, implying that much of the variability in the join graph matrix can be captured by a small number of latent factors. Specifically, this redundancy suggests that the join graph matrix can be approximated using a low-dimensional representation, revealing the inherent structure and dependencies among columns. In the following section, we describe our novel join graph inference algorithm based on these two key observations.

5.3 Nexus Design

In this section, we provide an architecture overview of our join graph inference solution, NEXUS, and provide details on each core system components.

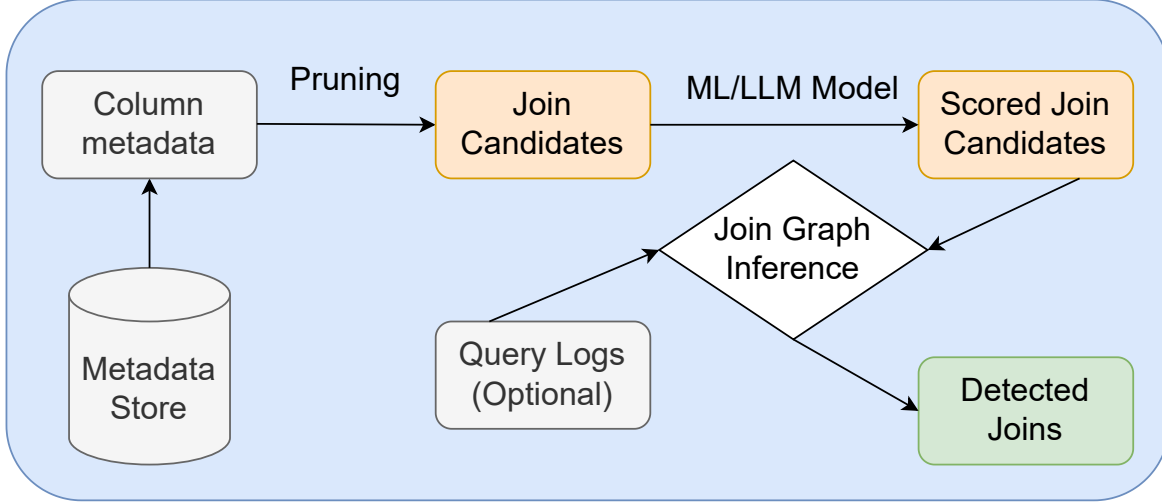


Figure 5.2: Overview of NEXUS pipeline that infers join graphs from only metadata.

5.3.1 Overview

Figure 5.2 illustrates the overall workflow of NEXUS. The process begins by retrieving schema information and column-level metadata for a given collection of tables from the metadata store. These metadata are then used to prune unlikely join column pairs, significantly reducing the number of potential join candidates. Next, each candidate is scored using a pre-trained model, which can be either an ML model or an LLM. These scored candidates, optionally supplemented with known joins extracted from query logs, form a join graph probability matrix. In this matrix, non-candidate pairs are assigned a probability of 0, known joins from query logs are given a probability of 1.0, and the remaining entries are populated with scores from the pre-trained model. The probability matrix is subsequently fed into a matrix completion algorithm, which is based on our observations of high sparsity and low-rank structure within the join graph matrix. This algorithm is further enhanced by an iterative Expectation-Maximization (EM) procedure designed to mitigate any inaccuracies from the pre-trained model. The output is the final inferred join graph matrix, which identifies the detected join relationships.

5.3.2 Metadata and Join Candidates Pruning

Exhaustively evaluating every pair of columns as a potential join candidate incurs a quadratic cost relative to the number of columns, making it impractical for large datasets with thousands of columns. To address this, we collect column-level metadata from the metadata stores and apply cheap filters to eliminate a large number of unlikely join candidates early

on.

We begin by discarding columns that are unlikely to serve as join keys, such as nested types (e.g., `JSON`, `ARRAY`, `MAP`), measure types (e.g., `FLOAT`, `DOUBLE`, `DECIMAL`), as well as `BLOB` and `BOOLEAN` types. For the remaining columns, we leverage the data type and the basic statistics from the metadata store, including cardinality (number of unique values), null value ratio, and min/max values. These statistics are typically maintained in metadata stores for query optimization purposes. Column pairs with incompatible data types (e.g., `INTEGER` vs. `VARCHAR`) are pruned immediately. Additionally, when inferring N:1 or 1:1 joins, we apply further pruning criteria, including ensuring uniqueness on the "1" side, verifying that the cardinality on the "N" side does not exceed that of the "1" side, and confirming that the value domain of the "N" side is a subset of the "1" side. These pruning criteria can be implemented in SQL syntax and executed efficiently in database systems.

While more sophisticated profiling and pruning techniques exist [99, 87, 89], they typically incur higher computational costs. We opt for these simpler, more efficient metadata-only filters because they effectively eliminate a large proportion of true negatives with minimal overhead.

5.3.3 Scoring Join Candidates

The join candidates produced by the initial pruning step may still contain many false positives. For instance, a user ID column and a product ID column may share compatible data types and value ranges, yet they are not semantically joinable. Prior works [143, 16, 104] have typically employed ML models that combine multiple signals, such as embedding similarities of column names and Jaccard similarities of column values, to predict join relationships and estimate the associated probability scores. However, prior works rely on computing ML features that require access to the actual data values. Such computations are expensive and become prohibitive for large tables, often containing millions or billions of rows in enterprise environments. In contrast to most prior work, we assume no access to data values. We employ two metadata-based strategies to estimate the probability scores of join candidates:

- *Pre-trained ML Model:* We pre-train an ML model exclusively on metadata-derived features. These include pairwise features derived from column names (max containment, Jaccard similarity, the ratio of edit distance to max length, Jaro-Winkler similarity, and the cosine similarity of weighted embeddings), and column-wise features (data type encodings, cardinality ratios, and null-value number ratios).
- *LLM:* We can leverage an LLM as a pre-trained model, prompting it directly for both predictions and associated confidence levels, categorized as low, medium, and high.

These qualitative confidence levels are then mapped to predefined probability scores (e.g., low $\rightarrow$ 0.1, medium $\rightarrow$ 0.6, and high $\rightarrow$ 0.9). This indirect elicitation of confidence is motivated by the known limitations of LLMs in generating reliable, well-calibrated confidence scores [108, 44, 132].

After this step, we construct a join graph probability matrix using the predicted scores from the pre-trained ML or LLM model, optionally augmented with confirmed join column pairs extracted from query logs, if available.

Despite the usefulness of pretrained models, they face two significant limitations: (1) their predictions are inherently local, considering only the features of individual column pairs and ignoring the global structure of the join graph, and (2) models trained on one dataset may not generalize effectively to others, and their confidence scores can be misleading. We address these critical challenges in Sections 5.3.4 and 5.3.5, respectively.

5.3.4 Join Graph Inference through Low-Rank Matrix Completion

After initial pruning and scoring, we obtain a join graph probability matrix, S . This matrix is partially observed: some entries are considered certain, having probabilities fixed at 0 (pruned) or 1 (from query logs), while the rest hold uncertain confidence scores from the pre-trained model. Let $\Omega = \{(i, j) \mid S_{i,j} \in \{0, 1\}, 0 \leq i, j < n\}$ denote the set of positions of certain entries in S .

The objective of join graph inference is to recover a matrix $M \in [0, 1]^{n \times n}$ that is a faithful approximation of the unknown ground truth join graph matrix A . This recovery process only affects the uncertain entries of S . Drawing from our key observations that real-world join graphs are highly sparse and low-rank, we formulate this inference task as a low-rank matrix completion problem.

The standard formulation of low-rank matrix completion seeks to minimize the rank of M while preserving the observed values. Formally, the optimization objective can be written as:

$$\begin{aligned} & \text{minimize} \quad \text{rank}(M) \\ & \text{subject to} \quad M_{i,j} = A_{i,j} \text{ for } (i, j) \in \Omega \end{aligned} \tag{5.2}$$

where $\text{rank}(\cdot)$ gives the rank of a matrix.

This is a non-convex optimization problem due to the rank function and is known to be NP-hard [37]. A well-established convex relaxation approximates the rank function with the

nuclear norm of the matrix [27], resulting in the following optimization objective:

$$\begin{aligned}
& \text{minimize} \quad ||M||_* = \sum_{k=1}^n \sigma_k(M) \\
& \text{subject to} \quad M_{i,j} = A_{i,j} \text{ for } (i,j) \in \Omega
\end{aligned} \tag{5.3}$$

where $\sigma_k(\cdot)$ denotes the k -th largest singular value of a matrix and the nuclear norm $||\cdot||_*$ is the sum of all singular values.

While alternative approaches to low-rank matrix completion exist, such as matrix factorization [94, 85], they require pre-specifying the rank of the matrix to be recovered. Such prior knowledge of the matrix rank is not readily available in our setting. On the other hand, nuclear norm relaxation is generally considered less scalable than matrix factorization, as it needs to estimate all the parameters corresponding to the latent entries, which increase quadratically with respect to the matrix dimension. However, this concern is largely mitigated in our context due to a crucial observation: real-world join graphs are extremely sparse. Specifically, we observed that the detected join candidates (after pruning) constitute less than 5% of the entries in the matrix we need to recover. Consequently, the number of parameters in our optimization is dramatically reduced, making the nuclear norm approach both feasible and efficient. This key insight not only addresses the scalability issue but also motivates us to explicitly incorporate a sparsity-inducing constraint into our recovery objective, ensuring the final solution reflects this fundamental property of join graphs.

Additionally, we have access to the confidence scores for join candidates in the join graph probability matrix S , provided by the pre-trained model. These scores serve as prior information to guide the recovery of the latent entries in M .

Let $\bar{\Omega}$ denote the complement of Ω , which contains the set of positions for the uncertain entries in our probability matrix S . To integrate the observed sparsity and the prior beliefs, we propose the following optimization objective:

$$\begin{aligned}
& \text{minimize} \quad ||P_{\bar{\Omega}}(S - M)||_F^2 + \lambda_1 ||M||_* + \lambda_2 ||M||_1 \\
& \text{subject to} \quad M_{i,j} = A_{i,j} \text{ for } (i,j) \in \Omega
\end{aligned} \tag{5.4}$$

Here, $P_{\bar{\Omega}}$ is a projection operator that preserves the entries with positions in $\bar{\Omega}$ while zeroing out all others. $||\cdot||_F$ and $||\cdot||_1$ denote the Frobenius norm and the 1-norm of a matrix, respectively.

The above object function can be broken down into the following three components:

- *Confidence Score Regularization:* The term $||P_{\bar{\Omega}}(S - M)||_F^2$ encourages the final matrix M to align with the confidence scores from the pre-trained model, but only for the

uncertain entries in $\bar{\Omega}$.

- *Low-Rank Regularization*: The term $\lambda_1 \|M\|_*$ uses the nuclear norm to encourage the solution M to be low-rank, aligning with our key observation.
- *Sparsity Regularization*: The term $\lambda_2 \|M\|_1$ uses the L1 norm to promote sparsity in M , effectively pushing the probabilities of unlikely joins towards zero.

λ_1 and λ_2 are hyperparameters weighting the regularization power of the low-rank term and the sparsity term, respectively.

Note that the recovered matrix M contains real values in $[0, 1]$, rather than binary indicators. To produce the final binary decisions, we apply a threshold θ to the latent entries of M . The resulting decision matrix $\hat{M} \in \{0, 1\}^{n \times n}$ is defined as:

$$\hat{M}_{i,j} = \begin{cases} M_{i,j} & \text{if } (i, j) \in \Omega \\ \mathbf{1}(M_{i,j} \geq \theta) & \text{otherwise} \end{cases} \quad (5.5)$$

where $\mathbf{1}(\cdot)$ is the indicator function.

5.3.5 Iterative Join Graph Inference via Expectation-Maximization Algorithm

As shown in Equation 5.4, we (partially) align the recovered matrix with the probability matrix determined by a pre-trained ML/LLM model. This approach assumes that the model accurately estimates the probability scores of join candidates. However, this assumption may not hold when applying a pretrained model to unseen datasets with very different distributions from the training datasets. Consequently, forcing latent entries in the recovered matrix to match potentially inaccurate estimates can degrade result quality. To address this issue, we propose an Expectation-Maximization (EM) algorithm (Algorithm 3), which iteratively improves the quality of low-rank matrix completion using guidance from an LLM.

EM is a widely adopted approach in ML for computing maximum likelihood estimates in models with incomplete or latent variables [45]. The algorithm, outlined in Algorithm 3, operates iteratively in two main steps. The *Expectation step* (E-step) involves low-rank matrix completion to estimate the values of latent entries. This estimation is based on the observed data Ω and the probability scores $S^{(t)}$ given by an ML/LLM model. This E-step can be interpreted as determining the posterior probabilities of the latent entries given the observed data and the prior knowledge from the ML/LLM model. Subsequently, in the *Maximization step* (M-step), we refine the probability matrix by incorporating a semantic

Algorithm 3: Expectation-Maximization Algorithm for Iterative Join Graph Inference

Input : $S^{(0)}$, initial probability matrix;
 $\Omega^{(0)}$, initial set of known certain entries in $S^{(0)}$;
 Γ , max number of iterations;
 ϵ , tolerance for early stopping

Output: M , recovered matrix

```

/* Map from column to its entity type (cache)                                */
1  $col\_entity\_type = \{\}$ ;
2 for  $t = 0, 1, \dots, \Gamma - 1$  do
    /* E-step: Recover the latent matrix  $M^{(t+1)}$                             */
    3  $M^{(t+1)} = \text{low\_rank\_matrix\_completion}(S^{(t)}, \Omega^{(t)});$ 
    /* Check for convergence                                                    */
    4 if  $t > 0$  then
        /* No candidates left or last iteration                                */
        5 if  $\bar{\Omega}^{(t)} == \emptyset$  or  $t == \Gamma - 1$  then
            6 break;
        7 end
        8 if  $\|M^{(t+1)} - M^{(t)}\|_F \leq \epsilon$  then
            9 break;
        10 end
    11 end
    /* M-step: Update the probability matrix                                    */
    12  $S^{(t+1)}, \Omega^{(t+1)}, col\_entity\_type =$ 
        update_prob_matrix( $M^{(t+1)}, \Omega^{(t)}, col\_entity\_type$ )
13 end
14 return  $M^{(t+1)}$ 

```

compatibility assessment of column entity types using an LLM. Specifically, we harness the semantic power of LLMs to enforce a key condition for valid joins: *two columns that can be joined must have semantically equivalent entity types*.

We define a *column entity type* as a free-text annotation that captures the specific semantics which a column represents within its relational table. For instance, an ID column in a table of food orders should ideally be annotated as “food order ID”, whereas an ID column in a table of users should be annotated as “user ID”. Leveraging the advanced semantic understanding capabilities of LLMs, we employ an LLM to annotate column entity types and to determine the semantic equivalence between two entity types. The match of entity types constitutes a necessary condition for valid joins and substantially reduces the number

Algorithm 4: Update the Probability Matrix (**update_prob_matrix**)

Input : S , the probability matrix;
 Ω , the set of known certain entries in S ;
 col_entity_type , column to entity type map;
 $low_threshold$, the threshold to prompt LLM for
 entity types and type match check;
 $high_threshold$, the threshold to promote a latent
 entry to be a hard positive;
 δ , the penalty factor for entity type mismatch
Output: updated S , Ω , and col_entity_type

```
1 join_candidates = {(i, j) | (i, j) ∈ Ω̄ and i < j};
   /* Loop through join candidates */
2 for (i, j) ∈ join_candidates do
3   if S[i][j] < low_threshold then
4     | continue;
5   end
6   if i ∉ col_entity_type then
7     | col_entity_type[i] = prompt_llm_for_entity_type(col_i);
8   end
9   if j ∉ col_entity_type then
10    | col_entity_type[j] = prompt_llm_for_entity_type(col_j);
11  end
   /* Boost probability of type-matched entry */
12 if col_entity_type[i] == col_entity_type[j] or
   prompt_llm_for_soft_type_match(col_i, col_j) then
13   if S[i][j] ≥ high_threshold then
14     | S[i][j] = S[j][i] = 1.0;
15     | /* Add new known entries */
16     | Ω.add({(i, j), (j, i)});
17   else
18     | S[i][j] = S[j][i] = high_threshold;
19   end
20 else
21   /* Decay probability of unmatched entry */
22   | S[i][j] *= δ; S[j][i] *= δ;
23 end
24 end
25 return S, Ω, col_entity_type
```

of spurious join candidates (e.g., two ID columns have coincidental value overlap but refer to distinct types of entities). Critically, our approach does not rely on a fixed vocabulary of entity types, as column semantics can vary across different tables and databases, and no

```

1  # System prompt:
2  "You are a data management expert that only replies with JSON."
3
4
5  # Prompt for entity type inference:
6  "---- Column 1 ----
7  Table: {table_name}
8  Column: {col_name}
9  ---- Column 2 ----
10 Table: {table_name}
11 Column: {col_name}
12 ...
13 -----
14 What is the most likely entity type for each of these columns?
15 Give me a short label for each in the order they were provided.
16 Respond in a JSON format like {'entity_types': ['label1', 'label2',
17 'label3']}]"
18
19 # Prompt for soft entity type matching:
20 "---- Pair 1 ----
21 Left table: {table name}
22 Left column: {column name}
23 Left column entity type: {entity type}
24 Right table: {table name}
25 Right column: {column name}
26 Right column entity type: {entity type}
27 ---- Pair 2 ----
28 Left table: {table name}
29 Left column: {column name}
30 Left column entity type: {entity type}
31 Right table: {table name}
32 Right column: {column name}
33 Right column entity type: {entity type}
34 ...
35 -----
36 For each pair listed above, do the two entity types match?
37 Provide your answers ('yes' or 'no') in a JSON format like
38 {'match_results': ['yes', 'no', 'yes']}
39 **It is crucial that the order of the answers in the list matches
  exactly the order of the pairs as they were given in this prompt.**"

```

Figure 5.3: LLM prompts for column entity type inference and soft type matching.

existing entity type vocabularies have a comprehensive coverage of the nuanced spectrum of column semantics.

As detailed in Algorithm 4, the M-step iterates through latent entries (join candidates) and prompts an LLM to identify the entity types of columns based on column and corresponding table names (see Figure 5.3 for our prompts). For each candidate pair with a score exceeding a predefined *low threshold*, we prompt an LLM to infer their entity types if they do not have one. To reduce the computational and monetary cost of LLM calls, we cache the identified entity types. Subsequently, we conduct a character-wise comparison of their entity types. If this comparison fails, we further prompt the LLM for a soft check of whether the entity types match semantically (see Figure 5.3 for our prompts). The alignment of entity types is a prerequisite for a valid join. We have observed LLMs to be particularly effective in performing such nuanced semantic comparisons. If the entity types align and the

associated probability score exceeds a predefined *high threshold*, we promote the pair to be a hard positive, setting its probability score to 1 and removing it from the set of latent entries. If the types match but the probability score falls below the high threshold, we boost the probability score to this threshold value, ensuring the pair undergoes another LLM check in the next EM iteration. This design accounts for the low initial probability estimate of the pair from the pre-trained model and the inherent stochasticity of LLM outputs. If the entity type check fails, we penalize the pair by reducing its probability score by a specific factor δ , which we set to 0.5 for all experiments. In the implementation, to further reduce the latency of LLM calls, we batch multiple requests in one prompt with a batch size of 24.

The iterative process of the EM algorithm continues until either a predefined maximum number of iterations, Γ , is reached or convergence is achieved. The convergence is determined by the change in the recovered matrix (quantified by the Frobenius norm) falling below a specified tolerance, ϵ . We set ϵ to $1e^{-5}$ in all experiments.

For latency-sensitive scenarios, our system can be configured to perform a single iteration of the EM algorithm (only low-rank matrix completion). This fast mode, which we call NEXUS-Fast, is particularly useful when pre-trained models provide strong priors. It bypasses all LLM calls in the EM algorithm, ensuring maximum inference speed while still improving over the pretrained models and achieving competitive results compared to our full EM algorithm. In the following text, we refer to our full EM algorithm as NEXUS-EM to distinguish from NEXUS-Fast.

5.3.6 Optimization via Core Submatrix Reduction

The procedure described above recovers a matrix M of size $n \times n$, where n is the total number of columns in the table collection. For a large table collection, n can be very big. However, since our primary goal is to infer latent entries determined by join candidates that typically involve only a small subset of all columns, we can instead recover a smaller matrix, which we call the *core submatrix*:

Definition 9 (Core Submatrix). Given a symmetric $n \times n$ matrix M , the core submatrix M' is defined as the submatrix obtained by removing all rows and columns from M that contain only zero entries. Formally, let $I = \{i \mid \exists j, M_{i,j} \neq 0\}$ be the set of indices corresponding to non-zero rows (and equivalently columns, since M is symmetric). Then, the core submatrix can be denoted as $M' = M[I, I]$. The dimension of M' is $n' \times n'$, where $n' = |I|$.

Since most columns are not join keys, especially those measure-oriented columns ubiquitous in analytical workloads, typically the core submatrix M' is much smaller than M (i.e. $n' \ll n$). Importantly, our approach based on low-rank matrix completion remains

applicable, as we found that M' also exhibits high sparsity and a low-rank structure akin to the full adjacency matrix.

Recovering a smaller core submatrix M' offers significant practical benefits. Empirically, its dimension is often less than half the size of M , resulting in faster runtime and reduced memory usage. The main caveat is that the component of pruning join candidates is not perfect, and some relevant join keys may be excluded from M' and not recovered. Through experiments, we observe that this optimization gives almost 25% speedup when only running low-rank matrix completion (i.e., Nexus-Fast), so we enable this optimization by default for Nexus-Fast. For NEXUS-EM, this optimization is less significant as the main performance bottleneck lies in LLM calls (from a remote service). Hence, we choose to recover the full matrix without losing any relevant join keys.

5.3.7 Extension to Multi-Column Joins

While we have discussed our approach in the context of single-column joins, it’s easily adaptable for multi-column join scenarios. The core idea remains the same: we treat every set of k columns within a table as a single conceptual unit. These units are then subject to the same pruning rules outlined in Section 5.3.2. This extension leads to a larger probability matrix for the low-rank matrix completion task. Specifically, for a set of tables T , the matrix dimension becomes $m \times m$, where $m = \sum_{\tau \in T} \binom{n_\tau}{k}$ and n_τ is the number of columns in table τ . Each entry in this matrix represents the join probability of a pair of k -column units.

Crucially, this expanded matrix also exhibits high sparsity and a low-rank structure. This allows us to directly apply the same matrix completion and EM algorithms (Section 5.3.4 and Section 5.3.5) to detect multi-column joins. In fact, the presence of multiple columns in a unit increases the matrix size while the overall number of multi-column joins is typically smaller. This often leads to an even higher sparsity and lower rank in the join graph matrix, which can benefit our algorithms. Additionally, the core submatrix reduction optimization from Section 5.3.6 is also applicable and can be employed to further improve the efficiency.

Despite this capability, our analysis of the SchemaPile-Perm dataset shows that multi-column joins constitute only a small fraction (7.5%) of real-world schemas. Consequently, our experimental evaluation focuses on the more prevalent single-column joins.

5.4 Experiments

We report experimental results in this section.

Table 5.2: Summary of experimental dataset statistics.

Datasets	# Tables	# Columns	Density	Rank	Normalized Rank
TPC-DS	24	425	0.001	34	0.08
TPC-H	8	61	0.004	12	0.20
REAL	30	1156	0.001	19	0.02

5.4.1 Experiment Setup

Datasets. We evaluate our approach using two synthetic datasets, TPC-DS and TPC-H, and a commercial production dataset which we call REAL. Table 5.2 summarizes their statistics. For the TPC benchmarks, we establish ground truth from PK/FK relationships declared in data definition files and join relationships present in the benchmark queries. For the REAL dataset, we derive ground truth from join relationships observed in query logs.

Baselines. We consider four baselines for comparison, which we briefly describe below.

- **SemPy.** As an integral component of Microsoft Fabric’s semantic link feature, the SemPy Python library [114] offers many functionalities, including a production-level classical method for detecting join relationships.
- **XGBoost.** Many prior works have employed ML models in their approaches to detect join relationships [143, 16, 104]. Notably, [16] evaluated various ML models (e.g. logistic regression, gradient boost, and random forests) on metadata features and found their performance to be comparable, while Auto-BI [104] employs an XGBoost model on both metadata and value features to initialize its graph-based optimization. Following this precedent, we train an XGBoost classifier using an expanded set of metadata features that encompasses those employed in [16], on the BIRD-SQL dataset (ensuring no overlap with our evaluation datasets) as a baseline. Additionally, we use this model as a pre-trained ML model to set the probability scores in NEXUS.
- **Auto-BI.** Lin et al. [104] propose Auto-BI for predicting join relationships in BI models within data warehouses. They observe that the join graphs in BI environments often follow a snowflake structure, and frame join detection as a graph-based optimization problem using join candidates scored by an ML model. Auto-BI represents the current state-of-the-art in join detection.
- **GPT-4o.** We also assess GPT-4o [127], OpenAI’s latest LLM, for its ability to predict join relationships. We leverage GPT-4o in two additional ways: as a pre-trained model

Table 5.3: Quality comparison of all approaches on TPC-DS, TPC-H and REAL datasets, using metadata only.

	TPC-DS			TPC-H			REAL		
	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall
SemPy	0.43	0.50	0.38	0.41	0.26	1.0	0.26	0.29	0.23
XGBoost	0.02	0.01	0.22	0.11	0.06	1.0	0.20	0.11	0.70
Auto-BI	0.03	0.13	0.02	0.55	0.40	0.86	0	0	0
GPT-4o	0.81	0.68	0.99	0.78	0.64	1.0	0.49	0.34	0.88
Nexus-XGBoost	0.56	0.65	0.50	0.74	0.58	1.0	0.40	0.41	0.39
Nexus-GPT-4o	0.89	0.87	0.90	0.93	0.88	1.0	0.58	0.45	0.81

to provide priors for our approach, and within the EM process to mitigate the impact of a poorly performing pre-trained model. All interactions with GPT-4o use the Azure OpenAI API, version 2025-01-01-preview.

To ensure a fair comparison, all methods in our evaluation, including NEXUS and all baselines, operate on metadata only by default. The XGBoost model is trained exclusively on metadata features, and GPT-4o is prompted using only metadata. We also created metadata-only variants of SemPy and Auto-BI to contrast with their standard methods that require full data access. For the Auto-BI metadata-only variant, we configured its graph-based optimization to run on predictions from the metadata-only XGBoost model. For the SemPy metadata-only version, we disabled the value-based overlap check. Unless otherwise specified (as in Section 5.4.6), all of our experiments use these metadata-only versions.

Metrics. Due to the class imbalance in detected join candidates, we evaluate the quality of all approaches using F1 score, precision, and recall. The reported numbers represent the best results for each method after we performed best-effort tuning on each dataset. For efficiency comparison, we report runtime in seconds.

Hardware. All experiments were conducted on an Azure virtual machine (standard NC12s v3) with 12 vCPUs and 224 GiB of RAM.

5.4.2 Quality Comparison

Table 5.3 presents the F1 score, precision and recall for all evaluated approaches on the TPC-DS, TPC-H, and REAL datasets, under a metadata-only configuration, without leveraging any query logs. For NEXUS, we report results on NEXUS-EM using two distinct pre-trained models as priors: XGBoost (NEXUS-XGBoost) and GPT-4o (NEXUS-GPT-4o).

NEXUS consistently outperforms all baselines by a significant margin. For instance, NEXUS with GPT-4o priors (NEXUS-GPT-4o) surpasses Auto-BI, the previous SOTA, by

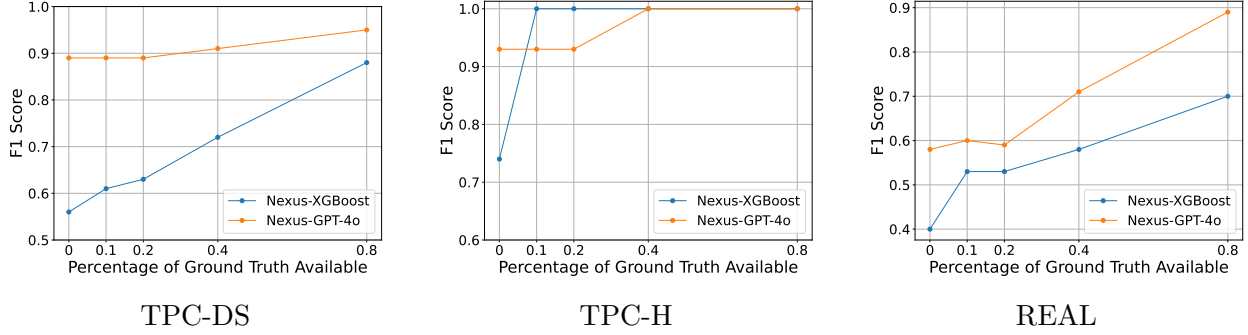


Figure 5.4: F1 score of NEXUS-EM w.r.t the percentage of ground truth joins available (assuming the availability of query logs) on TPC-DS, TPC-H, and REAL datasets.

86, 38, and 58 F1 points on the TPC-DS, TPC-H and REAL datasets, respectively. The classical solution, SemPy, also lags behind NEXUS-GPT-4o by 46, 52, and 32 F1 points on these three datasets.

XGBoost, since pretrained on the BIRD-SQL dataset, does not generalize well to any of the test datasets, yielding the lowest F1 scores on TPC-DS and TPC-H. In contrast, even with the weak priors from XGBoost, NEXUS-XGBoost significantly improves the F1 score by 54 points on TPC-DS, 63 points on TPC-H, and 20 points on the REAL dataset. The substantial improvement is attributed to NEXUS’s iterative optimization based on the low-rank structure of join graphs and column entity type checks via LLMs.

Similarly, NEXUS further improves on GPT-4o priors, even when GPT-4o shows strong results out of the box. GPT-4o’s strong result quality on the public TPC-DS and TPC-H benchmarks is expected, as it has likely learned their join ground truth during training. While it also achieves high recall on the unseen REAL dataset, demonstrating a robust ability to understand join semantics from schema, it nevertheless makes mistakes by giving a non-trivial number of false positives.

The pre-trained model is a modular component of NEXUS. While we use GPT-4o as an example of a strong prior in NEXUS-GPT-4o, any advanced ML model could be substituted. The results of NEXUS-XGBoost and NEXUS-GPT-4o highlight the robustness of our framework. Although a more accurate prior will naturally yield strong result quality, NEXUS proves capable of delivering adequate quality even when initialized with a significantly weaker model, as demonstrated by the XGBoost variant.

Auto-BI and SemPy underperform on all datasets. This outcome is not surprising, as both methods were originally designed with the assumption of data access. The lack of data access disables inclusion dependency check in join candidates pruning, leading to more false positives for both methods. Additionally, Auto-BI’s performance relies on a reasonably

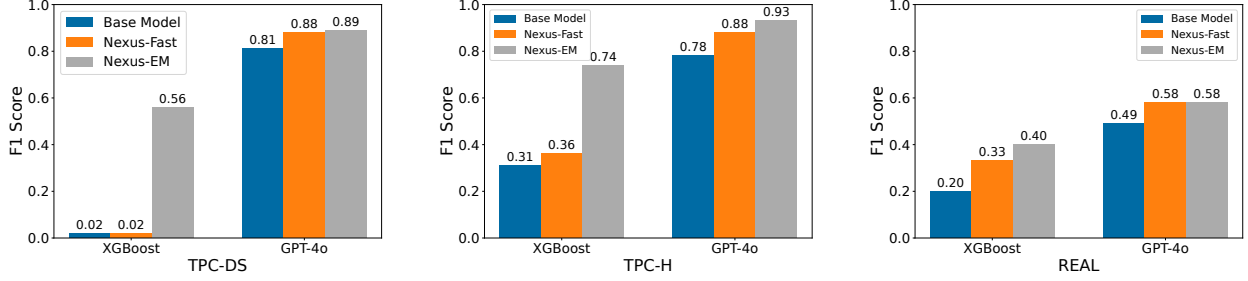


Figure 5.5: F1 score comparison of base models, NEXUS-Fast, and NEXUS-EM on TPC-DS, TPC-H, and REAL datasets.

effective base model, and its results are significantly compromised by the absence of features derived from data values. As we will show in Section 5.4.6, both methods demonstrate much-improved results when granted full data access. However, for the specific task of inferring a join graph from metadata alone, neither Auto-BI nor SemPy is competitive.

5.4.3 When Query Logs are Available

In practice, query logs are commonly available and can provide partial, if not complete, sets of true join relationships. We evaluate NEXUS’s result quality under varying percentages of available joins, which are randomly sampled from the ground truth. Figure 5.4 shows the F1 score of NEXUS-EM as the proportion of known joins increases across the TPC-DS, TPC-H, and REAL datasets.

Overall, increased availability of true joins via query logs leads to improved result quality for NEXUS. This is because sampled true joins offer valuable information about the structure and connectivity of the underlying ground truth adjacency matrix. For instance, on the TPC-DS dataset, NEXUS-GPT-4o achieves an F1 score of 0.91 and NEXUS-XGBoost reaches an F1 score of 0.72, when 40% of true joins are available. When the availability of true joins increases to 80%, NEXUS-GPT-4o’s F1 score rises to 0.95, and NEXUS-XGBoost’s F1 score reaches 0.88. On TPC-H, NEXUS-XGBoost achieves a perfect F1 score when just 10% of ground truth joins are available. The benefits of known joins are even more significant on the REAL dataset, where F1 scores increase by 30 points for both NEXUS-XGBoost and NEXUS-GPT-4o when 80% of true joins are known.

5.4.4 Fast Mode Inference

By default, NEXUS leverages LLMs within its EM algorithm to infer semantic entity types of columns and perform soft type matching. While powerful, these LLM API calls can incur

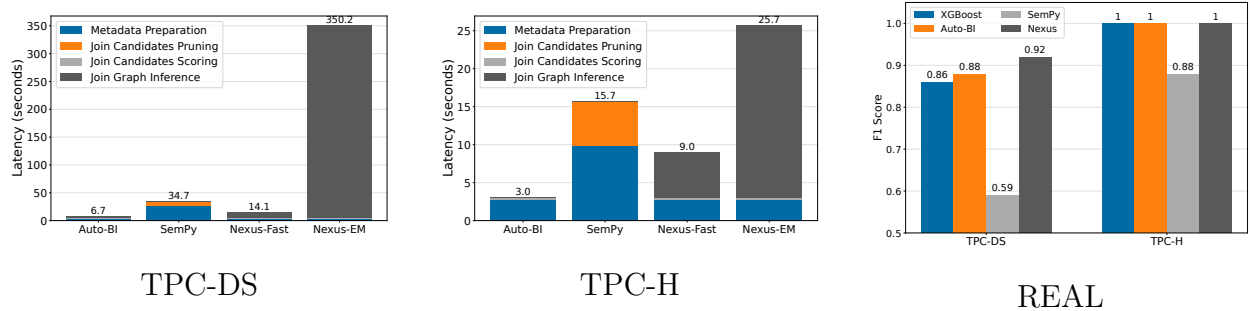


Figure 5.6: Efficiency comparison on TPC-DS, TPC-H, and REAL datasets. Auto-BI shares Nexus’s implementations of metadata preparation, join candidates pruning and join candidates scoring.

increased latency and considerable monetary costs. For latency- or budget-sensitive scenarios, we offer NEXUS-Fast, a fast mode of NEXUS that exclusively employs low-rank matrix completion (no iterative EM algorithm) on the core submatrix (as discussed in Section 5.3.6).

In Figure 5.5, we compare the result quality (F1 score) of NEXUS-Fast and NEXUS-EM, using XGBoost and GPT-4o as base models for scoring join candidates. Overall, NEXUS-Fast can improve result quality of both base models across three datasets while NEXUS-EM achieves the best results.

When a weak base model is used, or one that demonstrates poor generalization like XGBoost pretrained on the BIRD-SQL dataset, NEXUS-EM empowered by the LLM delivers stronger results than NEXUS-Fast, despite its longer runtime (as shown in the next section). Specifically, NEXUS-EM outperforms NEXUS-Fast by 54, 38, and 7 F1 points on the TPC-DS, TPC-H, and REAL datasets, respectively. When the weak model provides a useless prior, as seen with TPC-DS, NEXUS-Fast does not improve the result quality. Nevertheless, when the same base model has some predictive value as with TPC-H and REAL, NEXUS-Fast achieves a boost of 5 and 13 F1 points, respectively.

In contrast, when a strong base model like GPT-4o is employed, NEXUS-Fast yields result quality comparable to NEXUS-EM. The quality gap significantly shrinks, narrowing to only 1 F1 point on TPC-DS, 5 F1 points on TPC-H, and no difference on REAL. Compared to the base model alone, NEXUS-Fast yields improvements of 7, 10, and 9 F1 points on these three datasets. The subsequent section will elaborate on the latency advantages offered by NEXUS-Fast.

5.4.5 Efficiency Comparison

Figure 5.6 shows the efficiency comparison of four methods across all three datasets. We report the runtime of NEXUS-Fast and NEXUS-EM using the XGBoost base model here.

We categorize the end-to-end latency into four key components including metadata preparation, join candidates pruning, join candidates scoring, and join graph inference, as illustrated in our pipeline (Figure 5.2). Auto-BI, NEXUS-Fast and NEXUS-EM incorporate all four components, whereas SemPy only involves metadata preparation and join candidates pruning.

Note that we replaced the metadata preparation, join candidate pruning, and join candidates scoring components of Auto-BI with the more efficient implementation from NEXUS for a fair comparison due to our metadata-only adaption. Among all methods, Nexus-EM, which relies on LLM API calls, exhibits the highest latency, with join graph inference being the dominant factor although we have batched multiple requests in a single LLM prompt. The reported latency for Nexus-EM also depends on the number of iterations we run the algorithm and here we report the latency of Nexus-EM when it achieves the best result on each dataset. In contrast, NEXUS-Fast, by exclusively performing low-rank matrix completion on the core submatrix, achieves a significant speed-up: it is nearly 25 times faster on TPC-DS, 3 times faster on TPC-H and 14 times faster on REAL than NEXUS-EM. As demonstrated in Section 5.3.6, when we substitute the base model with a stronger model, NEXUS-Fast can also deliver comparable result quality.

5.4.6 When Data Values are Available

Although NEXUS is designed for enterprise settings where data access is restricted, it can be applied when data values are available. In fact, NEXUS remains highly effective in this setup. We compare NEXUS (referring to NEXUS-EM) against XGBoost, Auto-BI, and SemPy on the TPC-DS and TPC-H datasets. The REAL dataset is excluded from this experiment due to restricted access to the data values. With full data access, we switch to an XGBoost model trained on additional data-value features used in Auto-BI to score join candidates across all relevant methods, replacing the metadata-only models in previous experiments.

Figure 5.7 shows the F1 score comparison across all four methods. Unlike the metadata-only setting, all methods except SemPy now perform competitively, highlighting how heavily these existing approaches rely on data values. This also re-emphasizes the critical role of considering join graph structure when metadata alone provides limited signals of the joinability of individual column pairs. Nevertheless, NEXUS remains the top performer on both TPC-DS and TPC-H. Even though the TPC-DS and TPC-H datasets conform to the

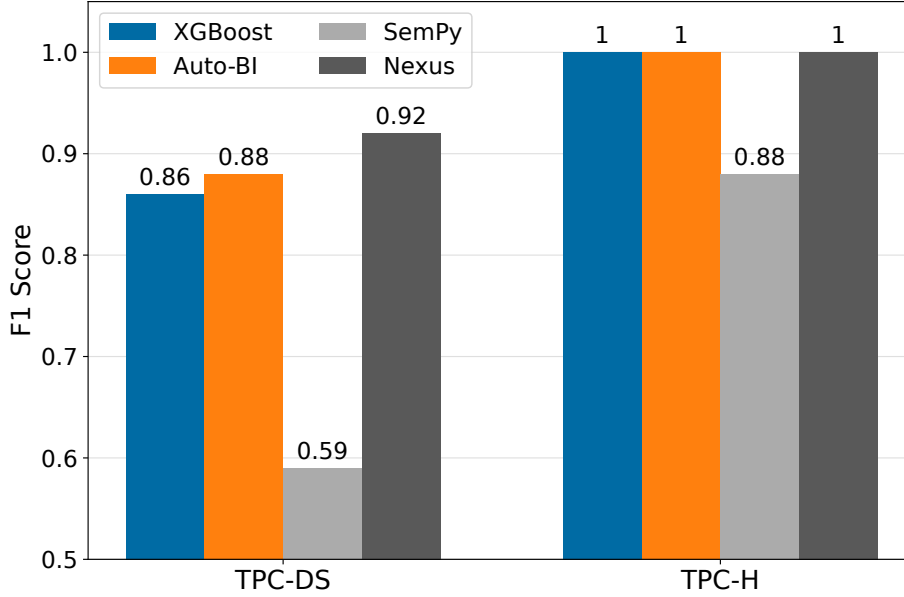


Figure 5.7: Comparison of F1 score on TPC-DS and TPC-H when data values are available.

strict snowflake schema that Auto-BI is optimized for, NEXUS still outperforms it. NEXUS’s more flexible assumptions about the join graph (i.e., high sparsity and low-rank structure) lead to a 4-point F1 score advantage over Auto-BI and a 6-point advantage over XGBoost on TPC-DS. SemPy trails behind all other methods as it simply relies on the column name similarity for pruning and value overlap to determine joins. Such heuristics can yield a significant number of false positives and/or false negatives.

In addition to better ML predictions using value-based features, access to data values enables inclusion dependency tests, an effective technique for pruning join candidates. This test effectively eliminates many false positives that might otherwise appear highly joinable based on metadata alone. On TPC-DS, the number of join candidates drops from 3351 to 1232. On TPC-H, the reduction is even more dramatic, from 183 candidates to just 30. This aggressive pruning is a key reason why XGBoost, Auto-BI, and NEXUS all achieve perfect F1 scores on TPC-H.

5.5 Related Work

Join detection has been studied extensively in the literature, in the context of PK/FK detection in relational databases [143, 182, 35, 87], BI model construction for analytical workloads [16, 104], and related table search in Web corpora [26, 25, 144, 135] and data lakes [192, 60, 191, 187, 53, 63, 54, 48, 110]. Two main scenarios are commonly studied:

(1) joinable table search, which finds tables joinable with a given input table, and (2) join graph inference, which discovers all join relationships within a collection of tables. This work focuses on the latter, which is more computationally demanding. While both scenarios share similar solution strategies, join graph inference places greater emphasis on computational efficiency due to its increased complexity.

Traditional methods on join detection rely on heuristics like name similarity and value overlap [114, 35]. For example, Chen et al. [35] combine signals from table and column names with data values to distinguish true relationships from spurious candidates, using containment and other constraints to prune unlikely join candidates. Similarly, Jiang et al. [87] jointly detect primary and foreign keys using a comprehensive set of pruning rules. However, these methods are often computationally expensive and miss semantic matches. Zhang et al. [182] detect PK/FK relationships by measuring distributional similarity between columns with the Earth Mover’s Distance, but this approach does not scale well, requiring over 2.5 hours to process 10 GB of data [35].

Embedding-based approaches represent table and column metadata as vectors and use approximate nearest neighbor (ANN) search to find join candidates [54, 42]. While these methods can uncover non-obvious semantic joins, their similarity metrics do not guarantee correctness, often leading to a high rate of false positives.

Recent work has shifted toward ML-based methods. Rostin et al. [143] propose learning PK/FK relationships from known foreign keys using various ML models. However, subsequent work has shown that such pretrained models often fail to generalize to unseen datasets [35, 104]. In NEXUS, we address this limitation, by adopting an EM algorithm that iteratively improve the predictions from the pretrained ML model with the guidance of LLMs.

Our work is most closely related to recent approaches by [16] and [104], which aim to automatically and efficiently construct complete join graphs with minimal user input. Similar to earlier work [143], the method from [16] relies on ML models trained on large-scale data. While effective on in-distribution datasets, their approach does not address out-of-distribution generalization and overlooks the global structure of the join graph. Auto-BI [104] target a specific BI setting, where it constructs a join graph by applying a k-Min-Cost-Arborescence optimization on top of ML predictions. However, its enforcement of a snowflake schema is overly restrictive for general workloads, particularly those found in modern lake-houses. In contrast, our approach is more general and makes no assumptions about workload characteristics. We base our method on the key observations that join graphs in real-world scenarios are highly sparse and exhibit a low-rank structure. By leveraging this insight, we demonstrate that our approach performs robustly across a diverse range of datasets and

workload types.

Finally, most prior work, except [16], assumes access to data values, but value-based feature computation on large tables is computationally expensive and often impractical in enterprise settings due to compliance and audit requirements [16]. In contrast, NEXUS is designed to operate solely on metadata, while still providing high-quality results in an efficient manner.

5.6 Conclusion

In this work, we introduce NEXUS, a novel, metadata-only solution for join graph inference designed specifically for enterprise environments with stringent data privacy requirements. Our approach is founded on key insights derived from analyzing numerous real-world database schemas: join graphs are not only highly sparse but also consistently exhibit a low-rank structure. We exploit these properties by formulating join graph inference as a low-rank matrix completion problem. To overcome the poor generalization of existing pre-trained models, we propose a novel EM algorithm that iteratively refines join probabilities by incorporating semantic compatibility assessments from LLMs. Extensive evaluations, including on a real production dataset, demonstrate that NEXUS achieves significant improvements in both effectiveness and efficiency.

CHAPTER 6

OpenForge: Probabilistic Metadata Integration

6.1 Introduction

Clean and consistent metadata is pivotal in enabling the FAIR Data Principles—findability, accessibility, interoperability and reusability—in data repositories [171, 116, 123]. While significant progress has been made in data discovery through techniques that leverage data values [74, 124, 23, 17, 29, 121, 42], real-world dataset search engines [162, 23, 82] rely heavily on metadata (e.g., keywords) for indexing and retrieval. However, the practice of metadata curation, for dataset publishing and indexing, remains labor-intensive [163]. Ensuring metadata consistency across disparate dataset sources is an ongoing challenge, especially as datasets grow in scale and diversity [163].

Metadata Granularity and Vocabulary Mismatch Most repositories rely heavily on metadata, on which they place significant requirements, including the vocabulary used. For example, Google dataset search engine [23] requires data publishers to annotate their datasets using the `schema.org` vocabulary; otherwise, their datasets would not be well indexed for search. Web Data Commons [41], a longstanding effort in curating web tables, publishes benchmarks [97] annotated with both Schema.org types and DBpedia classes [101]. In contrast, open data portals like CKAN [38] and Socrata [152] allow metadata to be provided in JSON format with open vocabulary fields (e.g., `description`, `publisher`, `theme`). As a result, metadata granularity and vocabulary—ranging from controlled schemas to open-ended terms—vary widely across datasets, organizations, and repositories.

Practice of Metadata Curation A close examination of curatorial work at ICPSR, the world’s largest social science data archive [82], reveals that (meta)data curation involves domain expertise and subjective judgments to resolve inconsistencies and ensure standardization [163]. Curators revise dataset descriptions and create metadata records, such as subject terms and geographic coverage, to improve dataset search. Such tasks are often

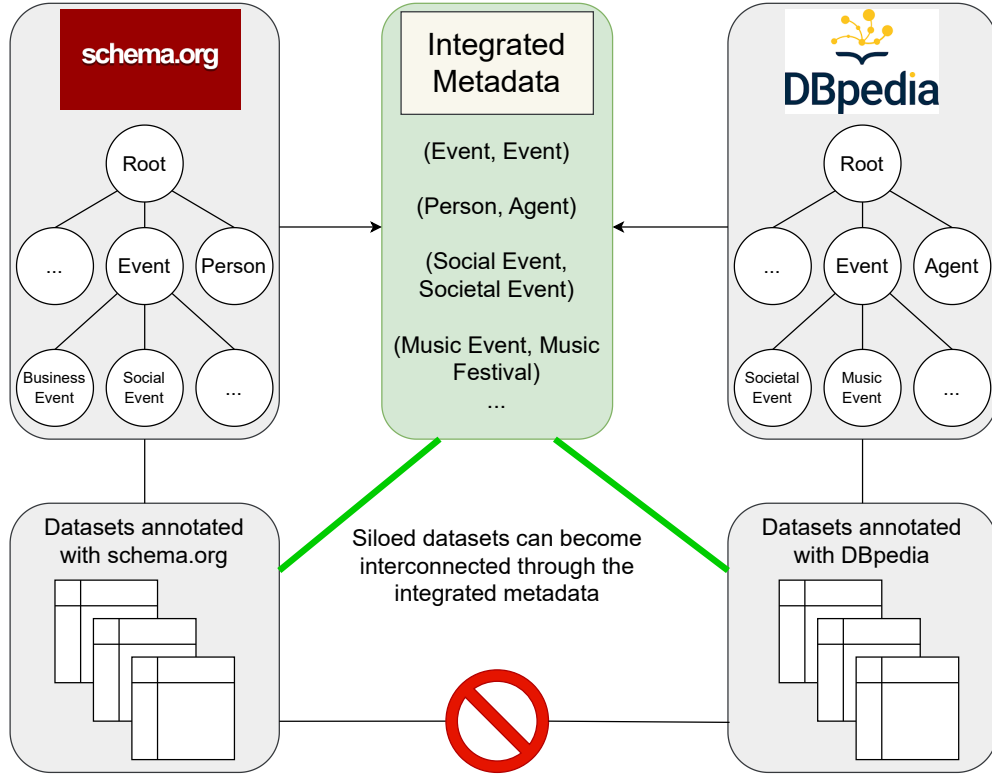


Figure 6.1: Illustration of metadata integration problem.

guided by metadata standards, like a manually curated vocabulary or thesaurus. However, experienced curators view such standards as flexible rather than rigid [163]. To remain relevant, curated vocabularies—and the metadata based on them—must evolve alongside changing data landscapes. Similarly, curators of platforms like `data.gov` must reconcile diverse metadata standards, as individual states and cities independently collect and manage their datasets.

Maintaining consistent and useful metadata, both within and across repositories, requires developing techniques to (1) unify and standardize metadata elements, (2) refine their granularity, and (3) perform holistic maintenance at scale. Motivated by these challenges, we study the problem of integrating metadata from disparate sources to build a consistent and granular view of metadata, by unifying equivalent metadata elements and defining metadata semantic hierarchies. Such hierarchies can be constructed using two primary types of relationships between metadata elements—*equivalence* and *parent-child*—as established in knowledge base literature. As illustrated in Figure 6.1, given a collection of potentially noisy metadata elements and associated data from various sources (optionally including dataset contents and descriptions), our goal is to identify all equivalence and parent-child relationships among element pairs.

We propose OPENFORGE, a data-driven framework that integrates and resolves inconsistencies among a collection of metadata elements. Conceptually, OPENFORGE’s output can be modeled as a graph of metadata elements (concepts) linked with two types of relationships. We call this graph, a *relationship assignment graph*. Since the relationship instantiations are not known apriori, we cast the problem of constructing this graph as a probabilistic inference problem. At the core of our approach lies a unified probabilistic formulation based on Markov Random Field (MRF), an undirected graphical model, which can integrate various types of relationships within a single framework. This formulation enables us to (1) incorporate diverse methods that leverage metadata elements and accompanying data (if available) to computing prior probabilities, (2) encode relationship axioms, such as transitivity, to prevent conflicting relationship instantiations and infer the most probable relationship assignment.

Our model represents relationships between elements as random variables within a MRF, framing relationship assignment as a joint probability distribution over these variables. The optimal relationship graph is then the assignment that maximizes this joint probability. By definition, MRFs decompose the joint probability into a product of factors which are non-negative functions defined over cliques of the graph. In particular, we leverage ternary factors (factors defined over ternary cliques of random variables) to encode transitivity where invalid configurations of random variables are assigned a probability of zero. This structural enforcement of transitivity is a key reason for adopting the MRF framework, alongside its principled handling of joint dependencies. However, the challenge lies in tuning the relative probabilities among valid (transitive) configurations, which critically influence model behavior. Manually setting these parameters risks poor generalization across datasets. To address this, we employ Bayesian optimization to learn optimal parameters in a data-driven manner, as described in Section 6.5.2. OPENFORGE also flexibly integrates a wide range of prior models—including prompting and fine-tuning large language models (LLMs), as well as traditional machine learning models—allowing the MRF to combine learned priors with structured inference.

Another key challenge in applying probabilistic graphical models to real-world data is inference scalability [166, 4, 190]. This challenge is exacerbated for OPENFORGE when operating on large datasets with sparse relationships. OPENFORGE addresses this challenge through an embedding-based strategy that decomposes the MRF into smaller, independent subgraphs based on metadata semantics, enabling efficient parallelized inference over these subgraphs.

In summary, we make the following contributions:

- We introduce the metadata integration problem as crucial for unification and integrity

maintenance of metadata repositories, identifying equivalence and parent-child relationship determination as central tasks to address this problem (Section 6.2.1).

- We propose a probabilistic formulation for resolving equivalence and parent-child relationships among metadata elements (Section 6.2.2), framing the task as maximum a posteriori inference in a Markov Random Field (Section 6.3.2).
- We develop OPENFORGE, a data-driven technique that integrates various prior models, including LLMs, and refines their predictions through probabilistic inference (Section 6.3).
- To address the scalability challenge of probabilistic inference on large sparse MRFs, we design an embedding-based strategy that decomposes sparse MRFs into smaller independent subgraphs, enabling parallelized inference for large datasets (Section 6.4).
- Experiments on four real-world datasets show that OPENFORGE outperforms task-specific baselines and the latest LLMs (including GPT-4) significantly. We also demonstrate efficient MRF inference completes efficiently on the largest dataset and scalability to synthetic graphs with millions of random variables (Section 6.6).

6.2 Metadata Integration Problem

6.2.1 Problem Statement

A metadata repository is the union of the metadata of a collection of datasets. To formalize a metadata repository, we consider metadata elements and their relationships as the building blocks of a metadata repository. Suppose a set of elements $\mathcal{V} = \{c_1, c_2, \dots, c_n\}$, where each element represents a unit of metadata information for integration such as column annotations, keywords, entity mentions, or dataset metadata attributes. We call each element in $\mathcal{V}$ a metadata element or concept, interchangeably. Consider the relationship $R : \mathcal{V} \times \mathcal{V} \rightarrow \{0, 1\}$ that can hold between any pair of concepts, e.g., equivalence or parent-child. Our concrete goal is to enrich the metadata of a data repository with a given relationship R . More specifically, we want to infer the presence or absence of R between each pair of concepts in $\mathcal{V}$. The concepts in $\mathcal{V}$ are raw metadata units extracted from metadata and dataset files. However, discovering equivalence relationships between pairs of concepts will allow us to unify and standardize these raw concepts. Note that data repositories may contain various types of metadata (e.g., `keyword`, `theme`, `column annotation`). A data curator can choose to integrate specific types or include all metadata, defining $\mathcal{V}$ accordingly. In addition to metadata elements in $\mathcal{V}$, data repositories may also include accompanying data content or

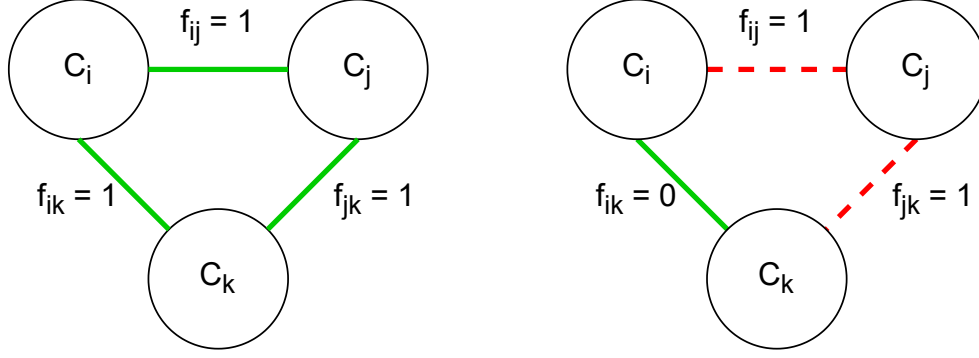


Figure 6.2: Illustration of the relationship transitivity (left) and inconsistent relationship assignments that violate the transitivity (right). Green solid edges indicate correct predictions and red dashed edges indicate conflicting predictions.

descriptions. We refer to this accompanying information, if present, as *evidence*, as described in Section 6.2.2.

In this paper, we focus on two types of relationships: equivalence and parent-child relationships. We note that identifying equivalent elements and parent-child elements is similar to tasks in data matching [164] and taxonomy induction [151], respectively. Although these two relationships are typically addressed with task-specific methods [50, 183, 142, 103, 164, 180], we demonstrate below that they can be effectively modeled within the same probabilistic formulation. This unified approach reduces complexity and maintenance overhead compared to using separate methods, and our experiments on four tasks empirically validate our approach’s ability to handle both relationship types.

6.2.2 Probabilistic Formulation

We first define a relationship assignment graph for relationship R , namely G_R , where each node represents a metadata concept in $\mathcal{V}$. The edges in G_R encode whether and how R holds between pairs of concepts. Specifically, for any two distinct concepts c_i and c_j , we define a random variable r_{ij} that characterizes the relationship:

- If R is bi-directional, like equivalence, r_{ij} is a binary random variable taking values in $\{0, 1\}$ where $r_{ij} = 1$ indicates that the relationship is present in both directions and $r_{ij} = 0$ means that no relationship exists.
- If R is uni-directional, like parent-child, r_{ij} is a categorical random variable taking values in $\{0, 1, 2\}$ where $r_{ij} = 1$ indicates that the relationship is directed from c_i to c_j , $r_{ij} = 2$ indicates that the relationship is directed from c_j to c_i , and $r_{ij} = 0$ means that there is no relationship.

The relationship assignment graph G_R can therefore be undirected or directed, depending on the nature of R . In both cases, each random variable follows a discrete probability distribution. When it is clear from the context, we refer to G_R simply as G .

Our goal is to find the most probable relationship assignment graph G defined over $\mathcal{V}$. We define the probability of any given relationship assignment graph G as the joint probability of all random variables r_{ij} formed for $\mathcal{V}$ of G , capturing both the presence and, if applicable, the directionality of relationships.

Definition 10 (Probability of a Relationship Assignment Graph).

$$P(G) = P(\{r_{ij} \mid (c_i, c_j) \in \mathcal{V} \times \mathcal{V}, i < j\}).$$

We consider only one ordering of concept pairs, meaning that for a pair of concepts (c_i, c_j) , we model only (c_i, c_j) as a random variable and not both (c_i, c_j) and (c_j, c_i) as indicated in Definition 10.

Prior Beliefs. Although probabilities $P(r_{ij})$ are not known a priori, we assume that for each r_{ij} , there exists evidence e_{ij} , consisting of features that can be observed (computed) to infer the probability of the concept pair (c_i, c_j) having the relationship R . This allows us to compute a prior for $P(r_{ij} \mid e_{ij})$. We denote the set of observed evidence over all random variables of G as $\mathcal{E}$ and the prior prediction of r_{ij} as f_{ij} where $f_{ij} = \arg \max_{r \in \text{dom}(r_{ij})} P(r_{ij} = r)$. We defer the discussion of obtaining these prior beliefs to Section 6.3.1.

Transitivity: Axiom on Relationships. In addition to the evidence used as priors for relationships, the considered types of relationships in our problem setting, equivalence and parent-child, impose additional structural properties, which can be leveraged for improving relationship predictions. Let us focus on transitivity which introduces dependencies among pairwise relationships. For example, the transitivity condition requires that if an equivalence relationship exists between concept pairs (c_i, c_j) and (c_j, c_k) , then it must also exist between (c_i, c_k) . Formally, considering equivalence or parent-child, for given predictions f_{ij} , f_{ik} , and f_{jk} evaluated for three concepts c_i , c_j , and c_k , if $f_{ij} = f_{jk} = 1$, transitivity requires that f_{ik} must also be 1. However, if these predictions are made independently, it is possible for f_{ik} to be erroneously predicted, violating the transitivity constraint. The transitivity property of the relationship further implies that certain relationship assignments are inherently inconsistent. Specifically, if we aim to infer f_{ij} and f_{jk} while knowing $f_{ik} = 0$, transitivity dictates that f_{ij} and f_{jk} cannot both be 1, as illustrated in Figure 6.2. This restriction prevents certain relationship assignments from coexisting.

Conflicting predictions can arise if pairwise decisions are made independently and it requires resolution of inconsistencies to maintain transitivity. For example, flipping either f_{ij}

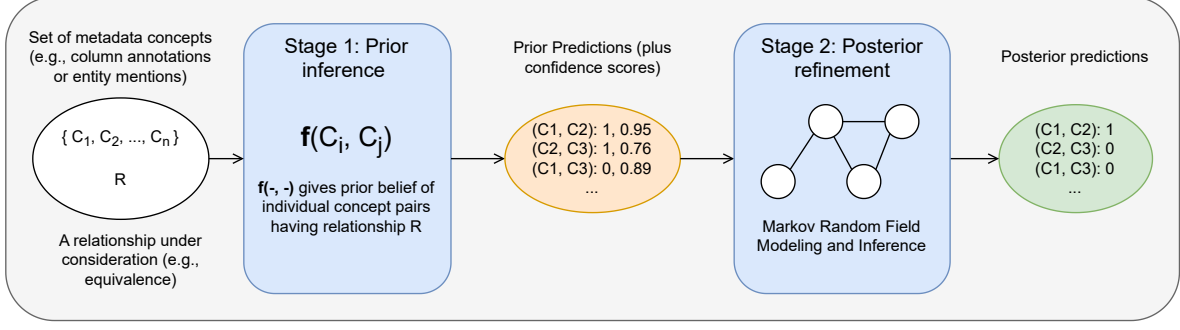


Figure 6.3: Overview of the proposed two-stage prior-posterior framework for integrating metadata concepts.

or f_{jk} would resolve the conflict by ensuring that the assignments align with the transitivity constraint. Yet, determining which prediction to alter is a non-trivial task, as it may require additional context. To address these challenges, we introduce a probabilistic formulation and a dependency-aware solution below.

Provided that prior beliefs for each r_{ij} , namely $P(r_{ij} | e_{ij})$, are available and we are given some axiom $\mathcal{A}$ on the relationships, the problem of metadata integration can be cast as an optimization problem. More specifically, we consider the common and intuitive transitivity axiom discussed above for our setting. The optimization objective is to find the relationship graph assignment $\tilde{G}$ with the maximum probability, conditioned on observed evidence $\mathcal{E}$, while satisfying the constraints implied by the relationship axiom $\mathcal{A}$.

Definition 11 (Optimal Relationship Graph). Given a relationship R , a set of concepts $\mathcal{V}$, and axiom $\mathcal{A}$, the optimal relationship assignment graph $\tilde{G}$ is the graph with the maximum joint probability of random variables $\{r_{ij}\}_{i < j}$, conditioned on evidence $\mathcal{E}$ and being subject to axiom $\mathcal{A}$.

$$\tilde{G} = \arg \max_G P(G | \mathcal{E}, \mathcal{A}) = \arg \max_{\{r_{ij}\}_{i < j}} P(\{r_{ij}\}_{i < j} | \mathcal{E}, \mathcal{A})$$

This formulation permits a theoretically optimal solution by exploiting a probabilistic graphical model called Markov Random Field, which also allows for incorporating the relationship transitivity and dependencies during the probabilistic inference. We give a brief introduction to Markov Random Field in Section 6.3.2.

6.3 OpenForge Framework

As illustrated in Figure 6.3, we introduce OPENFORGE, a prior-posterior framework for integrating metadata concepts. Given a set of concepts and a relationship, the framework operates in two stages. In the first stage, local predictions are generated as prior beliefs for each pair of concepts. Various methods may be used to obtain these prior predictions (Section 6.3.1). In the second stage, we address the relationship transitivity and dependencies between prior predictions through a probabilistic modeling approach using Markov Random Field (Section 6.3.2). This graphical model encodes the transitivity constraints and leverages labeled data to learn dependencies in a data-driven manner, effectively capturing the underlying relationship structures and refining the prior predictions.

6.3.1 Obtaining Prior Beliefs

Our probabilistic formulation requires a probability distribution for each random variable. We explore three different methods for obtaining these prior beliefs.

6.3.1.1 Prompting Large Language Models

Recent advances in large language models (LLMs) have shown human-expert performance on a broad array of natural language processing and multi-modality tasks [126, 6, 55]. These models are pre-trained on a vast amount of curated data and can be applied out-of-the-box to new contexts following a text-to-text paradigm with instruction tuning or prompting [137, 24]. Motivated by such success, we supply an LLM with a task description and prompt the model for predictions and confidence scores of concept pairs. Besides being straightforward, this method requires no labeled data or only few labeled data for demonstration (i.e., few-shot learning), making it feasible for scenarios where labeled data are scarce or there are not enough data for training/fine-tuning a model.

6.3.1.2 Fine-Tuning Large Language Models

Since LLMs are primarily pre-trained for generation tasks, we also experiment with fine-tuning them for domain-specific classification tasks if training data are available. Due to the large size of LLMs and the hardware constraints, we choose to fine-tune open-source models that have fewer than 10 billions of parameters. In particular, We employ Low-Rank Adaptation (LoRA) [77], a parameter-efficient fine-tuning technique that introduces a small set of trainable parameters (known as adapters) while leaving most of the model’s original parameters unchanged. This method enables effective and efficient adaptation of an LLM to

a specific relationship classification task with a single GPU. Additionally, LoRA allows for task flexibility: different tasks can share the base LLM, requiring only a switch in adapters rather than maintaining a fully fine-tuned model for each task.

6.3.1.3 Training Machine Learning Models

As compared to the previous methods based on LLMs, we also manually design task-specific features and train a machine learning model, such as Random Forest, to make predictions. This method serves as a baseline to evaluate the LLMs’ effectiveness in capturing dataset semantics.

Note that we fine-tune LLMs or train traditional machine learning models to obtain prior beliefs when training data are available. Nevertheless, all these methods treat each pair of concepts independently and make individual predictions ignoring relationship dependencies. There is no guarantee of avoiding conflicting predictions. Further details on the models and adaptations used in our experiments are provided in Section 6.5.1.

6.3.2 MRF Modeling of Metadata Integration

6.3.2.1 Markov Random Field Primer

Markov Random Field (MRF), also referred to as Markov Network in the literature, is a specific type of undirected graphical model for representing the joint probability distribution of a set of random variables [133].

Specifically, an MRF is defined by an undirected graph $M = (\mathcal{N}, \mathcal{L})$ where $\mathcal{N}$ denotes the set of nodes representing random variables and $\mathcal{L}$ denotes the set of edges representing the statistical dependencies between the random variables.

In MRFs, the joint probability distribution of random variables is factorized as a product of potential functions, each of which is associated with a clique in the graph. The potential function, also known as factor, is a function that assigns a non-negative value to each possible configuration of the random variables in the clique. Here, a configuration of random variables is an assignment of values to random variables. The joint probability of random variables is then proportional to the product of the potential functions over all cliques in the graph, defined as follows.

$$P(x_1, \dots, x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c) \quad (6.1)$$

where x_i denotes a random variable in $\mathcal{N}$, C denotes the set of cliques in G , while ϕ_c denotes a factor defined over a set of variables x_c in a clique $c \in C$, and Z is a normalizing constant.

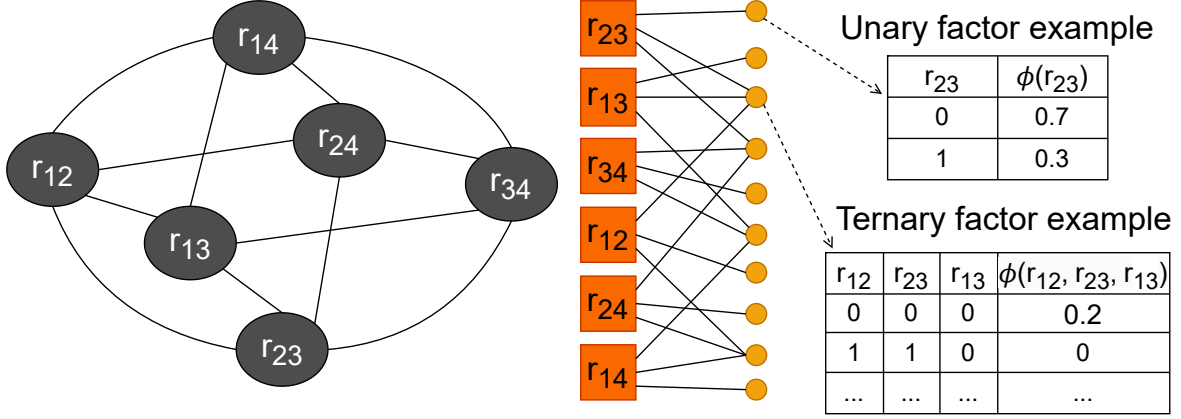


Figure 6.4: The plot on the left demonstrates an instance of our proposed MRF model containing six nodes/random variables and their dependencies; the plot on the right, known as a factor graph, visualizes the correspondence between factors and random variables in the MRF.

Note that clique set C and potential functions $\phi(c)$ need to be specified per application context.

An MRF can also be represented by a factor graph which explicitly shows the correspondence between random variables and factors. Specifically, a factor graph is a bipartite graph where the nodes are divided into two disjoint sets, one for the random variables and the other for the factors. An edge exists between a factor node and a variable node if the factor depends on that variable. This representation simplifies the computation of probability distributions in MRFs, such as Maximum a Posteriori (MAP) inference, which determines the most likely assignment of values to the variables in the MRF. We introduce our MRF design below and discuss MAP inference in Section 6.4.1.

6.3.2.2 Our MRF Design

MRFs are powerful models for capturing prior beliefs about and the dependencies between random variables. This makes MRFs a natural fit for our problem formulation, which require modeling existing knowledge about concepts and their relationships including encoding the transitivity axiom.

Let us first describe how the relationship assignment graph, its probability of Definition 10, as well as the transitivity axiom are modeled as a discrete MRF described in Section 6.3.2.1. We treat the presence of a specified relationship between a pair of concepts (c_i, c_j) , an edge between nodes c_i and c_j (directed if applicable) in a relationship graph, as a categorical random variable r_{ij} . We consider two types of potential functions, or factors, in our problem. The first type is unary cliques C_u to capture prior beliefs, each of which contains an individual

random variable r_{ij} . The second type is ternary cliques C_t to capture the relationship transitivity and dependencies, each of which consists of three random variables (r_{ij}, r_{jk}, r_{ik}) for $i < j < k$. Note that ternary cliques are the cliques of the smallest size on which we can encode the relationship transitivity. Although encoding the transitivity with higher-order cliques is possible, it will make the model more complex (e.g., the total number of possible configurations of a factor grows exponentially with respect to the size of a clique) and incur additional computational cost (we show that the time complexity of inference over unary and ternary cliques is already quartic with respect to the number of concepts in Section 6.4.1). Admittedly, the design choice of using ternary cliques instead of higher-order cliques weighs the efficiency over the potential benefit of having a more complex while expressive model.

The left plot of Figure 6.4 demonstrates an instance of MRF consisting of six nodes/random variables $\{r_{ij} \mid 1 \leq i < j \leq 4\}$ induced by four concepts $\{c_1, c_2, c_3, c_4\}$. The MRF is densely connected but not fully connected where there is an edge between a pair of nodes if they involve a common concept. For instance, there is an edge between r_{12} and r_{13} because they both involve concept c_1 whereas there is no edge between r_{12} and r_{34} . Under the hood, the edge connectivity is attributed to ternary cliques we consider in the MRF. We do not include all ternary cliques of three random variables in the MRF (e.g., the clique of r_{12} , r_{13} and r_{34}) but ternary cliques that involve exactly three concepts for the purpose of modeling the relationship transitivity. The right plot of Figure 6.4 shows the factor graph representation of the MRF that visualizes the correspondence between factors and random variables. This factor graph consists of six random variables and ten factors. Each variable is associated with three factors while each unary factor is associated with one random variable and each ternary factor is associated with a clique of three variables.

Since an MRF computes the joint probability of the graph based on cliques in the graph, now we can decompose the probability of our relationship assignment graph following Equation 6.1 as below.

$$\begin{aligned}
P(G \mid \mathcal{E}, \mathcal{A}) &= P(\{r_{ij}\}_{i < j} \mid \mathcal{E}, \mathcal{A}) = \frac{1}{Z} \prod_{c \in C} \phi_c(\{r_{ij}\}_{i < j, r_{ij} \in c} \mid \mathcal{E}, \mathcal{A}) \\
&= \frac{1}{Z} \prod_{c \in C_u} \phi_c(r_{ij} \mid \mathcal{E}) \cdot \prod_{c \in C_t} \phi_c(r_{ij}, r_{jk}, r_{ik} \mid \mathcal{A})
\end{aligned} \tag{6.2}$$

The potential function for each unary clique $\phi_c(r_{ij} \mid \mathcal{E})$ is defined as the conditional probability of the relationship's presence given the observed evidence of the concept pair, allowing us to capture the prior belief of each individual random variable.

While each unary clique has its own potential function, all ternary cliques share a common potential function, which is parameterized to capture relationship dependencies. Specifically,

configurations that violate transitivity (i.e., invalid configurations) are assigned a potential value of zero, while each valid configuration is parameterized to represent its likelihood. Higher parameter values indicate a greater probability for the corresponding assignment within the ternary clique. Table 6.1 gives an example of the parameterized potential function for ternary cliques of binary random variables, where 0 represents no relationship and 1 represents an equivalence relationship. Among the eight possible configurations of three binary variables, three configurations (i.e., $[0, 1, 1]$, $[1, 0, 1]$, and $[1, 1, 0]$) are invalid and thus set to zero. The remaining configurations are parameterized to determine their potential function values. Although it is technically feasible to assign unique potential functions to each ternary clique, this would result in a model with a linear increase in parameters relative to the number of ternary factors, making the model more complex. To maintain simplicity and prevent over-parametrization, we instead use a shared potential function across all ternary cliques. Hence, Equation 6.2 can be written as follows.

$$P(G|\mathcal{E}, \mathcal{A}) = \frac{1}{Z} \prod_{r_{ij} \in G} P(r_{ij} | e_{ij}) \cdot \prod_{c \in C_t} \phi(r_{ij}, r_{jk}, r_{ik} | \mathcal{A}) \quad (6.3)$$

where probabilities $P(r_{ij} | e_{ij})$ are given by some prior model and ϕ denotes the shared parameterized potential function for ternary cliques, which is independent of prior beliefs.

The parametrization of the shared potential function for ternary cliques enables us to learn relationship dependencies such as the label distributions in the dataset. In practice, datasets often exhibit skewed class proportions. For example, in an entity matching dataset, there are typically far more non-equivalent pairs of entities than equivalent ones. This imbalance affects the configuration of random variables in ternary cliques. In the parameterized potential function example in Table 6.1, the configuration $[0, 0, 0]$ is expected to have a higher potential value than other configurations, reflecting the larger number of concept pairs with no relationship. To learn the parameters in the shared potential function, we treat it as a hyperparameter tuning problem. Rather than manually setting these parameters, we search for well-performing parameters on a labeled validation set using a Bayesian Optimization technique [105]. We discuss automatic parameter tuning in Section 6.5.2.

Provided the learned potential function for ternary cliques, performing Maximum a Posteriori inference over this MRF gives us posterior predictions, an assignment to random variables yielding the maximum joint probability. Algorithm 5 summarizes our MRF modeling approach. Line 1-2 initialize the MRF with random variables. Lines 3-7 utilize methods, from Section 6.3.1, to estimate the conditional probabilities of random variables, and add unary factors to the MRF. In lines 8-14, we enumerate ternary cliques to create the ternary factors and line 15 executes the inference algorithm.

Table 6.1: Example of the parameterized potential function ϕ for ternary cliques of binary random variables r_{ij} , r_{jk} and r_{ik} where 0 represents no relationship and 1 represents an equivalence relationship. The potential values $\{\theta_i\}_{i=1}^5$ are learnable parameters.

r_{ij}	0	0	0	0	1	1	1	1
r_{jk}	0	0	1	1	0	0	1	1
r_{ik}	0	1	0	1	0	1	0	1
$\phi(r_{ij}, r_{jk}, r_{ik})$	θ_1	θ_2	θ_3	0	θ_4	0	0	θ_5

We remark that for a set of n concepts, the MRF we propose consists of $\binom{n}{2} = \frac{n(n-1)}{2}$ nodes and $O(n^3)$ factors (more precisely, $\binom{n}{3}$ factors for ternary cliques plus $\binom{n}{2}$ factors for unary cliques). Section 6.4.1 describes MRF inference and presents an analysis on the time and space complexity of running inference on our proposed MRF model. Section 6.4.2 discusses how we scale our MRF modeling for large and sparse datasets.

6.4 Scalable Metadata Integration

6.4.1 Inference in Markov Random Field

In our problem, we are interested in finding the most probable relationship graph or equivalently, the most probable assignment to random variables given the observed evidence, i.e., Maximum a Posteriori (MAP) inference. Performing exact MAP inference on complex non-tree structured MRFs is known to be NP-hard and computationally intractable when the graph is large and contains many loops [93, 58]. For example, a dataset containing 1000 concepts results in half a million of random variables and over 166 millions of factors. We thus resort to approximate inference algorithms, which mainly fall into three categories: linear programming-based inference, sampling-based inference, and variational inference [58].

6.4.1.1 Approximate MAP inference.

To find the most efficient existing inference algorithms, we evaluated various implementations from each category of approximate inference algorithms (more details in Section 6.5.2). We found that Loopy Belief Propagation (LBP) [119, 93], a special case of variational inference algorithms, is most efficient and scalable for our MRF model design. Our finding is consistent with the literature where variational inference methods often scale better and are more amenable to optimizations such as parallelization over multiple processors and hardware

Algorithm 5: MRF Modeling

Input : V , a set of concepts; f , a model that gives prior beliefs; ϕ , a shared potential function for ternary cliques; g , a MRF inference algorithm

Output: G , a relationship assignment graph

```
1  $mrf \leftarrow init\_mrf()$  ;  
   /* Create random variables for each ordered pair of concepts from the  
   initial vocabulary */  
2  $rvs \leftarrow V$  ;  
   /* Add nodes and factors for unary cliques */  
3 foreach  $var \in rvs$  do  
4    $mrf.add\_node(var), prior\_prob \leftarrow f(var)$  ;  
5    $unary\_factor \leftarrow init\_factor([var], prior\_prob)$  ;  
6    $mrf.add\_factor(unary\_factor)$  ;  
7 end  
   /* Add edges and factors for ternary cliques */  
8  $ternary\_cliques \leftarrow generate\_ternary\_cliques(rvs)$  ;  
9 foreach  $(var1, var2, var3) \in ternary\_cliques$  do  
10   $mrf.add\_edges\_from($   
11     $[(var1, var2), (var2, var3), (var1, var3)])$  ;  
12   $ternary\_factor \leftarrow init\_factor([var1, var2, var3], \phi)$  ;  
13   $mrf.add\_factor(ternary\_factor)$  ;  
14 end  
15  $G \leftarrow g(mrf)$ ; /* Run MRF inference algorithm */  
16 return  $G$ 
```

acceleration using GPUs [190, 58].

LBP is a message-passing algorithm on a loopy graph that iteratively updates the belief of each random variable based on the beliefs of its neighbors in the graph. More precisely, we use the max-product LBP algorithm on factor graphs to perform MAP inference and handle high-order potentials (i.e., potentials of ternary cliques). With respect to the factor graph, LBP computes the message $m_{x \rightarrow w}(x_i)$ from a variable node x to a factor node w (here factor w contains variable x so there is an edge between the two nodes for message passing) as $m_{x \rightarrow w}(x_i) = \prod_{y \in Nb(x) \setminus w} m_{y \rightarrow x}(x_i)$, where $Nb(x)$ denotes the set of neighbors of node x , and x_i is a value that x can take. Conversely, the message $m_{w \rightarrow x}(x_i)$ from factor node w to variable node x is computed as $m_{w \rightarrow x}(x_i) = \max_{I, I_x = x_i} (\phi(I) \prod_{y \in Nb(w) \setminus x} m_{y \rightarrow w}(y_i))$, where I is a valid assignment to all variables in factor w , tuple $(I, I_x = x_i)$ denotes a valid assignment with $x = x_i$, and ϕ is the potential function corresponding to w . Once a variable x receives messages from all its neighbors in an iteration, the maximal belief of x can be computed as $x^* = \arg \max_{x_i} \prod_{y \in Nb(x)} m_{y \rightarrow x}(x_i)$.

6.4.1.2 Time Complexity Analysis.

Having described the MRF approximate inference algorithm we apply for finding the optimal relationship assignment graph, let us describe its complexity. Let s and d be the number of possible states and degree of variable x , respectively. Computing a variable-to-factor message $m_{x \rightarrow w}(x_i)$ requires combining all incoming messages from neighboring factor nodes excluding w for each state of x . Then the cost of computing a single variable-to-factor message is $O(s \cdot (d - 1))$. It follows that the time complexity for computing all variable-to-factor messages in one iteration is $O(\sum_x d \cdot s \cdot (d - 1)) = O(|V| \cdot s \cdot d^2)$ where $|V|$ denotes the number of variable nodes in the factor graph.

Let d_w be the degree of factor node w . Computing a factor-to-variable message $m_{w \rightarrow x}(x_i)$ requires maximizing over the values of all variables connected to w excluding x . Then, the cost of computing a single factor-to-variable message is $O(s^{d_w-1})$ as there are s^{d_w-1} configurations of connected variables excluding x . It follows that the time complexity for computing all factor-to-variable messages in one iteration is $O(\sum_w d_w \cdot s^{d_w})$. Combining the complexities for both types of messages, the time complexity of running max-product LBP for l iterations is $O(l \cdot (|V| \cdot s \cdot d^2 + \sum_w d_w \cdot s^{d_w}))$.

In our proposed MRF model, there are $\binom{n}{2}$ binary variable nodes with degree $n - 1$, $\binom{n}{2}$ unary factors with degree 1 and $\binom{n}{3}$ ternary factors with degree 3 given a set of n concepts. By plugging in these numbers, the time complexity above can be simplified to $O(l \cdot n^4)$.

6.4.1.3 Space Complexity Analysis

Recall from Figure 6.4 that factors are modeled as bipartite graphs between variables and factors. Each edge in the factor graph requires two messages—one message from the variable node to the factor node and another from the factor node to the variable node. Storing a message requires space proportional to the number of states of a variable. Then, the space complexity of storing messages is $O(|E| \cdot s)$, where $|E|$ denotes the number of edges in the factor graph. Additionally, for each variable node, we store a belief over its possible states, which requires $O(s)$ space per variable. For $|V|$ variable nodes, the space required for storing beliefs is $O(|V| \cdot s)$. Thus, the overall space complexity for max-product LBP is $O(|E| \cdot s + |V| \cdot s)$ which translates to $O(n^3)$ in our proposed MRF model.

The quartic time complexity and cubic space complexity is forbidding for large datasets with thousands of concepts. Next, we introduce a solution to scale up the MRF inference.

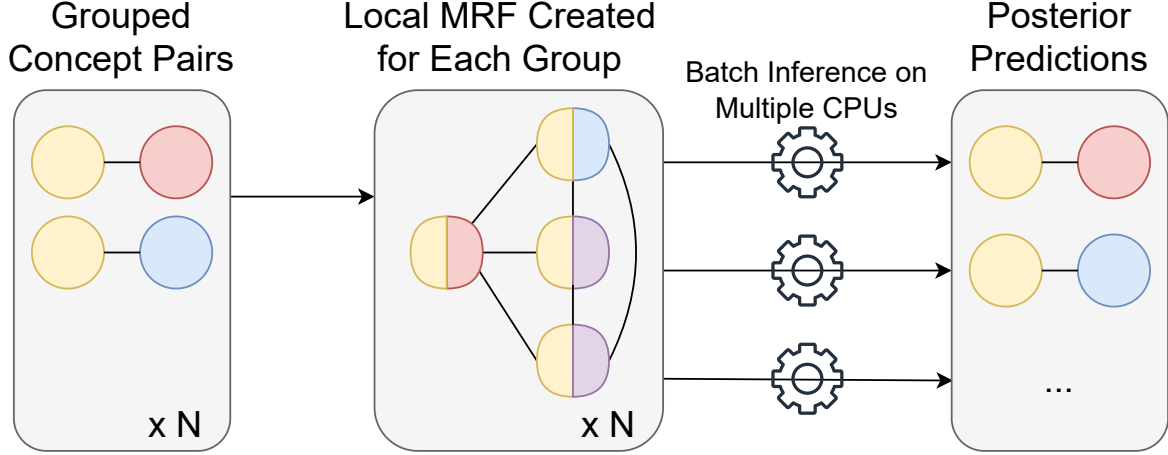


Figure 6.5: Creating independent MRFs for concept pairs in large datasets with sparse relationships. Ordered pairs are first grouped by the left concept and top- k neighbors (represented by the purple semi-circles) are retrieved for the left concept to construct a local MRF of random variables. Inference over independent MRFs is parallelized on available CPUs and posterior predictions of test pairs are collected.

6.4.2 Scaling MRF Modeling

The high time and space complexity stems from the densely connected graph we construct. For a set of n concepts, each node in the resulting factor graph connects to $n - 1$ neighbors including one unary and $n - 2$ ternary factors. This dense connectivity leads to significant computational demands and memory consumption.

However, we empirically observe that many connections in the dense graph are unnecessary for ensuring relationship transitivity and learning dependencies. Specifically, numerous connections are either redundant or represent weak dependencies that do not significantly impact inference accuracy. We propose to systematically reduce the connections for each node. By reducing the connections from $n - 1$ to a constant k , where $k \ll n$, we effectively decrease the number of message-passing operations required by the LBP algorithm, enabling faster execution and significantly lower memory usage. Our intuition is that concepts with more similar embeddings are more likely to share strong relationship dependencies. Therefore, to select these k neighbors for each concept, we employ a text embedding model, NV-Embed-v2 [100] to represent concepts in embedding space and search for k -nearest neighbors based on Cosine similarity. By focusing only on the top- k neighbors, we maintain the essential relationship dependencies while disregarding redundant connections that contribute little to inference quality.

Reducing the number of connections for each node has the advantage of dividing the MRF into disconnected local graphs. Our solution first groups ordered concept pairs by

the left concept and constructs a local MRF for each group. This ensures that the same concept pair does not appear in multiple MRFs and local MRFs are independent of each other. The inference for each local MRF can then be performed independently in constant time. Furthermore, we can parallelize the inference for local MRFs across the available CPUs, which significantly reduce the inference time. Figure 6.5 provides an illustration of the idea.

6.5 Implementation

6.5.1 Prior Models

6.5.1.1 Large Language Models

We employ two open-source LLMs as prior models for prompting and fine-tuning: `gemma-2-9b-it` from Google [113] and `Qwen2.5-7B-Instruct` from Alibaba [175]. Both models can be accessed from Hugging Face’s model hub. We choose these two models as we are able to deploy them locally and they can closely follow our straightforward prompts to generate both predictions and confidence scores. We refer to both models as `gemma-2` and `qwen2.5` respectively for convenience below.

Due to the space limitation, we have made our prompts and fine-tuning hyperparameters publicly available in our GitHub repository. Both models can be fine-tuned efficiently under an hour (for 20 epochs). One noteworthy issue we encountered when fine-tuning `gemma-2` and `qwen2.5` is that both models tend to exhibit excessive confidence in their predictions, with most predictions yielding a probability score above 0.99. This overconfidence hinders the message-passing process in MRF inference, as only information from the dominant class gets propagated. To mitigate the issue, we incorporate three techniques for calibrating model confidence. Specifically, we use a weighted loss function (with weights inversely proportional to the frequency of the respective classes) when fine-tuning models on imbalanced datasets and enable label smoothing [118] in the loss calculation, which assigns a small probability to incorrect classes to reduce overfitting. Additionally, we apply temperature scaling [73] to adjust the sharpness of the output logits during inference, which involves dividing the logits by a temperature value greater than 1 before applying the softmax function, thereby softening the resulting probabilities.

6.5.1.2 Machine Learning Models

We train multi-class classifiers on a predefined set of features to predict the presence of a relationship for each concept pair. We employ three classifiers including Ridge classifier, Random Forest, and Gradient-Boosted Decision Tree from scikit-learn [134]. These models were chosen for their built-in regularization mechanisms, which help reduce overfitting and improve generalization on class imbalanced datasets. We define features using concept names and associated data values if available. These features include string similarities (Q-gram, Jaccard, and edit distance), embedding similarities (fastText [19]), word and character count ratios between concept names, and Jaccard and embedding similarities between column values.

6.5.2 MRF Inference and Parameter Tuning

We evaluated five implementations of MAP inference for MRFs from three libraries: Shafer-Shenoy [147] from `pyAgrum` [56], Belief Propagation [133], Max-Product Linear Programming (MPLP) [68], and Gibbs Sampling [66] from `pgmpy` [7], and Loopy Belief Propagation (LBP) [119] from `PGMax` [190]. LBP from `PGMax` proved the most efficient and scalable due to their flat array-based implementation which leverages just-in-time compilation and parallelization for optimized execution on multi-processors and GPUs.

There are two sources of parameters we tune for our problem: those in the potential function of ternary cliques and those for LBP inference (e.g., the damping factor used to improve the convergence of LBP). In particular, we tune the parameters using SMAC [105], a robust and flexible Bayesian Optimization framework for hyperparameter optimization. SMAC treats MRF inference as a black-box function and iteratively searches for parameter configurations that maximize a target function which gives accuracy or F1 score for example. Internally, SMAC employs a surrogate model to approximate the target function and in each iteration proposes a configuration with the maximum expected improvement (which is computed using the surrogate model) for evaluation. The optimization process balances between exploration (i.e., search an unknown region in the parameter space) and exploitation (i.e., search for configurations near the best-so-far configuration), and stops after a fixed number of iterations. SMAC is known to be effective and efficient for AutoML applications [105] and under active maintenance.

Table 6.2: Summary of dataset statistics in the training, validation and test splits. All four datasets exhibit class imbalance.

Datasets	SOTAB	Walmart-Amazon	ICPSR-Detection	ICPSR-Direction
Relationship Type	Equivalence	Equivalence	Parent-Child	Parent-Child
Number of Concepts	46 / - / 46	5126 / 2507 / 2484	140 / 54 / 54	79 / 46 / 46
Number of Pairs	1035 / - / 1035	6144 / 2049 / 2049	9730 / 1431 / 1431	3081 / 1035 / 1035
% of Minority Class	1.26% / - / 1.26%	9.38% / 9.42% / 9.42%	1.63% / 4.05% / 3.98%	2.08% / 3.29% / 3.29%

6.6 Experiments

6.6.1 Experimental Setup

6.6.1.1 Datasets

We evaluate OPENFORGE on four datasets involving two relationship types: equivalence (SOTAB dataset for column types matching and Walmart-Amazon dataset for entity matching) and parent-child (ICPSR-Detection and ICPSR-Direction datasets for taxonomy induction). We have sourced SOTAB and ICPSR datasets particularly for metadata integration tasks.

SOTAB Dataset. The Schema.org Table Annotation Benchmark (SOTAB) [97, 96] provides manually verified column type annotation using two vocabularies, `schema.org` [70] and DBPedia [101]. We merge concepts from both vocabularies to form an initial collection of concepts. If a table column is annotated with concepts from both vocabularies, we consider the two concepts equivalent. The many-to-many correspondences from columns to concepts can challenge the priors. This allows us to test how the proposed MRF formulation can identify conflicting relationships. Given the limited vocabulary size, we split it evenly into training and test sets.

Walmart-Amazon Dataset. This dataset has been widely used to evaluate entity matching algorithms [117, 164]. It provides a challenging setting (e.g., noisy attributes and missing data) for matching product entities between two major e-commerce platforms. We use the same training, validation, and test splits as prior work.

ICPSR-Detection and ICPSR-Direction Datasets. ICPSR, the institute maintaining the world’s largest social science data archive [82], makes their controlled vocabularies publicly available. We create two datasets of parent-child relationships from ICPSR’s Subject Thesaurus [81]. Following the task definitions in [142], the ICPSR-Detection dataset is for hypernymy detection where one predicts whether the hypernymy relationship exists between a pair of concepts (i.e., binary classification) while for the ICPSR-Direction dataset,

one needs to identify which concept is broader in a given pair of concepts if the relationship exists (i.e., multi-class classification). The dataset comprises social science terms from various disciplines such as political science and economics. We manually identify concepts with transitive parent-child relationships (e.g., `credentials` $\rightarrow$ `academic degrees` $\rightarrow$ `doctoral degrees`) and divide the collection of transitive concept tuples into training, validation, and test sets at ratio of 0.6:0.2:0.2. The ICPSR datasets have no column data associated with each concept as ICPSR does not use subject terms to annotate table elements. The prior beliefs of these two datasets can only be learned from concept names, which is a more challenging setting.

Table 6.2 summarizes dataset statistics. Notably, all datasets exhibit highly imbalanced class distributions, with minority class proportions in test splits ranging from 1.26% to 9.42%, which poses a significant challenge for any solution.

6.6.1.2 Baselines.

For each task defined by a relationship, we compare our approach with the previous state-of-the-art (SOTA) methods as well as latest LLMs, GPT-3.5 and GPT-4, from OpenAI.

Unicorn. Unicorn [164] is a unified multi-task model for data integration tasks like entity and schema matching. It leverages multi-task learning and a Mixture-of-Experts architecture to enable knowledge sharing across tasks and datasets. Additionally, Unicorn claims to support zero-shot prediction for new tasks without requiring labeled data. We compare OPENFORGE with Unicorn on datasets involving equivalence relationships.

COMA. COMA [50] combines multiple match algorithms to improve the schema matching accuracy. Later extensions of COMA offer ontology matching among other features.

ADA. [183] propose a data-driven solution for automatically discovering attributes in relational databases, which we refer to as ADA for convenience. ADA clusters columns into attributes based on data distribution similarities where each attribute represents a unique semantic concept (e.g., telephone numbers).

Ditto. Ditto [103] casts entity matching as a sequence-pair classification task and fine-tunes pre-trained language models with labeled data and optimizations to achieve strong performance.

COMA and ADA, recognized as strong schema matching baselines [98], are compared with OPENFORGE on the SOTAB dataset, where the setup is most aligned with schema matching. We include Ditto for comparison on the Walmart-Amazon dataset as it is the task-specific SOTA method for entity matching.

Chain-of-Layer. [180] propose Chain-of-Layer for automatic taxonomy induction from a given set of entities. It heavily relies on prompting a LLM to construct a taxonomy from

top to bottom in an iterative manner and uses another smaller language model to remove hallucinated entities and relationships.

HypernymySuite. [142] investigates hypernymy detection through two primary approaches: pattern-based methods, which utilize predefined lexico-syntactic patterns to identify hierarchical relationships between words, and distributional methods, which leverage word co-occurrence statistics in large corpora.

We compare OPENFORGE with Chain-of-Layer and HypernymySuite on datasets involving parent-child relationships.

GPT-3.5/4. We prompt GPT-3.5 and GPT-4 [24, 126] to predict relationships between concept pairs using task descriptions and few-shot learning. We use OpenAI APIs to interact with the models identified as `gpt-3.5-turbo-0125` and `gpt-4-turbo-2024-04-09`.

Due to class imbalance in the datasets, we evaluate the quality of all approaches using F1 score (macro F1 for multi-class classification) and report the best results of each approach. For example, LLMs can obtain higher scores without few-shot learning in some cases. Additionally, we report the runtime of our MRF modeling in seconds for efficiency and scalability analysis.

6.6.1.3 Hardware

Experiments involving GPU-accelerated MRF inference, fine-tuning and inference of open-source LLMs, and runtime measurement are done on a node with a NVIDIA A40 40GB GPU, 16 cores of Intel Xeon Gold 6226R processors and 256 GB of RAM on a shared computing cluster. The rest of experiments are conducted on a local server with 16 cores of Intel Xeon Bronze 3106 processors and 256 GB of RAM.

6.6.2 Quality Comparison

RQ1: How does OpenForge compare with baselines for various metadata integration tasks? As shown in Table 6.3, Open-

Forge consistently outperforms the baselines by significant margins in F1 score across all datasets (e.g., 4-36 F1 points improvement over the second-best baselines). These results highlight OPENFORGE’s flexibility and robustness in capturing different relationship types, outperforming both specialized and general-purpose baselines. We attribute the superior performance to three key factors:

1. Joint Probability Modeling: MRF effectively models the joint probability over random variables, especially with strong priors given by fine-tuned LLMs, enabling more accurate predictions.

Table 6.3: Comparison of F1 score across methods and datasets.

Datasets	Methods					
	Unicorn	COMA	ADA	GPT-3.5	GPT-4	OPENFORGE
SOTAB	0.48	0.59	0.10	0.73	0.75	1.00
	Unicorn	Ditto		GPT-3.5	GPT-4	OPENFORGE
Walmart-Amazon	0.87	0.87		0.67	0.84	0.91
	Chain-of-Layer	HypernymySuite		GPT-3.5	GPT-4	OPENFORGE
ICPSR-Detection	0.37	0.21		0.58	0.79	0.91
ICPSR-Direction	0.01	0.34		0.41	0.45	0.81

2. Preservation of Transitivity: By incorporating potential functions for ternary cliques, OPENFORGE ensures that invalid assignments violating relationship transitivity are avoided.
3. Dependency Learning: Fine-tuning hyperparameters of MRF modeling and inference on a validation dataset with a class distribution similar to the test set allows OPENFORGE to capture relationship dependencies more effectively.

We discuss some highlighted results on each dataset below.

6.6.2.1 SOTAB dataset

OPENFORGE achieves a perfect F1 score of 1.0 even though the prior beliefs used as inputs to MRF modeling are far from perfect. We report a detailed comparison of prior models and OPENFORGE (prior beliefs + MRF-based modeling) in Section 6.5.1. These results provide strong evidence for the advantage of modeling relationship transitivity and dependencies using MRFs. Among the baselines, GPT-4 is the second-most effective method but trails OPENFORGE by 25 F1 points. COMA employing multiple matching strategies exhibits decent F1 score of 0.59 while another traditional method ADA using only the Earth Mover’s Distance as the similarity measure appears to be the least effective. Unicorn, the SOTA for data matching tasks, performs less competitively on this dataset. The suboptimal performance suggests that Unicorn, despite leveraging multi-task learning, finds it challenging to generalize to the metadata integration tasks and datasets. This is most likely due to two reasons: (1) the SOTAB dataset is not seen in pretraining of Unicorn, and (2) equivalent pairs in Unicorn datasets often share common parts whereas column values associated with equivalent types in SOTAB are heterogeneous and may have no overlap.

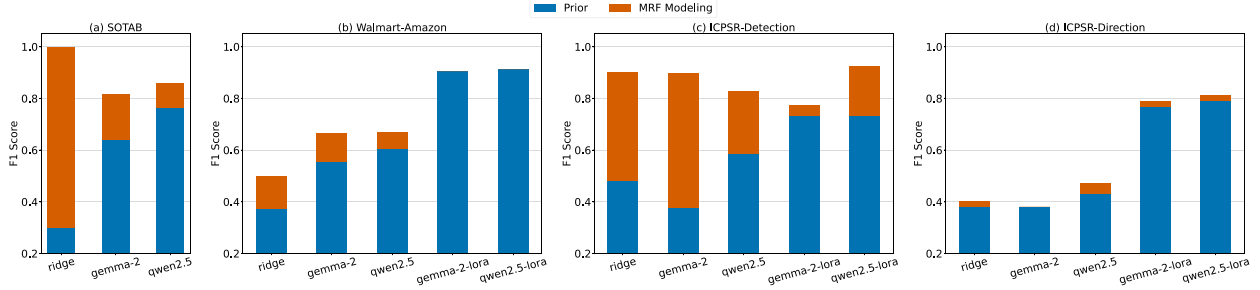


Figure 6.6: Comparison of F1 score between prior models and OPENFORGE on (a) SOTAB, (b) Walmart-Amazon, (c) ICPSR-Detection and (d) ICPSR-Direction datasets. We omit gemma-2-lora and qwen2.5-lora on SOTAB due to insufficient training data.

6.6.2.2 Walmart-Amazon dataset

OPENFORGE achieves an F1 score of 0.91, outperforming Unicorn, the second-best method, by 4 F1 points. The performance gap is smaller on this dataset as Unicorn was pre-trained on multiple entity matching datasets including the Walmart-Amazon dataset, making it a strong baseline. Additionally, the sparsity characteristic of this dataset (i.e., many entities appear only once in the test set and the predictions of many entity pairs are independent of others) diminishes the advantage of explicitly modeling relationship transitivity and dependencies. Ditto, the SOTA task-specific approach for entity matching, performs comparably to Unicorn due to its use of supervised learning. Among other baselines, GPT-4 falls behind OPENFORGE by 7 F1 points, while GPT-3.5 lags even further, trailing by 24 F1 points.

6.6.2.3 ICPSR-Detection and ICPSR-Direction datasets

OPENFORGE outperforms GPT-4 by 12 F1 points and GPT-3.5 by 33 F1 points on the ICPSR-Detection dataset, and by 36 F1 points and 40 F1 points, respectively, on ICPSR-Direction. The specialized approach, Chain-of-Layer, performs poorly in both tasks due to its assumption that all concepts belong to the same domain and fit within a single structured hierarchy, which does not hold for these datasets. The pattern-based approach, Hypernymy-Suite, performs the worst on ICPSR-Detection, as many concepts in the dataset are absent from the corpus it relies on for pattern extraction and it has limited capability in handling out-of-vocabulary concepts. While HypernymySuite obtains a moderate score on ICPSR-Direction, it still falls behind due to its reliance on extracted patterns, which struggle with directionality inference, particularly for unseen concepts.

6.6.3 Prior Models

RQ2: Can MRF modeling improve on various prior beliefs and by how much?

Figure 6.6 illustrates the impact of prior models on the result quality of OPENFORGE for the SOTAB, Walmart-Amazon, and ICPSR datasets. To save space, we present results for up to five prior models: the Ridge classifier (the top-performing ML model among three tested), gemma-2, qwen2.5, and their LoRA fine-tuned classifier variants, i.e., gemma-2-lora and qwen2.5-lora. On all datasets, our MRF modeling significantly improves the results quality over the prior models, with gains ranging from 2 to 70 F1 points. These results demonstrate the effectiveness of MRF modeling in enhancing prior predictions and correcting mispredictions that violate relationship transitivity.

We observe that overall LoRA fine-tuned LLMs outperform vanilla LLMs with few-shot learning by a large margin. For instance, the LoRA fine-tuned gemma-2 model achieves an F1 improvement of 52 points over the vanilla gemma-2 model, while the corresponding improvement on the Walmart-Amazon dataset is 45 F1 points. This highlights the potential of fine-tuning vanilla LLMs to obtain strong performance for classification tasks. Notably, on the Walmart-Amazon dataset, MRF modeling does not provide additional improvements for the gemma-2-lora and qwen2.5-lora models. This can be attributed to two factors: (1) the dataset’s sparsity, where many entity pairs in the test set are independent, reducing the occurrence of relationship transitivity violations; and (2) fine-tuned LLMs serving as strong priors, with manual checks confirming that none of their mispredictions violate relationship transitivity. Nevertheless, applying MRF modeling does not degrade the quality of the results, preserving the performance of these strong prior models. On the ICPSR-Direction dataset, the regularization effect of MRF modeling is diminished. This is because the ternary factor table has 27 configurations, only four of which violate transitivity, whereas in the binary classification setting, two out of eight configurations are invalid.

Guide for Practitioners. When sufficient training data (e.g., thousands of examples) are available, fine-tuning LLMs using parameter efficient fine-tuning techniques for domain-specific tasks yields very strong results and is highly recommended. In cases where training data are lacking, prompting a LLM to obtain prior beliefs offers a viable alternative.

6.6.4 Efficiency of MRF Modeling

RQ3: How efficient are MRF construction and inference? Figure 6.7 shows the MRF construction and inference time of different inference algorithms (i.e., pgmpy-MPLP, PGMax-CPU-LBP, and PGMax-GPU-LBP) on four datasets. All runtime numbers are reported using either a single CPU or GPU. We exclude the Shafer-Shenoy algorithm (im-

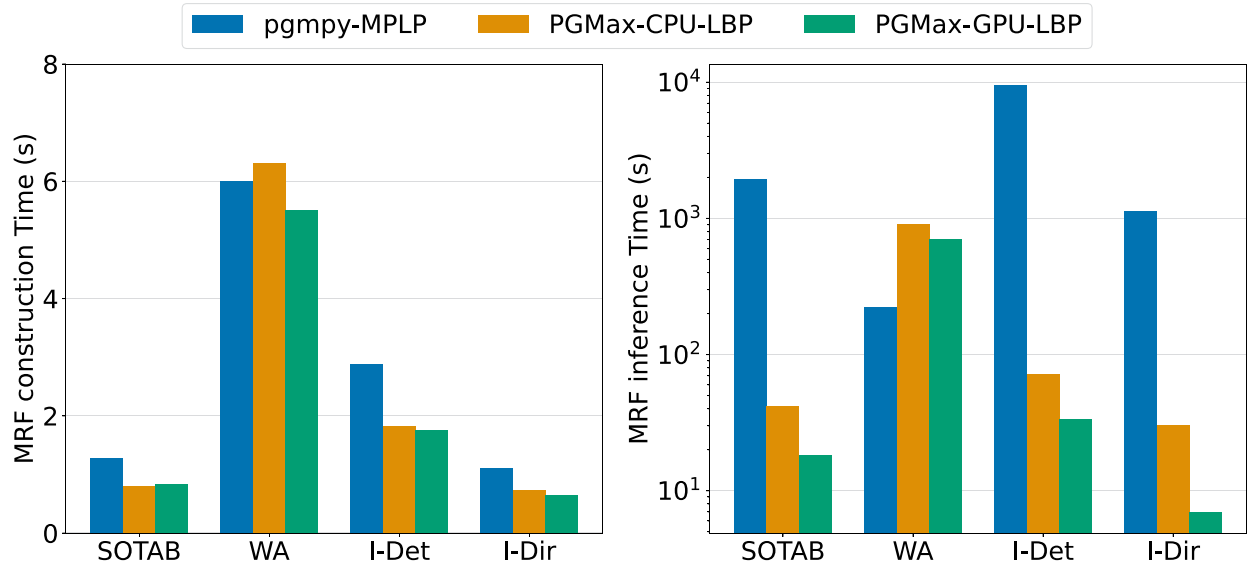


Figure 6.7: MRF construction and inference time of three inference algorithms using a single CPU or GPU. WA, I-Det, and I-Dir are shorthands for Walmart-Amazon, ICPSR-Detection, and ICPSR-Direction datasets, respectively.

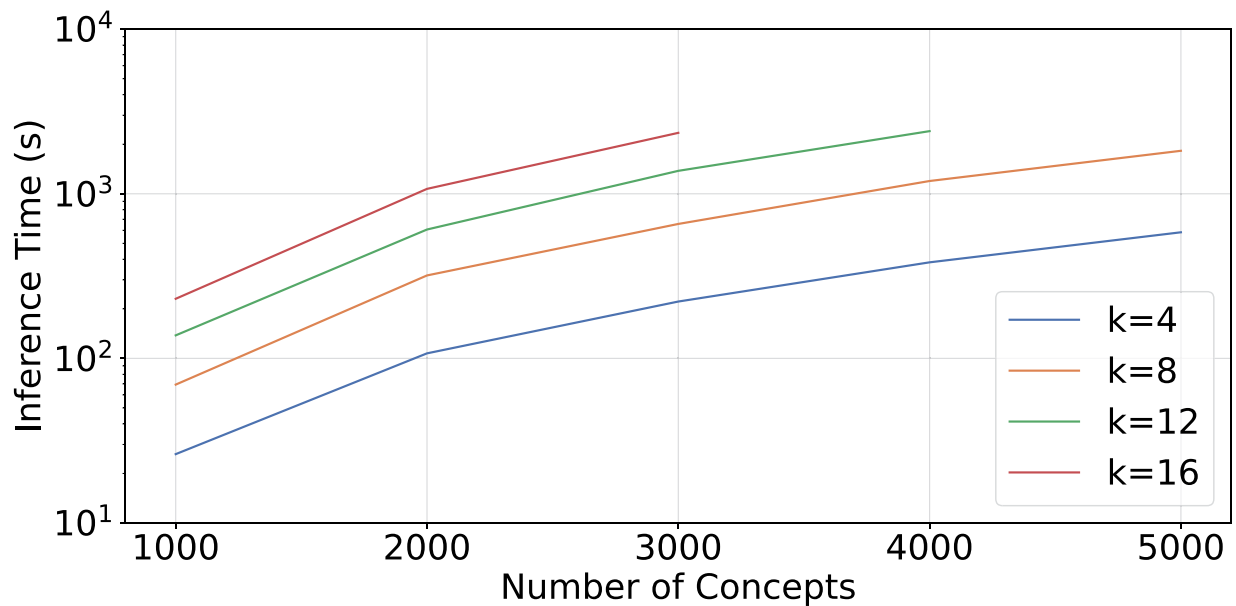


Figure 6.8: Scalability of MRF inference w.r.t. the number of concepts and connectivity k . $k = 12$ scales up to 4000 concepts (nearly 8 millions of random variables) and $k = 16$ scales up to 3000 concepts (about 4.5 millions of random variables).

plemented in pyAgrum) and Gibbs sampling (implemented in pgmpy) from the comparison as they do not complete execution within an hour on our smallest dataset.

The three algorithms under consideration exhibit efficient factor graph construction, with

completion time of a few seconds across all datasets. The primary distinction between these algorithms lie in their inference time. GPU-accelerated LBP (i.e., PGMax-GPU-LBP) is overall the most efficient, achieving more than 2x speedup compared to CPU-based LBP (i.e., PGMax-CPU-LBP) and over two orders of magnitude speedup compared to pgmpy-MPLP on CPU. The speedup is particularly significant on ICPSR datasets, which have over a thousand of random variables and more than twenty thousands of factors in the constructed graph. On ICPSR-Detection, pgmpy-MPLP does not complete running within two and a half hours. In contrast, PGMax-GPU-LBP finishes the inference in just 33.4 seconds and PGMax-CPU-LBP completes in 72.1 seconds. The superior performance of PGMax-GPU-LBP can be attributed to their flat array-based implementation of LBP in JAX [22] which leverages just-in-time compilation and parallelized array operations to optimize performance on hardware accelerators. However, for smaller graphs, such as the local MRFs in the Walmart-Amazon dataset, each of which has tens of random variables due to the dataset’s sparsity, pgmpy-MPLP appears to be more efficient while PGMax-GPU-LBP remains 1.3x faster than PGMax-CPU-LBP.

Although PGMax-LBP inference is slower on a single CPU compared to GPU, we can batch process independent local MRFs in the Walmart-Amazon dataset across multiple CPUs. When we use all 16 CPUs on the computing node, the runtime of PGMax-CPU-LBP can be reduced from nearly 1000 seconds to under 70 seconds.

6.6.5 Scalability

RQ4: How does MRF inference scale as MRF grows in size with a fixed amount of memory? Given that MRF inference is often the most time- and memory-intensive component of the solution, we evaluate its scalability under fixed resources: a single GPU (NVIDIA A40, 40 GB) and 256 GB of RAM. We synthesize sparse factor graphs with varying numbers of concepts (up to 5,000, translating to over 12 million random variables) and connectivity levels k (up to 16) as discussed in Section 6.4.2. These factor graphs consist of binary random variables with fixed prior probabilities and ternary factors with predefined potential values. We execute the PGMax-GPU-LBP algorithm using its default hyperparameters for 200 iterations.

Figure 6.8 illustrates the relationship between inference time and the number of concepts for varying connectivity levels. As expected, the inference time grows with both the number of concepts and the connectivity k . The inference times for all completed runs remain within 40 minutes. For connectivity levels $k = 4$ and $k = 8$, the algorithm completes inference on graphs with 5,000 concepts in under 30 minutes. At higher connectivity levels ($k = 12$ and

$k = 16$), the algorithm can handle graphs with up to 4,000 and 3,000 concepts, respectively, before exceeding the memory limits. Notably, a graph with 3,000 concepts corresponds to approximately 4.5 million pairs of concepts or random variables for prediction—significantly larger than the test set size of any existing public dataset for data matching or taxonomy induction. Hence, we consider the inference algorithm good enough for handling many use cases.

6.7 Discussion

More Types of Relationships. While OPENFORGE currently focuses on equivalence and parent-child relationships, its probabilistic framework can be extended to handle more diverse relationship types. For instance, directional relationships including part-whole and causal dependencies can be modeled similarly to parent-child relationships, with adjustments to the configurations of ternary factors to capture their unique properties.

Beyond Transitivity. While OPENFORGE primarily enforces transitivity in relationship inference, we acknowledge that some semantic correlations between concept pairs exist independently of transitivity. In Section 6.4.2, we leverage embedding similarities to identify semantically related concept pairs when constructing local MRFs, indirectly capturing such dependencies. Nevertheless, these semantic correlations are not explicitly modeled within the MRF model, which could further refine relationship assignments.

Future work could extend the model by incorporating pairwise semantic factors directly into the MRF to formally represent such dependencies. This would enable OPENFORGE to capture richer relational patterns beyond transitivity, further enhancing its expressiveness and robustness in complex metadata integration tasks. Meanwhile, it may introduce additional computational complexity, particularly for large datasets. Balancing model expressiveness with inference efficiency will be a key challenge to tackle in future work.

6.8 Related Work

We discuss data matching, from the data management community, and taxonomy induction, from the semantic web community, which are the most relevant to our work. Moreover, we review applications of probabilistic models in the broader area of data integration and summarize the role of metadata in dataset discovery.

Data Matching. Research efforts in generalizing matching techniques have primarily focused on solutions that generalize across domains [149] or tasks [164]. To the best of our knowledge, Unicorn, a multi-tasking model for data matching [164], represents the SOTA

in generalization across tasks. It supports matching tasks for data integration, such as entity matching and schema matching, by primarily focusing on equivalence relationships. For (meta)data integration, OPENFORGE not only goes beyond equivalence relationships to discover and resolve parent-child relationships, but also our experiments show that, despite Unicorn adopting a multi-task learning approach, it is still challenging for Unicorn to generalize to the new matching tasks and datasets of metadata integration. Parallel to multi-task learning, we demonstrate the promising potential of transfer learning using large language models with fewer than 10 billion parameters for various tasks. With parameter-efficient fine-tuning techniques such as LoRA [77], multiple tasks can share the same base model by incorporating lightweight adapters. These adapters are significantly smaller than the base model in size yet effectively enhance result quality across downstream tasks, making this approach highly efficient and scalable.

Taxonomy Induction. The taxonomy induction literature has extensively studied the problem of extracting parent-child relationships from text documents [75, 142, 148]. Various traditional frameworks have posed the taxonomy induction problem with a probabilistic formulation [151, 13]. For example, Snow et al. define the objective as finding a parent-child graph that maximizes the probability of observed evidence [151]. Their formulation relies on two additional independence assumptions to decompose the joint probability so that they can solve the problem with a heuristic search algorithm. This inspired us to design a simpler and more intuitive joint probability formulation, i.e., finding the most probable relationship graph given the existing evidence. This alternative formulation has two advantages. First, it allows us to incorporate various powerful priors, such as LLMs. Second, it allows us to cast our problem to MRF to incorporate relationship properties, such as transitivity, to refine the results. Bansal et al. also propose an MRF formulation for taxonomy induction [13]. OPENFORGE not only extends this formulation with an additional relationship type, it also explores different ways of obtaining prior beliefs and learning parameters for the MRF formulation. Chain-of-Layer [180], SOTA in the literature, iteratively prompts a LLM to construct the taxonomy while employing a smaller language model to reduce hallucinations of the LLM. However, our experiments reveal that Chain-of-Layer struggles to construct taxonomies effectively when working with a diverse set of metadata concepts from the social science domain. The deficiencies are mainly attributed to the prompts tailored for taxonomies with simple structures (e.g., country-state hierarchies) and the smaller language model’s inability to generalize effectively to entities within the social science domain.

Probabilistic Modeling. Due to the prevalence of uncertain and noisy data in modern applications, probabilistic models have been extensively studied for probabilistic databases [146], information extraction [166], and data integration [4]. [146] introduce

graphical models to represent and query correlated tuples (e.g., independence and implication relationships) efficiently, framing query evaluation as an inference problem over probabilistic graphical models. However, their approach relies on exact inference, limiting their scalability to smaller graphs with tens of thousands of random variables as opposed to millions of variables in our case. [166] integrate Conditional Random Fields (CRFs) into relational query processing, demonstrating how probabilistic inference improves both the accuracy and efficiency of querying uncertain information extracted from unstructured sources. Their approach focuses on tasks such as named entity recognition and sequence labeling, where the linear structure of input text sequences makes linear-chain CRFs a natural fit. In contrast, our integration problem requires a more general graph structure to model the transitivity of ternary concept groups. Consequently, their CRF-based techniques and optimizations, designed for linear sequences, do not directly apply to our setting. [4] establish the theoretical foundations for uncertain data integration within the Local-As-View framework, introducing novel containment notions—equality-containment and superset-containment—to model how uncertain data from diverse sources can be consistently merged. While their theoretical framework is sound, the paper leaves the development of efficient query processing techniques and real-world applications as open areas for future work.

Metadata and Dataset Discovery. Metadata has been extensively used in facilitating downstream tasks such as data discovery and data sharing. Google dataset search indexes the metadata of datasets (published by users in `schema.org` format) [23]. Auctus, a domain-specific and open-source dataset search engine, supports spatial and temporal data augmentation through indexing the data summaries describing dataset contents (e.g. grid size for spatial data) [29]. Raw metadata (e.g., column names and tags) [17, 121], concept annotations [116], and concept hierarchies [91] have been used as data summaries to be indexed for table union and join search. Besides data cataloging, the hierarchies inferred by OPENFORGE can further enhance the results quality of downstream data discovery tasks, particularly for domain-specific data repositories. For instance, datasets tagged with metadata of varying granularity could better align with user queries, improving retrieval accuracy [128, 121]. Exploring these applications represents a promising direction for future work.

6.9 Conclusion

We introduce the metadata integration problem and propose OPENFORGE, a data-driven solution that unifies metadata concepts for a given relationship. By combining advanced prior models with probabilistic modeling and inference using Markov Random Fields, OPENFORGE effectively resolves relationships between metadata concepts while enforcing

essential relationship properties such as transitivity. Our approach also demonstrates both efficiency and scalability across various datasets. A promising direction for future work is leveraging integrated metadata vocabularies to facilitate the discovery of siloed datasets.

CHAPTER 7

Observatory: Characterizing Embeddings of Relational Tables

7.1 Introduction

The advances of pretrained language models for NLP tasks such as summarization and dialog have sparked similar interest and progress in embedding relational tables for tasks such as table question answering [76], entity matching [47, 103], semantic column type annotation [181, 47, 157], and data integration and augmentation [28, 161, 43]. Most of these models are built on top of language models such as BERT [49] and specialized to take into account the structure of tables, for example, by leveraging vertical attention to incorporate information across rows [177].

As these table embedding models have shown strong performance on a variety of tasks, researchers and practitioners are also interested in using these pretrained models for new applications and in new domains. However, the process of identifying a suitable model typically involves trial and error due to a lack of understanding regarding the strengths and limitations of these models and their learned representations. This knowledge gap can produce inefficiency and even failures in downstream usage. Moreover, researchers have little visibility into the behaviors and generalizability of existing table embedding models beyond their performance on particular downstream tasks. Hence, there is a pressing need to understand the strengths and weaknesses of these models, especially in terms of the table embeddings they generate [52, 11].

To address this need, we propose OBSERVATORY, a formal framework for systematically analyzing language- and table embedding models from the perspective of what characteristics of relational tables these models do and do not capture in their learned embedding representations. OBSERVATORY presents eight primitive properties motivated both by invariants in Codd’s relational data model [39, 40] and by statistical considerations regarding data distributions in downstream tasks: for instance, if embeddings are sensitive to row and

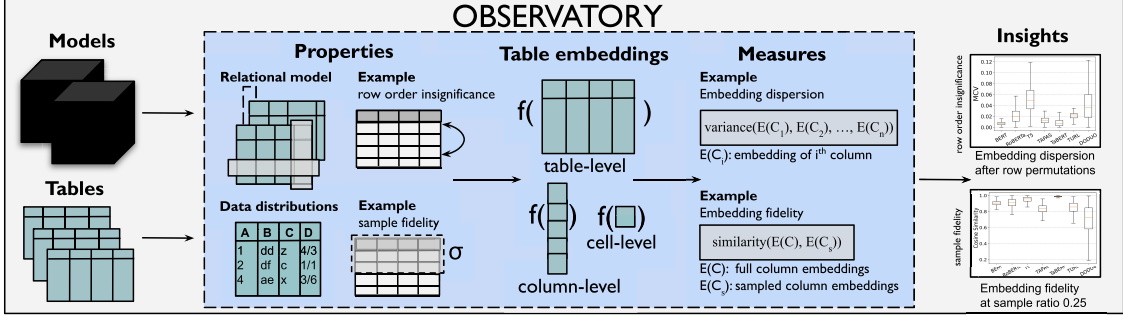


Figure 7.1: Overview of OBSERVATORY and how it solicits understanding of opaque table embedding models by measuring properties motivated by the relational data model and data distributions. We illustrate the framework for two out of eight properties: 1) row order insignificance, and 2) sample fidelity.

column order or sample size. Each of these properties is associated with a measure that quantitatively characterizes embedding representations over relational tables (see Figure 7.1 for an overview). Analogous to task-agnostic analyses of language models [141, 2], such data-specific evaluations of embeddings offer valuable insights into model behaviors, which are connected to various downstream applications (Section 7.6 gives more details). With OBSERVATORY, we 1) consolidate properties of relational tables important to reflect in table embeddings, 2) contribute a framework and implementation thereof, enabling researchers and practitioners to analyze the capabilities of existing and new models with respect to these properties, and 3) provide insights into the strengths and limitations of nine popular models through their learned representations over tabular data, which can inform researchers and practitioners of model selections and novel model designs.

Along with the implementation of OBSERVATORY, we collect and synthesize a suite of datasets for evaluation purposes, and present a comprehensive analysis of nine commonly used language and specialized table embedding models. Some key insights we surface in our analysis are that the embeddings of some models are sensitive to the order of rows and, in particular, the order of columns, while embeddings of some models are robust to uniform sampling. Moreover, we find that none of the models reflect functional dependencies among columns in tables. Although we do not aim to, and cannot, analyze all existing models, our implementation of OBSERVATORY is extensible such that researchers and practitioners can use OBSERVATORY for analysis of new models by specifying the procedure of embedding inference following the implemented interface. In summary, we make the following contributions:

- We propose OBSERVATORY, a framework including eight primitive properties and corresponding measures for systematically analyzing embedding representations over relational

tables.

- We implement and open-source a prototype of OBSERVATORY, which covers nine popular table embedding models while also being extensible for evaluation of new models.
- We present a comprehensive analysis with OBSERVATORY and provide novel insights into the strengths and limitations of evaluated models and their learned table representations.

7.2 Related Work

7.2.1 Language and Table Embedding Models

Language Models. BERT [49] is among the first transformer-based pretrained language models, generating contextual embeddings by predicting masked tokens. Subsequent optimizations like RoBERTa [107] and expansions in model size and tasks (e.g., T5 [137]) have driven rapid advancements. Language models soon progress from predictive tasks to sequence generation, exemplified by GPT models [136]. Beyond unstructured language tasks, investigations explore language models’ capabilities for structured inputs like tabular data. Narayan et al. [120] uses T5 for data wrangling, and recent GPT-based conversational models directly handle table understanding tasks [95, 90].

Table Embedding Models. TaBERT [177] pioneers extending pretrained language models to tabular data. It employs token-level embeddings with additional positional embeddings, incorporating vertical attention for inter-row information and a masked column name prediction objective inspired by BERT. Subsequent models, including TURL [47], TAPAS [76], and TaPEX [106], facilitate applications like table question answering, table understanding, and data preparation. For a comprehensive overview, we refer readers to surveys by Dong et al. [52] and Badaro et al. [11]. The latter emphasizes the need for intrinsic analysis of table embedding models, which we take a first step towards addressing with OBSERVATORY.

7.2.2 Analysis of Embedding Models

Analysis of Language Embedding Models. Efforts to comprehend and evaluate LMs involve task-specific [165] and task-agnostic analyses [141]. Task-agnostic investigations, exemplified by CheckList [141], explore internal LM behavior and capacities through unit-test-like assessments (e.g., whether a LM can handle negation). In line with this, OBSERVATORY proposes relational data model-inspired properties, considering practical data distribution

factors for downstream applications. Recently, Sui et al. [158] introduces a benchmark evaluating LMs on seven table tasks (e.g., cell lookup) while varying, among others, prompt designs and table input formatting. However, it falls short in examining fundamental properties of relational tables and data distributions, and excluding specialized table embedding models.

Analysis of Table Embedding Models. Limited analyses exist on table embedding models. Wang et al. [169] assess the impact of explicitly modeling table structure in transformer architectures for table retrieval, revealing the modest contribution of table-specific model design. However, this evaluation is confined to retrieval tasks and lacks insights into intrinsic model limitations affecting downstream performance. Dr.Spider [30] benchmarks text-to-SQL models for perturbation robustness, while OBSERVATORY introduces novel properties, including perturbation robustness, unexplored until now. Recent work [154] introduces LakeBench, highlighting performance gaps in specialized table embedding models for data discovery. In contrast, OBSERVATORY evaluates embedding representations based on broader table-specific properties relevant to diverse downstream tasks. Koleva et al. [92] examines patterns in table-specific attention mechanisms, remaining task-agnostic. Unlike OBSERVATORY, it doesn’t link model analysis with relational and data distribution properties of tables.

7.3 Observatory

In this section, we present OBSERVATORY, our methodology for characterizing embedding representations over relational tables. OBSERVATORY features two sets of properties that are agnostic to downstream tasks and motivated by the relational model [39, 40] and data distributions. For each property, OBSERVATORY proposes a measure to quantify how well embedding representations align with the property specification. This allows users to gain insights into the strengths and weaknesses of different models and to even compare models through a consistent lens.

7.3.1 Problem Statement

Various downstream applications may need different kinds of embeddings. For example, semantic column type detection is based on column embeddings whereas entity matching requires entity embeddings. Given that these embeddings look at different *levels of aggregation* of the table structure, we refer to these kinds of embeddings as levels of embeddings.

Definition 12 (Table Embedding Characterization). Given a pretrained model f , a corpus

of tables $T \in \mathcal{T}$, and a property $\mathcal{P}$ that characterizes a certain level of embeddings $\mathbf{E}_{\mathcal{P}}$ with a measure $\mathcal{M}$, table embedding characterization infers $\mathbf{E}_{\mathcal{P}}$ with f over each $T \in \mathcal{T}$ and computes $\mathcal{M}$ over the distribution of $\mathbf{E}_{\mathcal{P}}$.

A property $\mathcal{P}$ can characterize one or more levels of embeddings (e.g., it can apply for both row- and column-level embeddings). Properties in OBSERVATORY span five levels of embeddings: table, column, row, cell, and entity (so called *table embeddings*) while many of them are relevant to column-level embeddings. OBSERVATORY also focuses on Transformer-based embedding models. Technically, any pretrained model f , regardless of the architecture (encoder or encoder-decoder or decoder-only) can be integrated to and evaluated with OBSERVATORY, as long as f either natively exposes certain level of embeddings $\mathbf{E}_{\mathcal{P}}$ specified by $\mathcal{P}$ or exposes token-level embeddings that can be further aggregated to the level of $\mathbf{E}_{\mathcal{P}}$.

7.3.2 Relational Properties

The relational data model specifies both structural invariants and semantics. We first introduce two properties from structural invariants (namely, Row- and Column Order Insignificance), followed by two properties from structural semantics (namely, Join Relationship and Functional Dependencies).

Property 1 (Row Order Insignificance). A relational table can be viewed as a set of rows of which, in principle, the order is insignificant [39]. Tables may be stored in an ordered way, that is, rows may be ordered by dates, or ascending/descending values of a given column. Models that explicitly encode the table structure with position embeddings might reflect this order in the output embeddings. Awareness of the influence of row order on table embeddings is key to using them in a context of unordered tables. We consider column/row/table-level embeddings in this property.

Measure 1. Given a table T , let $\mathbf{E}(D^{(i)})$ denote the embedding of column/row/table D in the i -th row-wise shuffle of T for $1 \leq i \leq n$ (i.e., there are n row-wise permutations). We define the row order sensitivity as a high-dimensional dispersion measure $\mathcal{M}$ of n samples drawn from the embedding distribution, i.e.,

$$\mathcal{M}(\mathbf{E}(D^{(1)}), \mathbf{E}(D^{(2)}), \dots, \mathbf{E}(D^{(n)})).$$

The coefficient of variation (CV), the ratio of the standard deviation to the mean in the univariate setting, is a well-known measure of variability relative to the mean of a population. It has the merit of allowing for the comparison of random variables with different units or different means. Thus, we consider multivariate extensions of CV (MCV) that summarize

ID	year	competition		ID	year	competition
1	1993	Asian Championships	row permutation →	2	1994	Asian Games
2	1994	Asian Games		1	1993	Asian Championships
3	1997	World Championships		4	1997	Central Asian Games
4	1997	Central Asian Games		3	1997	World Championships
5	1998	Asian Games		6	1999	World Championships
6	1999	World Championships		5	1998	Asian Games

Figure 7.2: Illustration of row permutations.

relative variation of a random vector (instead of a random variable) into a scalar quantity. In particular, we use Albert and Zhang’s MCV [5] to compare row order sensitivity across models for the reasons that it takes into account correlations between variables and does not require the covariance matrix to have an inverse [5, 3], which is especially convenient when the number of observations (number of embeddings) is smaller than the number of variables (embedding dimensionality). Albert and Zhang’s MCV of embeddings $\{\mathbf{E}(C^{(i)})\}_{i=1}^n$ is computed as

$$\gamma_{AZ} = \sqrt{\frac{\mu^t \Sigma \mu}{(\mu^t \mu)^2}} \quad (7.1)$$

where μ is the mean vector and Σ is the covariance matrix.

In practice, the number of possible permutations can be large (i.e., factorial of the number of rows) for tables with high cardinality. For computational efficiency in the experiments, we use at most 1000 randomly generated permutations of each table.

Example. Figure 7.2 gives an example of row permutations. Given 6 data rows, there are in total $6! = 720$ possible permutations. Then for each column, we have 720 observations of embeddings, which is smaller than some embedding dimensionality (e.g., 768 of BERT). In this case, the covariance matrix derived from observations is singular. Nevertheless, Albert and Zhang’s MCV can be calculated whereas other MCVs surveyed in [3] can not.

Property 2 (Column Order Insignificance). Besides row order, some models exploit neighboring columns as context when learning representations based on the intuition that neighboring columns can provide local context [181, 186]. Analogous to row order insignificance, relational tables usually store data without preserving a particular column order. The (in)sensitivity of embeddings regarding the column order informs their suitability for tasks

such as join discovery and table understanding in relational databases with unordered tables versus views on Web and other media that may present data with related attributes next to each other. As in Property 1, we assess column/row/table embeddings.

Measure 2. Given a table T , let $\mathbf{E}(D^{(i)})$ be the embedding of column/row/table D in the i -th column-wise shuffle of T . Similarly, we measure the embedding variance using MCV in equation 7.1.

Property 3 (Join Relationship). The join operation combining tuples from two or more relational tables, is one of the essential operations for data analysis. Thus the problem of finding join candidates in a table repository has been extensively studied [182, 174, 192, 191, 63, 42]. Join candidates are typically identified by some notion of value overlap similarity such as Jaccard and containment [192, 191, 63] while the embedding approach has also been explored [42]. Their findings indicate that columns with significant value overlap are also close to each other in the embedding space. We investigate this postulate by assessing if there is a monotonic relationship between value overlap and embedding similarity.

Measure 3. Consider pairs of query and candidate columns (C_q, C_c) and their corresponding embeddings $(\mathbf{E}(C_q), \mathbf{E}(C_c))$. Two random variables can be derived, the embedding similarity measure $\mathcal{M}(\mathbf{E}(C_q), \mathbf{E}(C_c))$ and the value overlap measure $\mathcal{R}(C_q, C_c)$. In experiments, we use cosine similarity for $\mathcal{M}$ and containment for $\mathcal{R}$ where $\mathcal{R} = \frac{|C_q \cap C_c|}{|C_q|}$ and is not biased towards small sets [192, 191]. For completeness, we also experiment with Jaccard similarity (i.e., $\frac{|C_q \cap C_c|}{|C_q \cup C_c|}$) and multiset Jaccard similarity (i.e., $\frac{|C_q \cap C_c|}{|C_q| + |C_c|}$) for measuring value overlap.

With embedding similarity measure $\mathcal{M}$ and value overlap measure $\mathcal{R}$ calculated over n pairs of query and candidate columns $\{(M_1, R_1), (M_2, R_2), \dots, (M_n, R_n)\}$, we compute the Spearman's

rank correlation coefficient between $\mathcal{M}$ and $\mathcal{R}$ as

$$\rho = \frac{\text{cov}(R(\mathcal{M}), R(\mathcal{R}))}{\sigma_{R(\mathcal{M})}\sigma_{R(\mathcal{R})}} \quad (7.2)$$

where $R(\cdot)$ denotes the rank of a sample, $\text{cov}(\cdot, \cdot)$ is the covariance of the rank variables, and $\sigma_{(\cdot)}$ denotes the standard deviation.

Note that the Spearman coefficient ranges between -1 and 1, and considers the ranking values of two variables instead of raw variable values. A coefficient of 1 means the rankings of each variable match up for all data pairs and indicates there is a very strong positive monotonic relationship between two variables. We adopt the Spearman coefficient since it does not make any assumption of the underlying variable distributions.

Property 4 (Functional Dependencies). Let T be a relation with a set of attributes U . Relation T over U is said to satisfy a functional dependency, denoted $T \models X \rightarrow Y$ where $X, Y \subset U$, if for each pair s, t of tuples in T , $\pi_X(s) = \pi_X(t)$ implies $\pi_Y(s) = \pi_Y(t)$ [1]. Functional dependencies between columns provide a formal mechanism to express semantic constraints to the stored data, which is useful in many applications such as improving schema design, data imputation, and query optimization.

This property surfaces if models implicitly capture the relationship of functional dependencies in their representations (we are not aware of any model that explicitly takes functional dependencies into consideration in pretraining). Analogous to relationships between words [115] and entities in knowledge bases [21], the functional dependency relationship can be interpreted as a translation in the embedding space. Consider the relation triple $(\pi_X(s), r, \pi_Y(s))$, where r is the functional dependent relationship between value pair $\pi_X(s), \pi_Y(s)$. As demonstrated in [21], such relationship reflects as a *translation* between the embeddings $\mathbf{E}(\pi_X(s))$ and $\mathbf{E}(\pi_Y(s))$. The translation vector represents relationship r , which can be expected to remain equal in direction and magnitude across tuples if the relationship is preserved [21, 115]. More precisely, consider any pair s, t of tuples in T with a functional dependency $X \rightarrow Y$. We say that this functional dependency is preserved in an embedding space determined by a model f if

$$d(\mathbf{E}(\pi_X(s)), \mathbf{E}(\pi_Y(s))) = d(\mathbf{E}(\pi_X(t)), \mathbf{E}(\pi_Y(t)))$$

given $\pi_X(s) = \pi_X(t)$ where $\mathbf{E}(\cdot)$ is the embedding inferred with f and d denotes a distance metric preserving direction and magnitude.

Example. Consider a table T containing four columns in Figure 7.3. There exists a functional dependency between non-key attributes `country` and `continent`, i.e., `country` $\rightarrow$ `continent`. T satisfies this functional dependency because every instance of a specific value in column `country`, *Netherlands* for example, corresponds to the same value, i.e. *Europe*, in the corresponding tuples under column `continent`. By our definition, if an embedding space preserves functional dependencies, the squared Euclidean distances between embeddings generated for these specific value pairs will be (approximately) equal, despite influence of context on the embeddings.

Measure 4. Given a table T with functional dependency $X \rightarrow Y$, we refer to the group of tuples $\pi_{X \cup Y}$ with the same value v_X of determinant X as FD-group $\mathcal{G}_{v_X}$, to the value associated with v_X in the dependent attribute set Y as v_Y , and to the embeddings of these

ID	name	country	continent
1	Kathryn	Netherlands	Europe
2	Oscar	Netherlands	Europe
3	Lee	Canada	North America
4	Roxanne	USA	North America
5	Fern	Netherlands	Europe
6	Raphael	USA	North America
7	Rob	USA	North America
8	Ismail	Canada	North America

Figure 7.3: Table with a functional dependency `country` $\rightarrow$ `continent`. The colors illustrate different FD groups determined by the unique values in the `country` column.

values of the i -th entry in the group as $\mathbf{E}(v_{X,i})$ and $\mathbf{E}(v_{Y,i})$, respectively. For instance, there are three FD-groups under the functional dependency `country` $\rightarrow$ `continent` in the table shown in Figure 7.3, i.e., (Netherlands, Europe), (Canada, North America), (USA, North America) where the FD-group (Netherlands, Europe) has three entries.

Within each FD-group $\mathcal{G}_j$ of size $m_{\mathcal{G}_j}$, we calculate distance metric d for each embedding pair $(\mathbf{E}(v_{X,i}), \mathbf{E}(v_{Y,i}))$, denoted as d_{ji} . The average group-wise variance over all n FD-groups is calculated as:

$$\overline{S^2} = \frac{1}{n} \sum_{j=1}^n \frac{\sum_{i=1}^{m_{\mathcal{G}_j}} \|d_{ji} - \overline{d_j}\|_2^2}{m_{\mathcal{G}_j} - 1}$$

In our experiments, we take as distance metric d the L_1 - or L_2 -norm following [21], while other distance metrics preserving norm direction and magnitude are valid too. $\overline{S^2}$ approaches 0 if the *translation* between the group-wise FD value pairs in X (`country`) and Y (`continent`) remains approximately equal for each FD group. We note that this does not require a strictly injective model. That is, the same value across different table contexts is not necessarily mapped to exactly the same vector in the embedding space in order for this measure to approach 0.

In addition, it is expected that this measure shows higher value ranges over column sets without functional dependencies. We collect a set of functional dependencies over tables $\mathcal{T}_{FD}$ and a set of tables $\mathcal{T}_{-FD}$ in which no table contains functional dependent columns. We calculate the measure for all tables in the sets $\mathcal{T}_{FD}$ and $\mathcal{T}_{-FD}$. This yields two distributions of $\overline{S^2}$ values. If the embeddings preserve functional dependencies, $\overline{S^2}$ values over $\mathcal{T}_{FD}$ will be close to 0 and in general smaller than those over $\mathcal{T}_{-FD}$.

7.3.3 Data Distribution Properties

In practice, many aspects need to be considered when using embeddings including but not limited to the sample size, domain generalizability, robust representations of semantically similar values, and context. We introduce four properties involving data distributions that concern these four aspects.

Property 5 (Sample Fidelity). Large relational tables can easily have millions or even billions of rows. Embedding an entire table or even a single large column with a model is often infeasible due to constraints on the input length of models or memory constraints of computing resources. On the other hand, it may not be necessary to embed the full table for a downstream task [177, 157, 42]. In practice, existing work resorts to sampling, either up to the input limit or based on content relevance, as a straightforward workaround. While sampling provides a feasible solution, it also introduces a trade-off between computational cost and the fidelity of the embedding inferred from a smaller sample compared to the embedding that would have been obtained if the entire dataset were used. It is then essential to understand the fidelity of sample embeddings from a model by evaluating the extent to which sample embeddings deviate from the embeddings of full values.

Measure 5. Given a full column C and a sample C_S , we define sample fidelity as a similarity measure $\mathcal{M}$ between the embedding of the full column $\mathbf{E}(C)$ and the sample embedding $\mathbf{E}(C_S)$ where $\mathcal{M}$ can be cosine similarity for instance. Similar to [168], we split a full column into chunks with the shared header and obtain the full embedding by aggregating the chunk embeddings. This is because a full column may not fit into a single sequence for model ingestion.

For each column C , we perform uniform random sampling to get n distinct samples $\{C_1, C_2, \dots, C_n\}$ from C and report the average column sample fidelity

$$\frac{1}{n} \sum_{i=1}^n \mathcal{M}(\mathbf{E}(C_i), \mathbf{E}(C_{iS}))$$

as well as the multivariate coefficient of variation over the embedding set $\{\mathbf{E}(C), \mathbf{E}(C_1), \dots, \mathbf{E}(C_n)\}$. Since tables in a corpus may have various sizes, we experiment with different sampling fractions (e.g., 0.25, 0.5, and 0.75) instead of varying the absolute number of samples in evaluations.

This simple measure gives a good indication of computing efficiency and monetary cost. For example, provided that cloud vendors take a pay-as-you-go model, users do not need to pull out all their data to infer embeddings and pay the full scanning cost.

Property 6 (Entity Stability). Stability is a notion in NLP [170, 8] that indicates the variability of word embeddings relative to training data, training algorithms, and other factors in embedding model training. The idea is to use the overlap between K nearest neighbors of queries (i.e., words) found in different embedding spaces¹ as a proxy of agreement between embedding spaces. We borrow this notion to explore the (in)stability of entity embeddings.

Given n embedding spaces determined by embedding models $\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_n$, consider an entity cell $\mathbf{e} = (e_m, e_{md})$ in a relational table where e_m is the entity mention and e_{md} is associated metadata if exist (such as the entity linked to the cell from a knowledge base, the column name, and the table caption). Retrieve K nearest neighbor entities of $\mathbf{e}$ in each embedding space. The stability of entity $\mathbf{e}$ across n embedding spaces is defined as the average over all pairwise percent overlap between two embedding spaces.

Example. Take the entity column *competition* in Figure 7.2 for example. *World Championships* is an entity mention that links to a Wikipedia entity *1997_World_Championships_in_Athletics_-_Men's_Decathlon*. Depending on the context, the same entity mention may link to another distinct entity, for instance, *BWF_World_Championships*.

Measure 6. We consider the case when $n = 2$ (i.e., two embedding models $\mathbf{f}_1$ and $\mathbf{f}_2$). We randomly sample m entities, and for each entity $\mathbf{e}_i$, let s_1^i and s_2^i be the sets of K nearest neighbors of $\mathbf{e}_i$ in two embedding spaces, respectively. We compute the average entity stability as

$$\frac{1}{m} \sum_{i=1}^m \frac{|s_1^i \cap s_2^i|}{K}$$

which ranges between 0 and 1. A value of 1 indicates a perfect agreement between two embedding spaces while 0 indicates a complete disagreement.

For entity-centric downstream tasks, one can run this experiment over a model $\mathbf{f}_1$ to first see if the retrieved sets of K nearest neighbors to entities of interest fit their task domains. If not, one may want to try a different model $\mathbf{f}_2$ with a low entity stability relative to $\mathbf{f}_1$. This is because a model with high entity stability relative to $\mathbf{f}_1$ will be more likely to retrieve a set of entities similar to that of $\mathbf{f}_1$ and fail to fit task domains as well.

Property 7 (Perturbation Robustness). Neural model performance has been found vulnerable to input perturbations. For example, state-of-the-art text-to-SQL models are shown

¹An embedding space refers to a vector space that represents an original space of inputs (e.g., words or table columns).

to suffer from nuanced perturbations to database tables, natural language questions, and SQL queries [30]. Such perturbations are designed to preserve semantics and can reveal a model’s capacity to capture semantics. We hypothesize that preserving semantic similarities in the embedding space is key, especially, for downstream tasks such as retrieval, text-to-SQL and question answering. We therefore inspect the impact of input perturbations in the embedding space by measuring the robustness of column-level embeddings with respect to semantics-preserving perturbations.

Example. Three database perturbations curated by [30] include `schema-synonym`, `schema-abbreviation`, and `column-equivalence`. `schema-synonym` and `schema-abbreviation` replace the name of a column with its synonym (“country” → “nation”) and abbreviation (“CountryName” → “entry_name”), respectively. `column-equivalence` further perturbs both column names and contents, and may replace numerical columns with semantic-equivalent ones (“age” → “birthyear”).

Measure 7. Given a set of original columns $\{C_i\}_{i=1}^n$, we consider a set of perturbed variants $\{C'_{ij}\}_{j=1}^{m_i}$ for each C_i . The perturbations are semantics-preserving and can be at the schema level or data level or both. We compute the embedding cosine similarity of $(\mathbf{E}(C_i), \mathbf{E}(C'_{ij}))$ and average over all m_i pairs for each C_i . We draw a distribution plot of average cosine similarity over $\{C_i\}_{i=1}^n$ across models and also report a single number of cosine similarity averaged over all $\sum_{i=1}^n m_i$ pairs for each model.

Property 8 (Heterogeneous Context). Unlike coherent natural language sequences, tables are typically more heterogeneous comprising various types of data such as numeric, categorical, and datetime. As table embedding models mostly extend the architecture of language models and by default take context into consideration, it is less clear how much influence context has on embedding representations, especially for numeric data [79, 157]. Without context (e.g., subject columns² or neighboring columns), non-textual types of data, especially numerical columns, are typically hard to discriminate. Thus, it is important to understand the impact of context for many downstream tasks like semantic type prediction and relation extraction. In this property, we probe into the difference between contextual column embeddings and single-column embeddings for both textual and non-textual types of data.

Example. Figure 7.4 shows a table from the SOTAB benchmark [97]. The table does not have a header and consists of both textual and non-textual data columns. Without

²The subject column of a table, if exists, contains the entities the table pertains to.

context, `column 4` is hard to interpret on its own, which could be percentages, prices or any metric numbers. However, the neighbor column to the right, namely `column 5`, which refers to the currency of Romania, can provide clues to the semantic meanings of `column 4`. In this context, it is more likely that `column 4` contains price values.

<code>column 0</code>	<code>column 1</code>	<code>column 2</code>	<code>column 3</code>	<code>column 4</code>	<code>column 5</code>
Plan D	2013	448	Simon Urban	45.00	RON
Exams Dictionary for Upper ...	2010	1883	Engleza	95.95	RON
The greek connection	2011	180	Bogdan Hirt	20.00	RON

Figure 7.4: A table (without header) comprising textual and non-textual data columns.

Measure 8. To measure the effect of context, we consider four different input settings to get column embeddings as specified below. We compare embeddings of single columns with contextual column embeddings using their cosine similarity.

- (a) Only the column itself;
- (b) Subject column as context (if not exist, use the first textual column from the left of a table as the proxy);
- (c) Immediate neighboring columns on both sides as context;
- (d) The entire table as context.

7.4 Experiment Setup

7.4.1 Embedding Models

We consider well-established models and their variants that have been adopted for data management problems and open-sourced for public access. In particular, we select representative models from two categories: LMs and specialized table embedding models. Vanilla LMs are those designed for modeling natural language sequences and thus do not take into account the structure of tables or tabular data distributions. We include them in OBSERVATORY for comparison as many table embedding models share very similar architectures with weights initialized from LMs.

Language Models. We take in BERT [49], RoBERTa [107], and T5 [137]. BERT is a pioneer transformer-based model that learns contextual representations from unlabeled text. RoBERTa builds on top of BERT and systematically studies the impact of key hyperparameters and training data size. Both models are go-to options for a wide range of NLP tasks and are bases for many tabular language models. T5 is a representative of large language models whose largest variant has 11 billion parameters. We use base versions of all three models from the HuggingFace library [172] in the experiments.

Table 7.1: Overview of table embedding models and their design specifications (Column is abbreviated to Col.).

Model	Input	Output Embedding	Downstream Task
TURL	Table + metadata	Entity / Col. / Col. pair	Table interpretation/augmentation
DODUO	Table	Col. / Col. pair	Column type/relation prediction
TAPAS	NL question + table	Question / Table	Semantic parsing
TaBERT	NL question + table	Col. / Table	Semantic parsing
TaPEX	SQL query + table	Row / Table	Table Question Answering
TapTap	Table	Row	Data augmentation/imputation

Table Embedding Models. We include TURL [47], DODUO [157], TAPAS [76], TaBERT [177], TaPEX [106], and TapTap [186]. TURL, TAPAS, TaBERT, TaPEX, and TapTap first pretrain models over tables in an unsupervised manner by, for example, predicting masked column names or query execution results. The pretrained models are then fine-tuned for particular downstream tasks. We use pretrained models in the experiments as prescribed in our problem statement. DODUO directly fine-tunes a BERT-based model with labeled data from downstream tasks. See Table 7.1 for an overview of model specifications. The models we assess in experiments cover all levels of output embeddings, i.e., column, row, cell, and table embeddings.

7.4.2 Datasets

We use both relational database tables and web tables for evaluation.

WikiTables. The WikiTables [15] corpus contains 1.6M HTML tables of relational data extracted from Wikipedia pages. TURL preprocesses WikiTables and obtains an entity-rich dataset of 670,171 tables. We use the test partition released by TURL [46].

Spider. Spider [178], a widely-used semantic parsing and text-to-SQL dataset, includes 5,693 SQL queries over 200 databases across domains. We use the development set [179] and run HyFD [129], a functional dependency discovery algorithm, to create a dataset with

annotated functional dependencies. To avoid mining a massive number of functional dependencies, we set the size of determinant to 1 and found 713 functional dependencies. We also collect an equal number of random pairs of columns without the relationship of functional dependencies for our experiments.

Dr.Spider. Dr.Spider [30] designs perturbations to databases, natural language questions, and SQL queries in Spider to test the robustness of text-to-SQL models. We take advantage of database perturbation tests in Dr.Spider [31] to evaluate the property of perturbation robustness.

NextiaJD. Flores et al. [63] collected 139 datasets from open repositories such as Kaggle and OpenML for predicting joinable columns. They also divided datasets into four testbeds based on dataset file size. For example, NextiaJD-XS includes datasets smaller than 1 MB while NextiaJD-L consists of datasets larger than 1 GB. Candidate pairs of columns are labeled with the join quality using a measure that takes account of both containment and cardinality proportion with empirically determined thresholds. For our evaluation, we use all pairs with join quality greater than 0.

SOTAB. The Schema.org Table Annotation Benchmark [97] provides about 50,000 annotated tables collected from the WDC Schema.org Table Corpus for both column type and column property annotation tasks. We extract a subset that contains 5,000 tables for 20 semantic data types. The subset is balanced in terms of the number of non-textual and textual data types. Non-textual types include DATE, ISBN, POSTAL CODES, MONEY (monetary values), and QUANTITY (measurements as of weight etc.). We use this subset for measuring the property of Heterogeneous Context.

Note that a dataset may not accommodate all the properties. For example, WikiTables does not have information of which two columns can be joined, so we do not measure the property of Join Relationship over WikiTables. On the other hand, properties such as Functional Dependencies and Heterogeneous Context require synthesized datasets for evaluation purposes. Table 7.2 summarizes the datasets and assessed models for each property. Also note that TURL, TaBERT, and TapTap are excluded from certain experiments. This is because TURL is designed and implemented to output embeddings from entity-rich tables like those in WikiTables; TaBERT yields only column embeddings after the fusion of the vertical attention mechanism; and TapTap encodes single rows independently using a text template serialization strategy and only gives row embeddings.

Table 7.2: Overview of datasets and models for each property.

Property	Dataset	Models in Scope
Row order insignificance	WikiTables	Except TapTap
Column order insignificance	WikiTables	All
Join relationship	NextiaJD	Except TURL and TapTap
Functional dependencies	Spider	Except TURL, TaBERT, and TapTap
Sample fidelity	WikiTables	Except TapTap
Entity stability	WikiTables	Except TaBERT and TapTap
Perturbation robustness	Dr. Spider	Except TURL and TapTap
Heterogeneous Context	SOTAB	Except TURL and TapTap

7.4.3 Implementation

In general, we follow the original papers and their implementations in our evaluation. However, there are subtleties where extra consideration is needed, such as aligning the input and output across models for fair comparison. We make (minimal) design decisions in our implementation as discussed below.

Table Serialization. As Transformer-based models expect to take sequence inputs, a key input processing step is to serialize two-dimensional tabular data into flattened sequences of tokens. Table embedding models considered in this analysis generally follow two common types of serialization methods.

1. Row-wise serialization. Tables are parsed by rows, which are further concatenated with optional insertions of special tokens as delimiters. TURL, TAPAS, and TaBERT fall under this category despite the difference that TAPAS uses dedicated positional embeddings to indicate the row and column in which a token appears while TaBERT explicitly adds [SEP] tokens to mark boundaries of cells in the sequences.
2. Column-wise serialization. Alternatively, tables can be serialized by column. For DODUO, [CLS] tokens (as many as the number of columns) are inserted to separate values from different columns and are effectively used as column representations.

For each table embedding model, we adopt the serialization method as proposed in the original papers. Since vanilla language models do not have a default serialization method for tabular data, we experimentally apply row/column-wise serialization as applicable. In practice, models also enforce a length limit to token sequences (e.g. 512 is a common maximum). To ensure that all models take in (almost) the same inputs regardless of serialization

methods, we keep all the columns for each table, if possible, and preserve as many rows as the length limit permits. We use binary search to find the maximum number of rows that can fit into the input limit.

Embedding Retrieval. We use the embeddings provided by a model, if they are available. However, due to designs for particular downstream tasks, a model may not readily expose certain levels of embeddings needed for measuring a property. For instance, TAPAS does not give row or column embeddings out of the box. We circumvent this obstacle by observing that all the models can output token-level embeddings and some table embedding models have additional mask embeddings or positional embeddings that indicate to which row and column a token belongs. Therefore, we can aggregate token embeddings (by averaging them for example) to embeddings on a level (e.g. row or column) as needed. In particular, we take advantage of different serialization methods and use special tokens to retrieve row or column or table embeddings. As to cell embeddings needed for the property Functional Dependencies and entity embeddings as needed for the Entity Stability property, we keep track of token positions in the table and aggregate them accordingly. We take this alternative since inserting special tokens for each cell quickly uses up the input limit.

The practice of inserting special tokens and aggregating lower level of embeddings is common in the literature [47, 103, 157, 43]. As noted in our problem statement (Section 7.3.1), we consider pretrained models in OBSERVATORY, thereby not fine-tuning any model for downstream tasks. We illustrate in Section 7.6 that the characterization of pretrained models remain effective for anticipating behaviors of finetuned models on various downstream tasks.

7.5 Results

In this section, we present the experiment results and our analysis and describe the characteristics of models over the eight properties.

7.5.1 Row Order Insignificance

We calculate cosine similarity and MCV (as defined in Measure 1) measures over column, row, and table embeddings and plot their distributions in Figure 7.5. Overall, table embedding models exhibit comparably lower cosine similarity while both language and table embedding models may exhibit high MCV.

On the top row of Figure 7.5, column embeddings of five models BERT, RoBERTa, T5, TAPAS, and TaBERT show strong evidence of being robust to row order shuffling in terms of

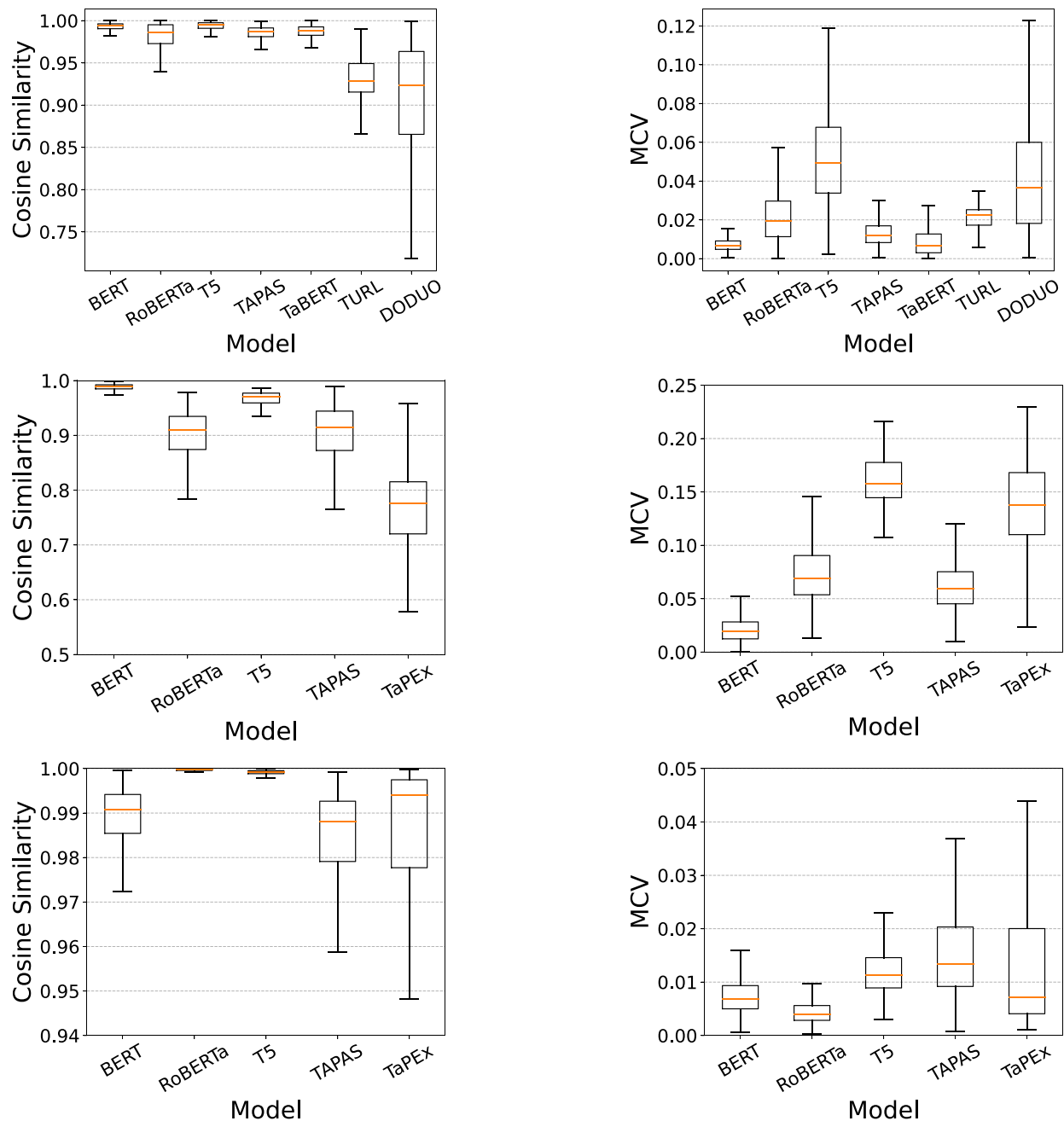


Figure 7.5: Cosine similarity and MCV distributions of column (top), row (middle), and table (bottom) embeddings from row shuffling. Across three levels of embeddings, table embedding models exhibit comparably lower cosine similarity while both language and table embedding models may exhibit high MCV.

cosine similarity. In particular, the first quartile (Q1) of these models is above 0.97 and the minimum (Q1 - $1.5 \times$ interquartile range) is above 0.95 except RoBERTa. TURL follows with Q1 above 0.92 and the minimum above 0.86. DODUO exhibits the largest spread with the

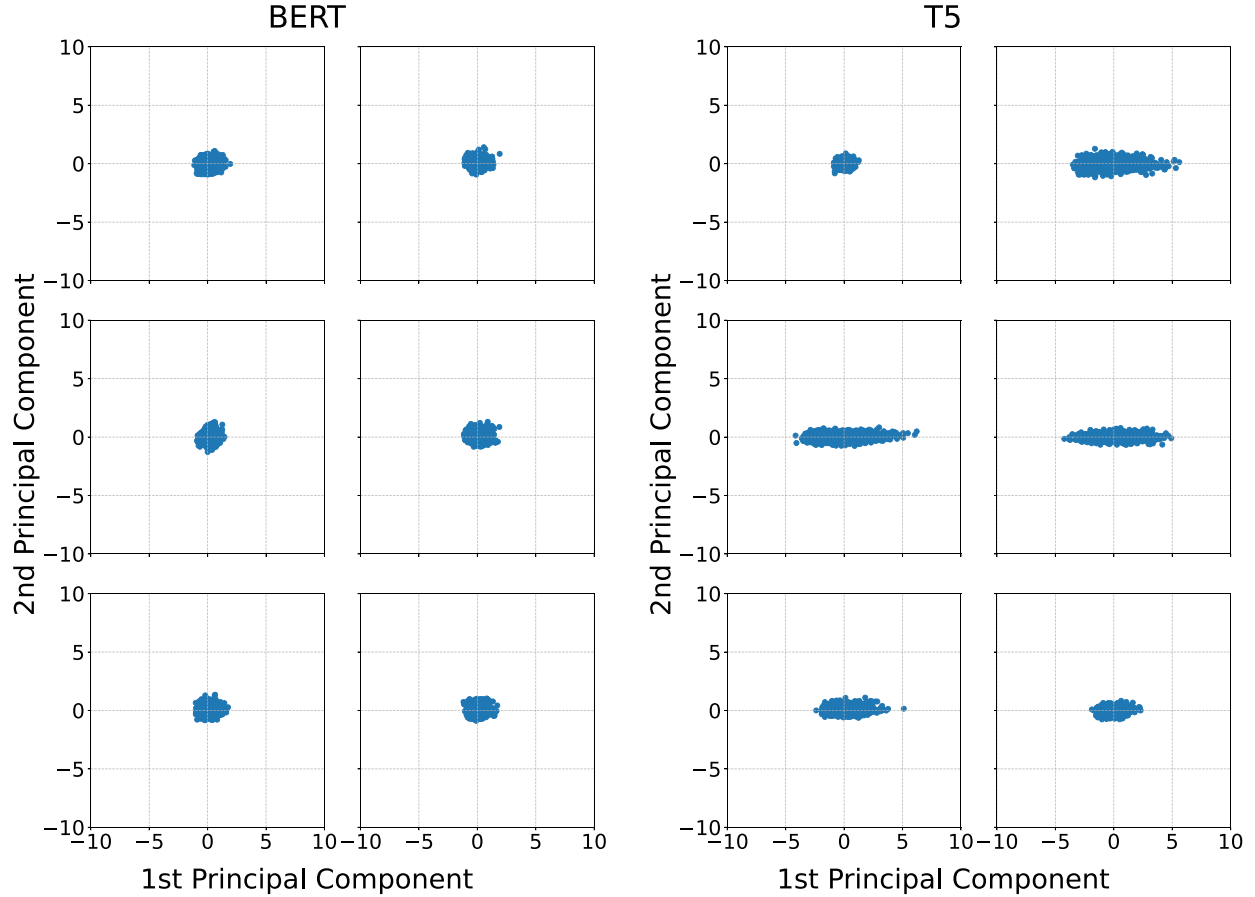


Figure 7.6: PCA visualization of high-dimensional column embeddings from a table of six columns, for BERT and T5. Each subplot draws $6!=720$ row-wise permutation variants of a column. While BERT embeddings are centered around the origin with some variation, the T5 embeddings are more stretched along the horizontal axis, resulting in the relatively high cosine similarity as well as high MCV value.

minimum below 0.75 while the median is over 0.91. This implies that DODUO is relatively sensitive towards row order, given that the content of these rows is not altered. We illustrate in Section 7.6 how DODUO’s sensitivity to row order shuffling translates to unstable predictions in a downstream task for which DODUO is proposed.

The MCV measure (the lower the better) indicates the variability of different populations (i.e., embedding distributions given by different models), especially when they have different means. It is notable that T5 has the largest third quartile (Q3) and second largest maximum ($Q3 + 1.5 \times \text{interquartile range}$) while T5 embeddings have high cosine similarities. We hypothesize that this is because T5 embeddings are more dispersed in a specific direction in high-dimensional space compared to models with low MCVs such as BERT. We verify this by visualizing the PCA projections of embeddings in two-dimensional space. For demon-

stration purposes, we use a table in which T5 embeddings of three columns yield high MCV scores (larger than 0.08, which is higher than Q3). Correspondingly, the projections of T5 embeddings of these three columns (top-right, middle left and middle right) are indeed more stretched along a specific direction than those of BERT as demonstrated in Figure 7.6.

With regard to row embeddings (the middle row of Figure 7.5), it is noticeable that BERT obtains high cosine similarity with the minimum above 0.95 and low MCV with Q3 below 0.03. On the other hand, row embeddings of RoBERTa, T5, TAPAS and TaPEX appear to vary more in MCV than their column embeddings. As seen in the bottom row of Figure 7.5, unlike column and row embeddings, table embeddings of assessed models manifest exceptionally high cosine similarity and low MCV. Precisely, the minimum of the cosine similarity of each model is above 0.94 and the range of MCV is also $5\times$ smaller than that of row embeddings.

These findings also underline the importance of combining the cosine similarity and MCV for measuring row order insignificance, as a single measure would give a limited perspective.

7.5.2 Column Order Insignificance

The results of column shuffling follow a similar trend as that of row shuffling. Nevertheless, column shuffling appears to cause more variations in all three levels of embeddings for both cosine similarity and MCV measures. In the interest of space, we only show column and row embeddings in Figure 7.7.

Considering, for example, the column embeddings in Figure 7.7, the median cosine similarity of RoBERTa embeddings drops by more than 5% and the same statistic of DODUO embeddings drops by more than 15%. The median MCV of both RoBERTa and T5 also increases by four times. To verify such large variations, we again visualize the PCA projections of T5 embeddings in Figure 7.8 for the same table as used in Figure 7.6. This figure confirms that the first principal component of T5 embeddings manifests larger spread, and illustrates the spread along the horizontal axis across all columns (instead of merely 3, as when rows are shuffled) indicating a higher sensitivity to column order than row order.

7.5.3 Join Relationship

Table 7.3 presents the Spearman coefficients between a value overlap measure and embedding cosine similarity over joinable pairs of columns from the NextiaJD-XS dataset. We find that, among the considered value overlap measures (containment, Jaccard, and multiset Jaccard), multiset Jaccard similarity is most positively correlated with embedding cosine similarity. For all models, the coefficient value between multiset Jaccard and embedding cosine similarity is above 0.5, which indicates a moderate positive correlation (TaBERT has a coefficient value

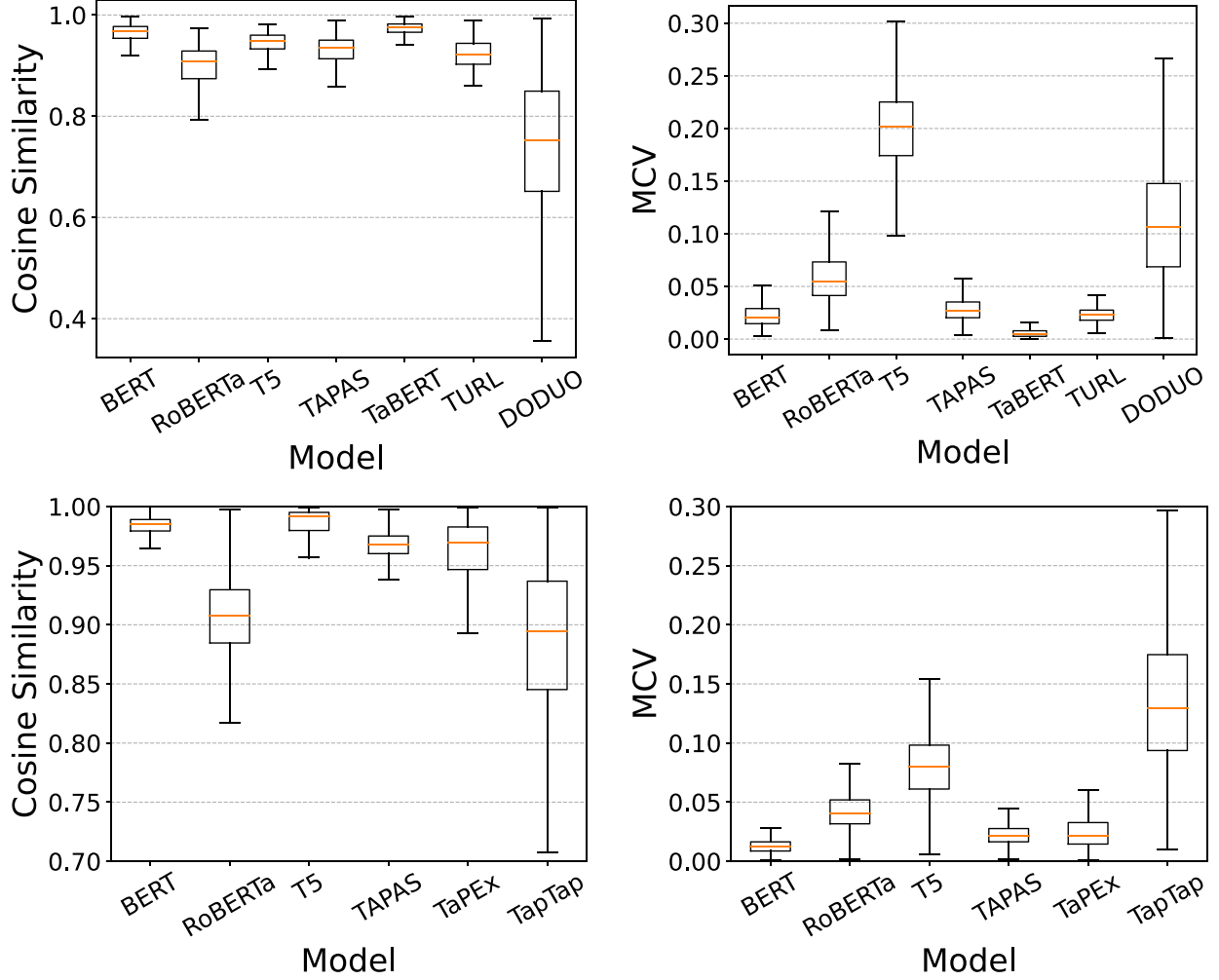


Figure 7.7: Cosine similarity and MCV distributions of column (top) and row (bottom) embeddings from column shuffling. Both column and row embeddings manifest similar patterns as in row shuffling.

of 0.72 which indicates a high positive correlation), and is significantly higher than that of the other two measures ($0.08 - 0.43 \uparrow$). This difference can be attributed to the fact that containment and Jaccard similarity do not take duplicate values into account while we use all values for embedding inference. In Figure 7.9, we also show scatter plots of embedding cosine similarity versus multiset Jaccard over pairs of joinable columns from NextiaJD-XS for each model, which demonstrates the moderate positive correlation between the two variables. Note that the maximum possible value of multiset Jaccard similarity is 0.5.

Both syntactic and semantic approaches have been employed for data discovery [124, 17, 43]. It is valuable to be aware of what syntactic measure is highly correlated with a semantic measure based on embeddings so that one can ensemble less correlated syntactic

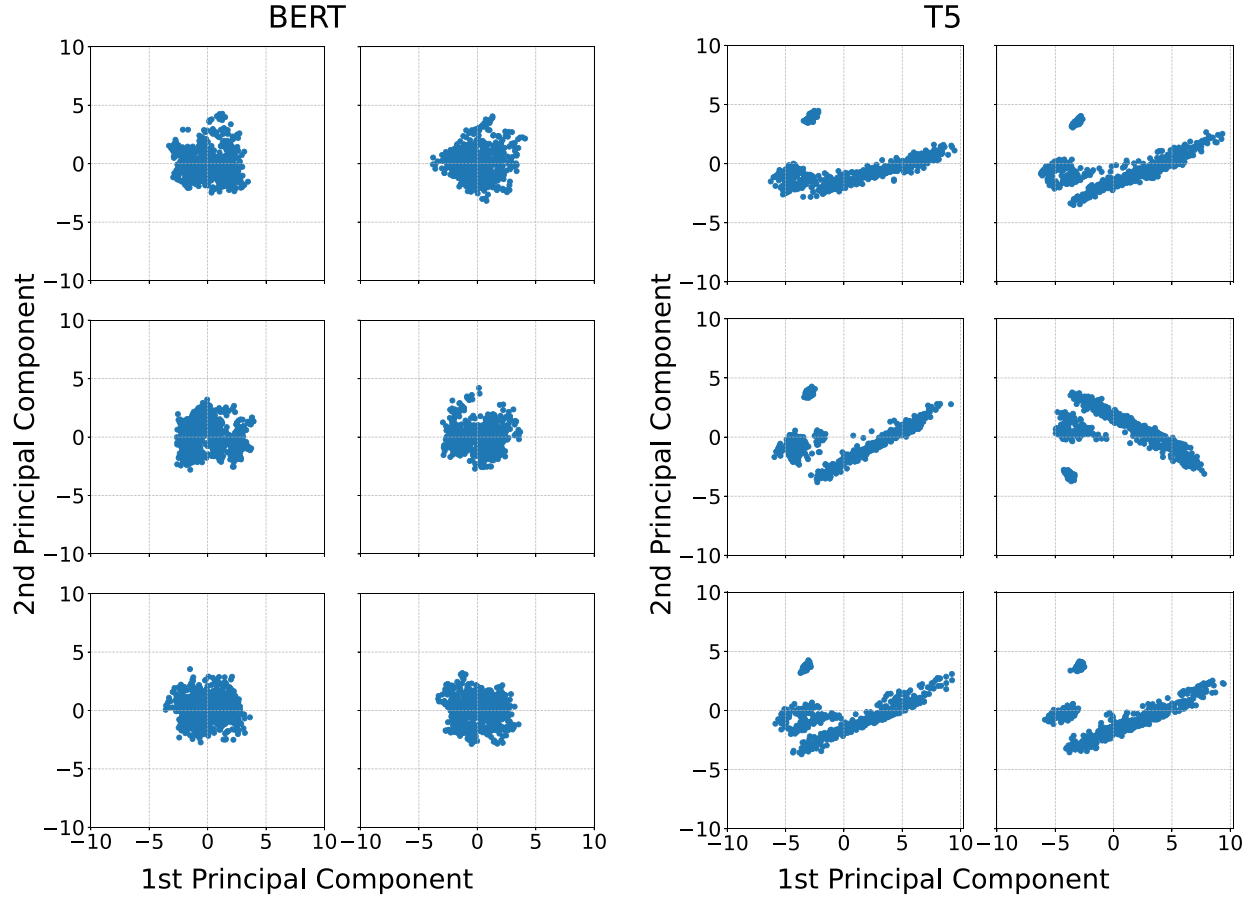


Figure 7.8: PCA visualization of high-dimensional column embeddings from the same table as used in Figure 7.6. Each subplot draws $6!=720$ variants of a column from column order shuffling. The embeddings exhibit similar patterns as in row order shuffling but show larger spread across all columns.

and embedding measures if they want to find more diverse candidates. For instance, consider the task of join discovery over NextiaJD-XS. Based on Table 7.3, it is recommended to use containment as the syntactic similarity measure when BERT embeddings are used to measure semantic similarities because these two measures show the least correlation. Similarly, it is recommended to use the Jaccard similarity when TAPAS embeddings are used.

7.5.4 Functional Dependencies

Table 7.4 displays the average variance of the L2 norm of translation embeddings over column pairs, comparing those with and without functional dependencies. Vanilla language models exhibit no significant reduction in variance for columns with functional dependencies, as expected, given their lack of consideration for table structure during pretraining. In contrast,

Table 7.3: Spearman coefficients between a value overlap measure and embedding cosine similarity on the NextiaJD-XS dataset. Multiset Jaccard is most positively correlated to embedding cosine similarity across all models. All coefficient numbers are statistically significant (p-value < 0.01).

	BERT	RoBERTa	T5	TAPAS	TaBERT	DODUO
Containment	0.241	0.412	0.649	0.438	0.506	0.438
Jaccard	0.288	0.339	0.563	0.368	0.553	0.441
Multiset Jaccard	0.670	0.512	0.647	0.655	0.721	0.696

Table 7.4: Average group-wise variances of embedding translations over columns with and without functional dependencies across five models. Only TAPAS yields $\overline{S^2}_{FD} < \overline{S^2}_{-FD}$ with $\overline{S^2}_{FD}$ close to 0, while language models and other table embedding models do not follow this pattern.

	BERT	RoBERTa	T5	TAPAS	DODUO
Columns w/ FD	0.87	0.39	1.80	0.88	83.34
Columns w/o FD	0.78	0.34	1.13	1.12	229.77

table embedding models, including DODUO and TAPAS, show variance patterns contrary to vanilla language models, though the average variance of DODUO is not close to 0. Despite TAPAS aligning with expected patterns, Figure 7.10 reveals that none of the models distinctly separate variance distributions for column pairs with and without functional dependencies. This lack of clear separation provides evidence that none of the models effectively capture the relationship of functional dependencies in their representations.

7.5.5 Sample Fidelity

Figure 7.11 depicts sample fidelity distributions of models across various sample ratios. As the ratio increases, sample embeddings tend to align more closely with those from full values in terms of cosine similarity, evident in the ascending quartile values of embedding cosine similarity in the box plots.

Vanilla language models consistently show high sample fidelity, reaching a median over 0.9 at a 0.25 sample ratio and exceeding 0.95 at a 0.75 ratio. Notably, T5 demonstrates strong robustness to sampling, with over 75% of tested pairs having cosine similarity surpassing 0.95 when half of the values are sampled. Table embedding models, excluding TaBERT, exhibit larger distribution spreads, particularly at a 0.25 sample ratio. TaBERT stands out as the

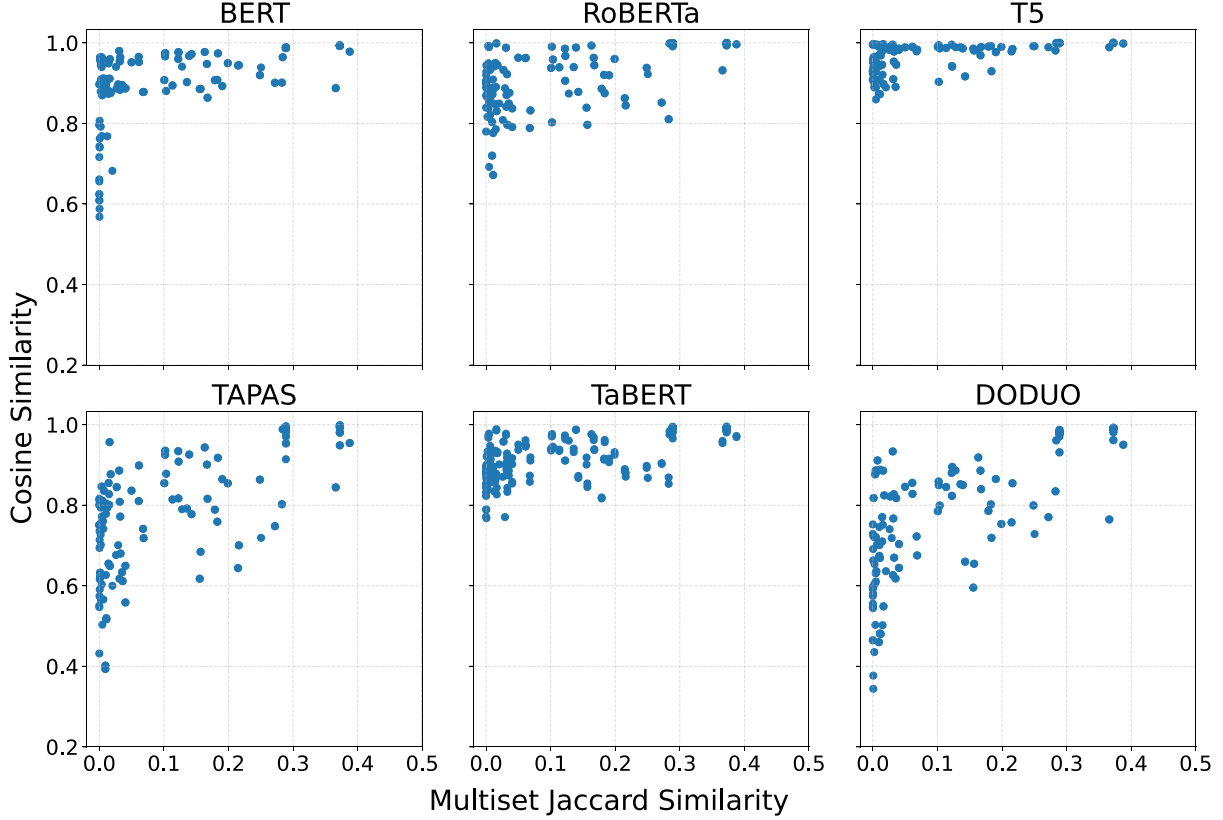


Figure 7.9: Scatter plots of embedding cosine similarity vs. multiset Jaccard similarity derived from pairs of joinable columns in the NextiaJD XS dataset, which illustrate a positive correlation between the two measures.

most sample-robust model, consistently maintaining cosine similarity over 0.95 across all sample ratios. This robustness stems from TaBERT’s internal practice of always considering the first three rows [176], increasing the likelihood of overlapping or identical inputs despite sampling. While TAPAS emerges as the next sample-robust model, achieving high fidelity comparable to vanilla language models at a 0.5 sample ratio, DODUO lags behind and proves more sensitive to sampling across all ratios, consistent with the results of row and column shuffling.

7.5.6 Entity Stability

We select query entities from five domains and compare their K -nearest neighbors between two embedding spaces: ten greatest men tennis players (Tennis Players), ten most popular movies (Movies), ten most essential nutrients for the body (Biochemistry), ten most valuable technology companies in the U.S., and ten largest countries in the world by area. We plot pairwise average entity stability using heatmaps in Figure 7.12. Due to space limits, we only

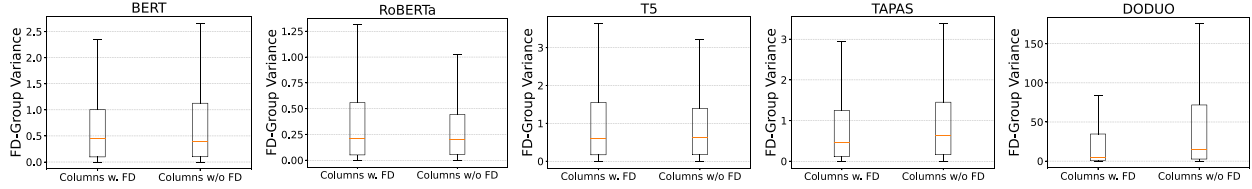


Figure 7.10: Distributions of the group-wise variances over embedding translations across column pairs with and without the relationship of functional dependencies. None of the models show clear separation between the two variance distributions.

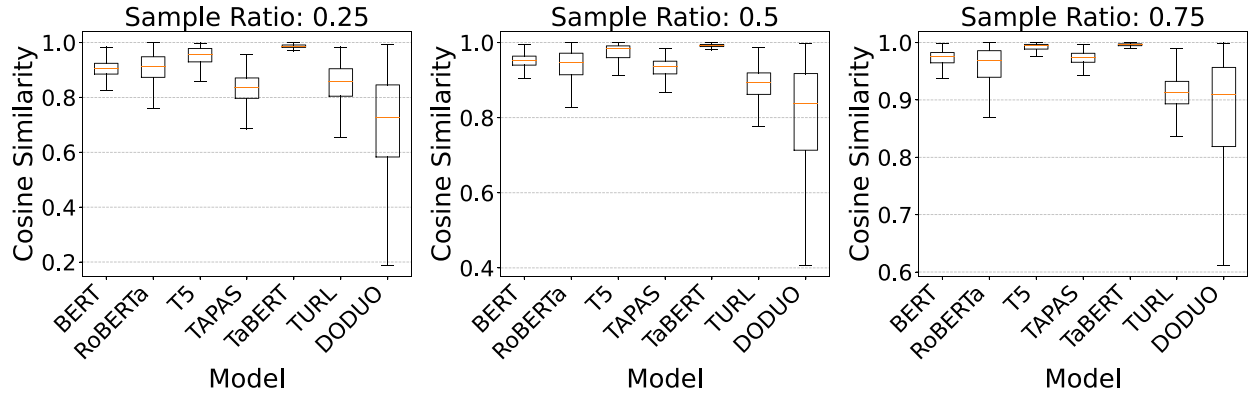


Figure 7.11: Distributions of sample fidelity of column embeddings under three sample ratios. Overall, vanilla LMs exhibit higher sample fidelity compared to table embedding models.

show heatmaps of Tennis Players, Movies, and Biochemistry with $K=10$. We observe that domain is a key factor in entity stability. In other words, for different domains, different pairs of models show high entity stability. For instance, BERT and TURL have the highest entity stability for movie entities while TAPAS and DODUO have the highest entity stability for biochemistry entities. This suggests for domain-specific tasks, if one finds model A is not feasible, they may want to try model B with relatively lower entity stability with respect to A.

7.5.7 Perturbation Robustness

Figure 7.13 shows distributions of embedding cosine similarities between pairs of an original column and a corresponding perturbed column. Even though both types of perturbations are at the schema level and data values remain unchanged, models exhibit different degrees of robustness, especially in terms of the spread and skewness of the distribution. Vanilla language models BERT and T5 are most robust to schema-level perturbations with first quartile above 0.97 while entire distributions are above 0.90. Despite being a language model,

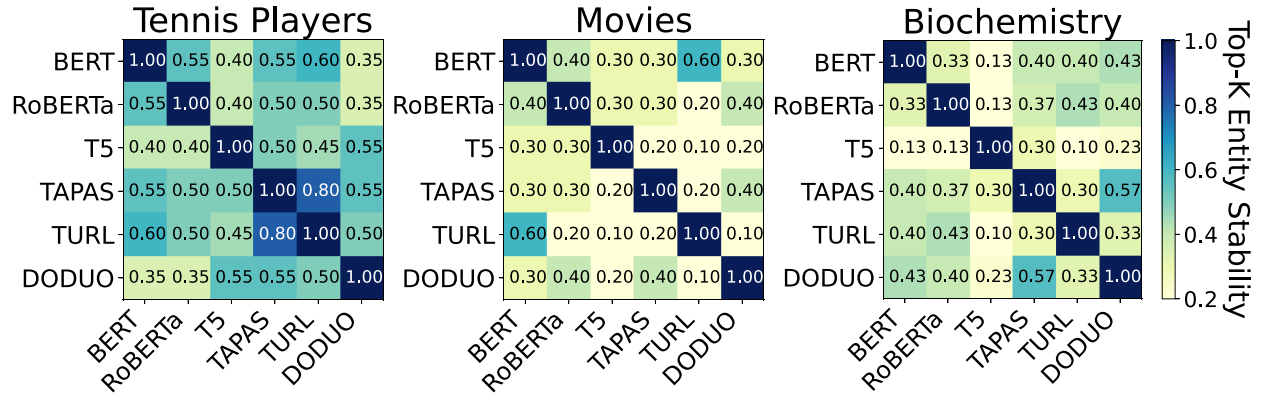


Figure 7.12: Pairwise top-10 entity stability with query entities from three distinct domains. Different pairs of models show high entity stability for different domains.

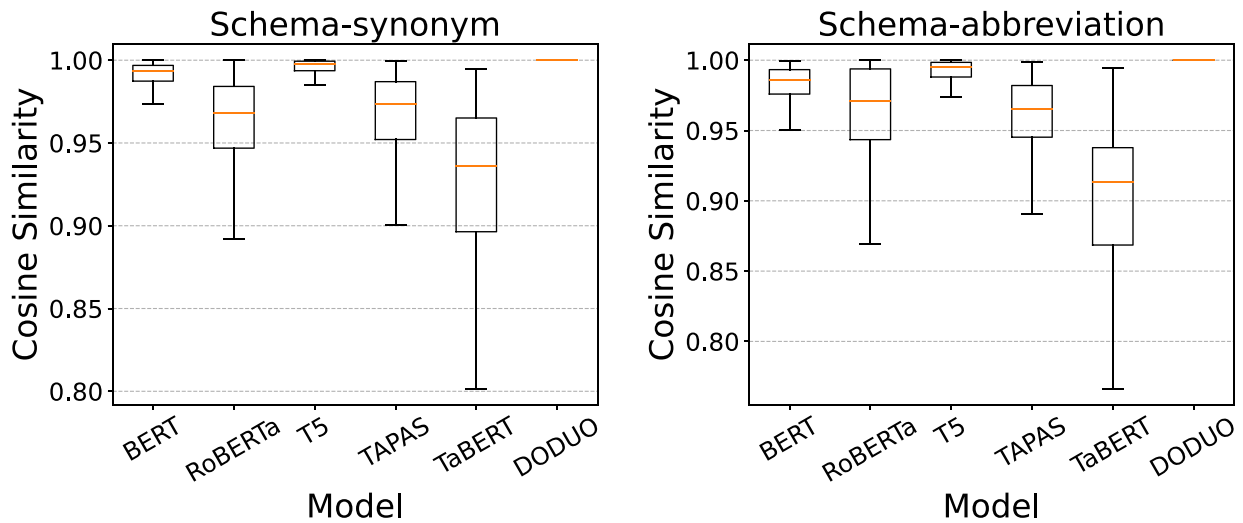


Figure 7.13: Distributions of embedding cosine similarities between original columns and perturbed columns. Perturbations only to schemas cause relatively small changes in cosine similarity (except for TaBERT).

RoBERTa surprisingly shows a larger spread with outliers down to 0.75 in synonym perturbations and to 0.65 in abbreviation perturbations. On the table model side, TaBERT is least robust to perturbations with the lowest median and first quartile among all models. In contrast, TAPAS is more robust with first percentile near 0.95 for both perturbations while it shows relatively large variance as well. DODUO does not show any variance because DODUO only takes in data values for representation inference and simply ignores changes to the schema. Overall, table embedding models in comparison are more sensitive to schema perturbations as they explicitly model the header component of tables and distinguish between headers and data values in representation learning.

7.5.8 Heterogeneous Context

Table 7.5: Summary statistics (min, median, and max) of cosine similarities between single column embeddings and contextual embeddings for non-textual and textual data types, on the first and second row, respectively. Incorporating context, especially the entire table, can change column embeddings significantly w.r.t cosine similarity (highlighted in bold).

Model	Subject Column	Neighboring Columns	Entire Table
BERT	0.72 / 0.89 / 0.99	0.62 / 0.86 / 0.99	0.57 / 0.78 / 0.96
	0.72 / 0.93 / 1.00	0.64 / 0.88 / 1.00	0.51 / 0.79 / 0.99
RoBERTa	0.76 / 0.83 / 0.89	0.71 / 0.82 / 0.93	0.75 / 0.84 / 0.92
	0.76 / 0.83 / 0.90	0.74 / 0.83 / 0.92	0.76 / 0.85 / 0.93
T5	0.77 / 0.85 / 0.93	0.75 / 0.88 / 0.97	0.74 / 0.83 / 0.92
	0.75 / 0.83 / 0.92	0.75 / 0.88 / 0.98	0.75 / 0.83 / 0.98
TAPAS	0.68 / 0.84 / 0.95	0.58 / 0.80 / 0.97	0.35 / 0.64 / 0.92
	0.52 / 0.83 / 0.98	0.50 / 0.80 / 0.98	0.31 / 0.67 / 0.92
TaBERT	0.94 / 0.97 / 1.00	0.93 / 0.97 / 1.00	0.89 / 0.95 / 0.99
	0.90 / 0.98 / 1.00	0.89 / 0.97 / 1.00	0.83 / 0.96 / 0.99
DODUO	0.25 / 0.62 / 0.99	0.14 / 0.59 / 0.99	0.06 / 0.45 / 0.87
	0.34 / 0.80 / 0.99	0.26 / 0.78 / 0.98	0.01 / 0.61 / 0.98

For both non- and textual data types, we infer from each model column embeddings using only the columns themselves, and adding 1) subject columns; 2) immediate neighbor columns; and 3) the entire tables as context respectively. We compute the cosine similarity between corresponding pairs of single column embeddings and contextual embeddings and show their three-number summary in Table 7.5. Unsurprisingly, adding different contexts to the inputs changes the embeddings to various degrees. For non-textual columns, among three context settings, models except DODUO preserve high cosine similarity when having subject columns as context (e.g., the median number of TaBERT is above 0.96 and that of BERT is close to 0.9) while they (except TaBERT) preserve relatively low cosine similarity when having the entire tables as context (e.g., the median number of TAPAS is below 0.65). We observe that TaBERT embeddings are insensitive to context (the median number is above 0.95 in all three settings) whereas DODUO embeddings are more sensitive to context (the median number is below 0.5 when having the entire tables as context and around 0.6 in the other two settings). We see a consistent trend for textual data. This can have implications that TaBERT may not be a good choice for context sensitive downstream tasks and a user may

want to try both single column embeddings and contextual embeddings when using DODUO.

7.6 Connection to Downstream Tasks

From the model characterization through their embedding representations as per the eight properties P1-8, we deduce below the model behaviors on downstream tasks. We illustrate three connections with experimental findings.

Column Type Prediction (P1/P2). In the experiments, DODUO is found sensitive to row/column shuffling and sampling, which are indicators of unstable predictions of DODUO over shuffled data in downstream tasks. To investigate this hypothesis, we randomly sample 1,000 tables from the WikiTables dataset used in the experiments and employ DODUO to predict semantic column types for all columns. For each table, we consider at most 1,000 distinct row-wise permutations for computational efficiency and keep track of how many predictions change per permutation relative to the original order. We find that, over this subset of tables with 5.8 columns on average, 34.0% of the permuted tables yield at least 1 changed column type prediction (averaged over all permutations). 12.8% of the tables have at least 2 changed type predictions while 5.4% of tables have at least 3 changed type predictions.

Join Discovery (P5). T5 exhibits high sample fidelity even when the sample ratio is low, leading us to anticipate T5 to be sample efficient in downstream tasks. We implement T5 in the task of join discovery following the approach and setup in [42]. Over the NextiaJD testbeds, sampled T5 embeddings obtain comparable precision and recall as those from full values while the indexing time and lookup time are significantly faster. For instance, on NextiaJD-XS with a sample size of 100 (which is about 5% of the average number of rows in NextiaJD-XS), there is less than $\pm 3\%$ variation in precision and recall between sampled T5 embeddings and full-value T5 embeddings. But the indexing time of using sampled values is more than 7x faster and the lookup time is more than $2\times$ faster.

Table Question Answering (P7). The task of table question answering (TableQA) refers to answering natural language questions based on information from given tables. In our experiments of the Perturbation Robustness property (Section 7.5.7), we found that TAPAS, among other models, was sensitive to semantics-preserving perturbations to the table schema. Based on this observation, we hypothesize that TAPAS may suffer performance degradation on perturbed tables in downstream tasks, such as TableQA, for which it is designed. As anticipated, the TableQA accuracy of TAPAS under synonym- and abbreviation perturbation drops by 6.2 and 8.3 points respectively on WikiTableQuestions [130], and 19.0 and 22.2

points respectively on WikiSQL [189] (see Table 2 and 7 in [188]).

We emphasize that, despite we focus on the characterization of pretrained models (e.g., pretrained version of TAPAS), our hypotheses predicated on such characterization propagate to finetuned models (in this case, TAPAS models fine-tuned for TableQA).

Additional Connections. Beyond the three empirically supported anticipations of model behaviors on downstream tasks, we also deduce informed expectations listed below as a result of the characterizations obtained with OBSERVATORY for the other properties. This list is not exhaustive as the connection between model characteristics and downstream tasks is not a one-to-one relationship.

- P3** Low Spearman’s coefficient between containment and embedding cosine similarity (e.g., BERT) → Join discovery: the containment-based method will complement the embedding-based method in finding join candidates.
- P4** Not preserving functional dependencies → Data imputation: imputed values may not maintain functional dependencies between attributes.
- P6** Relative to model A, model B has a lower entity stability than model C → Entity retrieval: model B will return fewer entities in common with model A than with model C.
- P8** Insensitive to context change (e.g., RoBERTa) → Join discovery: candidates found by single-column and contextual embeddings will largely overlap.

7.7 Discussion

Impact of Tables with Large Dimensionality. To assess the effect of table dimensions, we examine BERT and TAPAS regarding row- and column order insignificance on the NextiaJD-S dataset, averaging over 209k rows and 56 columns. No significant differences emerge on NextiaJD-S compared to tables from WikiTables. The partitioning of large tables into smaller ones, with aggregated embeddings, aligns with our practice for smaller tables.

Limitations. While OBSERVATORY encompasses crucial properties for various applications, implementing measures for all properties is unfeasible. For instance, assessing latent topics in tables, vital for retrieval tasks, lacks established measures and appropriate evaluation datasets. The challenge of evaluating models’ ability to capture signals across diverse data types, from numeric to textual, persists. Our model analysis is constrained to a representative selection, driven by the availability of code and pretrained model weights. However,

OBSERVATORY is extensible and open-sourced for analyzing additional models. We acknowledge the potential for future investigation into relationships between property metrics. Our analysis, like any empirical work, is subject to the inherent limitations of dataset specificity. Despite these considerations, we initiate the process of characterizing and understanding embedding representations across relational tables.

7.8 Conclusion

We introduce OBSERVATORY, a downstream-task agnostic analysis framework for table embeddings, gauging the alignment of pretrained embeddings with key relational data model and data distributions properties. Our assessment of nine language- and table embedding models reveals diverse capabilities of different models. Notably, some properties of the relational model and data distributions are not consistently reflected in table embeddings. OBSERVATORY provides a valuable tool for guiding model selection in various applications, aiding researchers in model evaluation, and informing future research on novel architectures for tabular data.

CHAPTER 8

Conclusion and Future Work

The FAIR Data Principles promise a world where data can be discovered and reused by both humans and machines. However, operationalizing these principles, particularly in large-scale, modern data stores, remains a major challenge. Achieving FAIR at scale requires a holistic, AI-driven automation of the entire data sense-making lifecycle—from finding the data, to understanding its relationships, to standardizing its description, and even to governing the tools used in the process. This dissertation has explored the critical intersection of AI and data management, demonstrating how modern AI techniques can significantly enhance the findability and reusability of tabular datasets by automating manual, “sense-making” tasks.

This work primarily focused on Findability and Reusability. The principles of Accessibility (e.g., automated access control and policy enforcement) and deeper Interoperability (e.g., querying data from any storage) remain significant open challenges. Our techniques were also designed for and evaluated on relational tabular data. Extending this automated sense-making to unstructured data (e.g., text, images) is a major next step. Below we discuss three promising directions for future research.

8.1 Future Work

8.1.1 Towards a Unified System for Hybrid Search

The current research landscape often treats different search paradigms—such as lexical, structural, and semantic search—as disparate problems solved by standalone systems. A promising avenue for future work is the design and implementation of a truly unified data system that seamlessly integrates different search needs. While this dissertation has proposed novel standalone algorithms, their integration into a single query processing engine presents a rich set of open research challenges. The goal is to enable expressive, hybrid queries that can combine, for example, exact keyword matching with semantic similarity search over vector embeddings within a single declarative interface.

Key research questions in this direction include:

- **Unified Query Language and API Design:** How can we extend existing query languages like SQL, or design new APIs, to allow users to express complex hybrid queries intuitively? This includes specifying clauses for keyword matching, vector similarity, and defining the logic for how results from different search needs should be combined and weighted.
- **Cost-Based Optimization for Hybrid Queries:** A unified system demands a sophisticated query optimizer. How can an optimizer model the cost and cardinality of semantic search operators? How would it decide the optimal execution plan? For instance, should it execute the keyword filter first to reduce the candidate set for a costly vector search, or should it use vector search to find a neighborhood of candidates and then apply lexical filters?
- **Principled Result Merging and Ranking:** Fusing results from different search paradigms is a non-trivial challenge. Each modality produces its own relevance score (e.g., BM25 for text, cosine similarity for vectors). Future work could develop principled fusion algorithms (e.g., based on Reciprocal Rank Fusion or learned ranking models) that combine these heterogeneous scores into a single, coherent relevance ranking for the end-user.

8.1.2 Multi-Modality Data Processing Systems

The work in this dissertation mostly focuses on tabular data but the advance of LLMs is shifting the landscape of data processing. Instead of the traditional ETL (Extract, Transform, Load) process where data is moved out of a database for complex processing (like feature extraction or inference) and then re-loaded, a new frontier is in-database multi-modal processing. Major vendors are already integrating AI functions into SQL (e.g., BigQuery ML, Snowflake’s Cortex), but this trend is in its infancy and presents deep research questions. This future direction explores moving beyond pre-processing data into uniform representations (like embeddings) and instead focuses on building systems that can apply powerful generative and analytical operators directly to raw, multi-modal data where it resides.

Key research challenges in this area include:

- **Designing First-Class AI Primitives:** What is the right set of AI/ML operators to expose within a declarative language like SQL? Beyond simple classification or summarization, this could include primitives for entity extraction, sentiment analysis,

question-answering, or even image-to-text generation. The challenge lies in designing these operators to be composable, deterministic (where possible), and seamlessly integrated with traditional relational operators like FILTER, JOIN, and GROUP BY.

- **Performance and Cost Modeling for AI Operators:** LLM inference is computationally expensive and often relies on external API calls, which introduces significant latency and monetary cost. A critical research area is developing a cost model for these AI operators within the database’s query optimizer. How can the system estimate the time and cost of running a SUMMARIZE function on a million text documents? This is essential for choosing an efficient query plan.
- **Novel Execution Strategies:** New execution strategies are needed to manage the high cost of these operators. This includes techniques like:
 - Batching: Grouping rows to make fewer, larger calls to a model endpoint.
 - Materialization and Caching: Caching the expensive results of AI functions to avoid re-computation.
 - Model-Aware Predicate Pushdown: Using cheaper models or heuristics to filter out rows that are unlikely to satisfy a condition before applying the expensive primary model. For example, filtering out reviews with a clearly positive sentiment before running a more nuanced model to detect specific complaints.
- **Handling Diverse and Unstructured Data Types:** A truly multi-modal system must efficiently manage and process various data types (text, images, audio) stored as BLOBs or in specialized formats. This involves developing mechanisms to route these different data types to the appropriate models and to integrate the structured output (e.g., a JSON object from an entity extraction model) back into the relational context for further querying and analysis.

8.1.3 Lakehouse Movement

Open data formats are the foundation of the modern “Lakehouse” architecture, built on two distinct layers [10, 112]. The first layer is open file formats, like Apache Parquet ¹, which define a standardized, columnar, and self-describing physical layout for data. This solves engine-level interoperability, allowing any query engine (e.g., Spark, DuckDB, Trino) to read the same file, decoupling compute from storage. The second layer is open table formats,

¹<https://parquet.apache.org>

such as Apache Iceberg ², Delta Lake [9], and Apache Hudi ³, which act as a “blueprint” or metadata layer on top. They organize collections of individual files into a single, logical table, providing database-style ACID transactions, versioning (time-travel), and schema evolution directly on the data lake.

This architecture’s core promise is ecosystem-level interoperability, eliminating data silos and enabling analytics and AI workloads on a single, reliable source of truth. However, this new paradigm introduces significant research and engineering challenges. These include managing the sheer scale of the metadata generated by table formats, efficiently handling real-time streaming ingestion without creating a “small file problem,” and designing intelligent compaction services to maintain performance [9, 84]. Furthermore, new query optimization and processing engines are required that are aware of evolving data and workloads, capable of processing raw data or complex data of nested types that lack statistics, and capable of handling the complex concurrency models needed for write-heavy workloads [84, 173, 111, 159].

8.2 Final Words

The journey from passive data repositories to active, intelligent, and autonomous data ecosystems is one of the most critical challenges in data management. The contributions in this dissertation provide the automated building blocks for this transition. By systematically replacing manual, error-prone human effort with reliable, scalable AI, this work lays the groundwork for a future where data can finally deliver on its full promise—not just to be stored, but to be truly found, understood, and reused.

²<https://iceberg.apache.org>

³<https://hudi.apache.org>

BIBLIOGRAPHY

- [1] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [2] Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *CoRR*, abs/1608.04207, 2016.
- [3] Stephanie Aerts, Gentiane Haesbroeck, and Christel Ruwet. Multivariate coefficients of variation: Comparison and influence functions. *J. Multivar. Anal.*, 142:183–198, 2015.
- [4] Parag Agrawal, Anish Das Sarma, Jeffrey D. Ullman, and Jennifer Widom. Foundations of uncertain-data integration. *Proc. VLDB Endow.*, 3(1):1080–1090, 2010.
- [5] Adelin Albert and Lixin Zhang. A novel definition of the multivariate coefficient of variation. *Biometrical Journal*, 52(5):667–675, 2010.
- [6] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023.
- [7] Ankur Ankan and Abinash Panda. pgmpy: Probabilistic graphical models using python. In *Proceedings of the 14th Python in Science Conference (SCIPY 2015)*. Citeseer, 2015.
- [8] Maria Antoniak and David Mimno. Evaluating the stability of embedding-based word similarities. *Trans. Assoc. Comput. Linguistics*, 6:107–119, 2018.
- [9] Michael Armbrust, Tathagata Das, Sameer Paranjpye, Reynold Xin, Shixiong Zhu, Ali Ghodsi, Burak Yavuz, Mukul Murthy, Joseph Torres, Liwen Sun, Peter Boncz, Mostafa

- Mokhtar, Herman Van Hovell, Adrian Ionescu, Alicja Luszczak, Michal Switakowski, Takuya Ueshin, Xiao Li, Michal Szafranski, Pieter Senster, and Matei Zaharia. Delta lake: High-performance ACID table storage over cloud object stores. *Proc. VLDB Endow.*, 13(12):3411–3424, 2020.
- [10] Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia. Lakehouse: A new generation of open platforms that unify data warehousing and advanced analytics. In *11th Conference on Innovative Data Systems Research, CIDR 2021, Virtual Event, January 11-15, 2021, Online Proceedings*. www.cidrdb.org, 2021.
 - [11] Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. Transformers for tabular data representation: A survey of models and applications. *Transactions of the Association for Computational Linguistics*, 2023.
 - [12] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Grégoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning. *CoRR*, abs/2304.12210, 2023.
 - [13] Mohit Bansal, David Burkett, Gerard de Melo, and Dan Klein. Structured learning for taxonomy induction with belief propagation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1041–1051. The Association for Computer Linguistics, 2014.
 - [14] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.
 - [15] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. Tabel: Entity linking in web tables. In *The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part I*, volume 9366 of *Lecture Notes in Computer Science*, pages 425–441. Springer, 2015.
 - [16] Sagar Bharadwaj, Praveen Gupta, Ranjita Bhagwan, and Saikat Guha. Discovering related data at scale. *Proc. VLDB Endow.*, 14(8):1392–1400, 2021.
 - [17] Alex Bogatu, Alvaro AA Fernandes, Norman W Paton, and Nikolaos Konstantinou. Dataset discovery in data lakes. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 709–720. IEEE, 2020.
 - [18] Alex Bogatu, Norman W. Paton, Mark Douthwaite, and André Freitas. Voyager: Data discovery and integration for onboarding in data science. In *EDBT 2022*, pages 2:537–2:548. OpenProceedings.org, 2022.
 - [19] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomáš Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017.

- [20] Rajesh Bordawekar and Oded Shmueli. Using word embedding to enable semantic queries in relational databases. In *DEEM@SIGMOD*, pages 5:1–5:4. ACM, 2017.
- [21] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems.*, pages 2787–2795, 2013.
- [22] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.
- [23] Dan Brickley, Matthew Burgess, and Natasha F. Noy. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 1365–1375. ACM, 2019.
- [24] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.
- [25] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. Data integration for the relational web. *Proc. VLDB Endow.*, 2(1):1090–1101, 2009.
- [26] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. *Proc. VLDB Endow.*, 1(1):538–549, 2008.
- [27] Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Found. Comput. Math.*, 9(6):717–772, 2009.
- [28] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1335–1349, 2020.
- [29] Sonia Castelo, Rémi Rampin, Aécio S. R. Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. Auctus: A dataset search engine for data discovery and augmentation. *Proc. VLDB Endow.*, 14(12):2791–2794, 2021.
- [30] Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang

- Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. Dr.spider: A diagnostic evaluation benchmark towards text-to-sql robustness. *CoRR*, abs/2301.08881, 2023.
- [31] Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. Dr.Spider Data Release. <https://github.com/awslabs/diagnostic-robustness-text-to-sql>, 2023. [Online; accessed October 12, 2023].
- [32] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. Dataset search: a survey. *The VLDB Journal*, 29(1):251–272, 2020.
- [33] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC 2002*, pages 380–388. ACM, 2002.
- [34] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- [35] Zhimin Chen, Vivek R. Narasayya, and Surajit Chaudhuri. Fast foreign-key detection in microsoft SQL server powerpivot for excel. *Proc. VLDB Endow.*, 7(13):1417–1428, 2014.
- [36] Zhimin Chen, Yue Wang, Vivek R. Narasayya, and Surajit Chaudhuri. Customizable and scalable fuzzy join for big data. *Proc. VLDB Endow.*, 12(12):2106–2117, 2019.
- [37] Alexander L. Chistov and Dima Grigoriev. Complexity of quantifier elimination in the theory of algebraically closed fields. In *Mathematical Foundations of Computer Science 1984, Praha, Czechoslovakia, September 3-7, 1984, Proceedings*, volume 176 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 1984.
- [38] CKAN. The world’s leading open source data management system, 2005. Accessed: 2025-2-2.
- [39] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, 1970.
- [40] E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397–434, 1979.
- [41] Web Data Commons. Extracting structured data from the common crawl, 2012. Accessed: 2024-11-29.
- [42] Tianji Cong, James Gale, Jason Frantz, H. V. Jagadish, and Çagatay Demiralp. Warp-gate: A semantic join discovery system for cloud data warehouses. In *13th Conference on Innovative Data Systems Research, CIDR 2023, Amsterdam, The Netherlands, January 8-11, 2023*. www.cidrdb.org, 2023.

- [43] Tianji Cong, Fatemeh Nargesian, and H. V. Jagadish. Pylon: Semantic table union search in data lakes. *arXiv preprint arXiv:2301.04901*, 2022.
- [44] Tianji Cong, Fatemeh Nargesian, Junjie Xing, and H. V. Jagadish. Openforge: Probabilistic metadata integration. *Proc. VLDB Endow.*, 18(9):2914–2927, 2024.
- [45] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.
- [46] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. TURL Data Release. <https://github.com/sunlab-osu/TURL#data>, 2020. [Online; accessed October 12, 2023].
- [47] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. TURL: table understanding through representation learning. *Proc. VLDB Endow.*, pages 307–319, 2020.
- [48] Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, Kaisen Jin, Chi Zhang, Yuqing Jiang, Yuanfang Zhang, Yuping Wang, Ye Yuan, Guoren Wang, and Nan Tang. Lakebench: A benchmark for discovering joinable and unionable tables in data lakes. *Proc. VLDB Endow.*, 17(8):1925–1938, 2024.
- [49] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019*, pages 4171–4186. Association for Computational Linguistics, 2019.
- [50] Hong Hai Do and Erhard Rahm. COMA - A system for flexible combination of schema matching approaches. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*, pages 610–621. Morgan Kaufmann, 2002.
- [51] Till Döhmen, Radu Geacu, Madelon Hulsebos, and Sebastian Schelter. Schemapile: A large collection of relational database schemas. *Proc. ACM Manag. Data*, 2(3):172, 2024.
- [52] Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. Table pre-training: A survey on model architectures, pre-training objectives, and downstream tasks. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 5426–5435. ijcai.org, 2022.
- [53] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach. In *ICDE 2021*, pages 456–467. IEEE, 2021.
- [54] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. Deepjoin: Joinable table discovery with pre-trained language models. *Proc. VLDB Endow.*, 16(10):2458–2470, 2023.

- [55] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.
- [56] Gaspard Ducamp, Christophe Gonzales, and Pierre-Henri Wuillemin. agrum/pyagrum : a toolbox to build models and algorithms for probabilistic graphical models in python. In *International Conference on Probabilistic Graphical Models, PGM 2020, 23-25 September 2020, Aalborg, Hotel Comwell Rebild Bakker, Skørping, Denmark*, volume 138 of *Proceedings of Machine Learning Research*, pages 609–612. PMLR, 2020.
- [57] Julian Eberius, Maik Thiele, Katrin Braunschweig, and Wolfgang Lehner. Top-k entity augmentation using consistent set covering. In *SSDBM 2015*, pages 8:1–8:12. ACM, 2015.
- [58] Stefano Ermon. Cs 228 - probabilistic graphical models. <https://ermongroup.github.io/cs228-notes/>, 2023. Accessed: 2024-3-16.
- [59] Raul Castro Fernandez. Data discovery is a socio-technical problem: the path from document identification and retrieval to data ecology. *IEEE Data Engineering Bulletin*, 2025.
- [60] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. Aurum: A data discovery system. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1001–1012. IEEE Computer Society, 2018.
- [61] Raul Castro Fernandez, Ziawasch Abedjan, Samuel Madden, and Michael Stonebraker. Towards large-scale data discovery: position paper. In *Proceedings of the Third International Workshop on Exploratory Search in Databases and the Web, San Francisco, CA, USA, July 1, 2016*, pages 3–5. ACM, 2016.

- [62] Raul Castro Fernandez, Essam Mansour, Abdulhakim A Qahtan, Ahmed Elmagarmid, Ihab Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. Seeping semantics: Linking datasets using word embeddings for data discovery. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 989–1000. IEEE, 2018.
- [63] Javier Flores, Sergi Nadal, and Oscar Romero. Towards scalable data discovery. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 433–438. OpenProceedings.org, 2021.
- [64] Juliana Freire, Grace Fan, Benjamin Feuer, Christos Koutras, Yurong Liu, Eduardo Peña, Aécio S. R. Santos, Cláudio T. Silva, and Eden Wu. Large language models for data discovery and integration: Challenges and opportunities. *IEEE Data Eng. Bull.*, 49(1):3–31, 2025.
- [65] James Gale, Max Seiden, Deepanshu Utkarsh, Jason Frantz, Rob Woollen, and Çagatay Demiralp. Sigma workbook: A spreadsheet for cloud data warehouses. *Proc. VLDB Endow.*, pages 3670–3673, 2022.
- [66] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741, 1984.
- [67] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB 1999*, pages 518–529, 1999.
- [68] Amir Globerson and Tommi S. Jaakkola. Fixing max-product: Convergent message passing algorithms for MAP lp-relaxations. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 553–560. Curran Associates, Inc., 2007.
- [69] Kathleen M Gregory, Helena Cousijn, Paul Groth, Andrea Scharnhorst, and Sally Wyatt. Understanding data search as a socio-technical practice. *Journal of Information Science*, 46(4):459–475, 2020.
- [70] Schema.org Community Group and Steering Group. Schema.org. <https://schema.org>, 2011. Accessed: 2023-10-23.
- [71] Ramanathan V Guha, Dan Brickley, and Steve Macbeth. Schema. org: evolution of structured data on the web. *Communications of the ACM*, 59(2):44–51, 2016.
- [72] Michael Günther, Maik Thiele, Julius Gonsior, and Wolfgang Lehner. Pre-trained web table embeddings for table discovery. In *aiDM@SIGMOD 2021*, pages 24–31. ACM, 2021.

- [73] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR, 2017.
- [74] Alon Y. Halevy, Flip Korn, Natalya Fridman Noy, Christopher Olston, Neoklis Polyzotis, Sudip Roy, and Steven Euijong Whang. Goods: Organizing google’s datasets. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 795–806. ACM, 2016.
- [75] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*, pages 539–545, 1992.
- [76] Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4320–4333. Association for Computational Linguistics, 2020.
- [77] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuezhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.
- [78] Madelon Hulsebos, Çağatay Demiralp, and Paul Groth. Gittables: A large-scale corpus of relational tables. *Proc. ACM Manag. Data*, 1(1):30:1–30:17, 2023.
- [79] Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zraggen, Arvind Satyanarayan, Tim Kraska, Çağatay Demiralp, and César A. Hidalgo. Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 1500–1508. ACM, 2019.
- [80] Madelon Hulsebos, Wenjing Lin, Shreya Shankar, and Aditya G. Parameswaran. It took longer than I was expecting: Why is dataset search still so hard? In *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics, HILDA 24, Santiago, Chile, 14 June 2024*, pages 1–4. ACM, 2024.
- [81] ICPSR. Icpsr subject thesaurus, 1962. Accessed: 2023-10-23.
- [82] ICPSR. Inter-university consortium for political and social research, 1962. Accessed: 2023-10-23.
- [83] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

- [84] Paras Jain, Peter Kraft, Conor Power, Tathagata Das, Ion Stoica, and Matei Zaharia. Analyzing and comparing lakehouse storage systems. In *13th Conference on Innovative Data Systems Research, CIDR 2023, Amsterdam, The Netherlands, January 8-11, 2023*. www.cidrdb.org, 2023.
- [85] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*, pages 665–674. ACM, 2013.
- [86] Xingyu Ji, Parker Glenn, Aditya G. Parameswaran, and Madelon Hulsebos. TARGET: benchmarking table retrieval for generative tasks. *CoRR*, abs/2505.11545, 2025.
- [87] Lan Jiang and Felix Naumann. Holistic primary key and foreign key detection. *J. Intell. Inf. Syst.*, 54(3):439–461, 2020.
- [88] Alekh Jindal, Shi Qiao, Hiren Patel, Abhishek Roy, Jyoti Leeka, and Brandon Haynes. Production experiences from computation reuse at microsoft. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT 2021, Nicosia, Cyprus, March 23 - 26, 2021*, pages 623–634. OpenProceedings.org, 2021.
- [89] Youri Kaminsky, Eduardo H. M. Pena, and Felix Naumann. Discovering similarity inclusion dependencies. *Proc. ACM Manag. Data*, 1(1):75:1–75:24, 2023.
- [90] Moe Kayali, Anton Lykov, Ilias Fountalis, Nikolaos Vasiloglou, Dan Olteanu, and Dan Suciu. CHORUS: foundation models for unified data discovery and exploration. *CoRR*, abs/2306.09610, 2023.
- [91] Aamod Khatiwada, Grace Fan, Roei Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. SANTOS: relationship-based semantic table union search. *Proc. ACM Manag. Data*, 1(1):9:1–9:25, 2023.
- [92] Aneta Koleva, Martin Ringsquandl, and Volker Tresp. Analysis of the attention in tabular language models. In *NeurIPS 2022 First Table Representation Workshop*, 2022.
- [93] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.
- [94] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [95] Ketil Korini and Christian Bizer. Column type annotation using chatgpt. *CoRR*, abs/2306.00745, 2023.
- [96] Ketil Korini, Ralph Peeters, and Christian Bize. Sotab v2 - table annotation benchmark, 2023. Accessed: 2025-2-27.

- [97] Keti Korini, Ralph Peeters, and Christian Bizer. SOTAB: the WDC schema.org table annotation benchmark. In *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab 2021, co-located with the 21st International Semantic Web Conference, ISWC 2022, Virtual conference, October 23-27, 2022*, CEUR Workshop Proceedings, 2022.
- [98] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. Valentine: Evaluating matching techniques for dataset discovery. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, pages 468–479. IEEE, 2021.
- [99] Sebastian Kruse and Felix Naumann. Efficient discovery of approximate dependencies. *Proc. VLDB Endow.*, 11(7):759–772, 2018.
- [100] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025.
- [101] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.
- [102] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. A large public corpus of web tables containing time and context metadata. In *WWW 2016*, pages 75–76. ACM, 2016.
- [103] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60, 2020.
- [104] Yiming Lin, Yeye He, and Surajit Chaudhuri. Auto-bi: Automatically build bi-models leveraging local join prediction and global schema graph. *Proc. VLDB Endow.*, 16(10):2578–2590, 2023.
- [105] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. SMAC3: A versatile bayesian optimization package for hyperparameter optimization. *J. Mach. Learn. Res.*, 23:54:1–54:9, 2022.
- [106] Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. TAPEX: table pre-training via learning a neural SQL executor. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

- [107] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.
- [108] Matéo Mahaut, Laura Aina, Paula Czarnowska, Momchil Hardalov, Thomas Müller, and Lluís Màrquez. Factual confidence of llms: on reliability and robustness of current estimators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 4554–4570. Association for Computational Linguistics, 2024.
- [109] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.
- [110] Marc Maynou, Sergi Nadal, Raquel Panadero, Javier Flores, Oscar Romero, and Anna Queralt. FREYJA: efficient join discovery in data lakes. *CoRR*, abs/2412.06637, 2024.
- [111] Dipankar Mazumdar. Concurrency control in open data lakehouse, 2025. Accessed: 2025-11-05.
- [112] Dipankar Mazumdar, Jason Hughes, and JB Onofre. The data lakehouse: Data warehousing and more. *CoRR*, abs/2310.08697, 2023.
- [113] Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295, 2024.
- [114] Microsoft. Sempy python library, 2024. Accessed: 2025-4-24.
- [115] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.
- [116] Renée J. Miller, Fatemeh Nargesian, Erkang Zhu, Christina Christodoulakis, Ken Q. Pu, and Periklis Andritsos. Making open data transparent: Data discovery on open data. *IEEE Data Eng. Bull.*, 41(2):59–70, 2018.

- [117] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 19–34. ACM, 2018.
- [118] Rafael Müller, Simon Kornblith, and Geoffrey E. Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4696–4705, 2019.
- [119] Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI '99: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, July 30 - August 1, 1999*, pages 467–475. Morgan Kaufmann, 1999.
- [120] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. Can foundation models wrangle your data? *Proc. VLDB Endow.*, 16(4):738–746, 2022.
- [121] Fatemeh Nargesian, Ken Q. Pu, Bahar Ghadiri Bashardoost, Erkang Zhu, and Renée J. Miller. Data lake organization. *IEEE Trans. Knowl. Data Eng.*, 35(1):237–250, 2023.
- [122] Fatemeh Nargesian, Ken Q. Pu, Erkang Zhu, Bahar Ghadiri Bashardoost, and Renée J. Miller. Organizing data lakes for navigation. In *SIGMOD 2020*, pages 1939–1950. ACM, 2020.
- [123] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. Data lake management: Challenges and opportunities. *Proc. VLDB Endow.*, 12(12):1986–1989, 2019.
- [124] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. Table union search on open data. *Proc. VLDB Endow.*, 11(7):813–825, 2018.
- [125] National Institute of Allergy and Infectious Diseases. Fair data principles at nih and niaid, 2025. Accessed: 2025-09-27.
- [126] OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.
- [127] OpenAI. Gpt-4o, 2024. Accessed: 2025-4-24.
- [128] Paul Ouellette, Aidan Sciortino, Fatemeh Nargesian, Bahar Ghadiri Bashardoost, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. RONIN: data lake exploration. *Proc. VLDB Endow.*, 14(12):2863–2866, 2021.
- [129] Thorsten Papenbrock and Felix Naumann. A hybrid approach to functional dependency discovery. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 821–833. ACM, 2016.

- [130] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics, 2015.
- [131] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035, 2019.
- [132] Yudi Pawitan and Chris Holmes. Confidence in the reasoning of large language models. *CoRR*, abs/2412.15296, 2024.
- [133] Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989.
- [134] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12:2825–2830, 2011.
- [135] Rakesh Pimplikar and Sunita Sarawagi. Answering table queries on the web using column keywords. *Proc. VLDB Endow.*, 5(10):908–919, 2012.
- [136] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. *OpenAI*, 2018.
- [137] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.
- [138] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
- [139] Vinay Rao. Extracting dark data. <https://developer.ibm.com/articles/ba-data-becomes-knowledge-3/>, 2018. Accessed: 2025-9-21.
- [140] David Reinsel, John Gantz, and John Rydning. Data age 2025: The evolution of data to life-critical. *Framingham: IDC Analyze the Future*, 2017.

- [141] Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 4902–4912, 2020.
- [142] Stephen Roller, Douwe Kiela, and Maximilian Nickel. Hearst patterns revisited: Automatic hypernym detection from large text corpora. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 358–363. Association for Computational Linguistics, 2018.
- [143] Alexandra Rostin, Oliver Albrecht, Jana Bauckmann, Felix Naumann, and Ulf Leser. A machine learning approach to foreign key discovery. In *12th International Workshop on the Web and Databases, WebDB 2009, Providence, Rhode Island, USA, June 28, 2009*, 2009.
- [144] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Y. Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. Finding related tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 817–828. ACM, 2012.
- [145] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [146] Prithviraj Sen and Amol Deshpande. Representing and querying correlated tuples in probabilistic databases. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, The Marmara Hotel, Istanbul, Turkey, April 15-20, 2007*, pages 596–605. IEEE Computer Society, 2007.
- [147] Glenn Shafer and Prakash P. Shenoy. Probability propagation. *Ann. Math. Artif. Intell.*, 2:327–351, 1990.
- [148] Jiaming Shen and Jiawei Han. *Automated Taxonomy Discovery and Exploration*. Synthesis Lectures on Data Mining and Knowledge Discovery. Springer, 2022.
- [149] Roei Shraga, Avigdor Gal, and Haggai Roitman. Adnev: Cross-domain schema matching using deep similarity matrix adjustment and evaluation. *Proc. VLDB Endow.*, 13(9):1401–1415, 2020.
- [150] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST 2012, Lake Tahoe, Nevada, USA, May 3-7, 2010*, pages 1–10. IEEE Computer Society, 2010.
- [151] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *ACL*, 2006.

- [152] Socrata. The socrata open data api, 2007. Accessed: 2025-2-2.
- [153] Splunk. The state of dark data, 2017.
- [154] Kavitha Srinivas, Julian Dolby, Ibrahim Abdelaziz, Oktie Hassanzadeh, Harsha Kokel, Aamod Khatiwada, Tejaswini Pedapati, Subhajit Chaudhury, and Horst Samulowitz. Lakebench: Benchmarks for data discovery over data lakes. *CoRR*, abs/2307.04217, 2023.
- [155] IDC Digital Universe study. Big data, bigger digital shadows, and biggest growth in the far east, 2012.
- [156] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 697–706. ACM, 2007.
- [157] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çagatay Demiralp, Chen Chen, and Wang-Chiew Tan. Annotating columns with pre-trained language models. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, pages 1493–1503, 2022.
- [158] Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Evaluating and enhancing structural understanding capabilities of large language models on tables via input designs. *CoRR*, abs/2305.13062, 2023.
- [159] Zhou Sun. No, iceberg won’t fix your data stack., 2025. Accessed: 2025-11-05.
- [160] Sahaana Suri, Ihab F. Ilyas, Christopher Ré, and Theodoros Rekatsinas. Ember: No-code context enrichment via similarity-based keyless joins. *Proc. VLDB Endow.*, 15(3):699–712, 2021.
- [161] Nan Tang, Ju Fan, Fangyi Li, Jianhong Tu, Xiaoyong Du, Guoliang Li, Sam Madden, and Mourad Ouzzani. Rpt: relational pre-trained transformer is almost all you need towards democratizing data preparation. *Proceedings of the VLDB Endowment*, 14(8):1254–1261, 2021.
- [162] Technology Transformation Services The U.S. General Services Administration. Data.gov. <https://opendata.cityofnewyork.us/overview/>, 2009. Accessed: 2023-10-23.
- [163] Andrea K. Thomer, Dharma Akmon, Jeremy York, Allison R. B. Tyler, Faye Polasek, Sara Lafia, Libby Hemphill, and Elizabeth Yakel. The craft and coordination of data curation: Complicating workflow views of data science. *Proc. ACM Hum. Comput. Interact.*, 6(CSCW2):1–29, 2022.
- [164] Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Guoliang Li, Xiaoyong Du, Xiaofeng Jia, and Song Gao. Unicorn: A unified multi-tasking model for supporting matching tasks in data integration. *Proc. ACM Manag. Data*, 1(1):84:1–84:26, 2023.

- [165] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.
- [166] Daisy Zhe Wang, Michael J. Franklin, Minos N. Garofalakis, and Joseph M. Hellerstein. Querying probabilistic information extraction. *Proc. VLDB Endow.*, 3(1):1057–1067, 2010.
- [167] Qiming Wang and Raul Castro Fernandez. Solo: Data discovery using natural language questions via A self-supervised approach. *Proc. ACM Manag. Data*, 1(4):262:1–262:27, 2023.
- [168] Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. TUTA: tree-based transformers for generally structured table pre-training. In *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*, pages 1780–1790. ACM, 2021.
- [169] Zhiruo Wang, Zhengbao Jiang, Eric Nyberg, and Graham Neubig. Table retrieval may not necessitate table-specific model design. *CoRR*, abs/2205.09843, 2022.
- [170] Laura Wendlandt, Jonathan K. Kummerfeld, and Rada Mihalcea. Factors influencing the surprising instability of word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2092–2102. Association for Computational Linguistics, 2018.
- [171] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3(1):1–9, 2016.
- [172] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics, 2020.
- [173] Maryann Xue, Yingyi Bu, Abhishek Somani, Wenchen Fan, Ziqi Liu, Steven Chen, Herman Van Hovell, Bart Samwel, Mostafa Mokhtar, Rk Korlapati, Andy Lam, Yunxiao Ma, Vuk Ercegovic, Jiexing Li, Alexander Behm, Yuanjian Li, Xiao Li, Sriram Krishnamurthy, Amit Shukla, Michalis Petropoulos, Sameer Paranjpye, Reynold Xin, and Matei Zaharia. Adaptive and robust query execution for lakehouses at scale. *Proc. VLDB Endow.*, 17(12):3947–3959, 2024.

- [174] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. Infogather: entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 97–108. ACM, 2012.
- [175] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yaqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report. *CoRR*, abs/2407.10671, 2024.
- [176] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. TaBERT Configuration. https://github.com/facebookresearch/TaBERT/blob/74aa4a88783825e71b71d1d0fdb6b338047eea9/table_bert/vertical/config.py#L23, 2020. [Online; accessed October 12, 2023].
- [177] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. Tabert: Pre-training for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 8413–8426. Association for Computational Linguistics, 2020.
- [178] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics, 2018.
- [179] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir R. Radev. Spider Data Release. <https://yale-lily.github.io/spider>, 2018. [Online; accessed October 12, 2023].
- [180] Qingkai Zeng, Yuyang Bai, Zhaoxuan Tan, Shangbin Feng, Zhenwen Liang, Zhihan Zhang, and Meng Jiang. Chain-of-layer: Iteratively prompting large language models for taxonomy induction from limited examples. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024*, pages 3093–3102. ACM, 2024.

- [181] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çagatay Demiralp, and Wang-Chiew Tan. Sato: Contextual semantic type detection in tables. *Proc. VLDB Endow.*, 13(11):1835–1848, 2020.
- [182] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M. Procopiuc, and Divesh Srivastava. On multi-column foreign key discovery. *Proc. VLDB Endow.*, 3(1):805–814, 2010.
- [183] Meihui Zhang, Marios Hadjieleftheriou, Beng Chin Ooi, Cecilia M. Procopiuc, and Divesh Srivastava. Automatic discovery of attributes in relational databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*, pages 109–120. ACM, 2011.
- [184] Shuo Zhang and Krisztian Balog. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, pages 1553–1562. ACM, 2018.
- [185] Shuo Zhang and Krisztian Balog. Web table extraction, retrieval, and augmentation: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(2):13:1–13:35, 2020.
- [186] Tianping Zhang, Shaowen Wang, Shuicheng Yan, Li Jian, and Qian Liu. Generative table pre-training empowers models for tabular prediction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14836–14854. Association for Computational Linguistics, 2023.
- [187] Yi Zhang and Zachary G. Ives. Finding related tables in data lakes for interactive data science. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1951–1966. ACM, 2020.
- [188] Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. Robut: A systematic study of table QA robustness against human-annotated adversarial perturbations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6064–6081. Association for Computational Linguistics, 2023.
- [189] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.
- [190] Guangyao Zhou, Nishanth Kumar, Miguel Lázaro-Gredilla, Shrinu Kushagra, and Dileep George. Pgmax: Factor graphs for discrete probabilistic graphical models and loopy belief propagation in JAX. *CoRR*, abs/2202.04110, 2022.
- [191] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. JOSIE: overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the*

2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019, pages 847–864. ACM, 2019.

- [192] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. LSH ensemble: Internet-scale domain search. *Proc. VLDB Endow.*, 9(12):1185–1196, 2016.