TabSketchFM: Sketch-based Tabular Representation Learning for Data Discovery over Data Lakes

Aamod Khatiwada[⋆▶] Harsha Kokel[⋆] Ibrahim Abdelaziz Subhajit Chaudhury Julian Dolby
Northeastern University *IBM Research* *IBM Research* *IBM Research* *IBM Research*
khatiwada.a@northeastern.edu harsha.kokel@ibm.com ibrahim.abdelaziz1@ibm.com subhajit@ibm.com dolby@us.ibm.com

Oktie Hassanzadeh Zhenhan Huang[▶] Tejaswini Pedapati Horst Samulowitz Kavitha Srinivas
IBM Research *Rensselaer Polytechnic Institute* *IBM Research* *IBM Research* *IBM Research*
hassanzadeh@us.ibm.com huangz12@rpi.edu tejaswinip@us.ibm.com samulowitz@us.ibm.com kavitha.srinivas@ibm.com

arXiv:2407.01619v4 [cs.LG] 26 Aug 2025

***Abstract*—Enterprises have a growing need to identify relevant tables in data lakes; e.g. tables that are unionable, joinable, or subsets of each other. Tabular neural models can be helpful for such data discovery tasks. In this paper, we present TabSketchFM, a neural tabular model for data discovery over data lakes. First, we propose novel pre-training: a sketch-based approach to enhance the effectiveness of data discovery in neural tabular models. Second, we finetune the pretrained model for identifying unionable, joinable, and subset table pairs and show significant improvement over previous tabular neural models. Third, we present a detailed ablation study to highlight which sketches are crucial for which tasks. Fourth, we use these finetuned models to perform table search; i.e., given a query table, find other tables in a corpus that are unionable, joinable, or that are subsets of the query. Our results demonstrate significant improvements in F1 scores for search compared to state-of-the-art techniques. Finally, we show significant transfer across datasets and tasks establishing that our model can generalize across different tasks and over different data lakes.**

I. Introduction

Enterprises store critical data in *data lakes*, large repositories of tabular data, for both governance and analytics [1]. It is essential to effectively find relevant tables (e.g., joinable, unionable, subsets) within lakes for reporting, decision-making, statistical analysis, and more [2], [3]. For instance, *Unionable table search* helps to augment an existing table with new rows, and to enrich analytics over it [4]. *Joinable table* search is useful to run analytics that needs data in columns from multiple tables, e.g., to identify how many customers in some region purchased a product due to an email campaign. Also, identifying *subsets* or *supersets* of a table is essential, for example, when searching for potential copies of personal data to comply with the General Data Privacy Regulation of the European Union[1]. Motivated by such use cases, we focus on the problem of identifying tables from the data lakes that can be *unionable, joinable, and/or subsets of each other* [4]–[6]. Specifically, in this work, we propose a pre-trained tabular model that could help in enhancing the effectiveness of data discovery techniques.

Central to the problem of identifying relevant tables in the data lakes is identifying similar columns in tables. However, the columns in a table could be ambiguous. For instance, a column called `Age` is different when it is in a table about buildings from when it is in a table about people. Furthermore, values within a column `Age` play a key role in determining semantics; `Age` of buildings in New York are unlikely to match `Age` of historical buildings in Egypt. Contextualizing columns could be useful in resolving such ambiguities. Language Models (LMs) are shown to be successful in contextualizing words in a sentence [7]. Consequently, we use them to contextualize columns and use such contextualization to determine similar columns or tables.

Existing pre-trained tabular models [8]–[11] based on LMs [7] do not focus on identifying relevant tables. Instead, they focus on language-style tasks such as fact checking [10], question answering [12], converting natural language to SQL [11], table cell content population [8], and table metadata prediction [8], [9]. In addition, input to these models is the actual cell values or a row because of the focus on language-style tasks. However, this has two limitations for understanding relevant tables: (a) only a small number of values can be passed in as input because of severe limits on token length in encoder-only models that can generate embeddings for search (e.g. 512 tokens in BERT [7]), and this potentially limits the information available to the model to understand the input table [2] ; (b) treating numerical values in the table as text tends to lose their semantics; it may be better to pass numerical information directly as vectors into the model.

To address these limitations on input length and numerical representation, we introduce a novel transformer-based tabular model—*TabSketchFM*—exploiting both table metadata and

⋆ Joint first-authors ▶ Work done while at IBM Research.

[1]https://gdpr-info.eu

[2] Decoder only models have larger context lengths. However, it has been difficult until recently to use decoder-only models for embeddings [13]. Moreover, these models are expensive to deploy and provide only a minimal improvement in accuracy over encoder-only models such as BERT. This is why embeddings for search are mostly based on BERT-style models. Further, even with large context lengths, it is not practical to serialize tables with millions of rows as sentences.

cell values that creates different sketches over them instead of linearizing cell values. Specifically, we abstract column cell values using data sketches, as has been used in the data discovery literature (e.g., MinHash sketches [14], and numerical sketches [15]). Such sketches capture tabular features, and we input them into the neural models. Inputting numerical sketches bypasses feeding table cell values to the model, addressing the token limit issue, as well as the numerical representation of cell values.

After pretraining using sketches, we finetune the models for individual data discovery tasks such as union, join, and finding subsets. The embeddings given by the finetuned models are then used for the respective retrieval tasks. While some in the literature [16] have used self supervision for fine tuning, self supervision applies better for some tasks such as unions compared to say joinability. To finetune models for the data discovery tasks, we used a suite of 8 benchmarks across the 3 tasks *LakeBench* that we have open sourced for use [17]. On the fine tuning tasks, *TabSketchFM* outperformed other neural methods, by as much as 55% on F1. We then used the fine tuned models for search tasks, and showed that *TabSketchFM* outperforms the state-of-the-art neural and traditional methods by as much as 70% on F1.

Importantly, we also compare the performance of specialized traditional and neural models for data discovery against an off-the-shelf pretrained model that encodes column values as a single sentence. To our surprise, the off-the-shelf model performs much better than existing systems - in fact, for certain tasks such as union, it is sufficient to use that model for discovering unionable tables; our tabular specific model does worse, as do all other models that target unionability. The insight from this finding is that while some data discovery tasks need values to match across columns (e.g. join or subsets), for union, it is sufficient when columns have the same semantic meaning; indeed there is no need for any value overlap. Sentence embeddings capture those semantics well. Because certain tasks seem to benefit from these sentence embeddings (e.g. union), and other do not (e.g., subset), we propose a way to concatenate the embeddings from sentence encoders with tabular model embeddings, and show that one can augment tabular embeddings with value embeddings to improve performance, at least in some cases.

In summary, our contributions are as follows:

- **A novel tabular pretrained model.** We are the first to present a sketch-based tabular pretrained model, *TabSketchFM*, which contextualizes columns based on their data sketches and tabular metadata. The sketches help to better handle input token limits and numerical cell values. To achieve this, we modified the text-based transformer architecture to take in numerical and MinHash sketches as input. A purely sketch based approach though misses the opportunity to include table semantics from column values. Given that the inputs to the transformer architecture in our system are numeric vectors, it is difficult to combine these numeric inputs with tokens derived from cell values. We therefore combine the sketch based model embeddings with column value embeddings from existing pretrained sentence transformer models. We show that this combination boosts performance on some data discovery tasks.
- **Finetuning models for downstream tasks.** We develop cross-encoders from pretrained TabSketchFM, which we fine-tune for a range of downstream tasks from the LakeBench collection [17]. We fine tune not only *TabSketchFM* but other tabular pretrained models to see if they can perform well on these downstream tasks and present the comparision.
- **Detailed ablation study.** We assess the effectiveness of different sketches for identifying different types of table relevance. Our study presents interesting insights such as Minhash sketches of column values are crucial for identifying joins, but numerical sketches are crucial for identifying subsets; once again highlighting the value of using sketches for table representation.
- **Demonstration of TabSketchFM on search.** We compared the performance of *TabSketchFM*'s fine tuned models against neural and non-neural baselines. In the case of join search, *TabSketchFM* outperformed most other systems geared for join search in terms of F1 scores.
- **Illustrate generalization.** Finally, we illustrate the main advantage of learning a pretrained foundational model by showing the generalization of search performance across tasks. For example, we fine tune the pretrained model on join identification on one dataset and use it for union search on another dataset. Our evaluation shows that our finetuned models generalize well.

II. Related Work

Tabular Pretrained Models. Neural models are pretrained on tabular datasets to either recover masked tokens in the table or to detect corrupt values [8]–[11]. [18] surveys neural models for tabular data representation and their applications. [8] combined the Masked Language Model (MLM) objective from BERT [7] with a novel Masked Entity Recovery to train TURL. [10] use MLM with novel Cell-level Cloze and Table Context Retrival for TUTA. [11] used Masked Column Prediction and Cell Value Recovery for TABERT while [9] repurposed ELECTRA's objective function [19] for TABBIE. These models are finetuned and evaluated for different downstream tasks such as table metadata prediction, table content population, fact-checking, question answering, semantic parsing, and so on. For example, TABERT is evaluated on neural semantic parsing; TABBIE for column, row, and column type population tasks; and TUTA is finetuned and evaluated for cell and table type classification tasks. These downstream tasks take either a table or a table with a question as input; they do not consider the inter-table tasks such as dataset discovery, which we consider in this paper. Furthermore, they adopt serialization techniques from LMs and represent table values as sentences, whereas we represent using sketches.

Dataset Discovery Methods. Different methods such as keyword search [20], [21] and table search [22]–[24] have been used for the dataset discovery over data lakes. Table Union

Search (TUS) [4] presented finding top-k data lake tables that are unionable with a query table. They determine the column unionability using three statistical tests based on value overlap, semantic overlap, and word embedding similarity, and aggregate them to infer table unionability. D^3L [25], adopted value overlap and word embedding measures from TUS and added three additional measures to measure column unionability: numerical column distributions, column header similarity, and regular expression matching. SANTOS [6] considers the relationships between the column pairs in both query and data lake tables. Fan et el. [16] developed a contrastive-learning-based table union search technique called Starmie that captures the context of the whole table in the form of column embeddings. AutoTUS [26] searches for unionable tables by contextualizing entire tables with embeddings of the relationship between the column pairs. Furthermore, there are other table search techniques that focus on finding joinable tables, i.e., given a query table marked with a query column, find the data lake tables that contain a column with overlapping values with the query column. For instance, LSH Ensemble [14] defines joinable table search as a problem of finding data lake tables that have a column having high set containment with the query column. They propose a novel index that approximates set containment computation. JOSIE [5] searches for top-k joinable tables based on exact set containment. AutoJoin [27], on the other hand, determines the fuzzy join between rows of a given pair of joinable tables by employing string transformations such as matching n-grams of cell values and concatenating cell values. PEXESO [28] efficiently searches for the top-k semantically joinable tables. Along with a join on exact values, they also consider fuzzy join, i.e., join on abbreviations and synonyms. DeepJoin [29] is a deep-learning-based technique that also searches for semantically joinable tables. WarpGate [30] is also a system for searching for top-k semantically joinable tables by embedding each column to vector space using pretrained embeddings. They index column embeddings using Locally Sensitive Hashing [31] for efficient search. Recently, Chorus [32] even demonstrated the ability of Large Language Models to identify joinable columns as a task of completing 'pd.merge' command when two tables are presented as pandas dataframe. Our work addresses the task of discovering tables from the data lakes that can be unionable, joinable, and/or subsets of each other. Notably, we specialize data discovery neural models based on a single pretrained model. The advantage of our approach over above mentioned task specific approaches is that, as in natural language processing, much smaller datasets are sufficient to finetune the model for new data discovery tasks; by leveraging the existing tabular knowledge in the pretrained model.

Benchmarks. Several tabular benchmarks exist in the literature [18], [33], [34]. Koutras et al. [35] created benchmarks for schema matching [36] using open data tables and evaluated the existing matching techniques over them. Mudgal et al [37] open-sourced 13 datasets such as the DBLP-Google Scholar author dataset and Walmart-Amazon product datasets for entity matching task [38]. The WDC Web Table corpus [39], containing around 233 million web tables extracted from different web pages, has been widely used for diverse tasks such as Question Answering [12], [40], Semantic Parsing [11], [40], [41], Table Retrieval [42], Table Metadata Prediction [8], [10], [43] and more [44]. Efthymiou et al. [45] used knowledge graphs to create tabular benchmarks for column type prediction and column-to-column binary relationship prediction tasks. The VizNet Benchmark [46] has been used to evaluate column type prediction tasks [47], [48]. GitTables [33] is a large table corpus in the literature. However, they contain web tables that are small entity tables with fewer rows and columns than enterprise tables and do not represent enterprise data lake scenarios [5]. But the most closely related benchmark to our work is the LakeBench collection of datasets [17] for identifying relation between pair of tables (Sec. III-D) Hence, we use that benchmark for finetuning our model. Further, there are publicly available table search benchmarks that are also relevant to our work which we use for illustrating the use of TabSketchFM for data discovery. TUS [4] and SANTOS [6] are table union search benchmarks which we include in our experiments. At the time of writing this paper, there were no search benchmarks for table join search and table subset search, so we generated *Wiki join search* and *Eurostat subset search* benchmarks (Sec. IV-C). However, recently two new benchmarks have appeared in this space. One is manually annotated table join and union search benchmark [49] (also called LakeBench[3]) and another is a semantic table search [50] benchmark that used wikipedia categories and navigational links to establish semantic join and union relation.

III. TabSketchFM

In this section, we detail the architecture of TabSketchFM. Like existing tabular models [10], [12], we pretrain the BERT model [7]. But, unlike them, we adapt the BERT input embeddings to accommodate the numerical inputs.

A. Sketches Preparation

While other models linearize table cells as text, we prepare *sketches*: numerical vectors that capture different features of the table. We use MinHash signature to convert sets of strings to numerical vectors. MinHash is a technique that uses hash functions to map similar sets to similar hash values, allowing for fast estimation of Jaccard similarity between two sets [51].[4] Specifically, we create one table-level sketch called a content snapshot and two types of column-level sketches: numerical sketch and minhash sketches.

Content Snapshot. Recognizing that row information could be crucial in detecting similarity of tables, we create a sketch from the first 10000 rows. We convert each row into a string and generate a MinHash signature from the set of rows. This is the only table-level sketch that we create.

Numerical Sketches. For each column, we extract a set of statistical features that provide insight into its characteristics. These features include the number of unique values and of

[3]Not to confuse with Srinivas et al. [17] LakeBench used for finetuning.

[4]We use datasketch library [52] to compute MinHash.

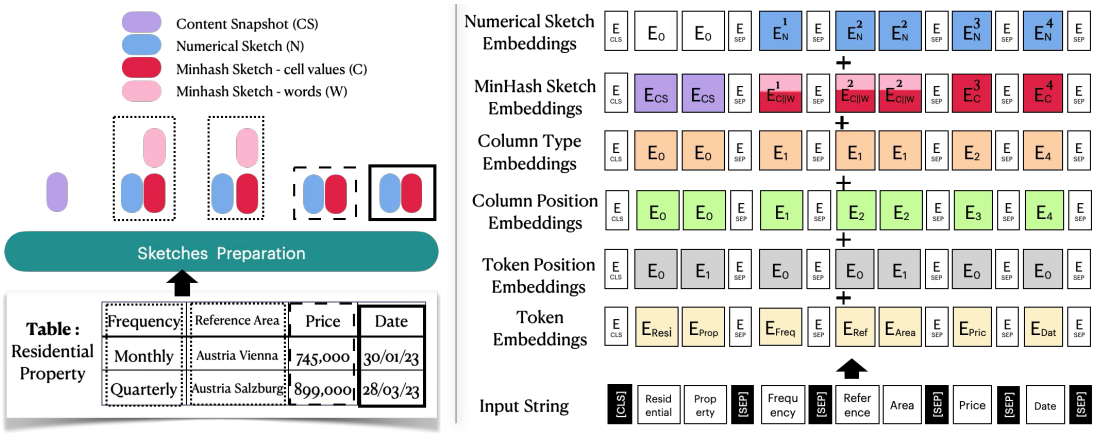


Fig. 1: TabSketchFM Table Representation. The left panel depicts the preparation of various sketches for each column of the input table, "Residential Properties". The right panel illustrates the constituent components of the input embedding, which are subsequently summed to form the input embedding for the 12-layered BERT encoder with self-attention mechanism.

nan values, each normalized by the total number of rows. For string columns, we record the average cell width in bytes, as this can impact the likelihood of the column being used as a join key. Notably, very long strings are unlikely to be suitable for joining. For numeric columns, we calculate additional statistics: percentile values, mean, standard deviation, min and max values. We combine these features into a single vector, which we refer to as a "numerical sketch". To summarize, a numerical sketch vector consists of the following elements: [unique count, NaN count, cell width, 10th percentile, 20th percentile, ..., 90th percentile, mean, standard deviation, min value, max value]. When possible, we convert date columns to timestamps and treat them as numeric columns.

MinHash Sketches. For each column, we treat cell-value as a string and compute a MinHash signature from the set of cell-values.[5] For string columns, we also compute a MinHash signature for set of words within the column. The rationale for this second MinHash is that the words can sometimes capture its semantics (e.g., if `street` appears in two different columns, they maybe both about address, which is useful if one considers a union of two tables that do not necessarily share cell values). Consider the Reference Area column in the table shown in Fig. 1. The cell-values MinHash Sketch treats "Austria Vienna" as an element in the set of strings, whereas "Austria" and "Vienna" are treated as separate strings in words-based MinHash signature. Both MinHashes are concatenated into a single input vector for string columns. For numerical and date columns, only the MinHash for the cell values is included in the input vector.

The left side of Figure 1 illustrates the process of preparing the content snapshot for the table, as well as generating numerical and MinHash sketches for each individual column.

[5]While it may not make sense to convert float or integer columns into MinHash signatures, we do so because it is often difficult to tell if a column is truly a float or an integer or is really a categorical value.

B. Input Representation

Now that we have these sketches, which are arrays of numbers (vectors), a key question is how to adapt BERT's architecture to consume these numeric values. Transformer models construct input embeddings from any given sentence by summing three different embeddings: 1. token embeddings—embedding of the input sentence broken into tokens, 2. token type embedding—that indicates if the token belongs to the first sentence or second, and 3. position embeddings—that indicates the position of the token in the input sentence. We adapt this to table-specific inputs in the following manner. We first generate an "input string" that is concatenation of all the table metadata and column names delimited by the separator token. The input string for table in Fig 1 is shown at the bottom right. The input embedding is constructed from this input string by summing the following different embeddings:

1. Token Embeddings. Similar to the BERT, the first embedding used for the input string is the token embeddings. We initialize the embedding weights from the pretrained BERT uncased model to leverage the extensive pretraining of BERT for its vocabulary. This is represented in Fig 1 with yellow boxes.

2. Token Position Embeddings. Considering each column analogous to a sentence in a paragraph, we re-purpose token positional embeddings to reflect a token's position within a column name. For instance, as shown in Fig. 1 with light gray boxes, the token position of the token `Area` is 1 because it is the second token in that column (after `Reference`).

3. Column Position Embeddings. Columns themselves have positions and are encoded through a new embedding layer which can range from 1 to the total number of columns,[6] as shown in green boxes in Fig. 1. The rationale for including column positions is that they sometimes do have semantics; e.g., a street address next to the city, followed by a zip code. Of course, table semantics do not change as a function of column

[6]Position 0 is reserved for any tokens in the table description.

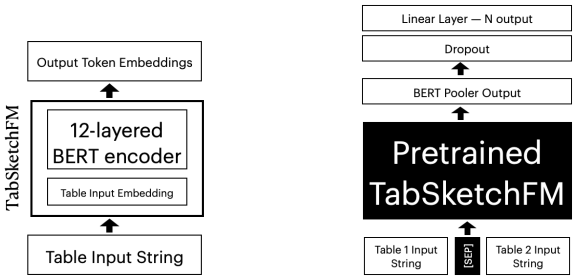
order; nevertheless, we included a position in case it helps the model understand column semantics, with the assumption that the attentional mechanism would disregard order if necessary.
4. Column Type Embeddings. The data types of columns: string, date, integer, or float are encoded through additional embedding. In order to determine column type, we made a best-case effort to parse the first 10 values of each column as dates, integers, or floats and defaulted to string if we could not convert them. In mixed-type columns, this can yield poor results, but at the very worst, at least one of the types was assigned to these columns. This is shown in Fig. 1 in orange, where string, integer, float and date types are represented with integers 1, 2, 3, and 4, resp.
5. MinHash Sketch Embeddings. This embedding encapsulates all the MinHash sketches, including those derived from cell values, words, and content snapshots. The embedding is obtained by passing the MinHash vectors through a linear transformation layer. Notably, the MinHash embedding for table meta-data tokens corresponds to the content snapshot embedding, denoted as E_{CS}, which is visually represented by purple boxes. For numerical columns, the MinHash embedding is the embedding of the MinHash signature derived from the set of corresponding cell values, denoted as E_C, and illustrated in red. In contrast, the MinHash embedding for string columns is constructed by concatenating the MinHash signatures from both the corresponding set of cell values and the set of words, resulting in the embedding $E_{C||W}$, shown with red and pink.
6. Numerical Sketch Embedding. Lastly, each column's numerical sketch vector is fed into a linear transformation layer, and the output is utilized as the numerical sketch embedding.

The sum of these embeddings is used as the input embedding for the 12-layered BERT encoder model with self-attention from HuggingFace[7]. The self-attention in BERT is bi-directional: each token can attend to the tokens on both its left and the right side. This bi-directional attention helps the TabSketchFM model to disambiguate column names like `Age` which can mean mean person's age in an `Employee` table vs age of a building in a `Residential Properties` table.

C. Pretraining

Dataset. Existing works generally use non-public data [18] or web tables [39] to pretrain the models. Unlike enterprise data, web tables often have few rows and columns to ease human consumption [20] and they focus on entities popular on the web (e.g., countries). Their performance may not generalize to enterprise data, where the tables have a large number of rows, the entities are often domain-specific, contain cryptic code words, and have a lot of numerical information. Therefore, we created a de-duplicated pretraining dataset of $197,254$ enterprise-like tables (CSVs) from CKAN and Socrata that are legally cleared to have the open licenses. This pretrain dataset contains 2234.5 rows and 35.8 columns in each table on average. About 66% of columns were non-string, resembling an enterprise datalake.

[7]https://huggingface.co/docs/transformers/model_doc/bert



(a) TabSketchFM Pretraining. (b) TabSketchFM Finetuning.

Fig. 2: (a) For pretraining, the table input embedding (see Section III-B) is passed through 12-layered best encoder to generate output token embeddings. The output token embeddings are used for MLM objective with cross entropy loss(Section III-C). We initialize the weight of the BERT encoder as well as the token embeddings from the pretrained base uncased BERT model. (b) Two input tables are concatenated and passed through the pretrained TabSketchFM. The BERT pooler output (embedding of the first token) is taken and passed through a dropout and a linear layer to generate output of size N.

Data Augmentation. In the image processing literature [53], different transformations are applied over an original image to generate new image samples for training. This makes the trained model robust on different variants of the same image. We want the model to be robust to the order of columns such that shuffling columns in a table does not impact its semantics. So, we created three different versions of the table, by changing the column order, which in turn, changed the table's content snapshot (see Section III). This increases the total number of pretraining tables to 290,948 and for each of them, we create signatures as explained in Section III.
Method. We employ the standard Mask Language Modeling (MLM) as the pretraining objective for our model, as in most text-based language models. First, a few tokens are randomly sampled according to the masked language probability. Then they are substituted with a [MASK] token to create an input sequence. The input sequence is fed to the model, and the model is asked to predict the substituted words. This can be viewed as a classification task where the original word is the label and the model's entire vocabulary is the set of possible labels. Cross-entropy loss is computed for each predicted word in the input text that was masked.

$$\frac{-1}{N}(\sum_{n=1}^{N} y_i * log(\hat{y_i})) \tag{1}$$

Equation (1) defines cross-entropy loss, where y_i is the class labels in the ground truth, $\hat{y_i}$ is the predicted class label and N is the number of training examples in the train set.

In our scenario, for each table, we mask a single column name such that all tokens corresponding to that column are masked. This is analogous to the natural language literature's

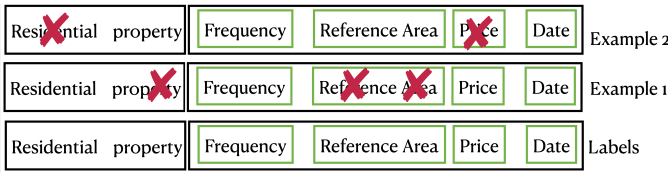


Fig. 3: A combination of whole column masking, and MLM probability masking to generate up to 5 examples. The box on the left hand side is the table description and the one on the right hand side is the column names

whole word masking where all tokens corresponding to a word are masked. In addition, we used MLM probability to mask the tokens in the table description as well. As shown in Fig. 3, we generate several samples from a single table. Note that each masking gives us an example. For small tables with less than 5 columns, we mask each of its columns one after another. However, if a table has a large number of columns, this produces a lot of examples for the same table, over-representing it. So, for large tables with more than 5 columns, we randomly select five columns and mask them. Following this strategy over the augmented pretraining dataset, we get $730,553$ examples in training, $54,430$ examples in validation, and $58,141$ examples in test sets. We provide the experimental setup for pretraining in Section IV. Figure 2a shows the TabSketchFM pretraining architecture.

D. Fine-tuning

To use our neural tabular model for identifying relevant tables in the data lakes, we finetune TabSketchFM on the *LakeBench* collection [17]. LakeBench is a collection of 8 benchmark that includes data from various sources such as open government data from CKAN and Socrata, economic data from the European Central Bank [54], Spider [55], and synthesized data from large knowledge graphs such as Wikidata [56]. The benchmarks cover binary classification, regression, and multi-label classification for the tasks of identifying table unionability, table joinability, and table subset. It contains 3 datasets for identifying unionable tables—TUS-SANTOS, Wiki Union, and ECB Union; 4 datasets for identifying joinable tables—Wiki Jaccard, Wiki Containment, Spider-OpenData, and ECB Join; and one dataset for identifying table subsets—CKAN Subset. Of these datasets, Wiki Jaccard, Wiki Containment and ECB Union are formulated as regression task to estimate jaccard similarity value, containment value, and the number of unionable columns, resp. (cf [17]); ECB Join is multi-class classification to predicts the joinable columns in the table; and remaining are classification tasks. Table I summarizes the main characteristics of the datasets to show the type of tasks we used for fine tuning.

For each finetuning task, we create a cross-encoder [57], which evaluates if a pair of tables can be joined, unioned or are subsets of each other. Figure 2b shows the architecture of the TabSketchFM cross-encoder used for finetuning. For the classification and regression tasks, we use the BERT pooler output, that is the embedding of the first [CLS] token and pass it through a dropout and a linear layer to generate output of size N. For binary classification tasks, the output layer has N=2, and we use cross-entropy loss. For regression tasks, the output layer has N=1 and we use mean-squared error loss. For multi-class classification tasks, the output layer has N = number of classes and we use binary cross-entropy with logit loss for finetuning. The goal of building multiple cross encoders for the same task across different datasets was to examine whether the cross encoders would generalize across datasets; allowing re-use to new data with no additional training.

E. TabSketchFM Overview

We propose TabSketchFM a novel sketch-based tabular representation learning architecture for dataset discovery tasks. Section III-B explains how we generate the sketches, and Section III-B describes how we adapt BERT embedding module to incorporate the numerical sketches. The model is pretrained on opendata using MLM objective to predict the columns with cross-entropy loss, described in Section III-C. After pretraining, we fine-tune the model for downstream tasks like union, join, and subset, as elaborated in Sec. III-D. Finally, we use the finetuned TabSketchFM for data discovery tasks, in the experiments. To do so, we extract the table embeddings from the finetuned TabSketchFM, and use that to create nearest neighbor indexes for search tasks. We evaluate all these aspects in the next section.

IV. Experiments

Our empirical work is organized around the following research questions:

- **Q1.** How does sketch-based TabSketchFM compare to existing pretrained tabular models when finetuned for relevant table identification tasks in LakeBench?
- **Q2.** How do different sketches of TabSketchFM impact different tasks?
- **Q3.** How well do these fine tuned TabSketchFM models perform on data discovery tasks?
- **Q4.** Can TabSketchFM fine tuned on a single task and domain, adapt and generalize to a new domain?

We pretrain TabSketchFM using 4 A100 40GB GPUs for 2 days, until the model converged.[8] Our pretrained model contains $118M$ parameters, similar to other row-based models in our experiments. We use patience of 5, i.e., we consider the model as having converged if the validation loss does not decrease for more than 5 epochs. As studied in the literature [58] this not only reduces training time but also helps us avoid overfitting. For finetuning the cross encoders, for all tasks, we use one A100 40GB GPU. Most finetuning tasks converged within 6 hours using the same 5 epoch patience as pretraining except for the Wiki Union benchmark which took 24 hours. Our code is accessible at https://github.com/ibm/tabsketchfm.

[8]The source code is available on https://github.com/ibm/tabsketchfm

TABLE I: Cardinality of all the datasets in LakeBench, as well as search benchmarks in this paper.

Benchmark	Task	# Tables	Avg. Rows	Avg. Cols	# Table Pairs			Data type distribution (%)			
					Train	Test	Valid	String	Int.	Float	Date
TUS-SANTOS	Binary Classification	1127	4285.17	13.04	16592	3566	3552	77.94	8.62	7.51	5.93
Wiki Union	Binary Classification	40752	51.05	2.66	301108	37638	37638	57.97	14.38	24.25	3.4
ECB Union	Regression	4226	292.47	36.3	15344	1910	1906	47.72	14.05	36.31	1.92
Wiki Jaccard	Regression	8489	47.3	2.8	12703	1412	1572	57.5	15.66	19.76	7.07
Wiki Containment	Regression	10318	47.15	2.79	21007	2343	2593	57.26	15.26	20.58	6.9
Spider-OpenData	Binary Classification	10730	1208.87	9.09	5146	742	1474	42.22	18.51	32.62	6.66
ECB Join	Multi-label Clasification	74	8772.24	34.47	1780	222	223	52.14	7.8	37.79	2.27
CKAN Subset	Binary Classification	36545	1832.58	25.37	24562	2993	3010	31.75	17.53	46.14	4.58
Eurostat Subset	Search	38904	2157	10.46				64.63	9.03	7.83	18.50
Wikijoin	Search	46521	49.64	2.68				58.13	13.35	25.0	3.50

TABLE II: Performance (avg $\pm$ stdev) of TabSketchFM vs. other baseline models on LakeBench averaged across five random seeds. **Best** performance is highlighted in bold, and the second best is underlined.

Tasks	Vanilla BERT	TAPAS	TABBIE	TUTA	TaBERT	TabSketchFM (ours)
TUS-SANTOS (F1)	$\mathbf{0.99 \pm 0.00}$	0.34 ± 0.00	$\underline{0.75 \pm 0.22}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$	$\mathbf{0.99 \pm 0.00}$
Wiki Union (F1)	0.33 ± 0.00	0.41 ± 0.00	0.64 ± 0.12	0.33 ± 0.00	$\mathbf{0.97 \pm 0.00}$	$\underline{0.94 \pm 0.00}$
ECB Union (R2)	0.03 ± 0.03	-0.01 ± 0.01	0.02 ± 0.01	$\underline{0.87 \pm 0.01}$	0.35 ± 0.06	$\mathbf{0.90 \pm 0.03}$
Wiki Jaccard (R2)	0.00 ± 0.00	-0.03 ± 0.03	0.25 ± 0.02	$\underline{0.43 \pm 0.02}$	0.33 ± 0.03	$\mathbf{0.58 \pm 0.03}$
Wiki Containment (R2)	0.00 ± 0.00	0.00 ± 0.00	0.21 ± 0.01	$\underline{0.35 \pm 0.03}$	0.30 ± 0.03	$\mathbf{0.58 \pm 0.02}$
Spider-OpenData (F1)	0.71 ± 0.06	0.65 ± 0.00	0.57 ± 0.00	0.76 ± 0.12	$\mathbf{0.87 \pm 0.02}$	$\underline{0.83 \pm 0.01}$
ECB Join (F1)	0.63 ± 0.04	0.40 ± 0.00	0.42 ± 0.03	$\underline{0.81 \pm 0.01}$	0.79 ± 0.03	$\mathbf{0.86 \pm 0.01}$
CKAN Subset (F1)	0.43 ± 0.00	0.43 ± 0.00	0.43 ± 0.00	0.43 ± 0.00	$\underline{0.43 \pm 0.00}$	$\mathbf{0.98 \pm 0.00}$

A. Fine-tuning models on LakeBench Tasks

We evaluate our finetuned cross-encoders and the existing value-based tabular foundational models for the LakeBench unionability, joinability, and subset tasks.

1) Baselines: We compare TabSketchFM against *four* publicly available tabular foundational models: TUTA, TaBERT, TABBIE, and TAPAS. We adapted these models for Lakebench data discovery tasks by building a dual encoder architecture. Each encoder represents the pretrained model with shared parameters; and the same model is used to encode each element of a table pair. The embeddings from the last layer of the encoders were concatenated and passed through a two-layered MLP (multi-layered perceptron). We used this architecture as opposed to the cross-encoder architecture we used for TabSketchFM because in general it is unclear how to feed two different tables with different row sizes into a cross-encoder. It gets even more complicated when one considers models such as TUTA that model hierarchical information in the table as a tree. While we were able to finetune the TUTA and TaBERT code in this dual encoder architecture, the code of TAPAS and TABBIE were less amenable to finetuning. Hence, for TAPAS and TABBIE, we froze their pretrained models while finetuning, but allowed the two layers above the model to learn the weights. In this case, the pretrained model can be seen as producing an embedding that is then fed into two layered network that tunes for a specific task. We will make the code for all the baselines available. Below, we provide details on model-specific implementation issues.

- **TaBERT** provides two types of embeddings for a table: context embeddings corresponding to the tokens in the context, and column embeddings corresponding to each column in the table [11]. For each table, we compute both embeddings for top $10,000$ rows. In [11], a content snapshot approach is described which creates synthetic rows based on the n-gram matches of the input question with the cell values. We could not experiment with this feature, since 1. this technique is not part of the authors' provided source code[9] and 2. LakeBench tasks do not contain a question. So, we compute these embeddings for each table, mean-pooled over the context tokens and tabular columns respectively. The final embedding for the table pair is computed as the concatenation of context and column embeddings obtained for each table.
- **TUTA** creates a tree-based attention matrix over all the tokens in the table, which has significant memory requirements. To overcome this, we obtain the top 256 rows and columns of the table, and the first 256 tokens of the table sequence.
- **TAPAS** requires a natural language query and a table as input [12]. For our dataset discovery tasks, we sent an empty string as a natural language query and use 512 serialized row based version of the table as the embedding.
- **TABBIE** provides row and column embeddings of the table [9]. For our work, we obtained the row embeddings for each row in the table. Following the original work, we use the first 30 rows and 20 columns of the table. These row embeddings are combined using the mean operation and the resulting vector is frozen as table embedding.

We also added a naive baseline, **Vanilla BERT**, to examine to what extent LakeBench tasks can be performed based on column headers alone; this provides a measure of the difficulty

[9]https://github.com/facebookresearch/TaBERT

of a given task. For *Vanilla BERT*, column headers were provided for the two tables as two sentences, and the BERT model was finetuned on the specific LakeBench task as were other models.

2) Results: Table II compares the performance of TabSketchFM with baseline models on LakeBench. For regression tasks, we report R2 statistics, and for (binary and multiclass) classification tasks, we report a weighted F1 score to handle skew in classes[10]. The best performance is highlighted in bold and the second best is underlined. All the results presented are aggregated over 5 random seeds. We see that on *five of the eight tasks, TabSketchFM outperformed all baselines, performed as well as the best methods on one, and performed second best on the remaining two*.

The binary classification adaptation of the TUS-SANTOS benchmark, was too easy and could be solved on the basis of column headers alone[11], as evident from the performance of the Vanilla BERT model. Nevertheless, TAPAS and TABBIE perform relatively poorly on it, pointing to the importance of including the Vanilla BERT as a baseline.

On the CKAN Subset benchmark, we find that the performance of most models was comparable to random guessing since the column headers were exactly the same, and most systems (except TaBERT) did not have a view of the entire dataset. On Wiki Union and on Spider-OpenData, TaBERT outperformed TabSketchFM. Even so, TabSketchFM achieved the second best performance on these two tasks.

We also performed an error analysis on table pairs that TabSketchFM got wrong on Wiki-Union, Spider open data and CKAN. In all these cases, mean hamming distances of MinHash sketches for positive pairs were equivalent to those from negative pairs; suggesting that the sketches alone did not contain sufficient information to discriminate those examples. TaBERT's superior performance on the Wiki Union case, combined with the fact that lack of discrimination of MinHashes provides a clue to at least one weakness in the current version of TabSketchFM, which is that we do not consider any values from the column at all. As shown in Figure 5, two columns may be unionable when their values map to the same domain (e.g., different municipalities in Slovakia), with minimal overlap in actual values. In such cases, a system such as TaBERT can still determine that the cell values are similar, and belong to the same semantic type, whereas TabSketchFM cannot. We show how to incorporate cell values externally in the section on search which addresses this issue.

In summary, the answer to Q1 is that sketches-based TabSketchFM is in general superior to other pre-trained models for data discovery in LakeBench. Note that this evaluation does not directly speak to performance on search tasks in data discovery; but it does evaluate the usefulness of sketches for building a good model tuned for data discovery. We turn next to investigating how different sketch types help specific tasks, before returning to the question of how well these fine tuned models perform on search.

[10]Our metrics were implemented from `scikit-learn`: https://scikit-learn.org/stable/modules/model_evaluation.html

[11]We added column headers for this benchmark, unlike in the literature, but removing headers made little difference to the results of tabular models.

B. Effectiveness of sketch types on task types

To explore the relevance of the sketches and content snapshots for the different data discovery tasks, we conduct an ablation study where we use only one type of sketch. The results on LakeBench tasks with the random seed 0 are shown in Table III. In Table IV we present another ablation study where we remove one sketch and retained the rest. Not surprisingly, MinHash sketches are crucial for join tasks, where MinHash alone seems to dominate overall performance of *TabSketchFM*. Subset selection clearly relies on the overlap of the data distribution; numerical sketches alone match the performance of *TabSketchFM*. Given that almost 69% of the columns were numerical in the subset benchmark, this outcome might be expected. The effect of content snapshot was less clear. Looking at Table IV, it is evident that the removal of content snapshot does in fact contribute somewhat to the Wiki join tasks. Numerical sketches contribute to ECB union which is perhaps because approximately 50% of the columns are not strings. Removal of MinHashes mirrors the result from earlier, which is, join tasks are most affected. Surprisingly, the removal of MinHashes affected the subset task; even though numerical sketches by themselves produced good performance. It is possible that there are complex interactions between the sketches. **In summary,** we answer Q2 from our ablation studies that different sketches play different roles across tasks, and may even interact to determining data discovery performance.

C. Application to search

We evaluate the embeddings from our finetuned TabSketchFM for three data discovery tasks: join search, union search, and subset search.

1) Join Search: Here, the task is to find all the tables from the data lake that can be joined on one or more columns with the given query table. Note that prior join works [5], [28], [59] evaluate the efficiency of searching for data lake columns having value overlap with the query column(s), and assume that the columns with overlapping values are always joinable. However, we are interested in whether it makes sense to join tables. For example, two columns: (i) explaining people's Ages with values such as (50, 24, 66, 78, 92, 95) and (ii) explaining students' marks out of 100 (rounded to integer) with values such as (78, 92, 95, 100, 20) can have overlapping values but it is not sensible to join them because they have different semantics. Because TabSketchFM is built to generate contextual embeddings of columns that include value overlap, column headings, its context with other columns, and so on, it should be possible to get rid of such useless joins.

To our knowledge, there are no such benchmarks with annotated gold standards that consider if it is sensible to join[12]. So, we construct a search benchmark named **Wiki Join** from

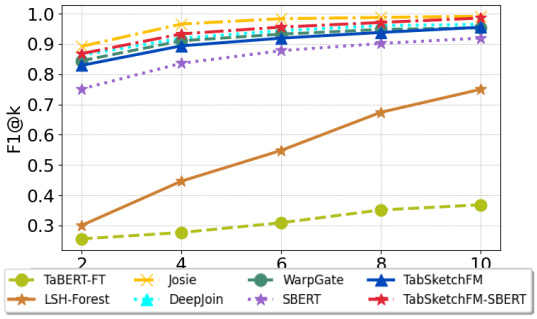
[12]A new benchmark very recently got published on this topic but that was after we had built the benchmark [49]

TABLE III: Importance of different sketches on *TabSketchFM* performance for the different tasks. Performance of sketches that are equivalent (within 2 points) to overall performance is highlighted in green. TUS-SANTOS is not considered because it can be performed based on column headers alone.

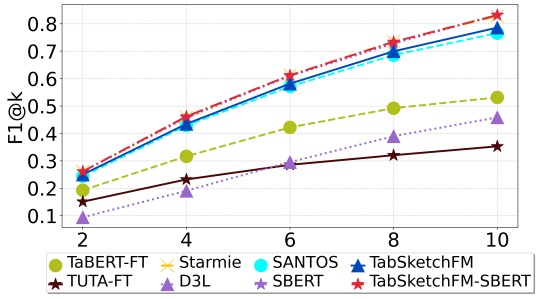
Tasks	Using only MinHash sketches	Using only Numerical sketches	Using only Content snapshot	*TabSketchFM* (w/ everything)
Wiki Union (F1)	0.914	0.804	0.897	0.940
ECB Union (R2)	0.829	0.498	0.752	0.897
Wiki Jacc. (R2)	0.537	0.318	0.314	0.577
Wiki Cont. (R2)	0.628	0.252	0.301	0.587
Spider-Op. (F1)	0.831	0.817	0.797	0.831
ECB Join (F1)	0.874	0.812	0.815	0.856
CKAN Subset (F1)	0.431	0.984	0.431	0.986

TABLE IV: Removing a sketch to study the impact of it on *TabSketchFM* performance for the different tasks. Performance of sketches that declined more than 2 points compared to overall performance is highlighted in red.

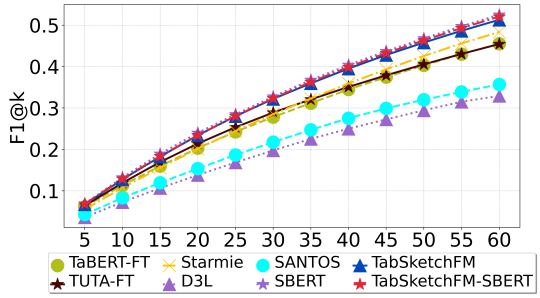
Tasks	Removing only MinHash sketches	Removing only Numerical sketches	Removing only Content snapshot	*TabSketchFM* (w/ everything)
Wiki Union (F1)	0.933	0.927	0.931	0.940
ECB Union (R2)	0.770	0.872	0.897	0.897
Wiki Jacc. (R2)	0.425	0.565	0.519	0.577
Wiki Cont. (R2)	0.358	0.598	0.559	0.586
Spider-Op. (F1)	0.814	0.851	0.847	0.831
ECB Join (F1)	0.812	0.855	0.846	0.855
CKAN Subset (F1)	0.431	0.431	0.980	0.986



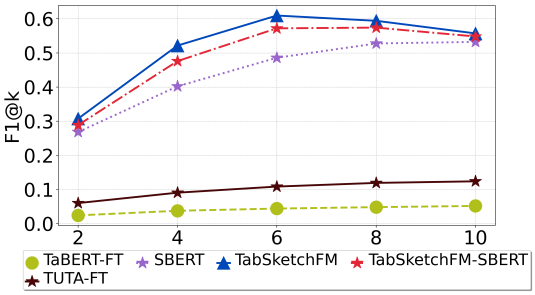
(a) F1 on *wiki join* search for varying k.



(b) F1 on *Santos* search for varying k.



(c) F1 on *TUS* search for varying k.



(d) F1 on *Eurostat* subset search for varying k.

Fig. 4: F1 plot for search on various join, union and subset datasets.

Wikidata (by following the cell-entity mapping approach used in the curation of the Wiki Jaccard dataset [17], see Table I for dataset characteristics). We first generated the tables and then used the ground truth cell-entity mapping to create a list of all the pairs of columns with the same entity annotations. For example, consider Figure 5. All entities in first column of table A and first column of table B are mapped to `Meteorite` where as entities in table C are mapped to `Locations`. So even when there is an overlap in cell values of all tables (`Aleppo` appears in all the tables), only table A and B are considered sensible to join. Once the list of sensible joinable column pairs is obtained, we assign each pair an overlap score, which is the Jaccard score of the entity annotation sets. The pairs with overlap score greater than 0.5 were considered as joinable in our dataset for a total of 5,512 queries. This dataset contains contains $46,521$ tables and $5,512$ query columns.

We compare the join search performance of TabSketchFM with two traditional join search methods—**LSHForest** and **Josie** [5]—and three neural methods–**WarpGate** [30], **DeepJoin** [29], and **TaBERT**. WarpGate is a recent embedding based method that performs joinable column search using SimHash LSH in the embedding space. It uses FastText embedding pretrained on WebTables to generate embeddings of column values. DeepJoin is a similar embedding base approach that finetunes a pretrained language model and uses HNSW [60] for indexing and efficient search. They

TABLE V: F1, Precision & Recall for Wiki-join search.

Baseline	Mean F1	P@10	R@10
TaBERT-FT	30.16	0.43	0.32
LSH-Forest	50.84	0.8	0.7
Josie	**94.86**	**0.99**	**1.00**
DeepJoin	91.59	0.96	0.97
WarpGate	90.34	0.95	0.95
SBERT	83.67	0.96	0.89
TabSketchFM	89.09	0.97	0.94
TabSketchFM-SBERT	92.81	0.98	0.99

TABLE VI: F1, Precision & Recall for SANTOS union search.

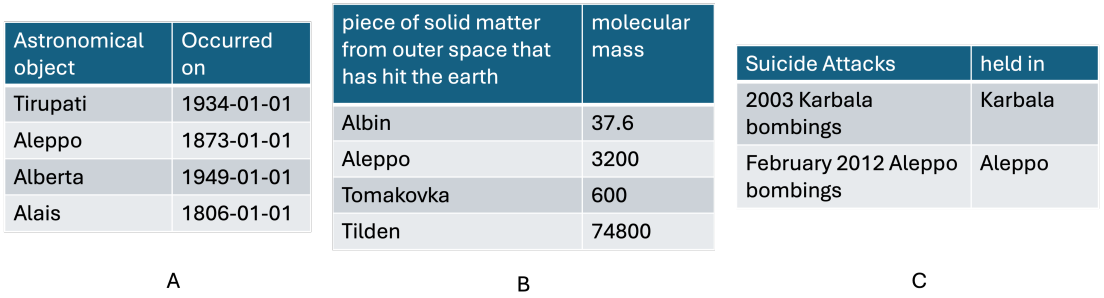
Baseline	Mean F1	P@10	R@10
TaBERT-FT	36.64	0.63	0.46
TUTA-FT	25.34	0.43	0.3
Starmie	**54.08**	**0.97**	0.72
D3L	26.44	0.54	0.4
SANTOS	50.36	0.89	0.67
SBERT	53.86	**0.97**	**0.73**
TabSketchFM	51.38	0.92	0.69
TabSketchFM-SBERT	**54.09**	**0.97**	**0.73**

TABLE VII: F1, Precision & Recall for TUS union search. *SANTOS results are reported only on 125 queries.

Baseline	Mean F1	P@60	R@60
TaBERT-FT	28.05	0.9	0.32
TUTA-FT	28.68	0.89	0.33
Starmie	28.79	0.9	0.33
D3L	20.77	0.6	0.23
SANTOS*	24.27	0.81	0.27
SBERT	**32.73**	**0.99**	**0.38**
TabSketchFM	32.0	0.97	0.37
TabSketch-SBERT	32.3	**0.99**	**0.38**

TABLE VIII: F1, Precision & Recall for Eurostat subset search.

Baseline	Mean F1	P@10	R@10
TABERT-FT	4.03	0.05	0.05
TUTA-FT	9.82	0.13	0.12
SBERT	43.12	0.56	0.51
TabSketch	**49.96**	**0.59**	**0.53**
TabSketch-SBERT	47.54	0.58	0.52



Astronomical object	Occurred on
Tirupati	1934-01-01
Aleppo	1873-01-01
Alberta	1949-01-01
Alais	1806-01-01

A

piece of solid matter from outer space that has hit the earth	molecular mass
Albin	37.6
Aleppo	3200
Tomakovka	600
Tilden	74800

B

Suicide Attacks	held in
2003 Karbala bombings	Karbala
February 2012 Aleppo bombings	Aleppo

C

Fig. 5: Sample subsets of four tables from our Wikidata-based table union and join benchmark (top), along with ground truth labels and mappings (bottom)

experiment with FastText, BERT, DistilBERT and MPNet. In our experiments we use the FastText embedding as it was found to be the best performing embedding when used without finetuning. In addition to column values, DeepJoin also uses column names, table names and column statistics (max, min and average character length) while embedding; much like our sketches. For TaBERT baseline, we use the column embeddings obtained from the TaBERT model which was finetuned on the Wiki-Containment dataset from LakeBench in Section IV-A (**TaBERT-FT**). We could not include TUTA as it does not provide column embeddings.

Recognizing the power of pretrained models trained on massive corpora of text, we added an off-the-shelf sentence encoder **SBERT** (`all-MiniLM-L12-v2`) as a powerful baseline. We include a very simple approach of concatenating the top 100 unique values in a column into a single sentence and encoding it to produced a column embedding. As seen in figure 4a this simple baseline was quite effective compared to many traditional hand crafted systems for data discovery, as well as other neural systems. For finding joins (and subsets), value overlap is a requirement, and data sketches do a very efficient job of finding overlap in large sets, whereas encoding many thousands of values in a sentence is inefficient and likely ineffective. To examine if there is any gain in combining the embeddings of columns derived from TabSketchFM models along with value embeddings for each column using sentence encodings, we also concatenated the two embeddings after normalizing them so the means and variances of the two vectors were in the same scale (**TabSketchFM-SBERT**).

Figure 4a shows an advantage of TabSketchFM over other join search baselines in Wiki Join Search benchmark. Table V summarizes these results considering Mean F1, Precision @ 10, and Recall @10. In this and other benchmarks, TabSketchFM gives comparable performance with the exact

match baseline Josie, and is among the top-2 best performing methods throughout the experiments. Specifically, the variation of TabSketchFM that adds SBERT column embeddings (TabSketchFM-BERT) improves performance of join search by around 3 % in terms of Mean F1.

2) Union Search: We use two publicly available benchmark datasets: $\mathbf{TUS_{Small}}$ and $\mathbf{SANTOS_{Small}}$ [4], [6] because the larger versions of the dataset are not labeled. As a baseline, we compare against state-of-the-art table union search techniques: $\mathbf{D^3L}$ [25], **SANTOS** [6], and **Starmie** [16] as well as the **SBERT** baseline explained earlier. We could not compare against $AutoTUS$ [26], a recent union search system, because its code was not public at the time of writing this paper.

To determine unionability, table embeddings from TabSketchFM did not yield strong results; perhaps because the union benchmarks had been created by creating subsets of the columns in a table or rows. Following the state-of-the-art unionability approach Starmie [16], we used column embeddings of TabSketchFM to determine if two tables are unionable. But we used a simpler technique than the bipartite graph matching algorithm introduced by Starmie. Our approach is defined in Figure 6, and works as follows:

1) For each column c in a given table t, KNNSEARCH returns the nearest $k * 3$ *columns* in the dataset. Note that we try to get a lot more columns than k simply to increase the number of candidate columns we consider at this step because multiple columns from a single table might match a given column.
2) For a column c, COLUMNNEARESTTABLES returns each table near c and the distance of its closest column.
3) For a table t, NEARESTTABLES gathers the near tables for each column in t.
4) We then choose the best, first using RANK_1 to select the tables with the largest number of matching columns, and then, for ones where that is equal, RANK_2 chooses the closest, i.e. the smallest sum of column distances.

For each technique, we use default parameters suggested in the respective papers. For TUTA we had to use table embeddings because that was what the model made available. For both the TUTA and TaBERT, we use the model that was fine tuned on TUS-SANTOS dataset from LakeBench in Sec. IV-A(denoted by **TUTA-FT** and **TaBERT-FT**). The results for TabSketchFM compared to the baselines show that TabSketchFM matched the best system [16] on SANTOS, when we added embeddings from SBERT, see Figure 4b, and Table VI for a summary across methods. For TUS, following the literaure [6], [16], [26], we report the results for tables having at least 60 unionable tables as we consider the value of k up to 60. Additionally, as SANTOS required manual annotation of intent, we only run this on a subset of $\mathbf{TUS_{Small}}$ queries used in the original paper [6]. The value embeddings from the off-the-shelf SBERT model performed very well, as did TabSketchFM when enhanced with value embeddings; see Figure 4c and Table VII. This suggests for union, column values alone are sufficient to find good unionable results; table specific embeddings do not seem to add much. Note that value embeddings out performed all the other systems, pointing to the power of existing pretrained models in embedding tabular data.

3) Subset Search: To our knowledge, there are no benchmarks for subset search. To ensure that the models we built generalize to search, we constructed a dataset for subset search, from Eurostat [13] of 3,072 CSV files comprising the variety of data collected by the EU. These files serve as query tables for search. Data characteristics for this benchmark are provided in Table I. For each file, as shown in Figure 7, we created the following variants reflecting different types of subsets of the file, sampling 25%, 50% or 75% or all of the rows or columns in the table. When all rows or columns were kept, we shuffled either the rows or the columns (shown in purple). These last two variants were added to measure whether the embeddings capture table semantics, that is order invariance for rows and columns of a table. The 11 variants per file made up a search dataset of 38,904 files.

We follows the same procedure described in Figure 6 to find tables that were subsets of the query tables. Figure 4d and Table VIII shows the results of the subset search on the fine tuned model trained on ckan-subset. As we see in the next section on transfer, this model turned out to be the weakest TabSketchFM models for the subset search problem, likely reflecting the quality of the dataset used for training The best performing TabSketchFM model was trained on union, on the European Central Bank data at 0.55. Still, we include it for consistency. In addition, the results suggest that for subsets, adding value embeddings of columns from SBERT actually made performance slightly worse. This highlights the importance of being able to customize embeddings for different data discovery tasks - the ability to add value embeddings to tabular embeddings seems to be useful for such customization.

One additional question we asked is whether the TabSketchFM models exhibited column order invariance, and row order invariance. Unsurprisingly, changing row order made no difference to TabSketchFM, all 3072 queried tables returned the row shuffled variants in their nearest neighbors set. In contrast, changing row order did influence SBERT embeddings; only 2809/3072 (91%) row shuffled variants were returned. Column order can matter to TabSketchFM models, since the encoder is a sequence model. For the column shuffled variants, 3059/3072 appeared in the nearest neighbors set (99.5%). The SBERT model of course did not consider the larger column context, and was hence unaffected by the shuffling of columns.

Overall, from our evaluations of 4 data discovery datasets across traditional and neural approaches answers Q3, that embeddings from finetuned TabSketchFM can be used for data discovery tasks. As the embeddings are pretty small vectors the search in embedding space can be made extremely efficient; making it an ideal candidate for large data lakes.

4) Generalization across tasks and domains: We now investigate Q4. How does our method, fine-tuned on a single task and domain, can adapt and generalize to a new domain.

[13] https://data.europa.eu/en

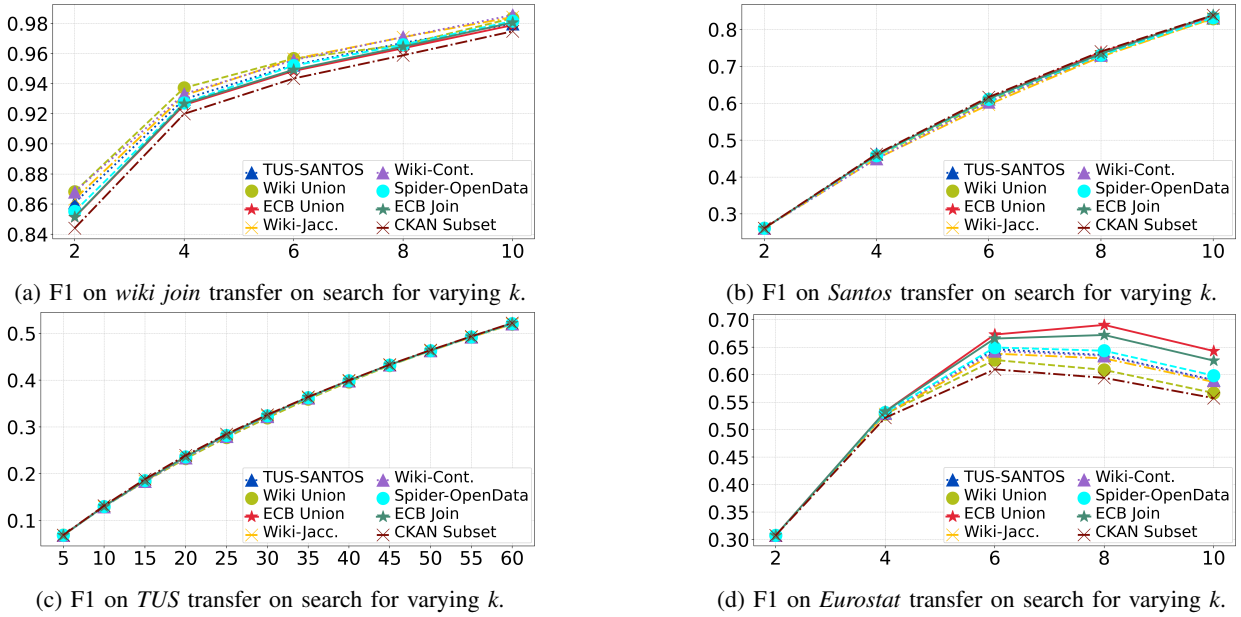
$$
\begin{aligned}
\textsc{KnnSearch}\,(c,k) &\equiv (\langle c_i, d_i\rangle \dots)\ (k\times 3) \text{ nearest columns } c_i \text{ by distance } d_i \\
\textsc{ColumnNearTables}\,(c,k) &\equiv \bigcup_{t_i\in\{\textsc{Table}(c_i)\mid c_i\in\textsc{KnnSearch}(c,k)\}} \left\langle t_i, \min_{\langle c_i,d_i\rangle\in\textsc{KnnSearch}(c,k)\wedge\textsc{Table}(c_i)=t_i} d_i \right\rangle \\
\textsc{NearTables}(t) &\equiv \left\{ \langle t_i, \{\langle c_i, d_i\rangle\}\rangle \;\middle|\; \left\langle \langle c_i, d_i\rangle \in \exists_{c\in\textsc{ColumnNearestTables}(t,k)} \textsc{ColumnNearestTables}(c,k) \right\rangle \right\} \\
\textsc{Rank}_1\left(\langle t_i, \{\langle c_i, d_i\rangle\}\rangle\right) &\equiv |\{c_i \dots\}| \\
\textsc{Rank}_2\left(\langle t_i, \{\langle c_i, d_i\rangle\}\rangle\right) &\equiv \textstyle\sum d_i
\end{aligned}
$$

Fig. 6: Ranking of nearest tables

DATAFLOW	LAST UPDATE	freq	itm_newa	indic_ag	unit	geo	TIME_PERIOD	OBS_VALUE
ESTAT:AACT_EAA01(1.0)	15/05/24 11:00:00	A	1000	PROD_BP	MIO_EUR	AT	1990	1089.16
ESTAT:AACT_EAA01(1.0)	15/05/24 11:00:00	A	1000	PROD_BP	MIO_EUR	AT	1991	1007.4
ESTAT:AACT_EAA01(1.0)	15/05/24 11:00:00	A	1000	PROD_BP	MIO_EUR	AT	1992	850.53
ESTAT:AACT_EAA01(1.0)	15/05/24 11:00:00	A	1000	PROD_BP	MIO_EUR	AT	1993	859.92

25% rows, 25% columns	50% rows, 50% columns	75% rows, 75% columns	100% rows, 25% columns	100% rows, 50% columns	100% rows, 75% columns	25% rows, 100% columns	50% rows, 100% columns	75% rows, 100% columns	100% rows, Shuffle columns	Shuffle rows, 100% columns

Fig. 7: Sample table and generated subsets from Eurostat



(a) F1 on *wiki join* transfer on search for varying k.

(b) F1 on *Santos* transfer on search for varying k.

(c) F1 on *TUS* transfer on search for varying k.

(d) F1 on *Eurostat* transfer on search for varying k.

Fig. 8: F1 plots for transfer across different K values. (a) WikiJoin (b) Santos (c) TUS (d) Eurostat

Figures 8a, 8b, and 8c, 8d, show that the fine tuned cross encoder models generalize rather well across tasks as well as domain, which is encouraging particularly because it shows that the models can be trained on a datalake that is quite different from the datalake that it is deployed to. In enterprise contexts, this is an important requirement, because it means that we can basically train models offline and apply it online. All models shown in the Figure are models that include the value embeddings from a column for maximal generalization, but using the models without values also produced the same type of generalization.

V. Conclusion and Discussion

We presented TabSketchFM, a novel transformer-based tabular representation model that inputs the table in the form of several sketches. We show that cross encoders based on TabSketchFM outperform existing systems on LakeBench related table identification task. We also show that finetuned TabSketchFM can be effectively used to search for tables that are unionable, joinable or are subsets of each other. We highlight that embedding values using sentence encoder models can be a useful addition for some discovery tasks. As far as the search efficiency is concerned, we recommend indexing the datalake offline and at query time only compute embeddings for the query table. This computation is similar to other embedding-based search system.

REFERENCES

[1] F. Nargesian, E. Zhu, R. J. Miller, K. Q. Pu, and P. C. Arocena, "Data lake management: Challenges and opportunities," *Proc. VLDB Endow.*, vol. 12, no. 12, pp. 1986–1989, 2019.

[2] A. Khatiwada, R. Shraga, W. Gatterbauer, and R. J. Miller, "Integrating data lake tables," *Proc. VLDB Endow.*, vol. 16, no. 4, pp. 932–945, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol16/p932-khatiwada.pdf

[3] A. Khatiwada, R. Shraga, and R. J. Miller, "DIALITE: discover, align and integrate open data tables," in *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, 2023, pp. 187–190.

[4] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller, "Table union search on open data," *Proc. VLDB Endow.*, vol. 11, no. 7, pp. 813–825, 2018.

[5] E. Zhu, D. Deng, F. Nargesian, and R. 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*, 2019, pp. 847–864.

[6] A. Khatiwada, G. Fan, R. Shraga, Z. Chen, W. Gatterbauer, R. J. Miller, and M. Riedewald, "Santos: Relationship-based semantic table union search," *Proc. ACM Manag. Data*, vol. 1, no. 1, p. Article 9, 2023.

[7] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423

[8] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu, "TURL: table understanding through representation learning," *Proc. VLDB Endow.*, vol. 14, no. 3, pp. 307–319, 2020.

[9] H. Iida, D. Thai, V. Manjunatha, and M. Iyyer, "TABBIE: pretrained representations of tabular data," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, 2021, pp. 3446–3456. [Online]. Available: https://doi.org/10.18653/v1/2021.naacl-main.270

[10] Z. Wang, H. Dong, R. Jia, J. Li, Z. Fu, S. Han, and D. 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*, 2021, pp. 1780–1790.

[11] P. Yin, G. Neubig, W. Yih, and S. Riedel, "Tabert: Pretraining 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*, 2020, pp. 8413–8426.

[12] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, and J. M. 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*, 2020, pp. 4320–4333. [Online]. Available: https://doi.org/10.18653/v1/2020.acl-main.398

[13] P. BehnamGhader, V. Adlakha, M. Mosbach, D. Bahdanau, N. Chapados, and S. Reddy, "LLM2vec: Large language models are secretly powerful text encoders," in *First Conference on Language Modeling*, 2024. [Online]. Available: https://openreview.net/forum?id=IW1PR7vEBf

[14] E. Zhu, F. Nargesian, K. Q. Pu, and R. J. Miller, "LSH ensemble: Internet-scale domain search," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1185–1196, 2016. [Online]. Available: http://www.vldb.org/pvldb/vol9/p1185-zhu.pdf

[15] A. S. R. Santos, A. Bessa, F. Chirigati, C. Musco, and J. Freire, "Correlation sketches for approximate join-correlation queries," in *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, 2021, pp. 1531–1544. [Online]. Available: https://doi.org/10.1145/3448016.3458456

[16] G. Fan, J. Wang, Y. Li, D. Zhang, and R. J. Miller, "Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning," *Proc. VLDB Endow.*, vol. 16, no. 7, pp. 1726–1739, 2023. [Online]. Available: https://www.vldb.org/pvldb/vol16/p1726-fan.pdf

[17] K. Srinivas, J. Dolby, I. Abdelaziz, O. Hassanzadeh, H. Kokel, A. Khatiwada, T. Pedapati, S. Chaudhury, and H. Samulowitz, "Lakebench: Benchmarks for data discovery over data lakes," 2023. [Online]. Available: https://arxiv.org/abs/2307.04217

[18] G. Badaro, M. Saeed, and P. Papotti, "Transformers for Tabular Data Representation: A Survey of Models and Applications," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 227–249, 03 2023. [Online]. Available: https://doi.org/10.1162/tacl_a_00544

[19] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=r1xMH1BtvB

[20] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: exploring the power of tables on the web," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 538–549, 2008.

[21] D. Brickley, M. Burgess, and N. Noy, "Google dataset search: Building a search engine for datasets in an open web ecosystem," in *The World Wide Web Conference*, 2019, p. 1365–1375.

[22] A. D. Sarma, L. Fang, N. Gupta, A. Y. Halevy, H. Lee, F. Wu, R. Xin, and C. 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*, 2012, pp. 817–828. [Online]. Available: https://doi.org/10.1145/2213836.2213962

[23] O. Lehmberg and C. Bizer, "Stitching web tables for improving matching quality," *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1502–1513, 2017. [Online]. Available: http://www.vldb.org/pvldb/vol10/p1502-lehmberg.pdf

[24] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A data discovery system," in *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, Paris, France, 2018, pp. 1001–1012. [Online]. Available: https://doi.org/10.1109/ICDE.2018.00094

[25] A. Bogatu, A. A. A. Fernandes, N. W. Paton, and N. Konstantinou, "Dataset discovery in data lakes," in *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*, 2020, pp. 709–720.

[26] X. Hu, S. Wang, X. Qin, C. Lei, Z. Shen, C. Faloutsos, A. Katsifodimos, G. Karypis, L. Wen, and P. S. Yu, "Automatic table union search with tabular representation learning," in *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, 2023, pp. 3786–3800. [Online]. Available: https://aclanthology.org/2023.findings-acl.233

[27] E. Zhu, Y. He, and S. Chaudhuri, "Auto-join: Joining tables by leveraging transformations," *Proc. VLDB Endow.*, vol. 10, no. 10, pp. 1034–1045, 2017. [Online]. Available: http://www.vldb.org/pvldb/vol10/p1034-he.pdf

[28] Y. Dong, K. Takeoka, C. Xiao, and M. Oyamada, "Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach," in *37th IEEE International Conference on Data Engineering, ICDE 2021*, 2021, pp. 456–467. [Online]. Available: https://doi.org/10.1109/ICDE51399.2021.00046

[29] Y. Dong, C. Xiao, T. Nozawa, M. Enomoto, and M. Oyamada, "Deepjoin: Joinable table discovery with pre-trained language models," *Proc. VLDB Endow.*, vol. 16, no. 10, pp. 2458 – 2470, 2023.

[30] T. Cong, J. Gale, J. Frantz, H. V. Jagadish, and Ç. Demiralp, "Warpgate: 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*, 2023. [Online]. Available: https://www.cidrdb.org/cidr2023/papers/p75-cong.pdf

[31] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, 1999, pp. 518–529. [Online]. Available: http://www.vldb.org/conf/1999/P49.pdf

[32] M. Kayali, A. Lykov, I. Fountalis, N. Vasiloglou, D. Olteanu, and D. Suciu, "Chorus: Foundation models for unified data discovery and exploration," *Proc. VLDB Endow.*, vol. 17, no. 8, p. 2104–2114, may 2024. [Online]. Available: https://doi.org/10.14778/3659437.3659461

[33] M. Hulsebos, Ç. Demiralp, and P. Groth, "Gittables: A large-scale corpus of relational tables," *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 30:1–30:17, 2023. [Online]. Available: https://doi.org/10.1145/3588710

[34] D. C. Fox, A. Khatiwada, and R. Shraga, "A generative benchmark creation framework for detecting common data table versions," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 5365–5369.

[35] C. Koutras, G. Siachamis, A. Ionescu, K. Psarakis, J. Brons, M. Fragkoulis, C. Lofi, A. Bonifati, and A. Katsifodimos, "Valentine: Evaluating matching techniques for dataset discovery," in *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*, 2021, pp. 468–479. [Online]. Available: https://doi.org/10.1109/ICDE51399.2021.00047

[36] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *VLDB J.*, vol. 10, no. 4, pp. 334–350, 2001. [Online]. Available: https://doi.org/10.1007/s007780100057

[37] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. 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*, 2018, pp. 19–34. [Online]. Available: https://doi.org/10.1145/3183713.3196926

[38] P. Konda, S. Das, P. S. G. C., A. Doan, A. Ardalan, J. R. Ballard, H. Li, F. Panahi, H. Zhang, J. F. Naughton, S. Prasad, G. Krishnan, R. Deep, and V. Raghavendra, "Magellan: Toward building entity matching management systems," *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1197–1208, 2016. [Online]. Available: http://www.vldb.org/pvldb/vol9/p1197-pkonda.pdf

[39] O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer, "A large public corpus of web tables containing time and context metadata," in *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, 2016, pp. 75–76. [Online]. Available: https://doi.org/10.1145/2872518.2889386

[40] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, and J. 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*, 2022. [Online]. Available: https://openreview.net/forum?id=O50443AsCP

[41] T. Yu, C. Wu, X. V. Lin, B. Wang, Y. C. Tan, X. Yang, D. R. Radev, R. Socher, and C. Xiong, "Grappa: Grammar-augmented pre-training for table semantic parsing," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, 2021. [Online]. Available: https://openreview.net/forum?id=kyaIeYj4zZ

[42] F. Wang, K. Sun, M. Chen, J. Pujara, and P. A. Szekely, "Retrieving complex tables with multi-granular graph representation learning," in *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, 2021, pp. 1472–1482. [Online]. Available: https://doi.org/10.1145/3404835.3462909

[43] Y. Suhara, J. Li, Y. Li, D. Zhang, Ç. Demiralp, C. Chen, and W. Tan, "Annotating columns with pre-trained language models," in *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, 2022, pp. 1493–1503.

[44] S. Zhang and K. Balog, "Web table extraction, retrieval, and augmentation: A survey," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 2, pp. 13:1–13:35, 2020. [Online]. Available: https://doi.org/10.1145/3372117

[45] V. Efthymiou, E. Jiménez-Ruiz, J. Chen, V. Cutrona, O. Hassanzadeh, J. Sequeda, K. Srinivas, N. Abdelmageed, and M. Hulsebos, Eds., *Proceedings of the Semantic Web Challenge on Tabular Data to Knowledge Graph Matching, SemTab 2022, co-located with the 21st International Semantic Web Conference, ISWC 2022, Virtual conference, October 23-27, 2022*, ser. CEUR Workshop Proceedings, vol. 3320. CEUR-WS.org, 2022. [Online]. Available: https://ceur-ws.org/Vol-3320

[46] K. Z. Hu, S. N. S. Gaikwad, M. Hulsebos, M. A. Bakker, E. Zgraggen, C. A. Hidalgo, T. Kraska, G. Li, A. Satyanarayan, and Ç. Demiralp, "Viznet: Towards A large-scale visualization learning and benchmarking repository," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, CHI 2019, Glasgow, Scotland, UK, May 04-09, 2019*, 2019, p. 662. [Online]. Available: https://doi.org/10.1145/3290605.3300892

[47] D. Zhang, Y. Suhara, J. Li, M. Hulsebos, Ç. Demiralp, and W. Tan, "Sato: Contextual semantic type detection in tables," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 1835–1848, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1835-zhang.pdf

[48] M. Hulsebos, K. Z. Hu, M. A. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. 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*, 2019, pp. 1500–1508.

[49] Y. Deng, C. Chai, L. Cao, Q. Yuan, S. Chen, Y. Yu, Z. Sun, J. Wang, J. Li, Z. Cao, K. Jin, C. Zhang, Y. Jiang, Y. Zhang, Y. Wang, Y. Yuan, G. Wang, and N. Tang, "Lakebench: A benchmark for discovering joinable and unionable tables in data lakes," *Proc. VLDB Endow.*, vol. 17, no. 8, p. 1925–1938, may 2024. [Online]. Available: https://doi.org/10.14778/3659437.3659448

[50] A. Leventidis, M. P. Christensen, M. Lissandrini, L. Di Rocco, K. Hose, and R. J. Miller, "A large scale test corpus for semantic table search," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1142–1151. [Online]. Available: https://doi.org/10.1145/3626772.3657877

[51] J. Leskovec, A. Rajaraman, and J. D. Ullman, *Mining of Massive Datasets, 2nd Ed*. Cambridge University Press, 2014.

[52] E. Zhu and V. Markovtsev, "ekzhu/datasketch: First stable release," Feb. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.290602

[53] T. Acharya, A. K. Ray, and A. C. Gallagher, "*Image Processing: Principles and Applications*," *J. Electronic Imaging*, vol. 15, no. 3, p. 039901, 2006. [Online]. Available: https://doi.org/10.1117/1.2348895

[54] "ECB Statistical Data Warehouse," https://sdw.ecb.europa.eu/, [Online, accessed October 3, 2023].

[55] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. 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, Oct.-Nov. 2018, pp. 3911–3921. [Online]. Available: https://aclanthology.org/D18-1425

[56] D. Vrandečić and M. Krötzsch, "Wikidata: A free collaborative knowledgebase," *Commun. ACM*, vol. 57, no. 10, p. 78–85, sep 2014. [Online]. Available: https://doi.org/10.1145/2629489

[57] Y. Du, Z. Liu, J. Li, and W. X. Zhao, "A survey of vision-language pre-trained models," in *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, 2022, pp. 5436–5443. [Online]. Available: https://doi.org/10.24963/ijcai.2022/762

[58] W. Zhou, C. Xu, T. Ge, J. J. McAuley, K. Xu, and F. Wei, "BERT loses patience: Fast and robust inference with early exit," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/d4dd111a4fd973394238aca5c05bebe3-Abstract.html

[59] M. Esmailoghli, J. Quiané-Ruiz, and Z. Abedjan, "MATE: multi-attribute table extraction," *Proc. VLDB Endow.*, vol. 15, no. 8, pp. 1684–1696, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol15/p1684-esmailoghli.pdf

[60] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, 2020. [Online]. Available: https://doi.org/10.1109/TPAMI.2018.2889473