Related Table Search for Numeric data using Large Language Models and Enterprise Knowledge Graphs

Pranav Subramaniam[1], Udayan Khurana[2], Kavitha Srinivas[2] and Horst Samulowitz[2]

[1] *The University of Chicago*
[2] *IBM Research*

Abstract

Searching related tables is a crucial part of enterprise data lake exploration. However, data lakes often contain numeric tables with unreliable column headers, and ID columns whose text names have been lost. Finding such related numeric tables in large data lakes is a challenging task. State-of-the-art related table search relies on text values in tables, and cannot be applied on numeric tables. On the other hand, the state-of-the-art for semantic labeling of numeric tables using enterprise knowledge graphs (EKGs) has clear sources of *semantic ambiguity* due to its heuristic and rule-based approaches for determining numeric types and EKG labels, leading to poor performance. In this paper, we propose a system, *NumSearchLLM*, that leverages LLMs alongside EKGs to alleviate the ambiguity in semantic labeling of numeric columns and facilitate both joinable table search, and more general table relatedness tasks. Specifically, we use LLMs to: (i) discover new relationships absent from EKGs; (ii) validate numeric types assigned by heuristics; and (iii) check whether the semantic labels assigned to columns of a table form a meaningful schema. We also show how EKGs can be used in conjunction with LLMs to fix labeling inconsistencies discovered by LLMs by finding alternate labels. We show that by an integrated use of LLMs with EKGs, we can achieve superior performance in joinable and related table search tasks in comparison to the current approaches.

Keywords

EKG- and LLM-based data discovery, numeric data discovery, paper formatting, tabular data

1. Introduction

When exploring enterprise data lakes, it is often crucial to find tables that can add valuable information in combination with a table already in possession. This could be done through a join operation, or even simply finding a table containing information that is related in some way, such as, information in the same domain, different information about the same event, etc. Finding such semantically related tables can involve searching over several kinds of tables in a data lake, including open data sources, which have proven useful in enterprise settings as well [1]. These sources include purely numeric tables with unreliable column headers. Purely numeric data is often found often in many domains, including finance, industrial measurements, and medical practice.

Related table search over purely numeric tables is a novel problem, and is challenging for a number of reasons: (i) the semantics of a numeric table are difficult to determine. For example, a column of positive integers could be IDs or counts; (ii) there are many ways in which tables can be related. For example, they may be joinable on columns representing the same type of information. Or the tables may capture related information, e.g., movie sales and movie cast finance tables.

Automatic procedures leveraging knowledge graphs (KGs) and semantic similarity using word embeddings have been developed to detect table relatedness [2, 3, 4, 5]. However, they each are specific to one type of relatedness (e.g., joinability/unionability) and rely heavily on text values contained in table cells and the column headers. Further, KGs are limited in finding relationships between entities because they are sparse and do not capture all possible relationships that may exist between entities [6].

Typically, methods for determining relatedness do not make the discovered relationship explicit: pairs of tables are returned with a score indicating a join cardinality or an embedding similarity, but no explanation.

On the other hand, existing works concerning purely numeric tables only focus on column semantic labeling. The state-of-the-art work, TTLA [7], for numeric semantic labeling suffers from *semantic ambiguities*: (i) TTLA [7] uses heuristics to assign numeric types to columns, and then uses these types to select subsets of KG candidate labels for each column. The rules implemented by these heuristics are chosen arbitrarily (e.g., checking whether the range is greater than the square root of total values), and must be adapted to different types of numeric data [7]. For this reason, TTLA [7] can assign the wrong type, ultimately resulting in incorrect labeling. For example, Figure 1 shows that a table column containing hospital bed counts can be incorrectly classi-

CIKM'23: Workshop on Enterprise Knowledge Graphs Using Large Language Models, October 21, 2023, Birmingham, UK
psubramaniam@uchicago.edu (P. Subramaniam); ukhurana@us.ibm.com (U. Khurana); Kavitha.Srinivas@ibm.com (K. Srinivas); samulowitz@us.ibm.com (H. Samulowitz)
0009-0005-4684-0501 (P. Subramaniam); 0000-0001-8113-1210 (U. Khurana); 0000-0003-4610-967X (K. Srinivas); 0000-0002-6780-3217 (H. Samulowitz)

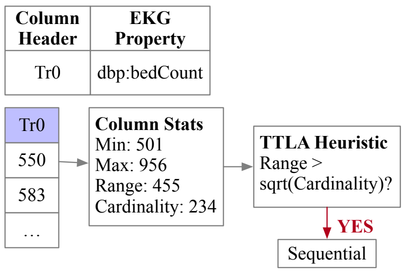


Figure 1: Ambiguous numeric types due to Heuristics.

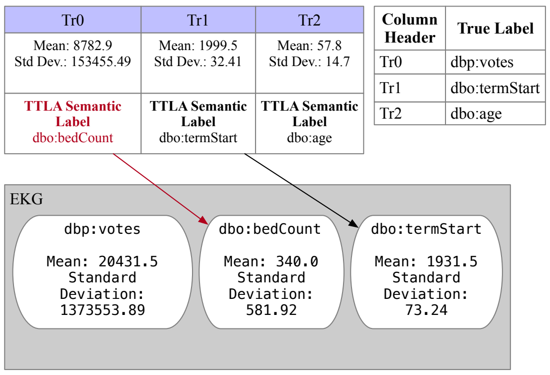
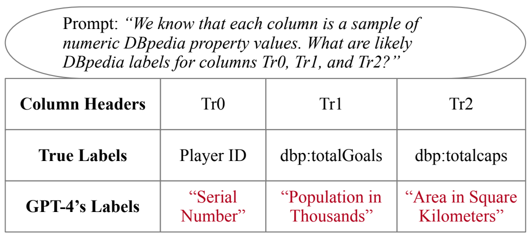


Figure 2: Ambiguous semantic labels due to Euclidean Distance.



Prompt: *"We know that each column is a sample of numeric DBpedia property values. What are likely DBpedia labels for columns Tr0, Tr1, and Tr2?"*

Column Headers	Tr0	Tr1	Tr2
True Labels	Player ID	dbp:totalGoals	dbp:totalcaps
GPT-4's Labels	"Serial Number"	"Population in Thousands"	"Area in Square Kilometers"

Figure 3: GPT-4's paraphrased answer. It cannot effectively detect semantics of soccer statistics table.

fied as "sequential" by following the heuristic rule that the range of values is greater than the square root of the total number of values in the column. (ii) TTLA directly compares column numeric distributions to numeric property distributions in KGs using Euclidean distance, which can often be inaccurate. Figure 2 shows an example of a numeric column such as *Tr*_0 can be mapped to the wrong EKG label (<http://dbpedia.org/bedCount>) simply because the summary statistics of *Tr*_0 are closer to those of dbo:bedCount than other labels.

Recently, many works have proven LLMs are effective at tabular tasks (e.g., joinability/semantic labeling [8], entity resolution [9]). However, LLMs currently cannot directly detect semantics of numeric tables (see Figure 3).

In this paper, we show how LLMs can be leveraged to overcome the semantic ambiguities of numeric typing and semantic labeling by detecting inconsistencies among semantic labels for the same table, and ensuring numbers match the type assigned by heuristics. We then use these semantic labels for table relatedness. We find that while LLMs cannot directly determine numeric table semantics, they are effective at these tasks.

LLMs are pretrained to perform classification, suggesting they can more accurately determine the types of numbers, which are leveraged by the state-of-the-art numeric semantic labeling method [7]. LLMs might also detect out-of-place semantic labels in a schema (e.g., do all labels explain a bus company, or does one of them concern a politician's votes?) These properties of LLMs make them useful in numeric semantic labeling. Further, LLMs are pretrained to answer text summarization, information extraction, and classification questions that can make them useful for determining relatedness of two blocks of text (such as two table schemas), if prompted correctly. And naturally, LLMs can explicate relationships between tables. These LLM properties are potentially useful for related numeric table search, provided proper handling of potential issues such as hallucinations.

We propose *NumSearchLLM*, a system that leverages these strengths of large language models by incorporating them into state-of-the-art numeric semantic labeling methods using EKGs for more accurate explainable related table search over purely numeric tables. Our preliminary evaluation shows that NumSearchLLM achieves superior performance in joinable and related table search tasks in comparison to the current approaches. Although we use DBpedia as a proxy for EKGs, NumSearchLLM can use any EKG as input. Also, giving DBpedia as input can allow NumSearchLLM to find related tables from open data, which has proven valuable in many settings, including the enterprise setting [1, 10, 11, 12].

2. Background

2.1. Related Table Search

Automatic procedures for determining table relatedness rely on *external knowledge*, in the form of heuristics, KGs, or word embeddings. Some works consider tables related based on the heuristics of overlap similarity, or identical datatypes [13, 14, 12]. More recent solutions leverage KG labels or word embedding vectors as external knowledge. Some such solutions map tables to KGs and determine table relatedness via KG relationships [5, 15], others use word embeddings for semantic similarity, both of column values [2, 3] and similarity of column headers [4, 5, 12, 16]. In the case of numeric tables, these solutions cannot be applied, as they rely on text appearing in tables.

2.2. Numeric Table Relatedness

There is no direct way to perform related table search over purely numeric tables. Perhaps the closest work is D3L [16], which uses row entities to contextualize numeric attributes when discovering joins and unions, but they assume a table will have a *subject attribute* they can use to determine the entities.

However, there has been modest success in leveraging KGs to perform column semantic labeling on numeric tables. TTLA is the state-of-the-art numeric column semantic labeling technique which designs and applies a numeric typology first to select subsets of KG labels more likely to contain the label that correctly describes a column, and then maps columns to KG properties by matching column and KG distribution parameters such as mean and standard deviation, using Euclidean distance.

2.3. LLMs for Numeric Related Search

Recent works have shown that generative pretrained LLMs have the key advantage that they can solve a wide variety of tasks including information extraction over tables [17]. Further, several methods have been developed to leverage LLMs and are useful for performing a wide variety of tasks, including entity resolution [9], tabular understanding [18, 19], and semantic annotation and joinability [8]. Considering their capabilities, LLMs can be used as another source of external knowledge to determine table relatedness. However, LLMs, like KGs and word embeddings, cannot directly be used to determine table relatedness for purely numeric tables because language models are likely not trained on examples of the semantics of numbers. Therefore, one key challenge is to devise a method of using LLMs that will enhance numeric semantic labeling and assess table relatedness.

2.4. Problem Statement

We consider the enterprise setting where an analyst has purely numeric input table I and wishes to find purely numeric tables related to I in data lake L. The enterprise has an EKG capturing all information collected by the enterprise. We consider the case where the tables in L are known to contain information in the EKG, but *it is unknown which KG labels describe each table.*

Formally, we assume that each table t is described by an EKG class C_t, each row of table t, r_t, is an EKG *entity* e_t that is an instance of class C_t, and each column is a numeric EKG *property* of one or more instances of C_t, p_t.

Then, given input table I, we want to find all tables $t \in L$ such that (i) I and t are *strongly-semantic joinable*, or (ii) I and t are *table-related.*

Definition 1: Strongly-semantic joinable There exists some join key column of I and some join key column of

USER: Consider the following sample of a list of values: ... Would you consider this list sequential? Begin your answer with YES or NO.

MODEL: NO, the sample list you've given is not sequential according to this definition...

Figure 4: NumSearchLLM's Type Verification Prompt

t such that $I \bowtie t$ can be taken, and for each row of $I \bowtie t$ consisting of pairs of rows from original tables I and t, (r_I, r_t), the EKG contains the same path between the EKG entities corresponding to r_I and r_t (that is, the join aligns pairs of entities such that all pairs of entities are related in the same way). *Definition 2: Table-related* I and t are *table-related* if there is an EKG path between the classes representing each table, C_I and C_t.

While the above definitions precisely describe the types of relatedness in this paper, LLMs allow us to relax these definitions: a path between entities/tables does not have to exist in the EKG for the LLM to determine they are related, and the LLM can describe classes and entities more precisely than the EKG. This allows us to discover a wider variety of joinable/table-related tables.

3. NumSearchLLM

To solve the above problem, we propose *NumSearchLLM*, an approach integrating LLMs and the EKG to recover the schemas of purely numeric tables, and then perform relatedness search. The approach leverages LLMs to (i) improve numeric typing and (ii) detect incorrect semantic labels among labels in a table schema and suggest alternatives. We then provide the LLM with semantic labels and use it to perform joinability and table relatedness.

The main steps of our NumSearchLLM approach are: *i*) Assigning Numeric Types, *ii*) Mapping types to semantic labels, and *iii*) Related table and joinability search. We describe how EKGs and LLMs are integrated to perform each of these steps in detail below.

3.1. Assigning Numeric Types to Columns

Given a collection of numeric values for a column, c, we use TTLA's rule-based heuristics [7] and the LLM to find the numeric type of c. For example, one such rule is that the numeric type "categorical" is assigned if DISTINCT VALUES IN C $< \sqrt{|c|}$ [7].

Given c and its type assigned by the heuristic y_c, we enhance the heuristics with LLMs for numeric typing by using an in-context learning step followed by the type verification step, followed by a type inference step if needed. The prompt for the in-context learning step defines y_c and includes an example list of numbers.

The in-context learning prompt is the following: *We define the following numeric type [definition of y_c]. Here is an example [example list].* The type verification step prompt is shown in Figure 4 (y_c is "sequential" here). If the type verification step returns "no", then we use a type inference prompt to choose a different numeric type. The type inference prompt is the following: *Consider the following alternate numeric types: [other definitions]. Of these, which do you think is most likely for this set of numbers?* If the heuristic and LLM types are identical then NumSearchLLM returns that type. If the inferred types differ, and the type inferred by LLM is not "categorical" then NumSearchLLM returns the type inferred by LLM. If the types disagree and LLM assigned type is "categorical", then since the LLM is unable to understand the numeric thresholds, NumSearchLLM does not rely on its answer and instead returns the type assigned by the heuristics.

3.2. Semantic Labeling of Columns

Given the numeric types of each column, we initially assign EKG semantic labels and validate them using LLMs to obtain the column semantic label (example below). The generation of this label involves reaching an agreement between the labels assigned using EKGs and LLMs and may involve a pre-defined number of iterations between the EKGs and LLMs to reach agreement.

USER: Consider the following table schema, represented using DBpedia labels: dbp:bedCount, dbp:votes,[...]. What might this table represent? Are there labels in the schema that seem out of place compared to the others?

MODEL: Yes, the label dbp:bedCount appears to be out of place[...].

In order to assign EKG semantic labels, the numeric properties in EKGs are first pre-processed to produce clustering models containing their distribution parameters.

Preprocessing the EKG. We find all properties P whose datatype is numeric according to the EKG backend. For each property $p \in P$, we search for nodes in the EKG that have a collection of values for these properties (e.g., node n has $p = \{4, 5\}$). We concatenate all collections of values from all nodes into one collection, S_p. This set of values is the distribution of p (e.g., the distribution of ages, heights, annual riderships, etc.). We use our method for numeric type assignment involving heuristics and LLMs on the distribution of p to determine the numeric type of p (e.g., counts, sequential, categorical, or other). Given the numeric type of p, we compute the appropriate distribution parameters, e.g., number of unique values and value histogram for categorical type, mean and standard deviation for counts etc. for that property. Then, for each numeric type, we cluster all property parameters of properties with that type in a fuzzy c-means clustering model. Each cluster is the vector of distribution parameters for one property (e.g., <mean, standard deviation>).

Algorithm 1 clusterEKG
```
1:  Input: EKG, LLM
2:  CLUSTERINGMODELS = ∅
3:  for p ∈ EKG.PROPERTIES do
4:      if p.isNumeric() == True then
5:          values = extractPropertyValues(EKG.entities, p)
6:          propertyType = computeType(values, LLM)
7:          distributionParameters = computeParameters(values, propertyType)
8:          clusteringModels[propertyType].cluster(distributionParameters, p)
9:      end if
10: end for
11: return clusteringModels
```

Given the numeric types and clustering models obtained from preprocessing the EKG, for each column of a table: we use its numeric type to select the clustering model containing numeric properties of the same numeric type. Then, we cluster the column's distribution parameters and assign the top EKG label from the subset. After performing semantic labeling on all table's columns, the output is a table schema comprised of EKG labels.

For each table t, NumSearchLLM gives the LLM table t's schema and asks the LLM the following question (paraphrased): *Consider the given DBpedia labels representing a schema[...] Do any of these labels seem out of place? If so, does a substitute for the wrong label from the following list apply? If not, suggest a label from the following alternatives [...].*. The LLM can return a list of labels that are out-of-place, O. In this case, for each label $l \in O$, we choose the top-k alternative labels based on the fuzzy c-means clustering model (k is specified by the user). If the LLM chooses a substitute from the top-k, NumSearchLLM chooses this label as substitute. Otherwise, the LLM can indicate none of the top-k labels apply. In this case, we ask the LLM to generate an alternative label. To avoid the possibility that the LLM hallucinated a label that is not actually in the EKG, NumSearchLLM embeds the LLM-generated labels, embeds the EKG labels, and then finds the most similar EKG label to the LLM-generated label. The most similar EKG label may still be unrelated. To account for this, we repeat the above procedure on the schema with the newly chosen semantic label. Algorithm 2 describes our procedure.

3.3. Related Table Search

Given input table and lake table I and t with semantic labels l_I and l_t respectively, we provide the following

Algorithm 2 semLabel

```
1: Input: clusteringModels, LLM, $t$
2: columnLabels = ∅
3: for column ∈ $t$ do
4:     columnType = computeType(column, LLM)
5:     distributionParameters = computeParameters(column, columnType)
6:     columnLabel = clusteringModel[columnType].cluster(distributionParameters)
7:     columnLabels.add(columnLabel
8: end for
9: correctedLabels = verifyLabels(LLM, columnLabels)
10: return correctedLabels
```

prompts for table relatedness and then joinability: (i) table relatedness: *Is it likely that these tables are related?* (ii) *I have determined that these tables can be joined, and this join aligns the following pairs of data instances represented in each table: [insert pairs]. Based on these pairs, do you think there is a relationship between the tables?* We show the overall NumSearchLLM algorithm in Algorithm 3.

Algorithm 3 NumSearchLLM

```
1: Input: $I$, $L$, EKG, LLM
2: clusteringModels = clusterEKG(EKG, LLM)
3: Related_Tables = ∅
4: inputSchema = semLabel(clusteringModels, LLM, $I$)
5: for $t \in L$ do
6:     tSchema = semLabel(clusteringModels, LLM, $t$)
7:     isRelated, Relationship = findRelationship(LLM, inputSchema, tSchema, $t$)
8:     if isRelated == True then
9:         Related_Tables.add($t$)
10:    end if
11: end for
12: return Related_Tables
```

3.4. Implementation Details

NumSearchLLM makes use of classification prompts whose answer is either binary (yes/no) or an element of a specified list (e.g., numeric type categories). The LLM must return responses that are parseable with respect to these categories. To ensure this, we suffix all prompts with "Begin your answer with YES or NO.", or "Begin your answer with the chosen category." So far, this has proven effective on ChatGPT as well as LlaMA2.

4. Evaluation

4.1. Setup

Our Benchmark. As there is no existing benchmark to our knowledge for related table search of purely numeric tables, we constructed our own benchmark of 100 tables, consisting of 50 pairs of joinable tables and 20 other pairs of non-joinable but related tables. We plan to make this benchmark available. We constructed pairs of joinable tables using DBpedia classes and their properties. We used a list of DBpedia classes available from Kaggle [20] to construct input and join table pairs. Each input table is constructed from numeric properties of instances of a DBpedia class (e.g., votes of a politician, which is an instance of class dbo:Politician). The join key is a KG edge from the class instance to another entity.

For each class, we sampled the numeric properties for the class, sampled the class property URIs for join keys, and then retrieved class instances with each numeric property-join key combination in a brute-force fashion, searching for combinations that returned a high number of class instances (more than 50 instances, in our case). The output is a set of tables where each table has a clearly labeled *ground truth* column indicating the DBpedia class instance each row represents, several *property* columns, and one or more *join key* columns. The property cells are numeric properties of the instance, the join key cells are the URIs for KG entities that are property values of the instance. The ground truth column is the DBpedia URI for the class instance. Of the 100 collected tables, we manually compare pairs of table schemas to determine table relatedness based on whether the information captured in both tables is related. For example, a table about U.S. politicians is related to a table about U.S. elections, but not to a table about birthplaces of European soccer players. Note that a pair of related tables can also be joinable, but there are tables in the dataset that are related but not joinable. We do not use the ground truth, but store it to report the accuracy of our approach.

Baselines. We compare against a recent system for finding numeric joins only using EKGs, called NumJoin [21].

4.2. Preliminary Results

In this section, we answer the central research question of this work: *do LLMs improve the precision and recall of joinable and related table search over a data lake for given input tables?* We explore a popular LLM, ChatGPT (gpt-3.5-turbo), and a popular open-source LLM used as a precursor to many other LLMs, LlaMA2.

We first discuss the overall performance of NumSearchLLM. Then, we evaluate the effectiveness of LLMs on each component, namely NumSearchLLM's numeric typing and labeling accuracy, and table relatedness precision and recall assuming the correct labels. Our current results show that LLMs can greatly enhance table relatedness when the correct EKG labels are provided.

Overall Performance of NumSearchLLM. We observe that NumSearchLLM boosts the recall of related table search compared to NumJoin, a state-of-the-art system for discovering numeric joins. Specifically, NumSearchLLM finds 36 joinable table pairs out of 50, of which 20 are joinable, compared to only 10 found by NumJoin (NumJoin Precision: 10 / 10 = 100%, NumJoin Recall: 10 / 50 = 20%, NumSearchLLM Precision: 20 / 36 = 55.56%. NumSearchLLM Recall: 20 / 50 = 40%).

NumJoin determines whether a join exists by finding *entities* with KG properties labeled using semantic labeling, and *links* between those entities. Therefore, when semantic labeling is incorrect due to semantic ambiguity, this greatly limits its ability to correctly disambiguate entities and find existing links between them. NumSearchLLM's enhanced semantic labeling is more accurate, allowing it to discover more joinable tables.

We also report NumSearchLLM's overall performance for table relatedness: precision: 20 / 20 = 100% of pairs of tables, recall: 20 / 70 = 28.57% of pairs of tables. Note that the baseline NumJoin does not perform table relatedness.

LLM Choice: We also report NumSearchLLM results on LlaMA2. LlaMA2's training data involves conversations more than tabular data [22]. Naively using the earlier prompts leads to a much higher number of hallucinations and stock responses such as, "As a language model, I cannot answer this question...". We attempt to improve the performance by adding 5 negative and 5 positive few-shot training examples [23, 24] devised by randomly choosing 5 other DBpedia classes from which to generate joinable table pairs using the data generation process above (one example described below).

USER: Is it likely that these tables are related? [Example Table 1] [Example Table 2]

USER POSING AS MODEL: No, there doesn't appear to be a direct and inherent relationship between the two table schemas you mentioned.

USER: Is it likely that these tables are related? [Real Table 1] [Real Table 2]

MODEL: Yes, these two tables are related.

ChatGPT's performance still far exceeds LlaMA2 (joinability precision: 5 / 92, recall: 5 / 50 = 10%). This indicates that including more training examples of tabular tasks on open LLMs may improve performance.

While these results exceed the state-of-the-art (NumJoin [21]), there is still much room to improve. To isolate the area with the most opportunity, we ablate NumSearchLLM with respect to numeric typing and semantic labeling accuracies, and related table search.

Numeric Typing Accuracy We compare the heuristic rule-based numeric typing used by TTLA and our LLM-enhanced approach. Our metric is the accuracy of the numeric type assigned to each column. We find that the rule-based accuracy is 145 / 358 columns, whereas the LLM-enhanced approach is 195 / 358 columns.

Overall, using the LLM to verify and suggest alternative numeric types can effectively avoid mislabeling count columns as sequential and vice versa. However, LLMs can incorrectly agree with heuristics. For example, foreign key columns with high redundancy are categorical (they should be sequential).

Numeric Semantic Labeling Accuracy We compare TTLA to the LLM-enhanced approach for semantic labeling. Our metric is the accuracy of the DBpedia semantic label assigned to each column. We find that TTLA has accuracy 126 / 358 columns, whereas the LLM-enhanced approach has accuracy 181 / 358.

Overall, using the LLM effectively detects incorrect labels by detecting inconsistencies among semantic labels assigned to columns of the same table. However, the LLM frequently hallucinates alternate labels that are more vague than the true label. For this reason, when NumSearchLLM uses embeddings to map the LLM-generated labels to DBpedia labels, it frequently chooses similar but different labels. For example, the true label may be dbp:PopulationDensity, but NumSearchLLM may ultimately choose dbp:PopulationTotal.

The numeric typing and semantic labeling accuracy values show that LLMs are successfully able to detect and correct semantic ambiguities present in the state-of-the-art semantic labeling system, TTLA.

Related Table Search with Perfect Labels: We have shown that LLMs can improve semantic labeling, but there is still much room for improvement. We now answer whether there is room for LLMs to improve with regards to table relatedness. We achieve this by assuming all tables have been labeled with the correct semantic labels, and then running only the related table search portion of NumSearchLLM. We find that joinable table search has precision 45 / 45 = 100%, and recall 45 / 50 = 90%. Based on this benchmark this suggests that one improvement lies in using LLMs for numeric semantic labeling while improving related table search appears difficult. Additional benchmarks with uncommon semantic labels not well-known to LLMs are likely to make related table search more challenging as well. Other opportunities include scaling NumSearchLLM (LLM inferences are max 1.5s per prompt, 250 tokens, and 300 prompts) and varying LLM prompts to enable precision/recall tuning.

References

[1] N. Chepurko, R. Marcus, E. Zgraggen, R. C. Fernandez, T. Kraska, D. Karger, Arda: Automatic relational data augmentation for machine learning, Proc. VLDB Endow. 13 (2020) 1373–1387. URL: https://doi.org/10.14778/3397230.3397235. doi:10.14778/3397230.3397235.

[2] Y. Dong, K. Takeoka, C. Xiao, M. Oyamada, Efficient joinable table discovery in data lakes: A high-dimensional similarity-based approach, CoRR abs/2010.13273 (2020). URL: https://arxiv.org/abs/2010.13273. arXiv:2010.13273.

[3] Y. Dong, C. Xiao, T. Nozawa, M. Enomoto, M. Oyamada, Deepjoin: Joinable table discovery with pre-trained language models, 2022. URL: https://arxiv.org/abs/2212.07588. doi:10.48550/ARXIV.2212.07588.

[4] G. Fan, J. Wang, Y. Li, D. Zhang, R. Miller, Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning, 2022. URL: https://arxiv.org/abs/2210.01922. doi:10.48550/ARXIV.2210.01922.

[5] R. Castro Fernandez, E. Mansour, A. A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, N. Tang, Seeping semantics: Linking datasets using word embeddings for data discovery, in: 2018 IEEE 34th International Conference on Data Engineering (ICDE), 2018, pp. 989–1000. doi:10.1109/ICDE.2018.00093.

[6] M. Wang, L. Qiu, X. Wang, A survey on knowledge graph embeddings for link prediction, Symmetry 13 (2021). URL: https://www.mdpi.com/2073-8994/13/3/485. doi:10.3390/sym13030485.

[7] C. Faron, C. Ghidini, A. Alobaid, E. Kacprzak, O. Corcho, C. Faron, C. Ghidini, Typology-based semantic labeling of numeric tabular data, Semant. Web 12 (2021) 5–20. URL: https://doi.org/10.3233/SW-200397. doi:10.3233/SW-200397.

[8] M. Kayali, A. Lykov, I. Fountalis, N. Vasiloglou, D. Olteanu, D. Suciu, Chorus: Foundation models for unified data discovery and exploration, 2023. arXiv:2306.09610.

[9] R. Peeters, C. Bizer, Using chatgpt for entity matching, 2023. arXiv:2305.03423.

[10] Z. Huang, P. Subramaniam, R. C. Fernandez, E. Wu, Kitana: Efficient data augmentation search for automl, 2023. arXiv:2305.10419.

[11] resources.data.gov, Business case for open data, 2023. URL: https://resources.data.gov/resources/open-data/.

[12] F. Nargesian, E. Zhu, K. Q. Pu, R. J. Miller, Table union search on open data, Proc. VLDB Endow. 11 (2018) 813–825. URL: https://doi.org/10.14778/3192965.3192973. doi:10.14778/3192965.3192973.

[13] E. Zhu, F. Nargesian, K. Q. Pu, R. J. Miller, Lsh ensemble: Internet-scale domain search, Proc. VLDB Endow. 9 (2016) 1185–1196. URL: https://doi.org/10.14778/2994509.2994534. doi:10.14778/2994509.2994534.

[14] E. Zhu, D. Deng, F. Nargesian, 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 '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 847–864. URL: https://doi.org/10.1145/3299869.3300065. doi:10.1145/3299869.3300065.

[15] A. Khatiwada, G. Fan, R. Shraga, Z. Chen, W. Gatterbauer, R. J. Miller, M. Riedewald, Santos: Relationship-based semantic table union search, 2022. URL: https://arxiv.org/abs/2209.13589. doi:10.48550/ARXIV.2209.13589.

[16] A. Bogatu, A. A. A. Fernandes, N. W. Paton, N. Konstantinou, Dataset discovery in data lakes, in: 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 709–720. doi:10.1109/ICDE48307.2020.00067.

[17] T. Brown, B. Mann, et al., Language models are few-shot learners, in: Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[18] J. Jiang, K. Zhou, Z. Dong, K. Ye, W. X. Zhao, J.-R. Wen, Structgpt: A general framework for large language model to reason over structured data, 2023. arXiv:2305.09645.

[19] Y. Sui, M. Zhou, M. Zhou, S. Han, D. Zhang, Evaluating and enhancing structural understanding capabilities of large language models on tables via input designs, 2023. arXiv:2305.13062.

[20] D. Ofer, Dbpedia classes: Hierarchical taxonomy of wikipedia article classes, 2019. URL: https://www.kaggle.com/datasets/danofer/dbpedia-classes.

[21] IBM, Numjoin: Discovering numeric joinable tables with semantically related columns, 2023. URL: https://tinyurl.com/u2upae2j.

[22] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, 2023. arXiv:2302.13971.

[23] S. M. Xie, A. Raghunathan, P. Liang, T. Ma, An explanation of in-context learning as implicit bayesian inference, in: International Conference on Learning Representations, 2022. URL: https://openreview.net/forum?id=RdJVFCHjUMI.

[24] S. Min, X. Lyu, A. Holtzman, M. Artetxe, M. Lewis,

H. Hajishirzi, L. Zettlemoyer, Rethinking the role of demonstrations: What makes in-context learning work?, arXiv preprint arXiv:2202.12837 (2022).