Scalable Table-to-Knowledge Graph Matching from Metadata using LLMs

Nathan Vandemoortele[1,*], Bram Steenwinckel[1], Sofie Van Hoecke[1] & Femke Ongenae[1]

[1]*Ghent University - imec, Technologiepark-Zwijnaarde 122, Gent, 9052, Belgium*

Abstract

Addressing the challenge of interoperability when integrating and interpreting large datasets from diverse sources is essential for businesses aiming to make informed, data-driven decisions. Therefore, the 2024 Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab) focuses on using metadata, such as column names, to map tables to semantic concepts within standardized vocabularies or Knowledge Graphs (KGs). The challenge involves mapping two datasets, one to the DBpedia ontology and the other to a custom vocabulary.

Our approach begins with applying Retrieval-Augmented Generation (RAG) for a broad search of relevant matches, ensuring scalability. We then refine these matches using a Large Language Model (LLM) with Chain-of-Thought (CoT) prompting and Self-Consistency (SC). Finally, we combine the results using Reciprocal Rank Fusion (RRF) to obtain a final ranking of the matches.

This method achieves hit rates of 62% (top 1) and 82% (top 5) for the first dataset, and 84% (top 1) and 98% (top 5) for the second. The LLM's strong semantic understanding and extensive knowledge base provide significant advantages over traditional human labeling, which is often laborious and time-consuming. Furthermore, the LLM's zero-shot capability removes the need for additional task-specific training data, making this solution applicable across various domains.

Despite limitations like computational costs and the need for well-defined concepts in the ontology or vocabulary, our approach remains cost-effective compared to extensive human labeling. Moreover, we leave room to trade performance for scalability if needed, pending further research.

Keywords

data linkage, knowledge graphs, large language models, retrieval augmented generation, zero-shot learning, metadata

1. Introduction

In today's data-driven world, many organizations and companies frequently face the challenge of integrating and interpreting large datasets from various sources to streamline their business operations and their ability to make data-driven decisions [1]. For example, consider a multinational corporation that has recently acquired several smaller companies, each with a different data management system. Merging such heterogeneous data sources while maintaining the intrinsic semantic meaning of the available information is a difficult task for humans.

SemTab'24: Semantic Web Challenge on Tabular Data to Knowledge Graph Matching 2024, co-located with the 23rd International Semantic Web Conference (ISWC), November 11-15, 2024, Baltimore, USA

*Corresponding author.

nathan.vandemoortele@ugent.be (N. Vandemoortele); bram.steenwinckel@ugent.be (B. Steenwinckel); sofie.vanhoecke@ugent.be (S. Van Hoecke); femke.ongenae@ugent.be (F. Ongenae)

0009-0009-0118-3519 (N. Vandemoortele); 0000-0002-3488-2334 (B. Steenwinckel); 0000-0002-7865-6793 (S. Van Hoecke); 0000-0003-2529-5477 (F. Ongenae)

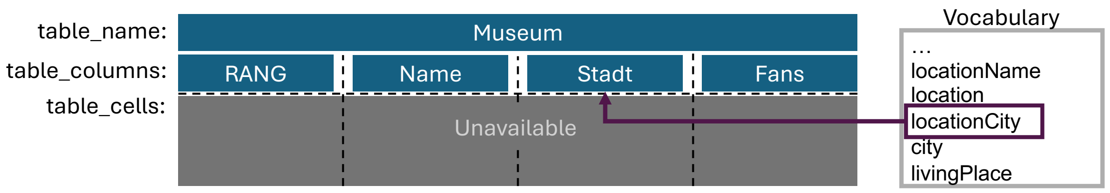


Figure 1: Matching table metadata only, i.e., table and column names, to a vocabulary or ontology without any access to table data. In this example, the column "Stadt" of the "Museum" table is matched to the "locationCity" property within the given vocabulary.

One possible solution to this problem involves matching tables to standardized vocabularies or Knowledge Graphs (KGs) [2, 3]. This process, known as semantic table linking or table matching, aims to align data within tables with predefined concepts. The "Semantic Web Challenge on Tabular Data to Knowledge Graph Matching (SemTab)[1]" has been hosted for several years at the International Semantic Web Conference (ISWC) to stimulate the creation of such automated semantic annotation systems. The goal is to automate the assignment of a DBpedia or Wikidata entity to a whole column based on table data, assign a DBpedia or Wikidata entity to different cells, infer relations between different columns where possible, or assign a KG class to an entire table. One of the objectives of this year's challenge[2] is to map columns to semantic concepts in a vocabulary or KG based solely on the information in the header of the columns, i.e. column and table names. The objective is illustrated in Figure 1.

This automated mapping presents considerable challenges, as the true meaning of the table relies on interpreting the data within the cells.

- For instance, a column named "ID" or "Value" provides little to no information about its semantic meaning to help linking it to a concept in a KG.
- Many column names are ambiguous because they can have multiple potential meanings depending on the context. For example, a column named "Date" could refer to, amongst others, a birth date, transaction date, or event date.
- Tables can employ different naming conventions, languages, abbreviations, or terminologies. For instance, one dataset might use "CustID" while another uses "CustomerID" to represent the same concept.
- As organizations are dynamic, new tables can be added and existing ones modified. This dynamic nature requires continuous updates and maintenance of data structures, which is resource-intensive if done manually.
- As organizations accumulate new tables, the scalability of semantic mapping becomes a concern. Mapping metadata to vocabularies or KGs for potentially thousands of tables is not humanly feasible, necessitating automated and scalable solutions.

We hypothesize that Large Language Models (LLMs) offer a promising solution to the challenges of semantic mapping when relying solely on metadata. Even in the absence of actual

[1]https://www.cs.ox.ac.uk/isg/challenges/sem-tab/
[2]https://sem-tab-challenge.github.io/2024/tracks/metadata-to-kg-track.html

data content (e.g., due to restricted access), LLMs can leverage their intrinsic knowledge and reasoning capabilities to tackle automated mapping problems. In this paper, we investigate the zero-shot capabilities of GPT-4o[3], a state-of-the-art LLM at the time of writing, for the "Table Metadata to KG" task in the 2024 "Semantic Web Challenge on Tabular Data to Knowledge Graph Matching" challenge. Zero-shot learning refers to the ability to transfer to an unseen problem without new task-specific training data. To remain cost-effective, we adapt an advanced Retrieval Augmented Generation (RAG) [4] solution that retrieves only the most relevant information within the vocabulary or KG. This ensures that the generated responses are contextually accurate while maintaining scalability. In this paper, we present the following key contributions:

- While RAG solutions exist in the context of ontology matching, we are the first to incorporate and apply (LLM-based) rerankers in this context.
- Our main innovation comprises a novel completion stage, where we uniquely combine two prompting techniques, Chain-of-Thought and Self-Consistency, with Reciprocal Rerank Fusion to find the best mappings.
- Our methodology efficiently scales to handle large and diverse datasets in a zero-shot manner.

The remainder of this paper is as follows. We first discuss related work in Section 2. Next, we present the table datasets and vocabularies provided by the SemTab organizers in Section 3, followed by a detailed explanation of our methodology in Section 4. In Section 5, we present the results. We conclude the paper in Section 6 and suggest additional paths for future work.

2. Related Work

Tasks regarding table topic annotation and column type annotation are closely related to the general problem of semantic table understanding [5]. The SemTab challenge [6] has always been formulated as an unsupervised table annotation problem. Systems like MTab [7], CSV2KG [8], and DAGOBAH [9], which participated in the SemTab challenge in previous years, tackle three main tasks in KG matching: creating cell entity annotations (CEA), column type annotation (CTA), and paired column property annotations (CPA). These pipelines typically begin by linking individual cell contents, such as the various column entries, to ontology entities, and then predicting the most likely column type based on these linkages. To do so, these systems need available cell values to be mapped to KG entities before column types can be identified, which is not applicable in this context as here the actual table data is not available.

Given that we can only rely on metadata information for the task addressed in this paper, the related work methodologies that are applicable align more closely with ontology alignment techniques. These can range from traditional string similarity methods [10] to semantic similarity measures utilizing resources like WordNet [11]. WordNet is, however, designed for general purposes and may not include specialized terms or jargon used in specific, custom vocabularies. More recent work includes vector representations for semantic similarity, using, e.g., word2vec

[3]https://platform.openai.com/docs/models

to get promising results in semantic textual similarity tasks [12]. Nowadays, also LLM-based approaches have achieved traction for tasks like ontology alignment [13], entity matching [14] and subject annotation [15]. Since LLMs are trained on large data corpora, they can identify complex relations and patterns between different objects in natural language, leading to more accurate matching. LLMs demonstrate their effectiveness to improve metadata matching performance, either through additional fine-tuning on table-based tasks [16] or without fine-tuning [3], and show how contextual information, such as the dataset description, can be incorporated in an LLM prompt to match a vocabulary to a table's column topics [17]. This last task is similar to the one we are trying to solve in this paper. However, providing the full vocabulary as context within each prompt would not be cost-effective at scale. To this end, RAG pipelines have been proposed [18, 19], though they have not yet been applied in this specific context. Furthermore, advanced prompt optimization strategies remain largely unexplored.

3. Metadata Datasets

Two datasets were provided in the challenge, which both contained a vocabulary on the one hand and table metadata (i.e., table and column names) for multiple tables without the actual cell data on the other hand:

- **Dataset 1** consists of metadata from a selected set of HTML-based tables that can be found on the web, and needs to get mapped to a subset of the DBpedia ontology[4]. The goal is to map each table column to a corresponding ontology property. While the ontology is hierarchically structured, it is treated as a simple vocabulary of properties, each defined by labels and short descriptions, with no consideration of the relationships between them. In total, metadata for 77 web tables were provided, of which 141 columns needed to be mapped. The vocabulary consists of 2881 different properties.
- **Dataset 2** contains a set of tables, available in a database accessible for public use. The metadata of these tables needs to be mapped to a custom vocabulary with clear descriptions. However, the vocabulary is not derived from an existing publicly available KG. The goal is, again, to map each table column to one vocabulary item. This dataset contains metadata for 75 tables, of which 1181 columns needed to be mapped, and has 1181 items in the vocabulary.

4. Methodology

Our architecture adapts an advanced RAG solution with a novel LLM-based completion stage. RAG is a technique used to incorporate new knowledge into LLMs through in-context learning (ICL), such as a vocabulary or a knowledge graph (KG). It promises a scalable approach by retrieving only relevant information, thereby significantly driving down costs. ICL enables LLMs to learn from examples within the context window without altering their weights.

Our solution's two-stage pipeline is schematically visualized in Figure 2 and consists of a retrieval stage and a completion stage. In this section, we explain each component of the

[4]http://dbpedia.org/ontology/

pipeline step by step. You can find the prompt templates used in the "appendix" directory of our GitHub repository: https://github.com/predict-idlab/Meta2Concept

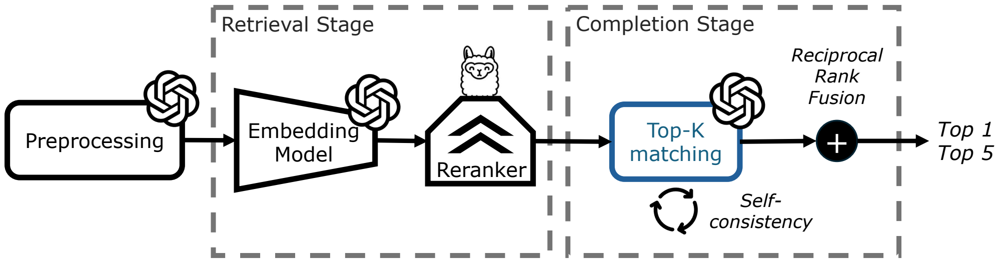


Figure 2: Overview of the two-stage methodology composed of a retrieval and completion stage after an initial preprocessing step. The hexagonal knot and lama icon refer to the OpenAI and Llama models, respectively.

4.1. Preprocessing

Both tables and vocabularies can vary in size, language, and quality. For example, the descriptions of the provided vocabulary items can be rather short (see Section 3), or the provided table metadata can contains abbreviations or be multilingual (see introduction). To address these variations, a preprocessing step is applied to both the vocabulary and metadata as a first step as detailed in the following subsections.

4.1.1. Enriching of vocabulary

An item within a vocabulary represents the information of a single concept or property, e.g., a DBpedia property label and its description for the first dataset. We ask GPT-4o to translate the item labels to English, fix grammar mistakes, convert proper names to their type (e.g., Harelbeke becomes ⟶ City (Harelbeke)) and write acronyms in full. Closed-source LLMs like GPT-4o enrich the embedding model in the next stage, as they are more knowledgeable and frequently updated with the latest information. We process the vocabulary in chunks of 100 items per prompt, which is overall inexpensive and enables the use of GPT-4o. In most cases, this results in property labels becoming self-explanatory. Below is an example of vocabulary enrichment using GPT-4o:

```
Input: prompt(label='ept itm')
Output: 'European Poker Tour (EPT) In The Money (ITM)'
```

This helps generate more accurate embeddings for the next stage. In this case, you might also find matches closer to poker, tours, or money.

Item descriptions are more fine-grained representations of the label in semantic space. However, for these descriptions to be effective as embeddings, they must be specific and not open to interpretation. For example, "season" is simply described as "season", however "season" can refer to the yearly seasons but also television seasons, or sports seasons. Attempting to assign one interpretation without knowing the ground truth could skew the embeddings vector in a specific direction, whereas it might be preferable to leave such terms undefined at this stage. This problem is specific to Dataset 1, hence, we focus on enriching and utilizing only its labels. For Dataset 2, we use and retain the original descriptions, as they are deemed to be of high quality.

4.1.2. Enriching of table metadata

The metadata consists of a table name and column names, one or more of which need to be mapped. We ask GPT-4o to transform the column and table name into a sentence in the form:

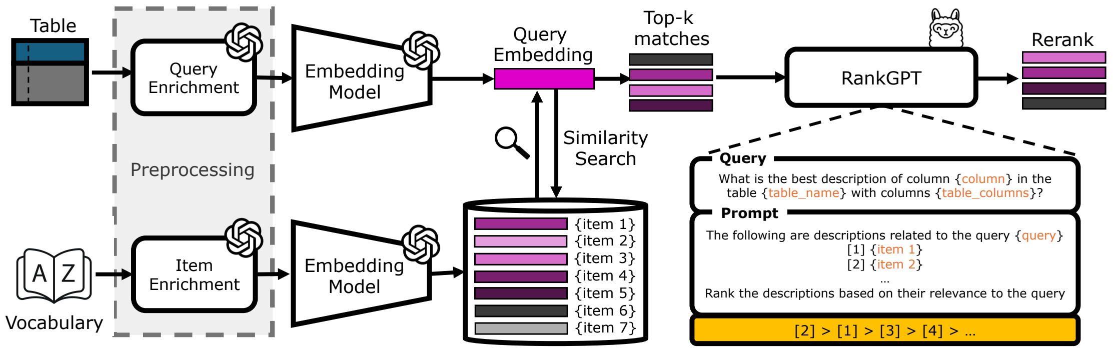
```
{column} {relation} {table_name}
```

No other information about the other column names in the table is included. Furthermore, we ask GPT-4o to translate words to English, fix grammar mistakes, convert names to their type and write acronyms in full, similar to what we did with the vocabulary labels. The following is an example of table metadata enrichment using GPT-4o:

```
Input: prompt(column='Height (m)', table_name='Mountain')
Output: 'Height in meters of a mountain'
```

4.2. Retrieval Stage

Figure 3 illustrates the Retrieval Stage. It consists of an advanced RAG solution including an embedding step and a reranking step, both of which will be further explained in the subsequent subsections.



What is the best description of column {column} in the table {table_name} with columns {table_columns}?

The following are descriptions related to the query {query}
[1] {item 1}
[2] {item 2}
...
Rank the descriptions based on their relevance to the query

[2] > [1] > [3] > [4] > ...

Figure 3: Detailed overview of the retrieval stage: Vocabulary items are enriched with an LLM and embedded with an embedding model, then stored. Table metadata is embedded similarly to create query embeddings. A similarity search retrieves the top-k matches for the query embedding, which are reranked by RankGPT based on relevance.

4.2.1. Embedding model

Dense embeddings provide a fast way to retrieve relevant documents at scale. The main idea is to perform a coarse search for the relevant items within the vocabulary or KG at a low cost, which are then refined further downstream.

The process begins with indexing the enriched vocabulary, where it is chunked by item and embedded using a dense embedding model, as shown in Figure 3. We use text-embedding-3-large (OpenAI)[5], which converts sentences into 3072-dimensional vectors. These embeddings are stored and can be reused for comparison with each new query. Next, the enriched table metadata is embedded similarly to generate query embeddings. We then perform a similarity search, comparing the embedded query with each item in the vocabulary using cosine similarity, which can be expressed as:

$$cosine\ similarity = dot(Item_{vector}, Query_{vector})$$

Since the vectors are normalized, there is no need to divide by their lengths.

Finally, we rank the items according to the cosine similarity and retrieve the top 150 matches to be reranked. This number was chosen based on cost considerations. Ideally, this number should be empirically validated using a validation set; however, in this challenge, we were provided with only 10 samples for Dataset 1 and none for Dataset 2. Table 2 indicates the number is sufficient for Dataset 1 with a 100% hit rate but may be inadequate for Dataset 2, which has a 95% hit rate. This likely stems from a larger vocabulary and a higher likelihood of encountering similar items. We believe there is a probable relationship between the size of the dataset and the optimal number of retrieval candidates, which we have yet to evaluate thoroughly.

4.2.2. Reranker

The top 150 candidates identified in the previous step show high semantic similarity with the given table query, but their order is highly dependent on the embedding model. Embedding models have several drawbacks: both the vocabulary item and query are embedded independently, resulting in some loss of information. They are also often trained for specific tasks and therefore require further fine-tuning to achieve the best results. This goes against our goal of making the pipeline generically reusable. Furthermore, they do not have the same knowledge or reasoning capabilities of an LLM.

This is where LLM-based rerankers come in. These rerankers find logical connections between several "passages" (our vocabulary items) and a query simultaneously, albeit at a higher computational cost, which is why we only use it on the top-ranked results in order to improve scalability. We use RankGPT [20], a library that provides a prompt template to transform LLMs into reranking agents. We selected Llama-3-70B (Meta), an open-weight LLM, to ensure cost-effectiveness given the size of our datasets. We utilized RankGPT's permutation generation with a sliding window (step size of 10 and window size of 20) to limit the number of candidates needing reranking in each prompt. Moreover, this measure decreases the prompt size and improves performance by mitigating context length sensitivity. RankGPT's sliding window is applied from back to front, allowing items to be promoted in rank but not demoted outside the window size. The reranking of vocabulary items is determined by their relevance to the following new query:

[5]https://platform.openai.com/docs/models/embeddings

```
Query:
        What is the best description of column {column} in
        the table '{table_name}' with columns {table_columns}?
```

Notice that this query now uses all the metadata of the table. Similarly, items now contain all the information of the original vocabulary, i.e. the labels with their descriptions, allowing for a more fine-grained ranking. We no longer use the information obtained during enrichment because the clean-up step in preprocessing could introduce errors that may confuse the LLM. However, for Dataset 1, we found the descriptions to be poor, which is why we included the domain and range based on the properties provided by DBpedia[6]. If the domain is an "owl:Thing" or the range is a datatype, the field is left empty. This results in the following items:

```
Item:
        Dataset 1: [<domain>, <property>, <range>] <description>
        Dataset 2: <label>, <description>
```

After reranking the items, we retrieve the top 30 candidates for further processing. This number was chosen based on similar limitations described in the initial ranking.

4.3. Completion Stage

Figure 4 illustrates the Completion Stage, which aims to make a final ranking of the retrieved candidates and will be further explained in detail in the following subsections.

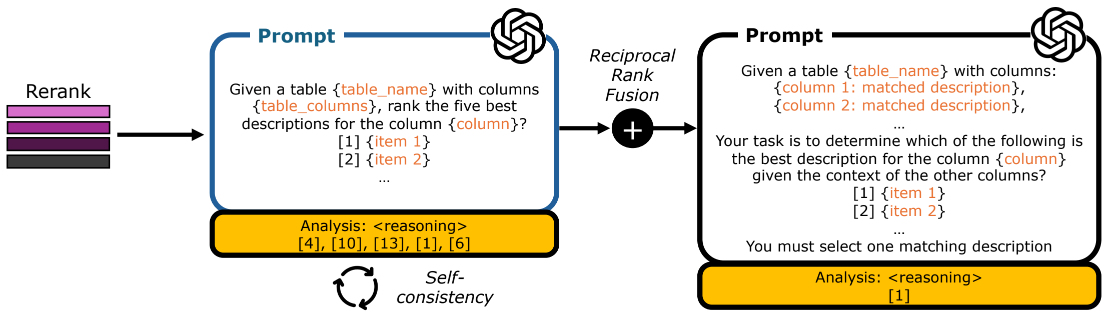

Rerank
Prompt
Given a table {table_name} with columns {table_columns}, rank the five best descriptions for the column {column}?
[1] {item 1}
[2] {item 2}
...
Analysis: <reasoning>
[4], [10], [13], [1], [6]
Self-consistency
Reciprocal Rank Fusion
Prompt
Given a table {table_name} with columns:
{column 1: matched description},
{column 2: matched description},
...
Your task is to determine which of the following is the best description for the column {column} given the context of the other columns?
[1] {item 1}
[2] {item 2}
...
You must select one matching description
Analysis: <reasoning>
[1]

Figure 4: Detailed overview of the completion stage: An LLM is prompted using CoT-SC to retrieve multiple rankings, which are then fused using RRF. The process includes a second prompt to ensure column descriptions are consistent across the entire table.

4.3.1. Top-k matching

To obtain the final top 1 and top 5 matches, which is the goal of the challenge, we ask GPT-4o in zero-shot setting to rank the top 5 best vocabulary items out of the top 30 candidates. We

[6]e.g. https://mappings.dbpedia.org/index.php/OntologyProperty:ReleaseDate

selected GPT-4o (OpenAI) as the LLM of choice as it is the most capable model based on MMLU and Chatbot Arena scores, according to lmsys.org[7].

We use Chain-of-Thought (CoT) [21, 22] with Self-Consistency (SC) [23] to retrieve multiple rankings, which are then fused using Reciprocal Rank Fusion (RRF) [24]. CoT-SC is a well-known technique for improving classification task performance by sampling multiple reasoning paths (n=10), i.e. repetitively performing the same task, with a high temperature (T=0.7), i.e. encouraging exploration, and taking the consensus of the answers from an LLM. During each resampling, the vocabulary descriptions are randomly shuffled to avoid the 'lost in the middle' problem, where the LLM tends to be biased toward the beginning or end of a document. We instruct the LLM to generate structured output in free text format and extract the answers using regular expressions.

RRF is a method in information retrieval for combining the results of multiple ranked lists. The core idea behind RRF is to leverage the rankings provided by different retrieval models and fuse them into a single, more robust ranking. The formula for RRF can be expressed as follows, where d is a document in a set of documents D and r is the rank in a set of rankings R:

$$RRFscore_{d \in D} = \sum_{r \in R} \frac{1}{k + r(d)}, \text{ where k=0.01 (a tuned parameter)}$$

4.3.2. Table matching

The methodology is further extended to improve the results of the second dataset. As visualized in Figure 4, the completion stage includes a second prompt containing the descriptions of the columns of the entire table identified earlier. The aim is to ensure that the column descriptions are consistent with one another, which potentially reveals a pattern. For instance, it is clear from the other descriptions that the table relates to a visitor visiting a webpage, this knowledge therefore reduces ambiguity when choosing between vocabulary labels "UserID" or "VisitorID" when mapping the column "ID".

Although the same CoT-SC process can be applied here, we limit it to one inference with a low temperature (T=0.0), focusing only on improving the top 1 out of the top 5 descriptions identified in the previous step. This limitation is due to the significantly longer prompt, and we expect the gains to be marginal.

4.4. Non-contextual Methodology

In addition to the two-stage pipeline, we also present a non-contextual method. This approach tests the LLM's ability to reason and recall properties from parametric memory rather than from context. As illustrated in Figure 5, we simply ask GPT-4o to rank the top 5 fine-grained properties from the DBpedia ontology and filter out those not included in the given subset. It has been shown that top LLM models like ChatGPT and GPT-4 are pretrained on DBpedia data [25]. Thus, we expect the non-contextual method to perform very well on Dataset 1. However, this approach is not usable on unseen vocabularies or KGs, such as those in Dataset 2.

We prompted GPT-4o using SC (n=20) without CoT and combined the results with RRF.

[7]https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

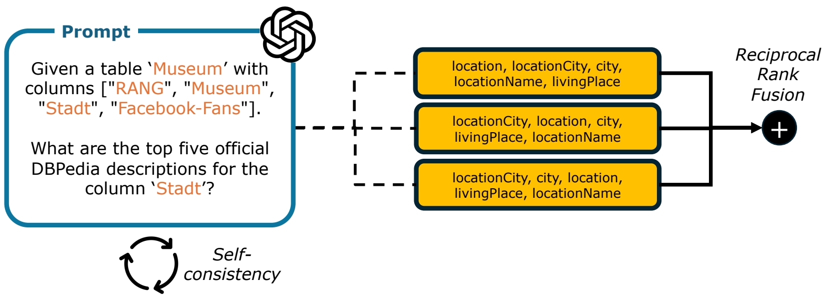

Prompt
Given a table 'Museum' with columns ["RANG", "Museum", "Stadt", "Facebook-Fans"].
What are the top five official DBPedia descriptions for the column 'Stadt'?
Self-consistency
location, locationCity, city, locationName, livingPlace
locationCity, location, city, livingPlace, locationName
locationCity, city, location, livingPlace, locationName
Reciprocal Rank Fusion

Figure 5: Detailed overview of the non-contextual methodology: An LLM is prompted using SC to retrieve multiple rankings, which are then fused using RRF.

5. Results

Both datasets were evaluated using the Hit@k metric for 2 different k: 1 and 5. The Hit@k metric indicates whether the true item appears in the first k items of the sorted rank list. This can be expressed as:

$$\text{Hit@k} = \max_{i=1..k} \begin{cases} 1, & r_i \in T \\ 0, & \text{otherwise} \end{cases}$$

in which, r_i is the item ranked at position i, and T is the set of items that are in the test set. We take the average of this metric across the entire dataset, which is also referred to as the "hit rate".

The results of both the non-contextual method and our approach are summarized in Table 1. Note that we did not evaluate the non-contextual method for Dataset 2, as it is not publicly available and therefore should not have been included in the LLM's training data. Additionally, we provide the scores from the embedding model (text-embedding-3-large), during the embedding phase of our method, as a baseline for comparison.

Table 1
Comparison of the baseline (text-embedding-3-large), contextual method, and our methodology for Dataset 1 & 2.

	Dataset 1		**Dataset 2**	
	Hit@1	**Hit@5**	**Hit@1**	**Hit@5**
Baseline	33%	52%	41%	72%
Non-contextual Method	**75%**	**92%**	-	-
Our Method	62%	82%	**84%**	**98%**

In Table 2, we provide a breakdown of performance across the different phases of the pipeline for Datasets 1 & 2. The "# Items" row indicates the number of vocabulary items remaining at the end of each phase. This also determines the metrics that we can evaluate.

Table 2
Performance metrics across different stages for Dataset 1 & 2.

		Prepro-cessing	Retrieval Stage (em-bedding)	Retrieval Stage (rerank)	Completion Stage (top-k matching)	Completion Stage (table matching)
<u>**Dataset 1**</u>	# Items	2881	150	30	5	–
	Hit@150	–	**95%**	–	–	–
	Hit@30	–	75%	**91%**	–	–
	Hit@5	–	52%	59%	**82%**	–
	Hit@1	–	33%	43%	**62%**	–
<u>**Dataset 2**</u>	# Items	1181	150	30	5	1
	Hit@150	–	**100%**	–	–	–
	Hit@30	–	96%	**99%**	–	–
	Hit@5	–	72%	95%	**98%**	–
	Hit@1	–	41%	77%	83%	**84%**

6. Discussion and Future work

Our two-stage pipeline performed well, achieving top 1 and top 5 hit rates of 62% and 82% respectively for the first dataset, and top 1 and top 5 scores of 84% and 98% respectively for the second dataset. LLMs significantly outperform embedding models, achieving increases of 29% and 43% for the top 1 ranking in the first and second datasets, respectively. The non-contextual method achieved the highest score on Dataset 1, confirming that the DBpedia ontology is indeed part of the original training data of the LLM. This suggests that the model formed a better understanding of the properties during pretraining, possibly because they were seen in the context of other documents, and/or is better at reasoning using parametric memory.

The results of our method show a high hit rate within the top 5 candidates, indicating that it retrieves almost all relevant labels. However, the score of the top 1 candidate is lower. Analysis suggests this is partly due to poor label definitions or subjective human preferences. For instance, in Dataset 1, a human labeler might categorize a city as "location", whereas our method might specify "locationCity". Moreover, based on the limited validation dataset, it is apparent that human labelers do not consistently choose the most fine-grained property. This has shifted our problem definition from identifying the "most fine-grained" property to "matching the human labels" to maximize our score. For this reason, we did not explicitly instruct the LLM to find the most fine-grained property.

We believe that the scores we report are generally understated. There is often a lack of consensus among humans regarding the ground truth. Our observations from the limited validation dataset suggest that, in some cases, the LLM's choice may be preferable to the human one. We aim to highlight these discrepancies, although we lack access to the definitive ground truth. If the true objective is to identify the "most fine-grained" property, further improvements could be achieved, such as refining the prompt, adding hierarchical relationships within the KG, etc.

This issue highlights three key advantages over human labelers:

- Humans must be aware of all possible items in the vocabulary or KG, which is a difficult task, whereas LLMs can ingest everything exhaustively.
- Our pipeline provides consistent mapping. While human labelers may have varying preferences, our approach ensures uniform assignments from an independent expert.
- Table metadata is often filled with specialized jargon, LLMs can eliminate the need for separate experts in each field.

We attribute this advantage to the strong semantic understanding of LLMs, specifically their ability to extract meanings from context—i.e., how words appear together in documents. Coupled with a comprehensive knowledge base of memorized facts, which includes concepts of names, places, and acronyms, giving them an edge over traditional embedding methods and, in some respects, even human capabilities.

Another advantage is the ability of LLMs to transfer to unseen tasks in a zero-shot manner. The two-stage pipeline removes the need for additional task-specific training data. In our tests, few-shot approaches offered only marginal improvements, making zero-shot learning a more practical and resource-efficient choice. This zero-shot capability also allows the pipeline to be applied to other datasets. As shown in the second dataset, a well-defined vocabulary suffices, making our solution applicable across other domains.

The use of LLMs is not without limitations. The accuracy of our approach is highly affected by the quality of the descriptions and how well the column names represent the corresponding table data. Some current mistakes stem from poorly defined descriptions or misleading column names. If possible, providing a single row of each table as additional context to the LLM could potentially go a long way to improve clarity and also benefit human labelers.

Table 3
Cost and token usage across different stages for Dataset 2

	# Tokens (in/out)	Cost per 1M tokens (in/out)	Total cost	Model	Inference Platform
Preprocessing	16K/18K	\$5.0/\$15.0	\$0.35	GPT-4o	OpenAI
Retrieval Stage (embedding)	125K	\$0.02	\$0.0025	text-embedding-3-large	OpenAI
Retrieval Stage (rerank)	22.7M/1.4M	\$0.59/\$0.79	\$14.50	Llama-3-70B-Instruct	DeepInfra
Completion Stage (top-k matching)	3.6M/939K	\$5.0/\$15.0	\$32.09	GPT-4o	OpenAI
Completion Stage (table matching)	555K/96K	\$5.0/\$15.0	\$4.22	GPT-4o	OpenAI
ALL	27.0M/2.5M	–	\$51.16	–	–

Cost remains another important factor. Table 3 provides a breakdown of the number of input and output tokens and their associated costs for each phase within our two-stage pipeline for Dataset 2, which has the highest token count of the two datasets. The cost of an LLM is determined by the volume of tokens provided (input) and generated (output), each at different price points. Running our solution for Dataset 2 is estimated to be around \$51 in total, or 4.33¢ per mapping of a column, with costs scaling linearly. This indicates that the function of a retrieval stage in this approach, as opposed to providing all items in the vocabulary or KG to the completion stage directly, is a very cost-efficient measure.

There are, however, opportunities to further reduce the cost of the provided pipeline and research the balance between cost and accuracy. For instance, during the rerank phase, one can further adjust the step and window size or use less expensive models for lower-ranked embeddings. During the completion stage, the self-consistency parameter can also be easily tweaked without making structural changes to the architecture. While these are stopgaps to the current limitations, the cost and performance of foundation models are set to continue to improve in the near future.

Looking forward, future research could further improve our pipeline at various stages. Embedding-based rankings could benefit from a hybrid approach that combines sparse retrieval models with dense models to help identify exact matches, such as names, IDs, and numbers. One could also suggest more fine-grained domain and ranges based on the usage of the property in axioms and data, or even towards the specific task or context at hand. Furthermore, we currently see an opportunity to make LLM-based rerankers more scalable through new algorithms or the aforementioned strategic use of cheaper LLMs. Additionally, their performance can potentially be improved by combining SC with RRF. As we look ahead, new solutions will need to be devised to maintain scalability if the vocabulary changes at test time.

The holy grail is still to embed knowledge into the LLM itself and retrieve the items from parametric memory. The advantages of this approach are revealed by the non-contextual method in this paper, where a single and simple prompt, together with a post-processing operation, results in very accurate table annotations. This solution is, however, still only applicable for DBpedia, as the GPT-4o model has the DBpedia properties stored in its parametric memory. Nonetheless, injecting new knowledge into LLMs through fine-tuning remains a challenging task and recent advancements such as Knowledge-based Model Editing are still in their emerging stage [26].

Acknowledgments

This work was partly funded by the SolidLab Vlaanderen project (Flemish Government, EWI and RRF project VV023/10), the FWO Project FRACTION (Nr. G086822N) and the BOF Project FRACTION (Nr. BOF.24Y.2021.0048.01).

Code availability

The code and additional resources used in this paper are available on GitHub at the following repository: https://github.com/predict-idlab/Meta2Concept

References

[1] R. H. Hariri, E. M. Fredericks, K. M. Bowers, Uncertainty in big data analytics: survey, opportunities, and challenges, Journal of Big Data 6 (2019) 44. doi:10.1186/s40537-019-0206-3.

[2] J. Liu, Y. Chabot, R. Troncy, V.-P. Huynh, T. Labbé, P. Monnin, From tabular data to knowledge graphs: A survey of semantic table interpretation tasks and methods, Journal of Web Semantics 76 (2023) 100761. doi:10.1016/j.websem.2022.100761.

[3] E. Lobo, O. Hassanzadeh, N. Pham, N. Mihindukulasooriya, D. Subramanian, H. Samulowitz, Matching table metadata with business glossaries using large language models, arXiv preprint arXiv:2309.11506 (2023).

[4] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, H. Wang, Retrieval-Augmented Generation for Large Language Models: A Survey, 2024. doi:10.48550/arXiv.2312.10997, arXiv:2312.10997 [cs].

[5] J. Pujara, P. Szekely, H. Sun, M. Chen, From tables to knowledge: Recent advances in table understanding, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 4060–4061.

[6] E. Jiménez-Ruiz, O. Hassanzadeh, V. Efthymiou, J. Chen, K. Srinivas, Semtab 2019: Resources to benchmark tabular data to knowledge graph matching systems, in: The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17, Springer, 2020, pp. 514–530.

[7] P. Nguyen, I. Yamada, N. Kertkeidkachorn, R. Ichise, H. Takeda, Semtab 2021: Tabular data annotation with mtab tool., in: SemTab@ ISWC, 2021, pp. 92–101.

[8] B. Steenwinckel, G. Vandewiele, F. De Turck, F. Ongenae, Csv2kg: Transforming tabular data into semantic knowledge, SemTab, ISWC Challenge (2019).

[9] V.-P. Huynh, J. Liu, Y. Chabot, F. Deuzé, T. Labbé, P. Monnin, R. Troncy, Dagobah: table and graph contexts for efficient semantic annotation of tabular data, in: The 20th International Semantic Web Conference (ISWC 2021), volume 3103, 2021, p. 2.

[10] M. Cheatham, P. Hitzler, String similarity metrics for ontology alignment, in: The Semantic Web–ISWC 2013: 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II 12, Springer, 2013, pp. 294–309.

[11] F. Lin, K. Sandkuhl, A survey of exploiting wordnet in ontology matching, in: IFIP International Conference on Artificial Intelligence in Theory and Practice, Springer, 2008, pp. 341–350.

[12] J. Liao, Y. Huang, H. Wang, M. Li, Matching ontologies with word2vec model based on cosine similarity, in: The International Conference on Artificial Intelligence and Computer Vision, Springer, 2021, pp. 367–374.

[13] Y. He, J. Chen, H. Dong, I. Horrocks, Exploring large language models for ontology alignment, arXiv preprint arXiv:2309.07172 (2023).

[14] R. Peeters, C. Bizer, Entity matching using large language models, arXiv preprint arXiv:2310.11244 (2023).

[15] S. Zhang, M. Wu, X. Zhang, Utilising a large language model to annotate subject metadata: A case study in an australian national research data catalogue, arXiv preprint arXiv:2310.11318 (2023).

[16] T. Zhang, X. Yue, Y. Li, H. Sun, TableLlama: Towards Open Large Generalist Models for Tables, 2024. URL: http://arxiv.org/abs/2311.09206, arXiv:2311.09206.

[17] M. Martorana, T. Kuhn, L. Stork, J. van Ossenbruggen, Text classification of column headers with a controlled vocabulary: leveraging llms for metadata enrichment, arXiv preprint arXiv:2403.00884 (2024).

[18] H. B. Giglou, J. D'Souza, F. Engel, S. Auer, LLMs4OM: Matching Ontologies with Large Language Models, 2024. doi:10.48550/arXiv.2404.10317, arXiv:2404.10317 [cs].

[19] S. Hertling, H. Paulheim, OLaLa: Ontology Matching with Large Language Models, in: Proceedings of the 12th Knowledge Capture Conference 2023, 2023, pp. 131–139. doi:10.1145/3587259.3627571, arXiv:2311.03837 [cs].

[20] W. Sun, L. Yan, X. Ma, P. Ren, D. Yin, Z. Ren, Is chatgpt good at search? investigating large language models as re-ranking agent, ArXiv abs/2304.09542 (2023).

[21] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, 2024, pp. 24824–24837.

[22] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large Language Models are Zero-Shot Reasoners, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 22199–22213. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.

[23] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, D. Zhou, Self-consistency improves chain of thought reasoning in language models, arXiv preprint arXiv:2203.11171 (2022).

[24] G. V. Cormack, C. L. A. Clarke, S. Buettcher, Reciprocal rank fusion outperforms condorcet and individual rank learning methods, in: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, SIGIR '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 758–759. doi:10.1145/1571941.1572114.

[25] P.-C. Lo, Y.-H. Tsai, E.-P. Lim, S.-Y. Hwang, On Exploring the Reasoning Capability of Large Language Models with Knowledge Graphs, 2023. URL: http://arxiv.org/abs/2312.00353, arXiv:2312.00353 [cs].

[26] S. Wang, Y. Zhu, H. Liu, Z. Zheng, C. Chen, J. Li, Knowledge Editing for Large Language Models: A Survey, 2023. doi:10.48550/arXiv.2310.16218, arXiv:2310.16218 [cs].