UTRECHT UNIVERSITY

Department of Information and Computing Science

Applied Data Science master thesis

Enhancing table discovery and similarity evaluation in data lakes

First examiner:
Enas Khwaileh

Candidate:
Víctor Díaz de Burgos Llaberia

Second examiner:
Yannis Velegrakis

July 5, 2023

Abstract

The discovery of relevant tables within a data lake is a crucial task for users seeking to expand their available data and gain deeper insights into specific topics. In this thesis, we propose several approaches to address the problem of table discovery using similarity metrics. Our main objective is to find the most similar tables in a data lake to a given query table.

We begin by comparing the columns of the query table with those of the candidate tables using Jaccard similarity. This pairwise comparison allows us to compute their similarity score. It presents efficiency challenges due to the extensive computational requirements. To overcome these limitations, we investigate the use of keyword-based approaches. We propose using the Yake and LDA algorithms to extract the keywords that best represent the tables and determine the similarity score with weighted Jaccard similarity. These keyword-based approaches yield comparable accuracy scores to the column-based methods while offering improved efficiency. However, comparing keywords works poorer when tables contain data about a similar but not exact same topic. We transform the keywords into embeddings using Word2Vec and BERT to be able to analyse the semantics of the words. The similarity score in this case is determined by the weighted cosine similarity between the vectorized keywords. Furthermore, we evaluate our models using the NDCG@10 evaluation metric, which assesses the ranking of the top tables based on a labelled data lake we annotated. We show that LDA combined with Word2Vec is the most efficient and accurate model when tables contain sufficient natural language textual data.

In conclusion, our research presents a comprehensive exploration of table discovery in data lakes, focusing on similarity-based approaches. We provide insights into the efficiency and accuracy of various methods, emphasising the use of keywords and embeddings for table comparison. Our findings contribute to the broader field of data discovery and serve as a foundation for future research in improving table discovery techniques.

Contents

1. Introduction

1.1 Motivation and context

The amount of data generated and collected in recent years has increased exponentially mainly due to the proliferation of digital devices and technologies and it is expected to continue growing as more and more aspects of daily life become digitised. This is certainly having a big impact in many fields, and it opens up great opportunities for many individuals and organisations. However, this data is collected by various systems at the same time, and they all do it differently, resulting in data being saved in different heterogeneous and disconnected datasets. Thus, when researchers are interested in some of it, they have to process it and put it together before being able to use it. In order to do that, first they must identify which data is more valuable for each specific use and this is not an easy task. It is known as Dataset Discovery (Bogatu et al., 2020) and it consists of finding meaningful datasets in a data lake of datasets. In this project we will focus on a part of it: given a table, how do we compare it with other tables in a data lake, how do we determine how similar they are and, how do we match it with the most similar ones.

It is common for data scientists to have a table and want to enrich it with new complementary data. To do so, they have to identify other tables that may have similar content and then select what they are interested in. The model that will be created during this thesis project will be helpful for this task. Another occasion when this model will come in handy is when data scientists have to employ synthesised data, since they will be able to determine how similar it is to original data before using it (Liang, 2021).

In this project we focus on finding tables given a table, known as query by example (Nargesian et al., 2018), but there are other methods to do it.

Two widely used are the ones that use strings and keywords (Park & Lee, 2011) or structures (Chen et al., 2020) as queries. Both methods work similarly, the only difference is how the query is defined. When using a string the query contains the values that the model has to look for in the tables while, when using a structure as a query, the information to look for is the schema or metadata of the tables such as their column headers or data types.

The challenge of dataset discovery, or table discovery in our case, should not be that complicated in essence. There are many metrics that can be used to compare two tables and determine their similarity. However, the main issue is that simple solutions such as a linear scan and comparison is very time consuming and computationally expensive, meaning that it requires a relatively large amount of time and computational power to be completed. Therefore, the main goal of any research on the subject is to find an alternative algorithm that can yield good results in a short time.

Overall, a model that can find similar tables can significantly enhance the efficiency and accuracy of any data related work. This search mechanism would become an important and interesting tool for data scientists in today's data-rich environment.

1.2 Problem definition

As aforementioned, the problem we try to solve in this research is known as table search with example as a query, or also referred to as table-based search. To better understand where exactly the main complications lie, and to provide a clear and comprehensive analysis of the factors contributing to the problem, we delve into the description of key concepts within the research domain below.

We assume the existence of an infinite set of *names* N. We assume the existence of an infinite set of *values* V. An *attribute* is a pair of <n,v>, where $n \in N$ and $v \in V$ where n is the attribute name and v is the attribute value. Let $\mathcal{A}$ denote the set of all possible attributes, i.e., $\mathcal{A} = N \times V$.

A *tuple* is a sequence $[a_1, a_2, ..., a_n]$, where $a_i \in \mathcal{A}$ for $i = 1..n$. The *cardinal-*

ity of a tuple is the number of attributes it contains in its sequence. A *schema* of a tuple $[a_1, a_2, ..., a_n]$ is the sequence $[n_1, n_2, ..., n_n]$ where n_i is the name of the attribute a_i, for $i = 1..n$.

A *relation* is a finite set of tuples that have all the same cardinality and schema, also referred to as *table*. We denote by $\mathcal{R}$ the set of all possible relations. The notion of schema extends naturally to the relation, to indicate the common schema that all the tuples of the relation are having. A *dataset* is a set of relations. A *data lake* is a finite set of datasets.

A *query* is a finite set of tables (t_1, t_2, ..., t_n), where $t_i \in \mathcal{V}$, for $i = 1..n$. A query describes a set of concepts that are of interest to the user. Let $\mathcal{Q}$ be the set of all possible queries. Given a query q and a data lake D we are interested in finding the datasets that are related to the query q. To define relatedness we assume a match function $match| \mathcal{D} \times \mathcal{Q} \rightarrow \mathbb{R}$ and we consider a dataset to be related if and only if, match(D,q)$\geq t$, where t is a threshold.

In this work, we consider only datasets consisting of one relation, although the setting can be generalised to multi-relation datasets. For this reason, the terms dataset and relation will be used interchangeably.

Having defined all the necessary concepts, we must define the research questions we want to answer during this research. These are the following:

- How to identify the most similar tables to a query table?
- What are the best similarity measures that can be used to identify the quality of the retrieved relations?
- How to evaluate the models and how to determine whether the output tables are the most similar ones to the query table?

To solve these questions there are some problems we may encounter. Data lakes can contain a huge number of datasets and tables, and each of them can, at the same time, also be enormous; therefore, to iterate over all of them and find the most similar ones is not a trivial task. The efficiency of the algorithm will be, hence, an aspect to bear in mind when designing it. Moreover, other elements to consider are the available data that the tables can have. Although some of them may include metadata describing their

structure and the values it contains, often in big data lakes the available data are only the cell values. For that reason, the proposed solution in this research uses only what can be extracted from those values directly, and not column headers or any other data. Other metadata could be useful but, since it is not always accessible, it would restrict the types of data lakes for which our model could be executed, and this research aims to find a solution that can work at any time the user requires it.

1.3 Literature review

The idea of finding how the similarity between two tables is very much related with the concepts of table unionability and joinability. These two terms refer to the concept of finding the best data that could enrich a given table and they are well defined by (Koutras et al., 2021). Unionability refers to adding complementary data as new rows using the same attributes; while, on the other hand, joinability refers to merging two tables that have data about the same entities but have complementary attributes, meaning that the new data is added as new columns. However, as introduced in the previous section, these tables, even containing similar data, usually have very different structure and it is not easy to automatically determine which columns or rows may be joinable or unionable. Moreover, as it can be seen in Figure 1.1, two tables may not have the exact same attributes but still be unionable (section b) and two cell values may not be written the same way but still mean the same and be joinable (section d). Another issue that data scientists encounter is that generally they use data extracted from data lakes, and, as aforementioned, one of the characteristics of this kind of data repositories is that they store raw data with no attribute names; therefore, the only information available to data scientists have to determine how similar two tables are is the information contained in cell values.

In this section we will analyse how researchers have tackled this problem in previous works and we will describe the current state-of-the-art solutions.

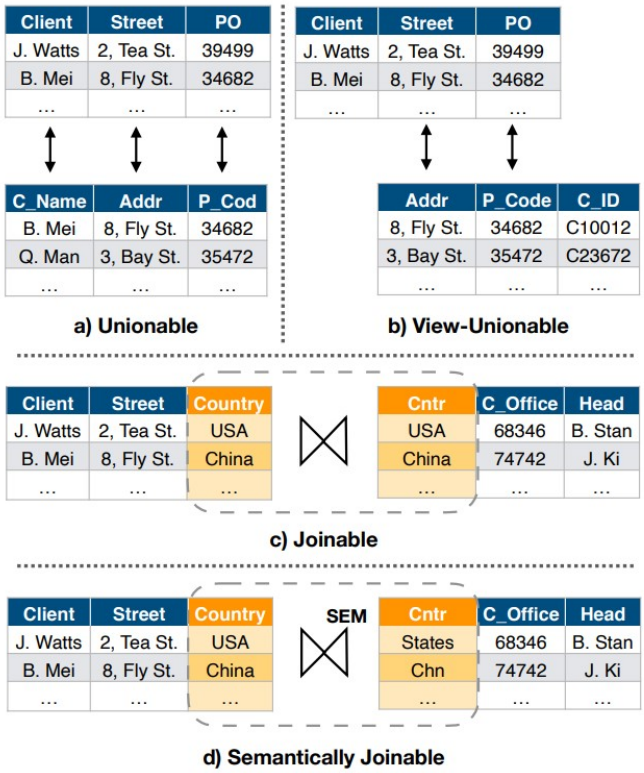

Figure 1.1: Four cases of table relatedness scenarios. Figure extracted from Koutras et al. (2021).

1.3.1 Unionability search

As Nargesian et al. (2018) describe, to find the unionability between two tables, we first have to find the unionability between each pair of attributes, and then determine the unionability of the whole tables. They define three different measures for comparing attributes: set, semantic, and natural language unionabilities. For the first one they check the intersection of the cell values just with some text preprocessing to have better results while for the second one they also consider the semantics of the values. For instance, the word Barcelona and its abbreviation BCN would be considered as two different values for the first measure but as the same value for the second one. They use an open ontology named YAGO (Suchanek et al., 2007) to do that. Ontologies provide formal specifications of the vocabularies of concepts and the relationships among them, in a domain of interest (Gagnon, 2007). Therefore, using ontologies data scientists can know, for instance, that Madrid and Barcelona are both Spanish cities and can be considered

similar. However, the coverage of these open ontologies may be quite poor in many fields, and Nargesian et al. (2018) decided to use natural language for their third unionability measure. More precisely, they use word embeddings. Word embeddings are vectorised representations of words that capture semantic and syntactic similarities between them (Levy & Goldberg, 2014). Therefore they consider nearby vectors in the embedding space to be words that share a similar context. However, since they have to use pretrained models for word embedding because of the dimensionality of the datasets, they may also yield poor results if the word to be embedded is not in the training corpus. Therefore, they suggest using a statistical method to decide which of the previous three unionability measures is the best in every specific case, they call it ensemble unionability.

As mentioned before, the main issue with this kind of model is that they are time consuming. The solution Nargesian et al. (2018) found is to use Locality Sensitive Hashing (Leskovec et al., 2014) to act as a proxy for Jaccard and cosine similarities, which they found are highly correlated to the measures they propose. With this, they can find the best candidate attributes to be united and then, compute for them the exact unionability score to see if they actually share a similar domain. Their goal is to find the most unionable tables to a query table and what they do is first to find the most unionable attributes from each table using the method previously described and then, compare only the attribute with the higher unionability score from each table. It saves a lot of computational time, and they showed that this approximation does not lead to missing many of the actual top-k most unionable tables. Once they have the candidate tables, they find the alignment that maximises the unionability score with the query table. Regarding the results and efficiency of the model, they affirm that they can get good precision and recall using natural language when the number of tables to find is large. When it is small, using the intersection of the cell values without considering the semantics is what yields better results. They assure they can achieve interactive response times. Using the semantics of the words did not work for them due to the small coverage of available ontologies.

To solve this problem of ontologies, Khatiwada et al. (2023) propose to create a synthesised knowledge base (KB). They include the ontology of the cell values and also the relations between the different attributes of the tables. They label the values with the type of information they contain and, an interesting aspect they include are different granular levels for each value; for instance, for a city they may label it as a city but also as a region and a place, to be more flexible when comparing it with other location types. To create the synthesised KB, they exploit the knowledge of the data lake itself. Data lakes are where all the possible tables to be united are stored. They look at the co-occurrence of information across the tables in it and they map the values from the query table to other columns in the table in the data lake where these values appear. With that they define both attribute and relationship semantics. Then, they represent each table as a graph where attributes are nodes and branches are the relation that connects them. Finally, they only have to look for similar trees to the query one to find the most unionable tables.

They called this model SANTOS and, after benchmarking it against other models they see a clear effect on using the relations between attributes and not only their semantics. Moreover, they proved that the synthesised KB works well when there is no existing KB that can describe the data in the table. However, this model takes three times what others take to index all the tables. However, once it is created for the specific data lake, it has a query time of only seven seconds.

Mountantonakis and Tzitzikas (2020) also propose a solution to the unionability search problem based on the semantics of the attributes and values. To deal with the time-consuming issue, they propose an algorithm that adapts to the input datasets to make the model more efficient. They use a lattice-based incremental algorithm based on set-theory properties and pruning and regrouping methods. To find the similarity between tables it uses the information coverage, enrichment, and uniqueness of their values. This solution is 1000 times faster than a normal straightforward linear-scan, and it is even faster when they do more pruning of tables before processing them and they proved that it still yields good results.

1.3.2 Joinability search

To recall, the aim of doing joinability search is to find tables that can add new information to the query table as new columns. To do so, what many researchers do is to first find the joinability with respect to an attribute of the query table and see how much the new table could contribute. Zhu et al. (2016) propose a method that analyses the containment of a column with respect to another which means to analyse how much from a set can be found in another set. As other aforementioned papers, they measure this value using Jaccard, in this case Jaccard set containment score. The main idea is to, given an attribute of the query table, find a column of another table that contains as much of the domain of the attribute of the query table as possible; they define domain as a set of values that characterises a data set. Therefore, what they are looking for in the attributes of the candidate tables are values that contain as much information about the same field as the query table as possible.

They do not use Jaccard similarity for this because it favours domains with smaller size and the resulting candidate attributes to be joined would be biassed to the cardinality of the sets. Set containment, on the contrary, is agnostic to the difference in the sizes of the domains. However, with LSH, which is the method they use to index the values and estimate the metrics, they can only estimate the Jaccard Similarity and not the set containment. To solve the cardinality problem is to first create a partition of the sets and then, they run the LSH algorithm for the values of each of the partitions. Finally, to estimate the set containment score they use the inclusion-exclusion principle. They call this procedure the LSH Ensemble. It improves the overall accuracy over the baseline of the benchmarking they did by as much as 25However, even if they use dynamic indexing, the accuracy decreases when new data arrives. And, since it is supposed to be used with real open data, it is not something we want for our model. Fernandez et al. (2019) solved it with LAZO. They propose a method that not only uses the containment but also the similarity between columns. They, again, use the Jaccard metrics to measure that and they use LSH to estimate them. They tweak

LSH in order to obtain not only the candidate attributes but also their scores with respect to the Jaccard similarity and containment. Thanks to that, they can do some error correction and generate more accurate estimations. They can do it because they know the containment score can, at most, be as large as the cardinality of the smaller table; therefore, if the estimation does not meet this condition, they know it is not correct and the algorithm can fine tune some parameters in order to get better results.

They affirm that LAZO is a practical method to use in data discovery scenarios because it achieves good estimation results and, unlike LSHEnsemble, it does not deteriorate when new data arrives. Moreover, they can match the accuracy of more complex methods but with less time-consuming algorithms without any quality loss.

Zhu et al. (2019) propose JOSIE which, unlike other models, is not based on estimations but on exact measures. They use the exact intersection size of the two columns. The columns to be compared are always one column from one of the tables of the data lake and one column of the query table which the user has chosen as the one from which the algorithm has to look for candidate pairs. This column from the query table is known as the join column. Most of the models for joinability search have this column as an input as well as a threshold that the algorithm uses to find columns whose similarity score is over it. However, JOSIE lets the user set the number of most joinable tables they want to have as an output. Researchers did it this way because they say it may be confusing for users that do not know exactly what the data is about to set a specific threshold. Another important feature JOSIE has is that it is adaptive to the data distribution and it is not sensitive to the characteristics of the data. This is interesting when using real data from big open data lakes which all have very different structures. Researchers showed that JOSIE out-performs other models on these real-world data lakes. Since it is based on exact metrics it is 3 to 4 times slower than other approaches based on estimations. However, it finds all the candidate pairs and does not miss the 10% to 40% that approximate techniques can be missing.

Other researchers such as Yang et al. (2019) also propose solutions to the joinability search based on set containment and intersection. In this case, they propose a sketch technique, namely GB-KMV, which approximates and estimates the measure but still out-performs LSH Ensemble in both accuracy and efficiency. However, the model only works when assuming some characteristics of the data that may not always be true. Therefore, it is a data dependent method that, although it has been tested with real-world data, may not always work fine. Santos et al. (2021) also propose a solution based on a sketch technique, they name it Correlation Sketches. They estimate correlations between columns from unjoined datasets with the descriptions of the columns they have precomputed. For that, they use hashing functions which may lead to some approximation error. Similarly to other work aforementioned, their algorithm addresses this issue but, in this case, it does not rely on assumptions. They showed that it is effective and derives high-quality rankings of most joinable tables.

1.3.3 Keyword extraction and word embeddings

Much of the knowledge from unionability and joinability research can be extrapolated to the problem we are trying to solve and this is what we will do in this project. However, we have to, naturally, analyse as well what has been achieved in dataset discovery when the main goal was to simply find similar tables without the need of merging them. To do so, the structure of the tables is not as important as with methods mentioned above, and the interest falls directly to their content.

S. Zhang and Balog (2021) propose treating tables as sets of terms that can be then compared to other sets of terms representing other tables. They propose two options to select the terms that represent the tables: either use all words or use only the entities present in the tables. They find these entities using a knowledge base as we have seen other models do. Once they have them, they find their semantic vector representation using word embeddings with Word2Vec (Mikolov et al., 2013). As mentioned above, word embeddings capture semantic and syntactic similarities between words; there-

fore, after this step, researchers are able to find similarities regarding the semantic meaning of the tables and not just the raw content. In this case, other data from the table apart from the core values in the cells such as table title or columns headings are used only to determine the most relevant terms; however, Trabelsi et al. (2019) propose a similar model that uses these additional data to find the embeddings in order to have a more context aware representation of the words.

Now, the terms are represented in the vector space and, therefore, the similarity metric they have to use is cosine similarity. However, each table is represented by several vectors and this metric only compares two vectors. They propose two solutions to solve this issue, find the centroid of the vectors representing each table and then compute the cosine similarity between these two vectors, or compute the pairwise similarity between all vectors and then aggregate the results.

For the evaluation of the outcomes they use the Normalised Discounted Cumulative Gain (NDCG). It is a measure of ranking quality that compares the relevance of the items returned by the search engine to the relevance of the item that a hypothetical ideal search engine would return (Chi & Roberts, 2023). They state that their results can improve retrieval performance substantially. Moreover, they compared the model using word embeddings to using other semantic representations of the terms and the former outperforms the others.

Similarly, the research made by Rossiello et al. (2017) also concludes that embeddings created with Word2Vec are the best semantic representation for terms when we are interested in their similarity. They propose a centroid approach for the similarity measure where they select the closest terms to the centroid of the document, and then they compare them with the word embeddings of other tables in the data lake. In their case, they seek for the similarity of text documents instead of tables, but since we are interested in only the words of the table, we can also treat them as such. However, we have to bear in mind that most of the tables do not contain structured sentences but only single words or concepts. Therefore, models such as

the one proposed by Boudin (2018) where keyphrases are extracted from documents instead of simple keywords.

This process of determining the terms that best describe the tables is known as keyword extraction. After the text first has gone through this operation, the table search model only has to deal with a few terms per table instead of the whole raw text. Thus, its efficiency can be improved. Miah et al. (2021) suggest using YAKE (Campos et al., 2018) or TopicRank (Bougouin et al., 2013). The former uses word features such as the word frequency while the latter uses topic modelling algorithms which cluster the text into topics and select the most relevant keywords for each of them. Both methods yield similar results. Then, they propose to, as other methods, use word embeddings and cosine similarity to determine the similarity between tables. However, in this case they also propose other approaches. Once they extract the keywords, there is no need to transform them into vectors, other similarity metrics can be applied to the keyword sets such as Jaccard similarity. The output of this approach is much worse than using word embedding, but, at the same time, it is much less complex and more efficient. Therefore, at the end, the decision of the method to be employed must always take into account the context of the dataset discovery to evaluate whether its efficiency or its accuracy is more valuable.

2. Data

2.1 Description of the data

The goal of this research is to develop a model that remains independent of the specific input data and can be universally applied to any data lake of interest to users. This type of models are known as data-agnostic models (Guidotti & Monreale, 2020). As such, the functionality of the model does not heavily depend on the selection of the input data. Instead, the focus lies on utilising data that can provide a robust evaluation of the model's accuracy and efficiency. Therefore, the data chosen for this study has been selected based on its potential to facilitate a thorough assessment of the model's performance. By leveraging tables that offer diverse challenges and opportunities for evaluation, we can effectively gauge the model's effectiveness in practical scenarios. For this purpose, three different table corpora have been used.

One of them is WikiTables (Bhagavatula et al., 2013, 2015) which contains all tables in Wikipedia, over 1.6M tables. For each of these tables different information is available: title, little caption or description, title of the Wikipedia page where it is included, number of columns, number of rows, column headers and the data of the cells of the table. It is a good corpus for testing purposes because it contains a wide variety of topics; however, since all tables are extracted from articles, they may generalise poorly due to their small dimensions (Langenecker et al., 2021). However, it has been widely used in previous researches and it helped compare our results with the state-of-the-art models for table search and dataset discovery. Its tables have, on average, 11 rows and 5 columns. Another table corpus used was GitTables (Hulsebos et al., 2023). It contains 1M tables extracted from GitHub and its tables are much bigger than the ones from WikiTables, they have, on average, 142 rows and 12 columns. For easier and more efficient testing, we

sampled 30 tables from both corpora that will act as our data lakes. We also sampled 5 other tables that will be the input queries for which the model has to find and rank the most similar tables. Then, to be able to discover the effectiveness of our model, we labelled the sample tables with respect to their similarity with the input tables. These labels will be used to determine how similar the output tables actually are to the query table.

Finally, we used a last table corpus taken from Nargesian et al. (2018). This corpus comprises 24 tables with highly diverse schemas, ranging from smaller tables with 25 rows to larger ones with 106,415 rows. The number of columns in these tables also varies, ranging from 4 to 21. However, its most significant value for our purposes lies in the fact that it predominantly consists of textual data. This particular corpus serves as a valuable resource to assess the significance of incorporating such textual values for the accurate functioning of our proposed model. Similar to the other two table corpora, we labelled the similarity between query tables and tables in the data lake to enable accurate evaluation of the model's performance. In this case, due to the limited number of tables in the corpus, query tables are selected from the data lake itself, and are part of it when they do not act as queries.

2.2 Preparation of the data

The model we propose only operates with the information contained in the cell values of the tables because data lakes often do not have any other metadata available and we do not want a model that is restricted to specific types of data lakes. Therefore, we discard all other metadata that WikiTables contains and we only keep the actual data of the tables. Once we have it, the first thing we do is to merge all values in one big text variable. We do it because, as it will be described in the method section, our model works with tables as texts which are then processed to find other texts about similar topics in the data lake.

In order to derive meaningful insights from the textual data, it is imperative to perform a preliminary data cleansing process wherein the tables are appropriately formatted to meet the requisite criteria. To do so, some ba-

sic text preprocessing steps are carried out, including lower casing, symbol and punctuation removal and lemmatization and tokenization. Moreover, we decided to not include numerical columns in our study because they contain very little information about what a table is about and can reduce the matching criteria. Similarly, we also get rid of all words that are neither verbs nor nouns because they do not help us determine the topic of a table either. Another step in our preprocessing pipeline is the separation of concatenated words into individual words. This process involves identifying and separating strings that contain a lowercase letter followed immediately by an uppercase letter. This step is particularly important when working with tables since many of them contain concatenated words rather than properly segmented words. By isolating and splitting these concatenated words, we can ensure that each component is treated as a distinct word. This preprocessing step significantly enhances the accuracy and effectiveness of subsequent analyses.

3. Method

3.1 Description of the methods used

This section presents the solution we propose for dataset discovery in terms of similarity matching between tables. To recall, the main goal of this research is to design an algorithm that finds and ranks the most similar tables in a data lake with respect to a query table given by the user. Different approaches are introduced in the following pages, starting with the most straightforward solutions and finishing with the most advanced methods. All the proposed approaches follow the same structure: we iterate through all the tables in the data lake determining their similarity score with the query table, we rank those values and we return the k most similar tables, being k a number established by the user which tells the model how many of them it has to output. The differences in the approaches we introduce are, therefore, in the methods used to determine the similarity between two tables.

3.1.1 Simple similarity

In our first proposed solution, we compare the columns of the query table with those of each candidate table in the data lake using Jaccard similarity. The similarity score between two tables is computed by taking the mean of the Jaccard similarity values of each column pair. This approach, known as pairwise Jaccard similarity, compares all pairs of columns, resulting in a time-consuming process with a complexity of $O(N \cdot M)$, where N and M represent the number of columns in the query and candidate tables, respectively. This comparison needs to be performed for every table in the data lake, further impacting the efficiency of the approach. Additionally, we can also extend this comparison to the rows of the tables, considering the shared entities, but given that tables typically have more rows than columns, the ef-

ficiency of this approach is even lower.

As mentioned in previous sections, other researchers have used different types of estimations for Jaccard similarity that can make the computational time of the algorithm decrease, for instance, min-Hash (Ji et al., 2013). However, we decided not to continue exploring this approach and try to, instead of comparing the whole tables, find the words that best describe the content of the tables and compare them to determine the similarity of two tables. These words are referred to as keywords and, as K. Zhang et al. (2006) describe, they summarise a document concisely and give a high-level description of the document's content.

3.1.2 Keywords similarity

As defined by Hulth (2004), keyword extraction is the task of selecting a small set of words/phrases from a document that can describe the meaning of the document; in our case, the documents are the tables. It is a natural choice to simply use word tokens to represent table content (S. Zhang & Balog, 2021). To do so, as mentioned in Section 2.2, the first step is to convert tables into texts and preprocess them. Once the tables are in the required format, we can apply the keyword extraction methods we use to determine the concepts that best describe the tables and compare them to find the most similar tables. In this research two methods have been used: Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Yake (Campos et al., 2020).

LDA is a popular topic modelling algorithm used in natural language processing (Putri & Kusumaningrum, 2017). It is designed to discover hidden thematic structures within documents. It assumes that each document is a mixture of various topics, and each word in the document is generated from one of these topics. By applying probabilistic inference techniques, LDA enables the identification and extraction of underlying topics, providing valuable insights on large text datasets. The output is normally a list of topics with the keywords that best represent each of them. However, since most of the tables do not contain more than just one topic, we set the number of topics to derive from each of them as one. Therefore, the output we

get is the list of keywords that best represent the table. Moreover, the algorithm outputs the weight each of the keywords has in the overall topic distribution of the table. Those values will be used when comparing two sets of keywords to determine their similarity.

Not all values in the tables have the same importance when detecting the topic of a table. For instance, categorical attributes with a short number of unique values in their cells typically have more meaningful information than attributes that contain open data where most of the cells have very diverse values. Therefore, giving more weight to specific columns could be a good idea. However, since LDA works with the number of times each word appears in the text, it already gives more importance to the columns with less varied values because they appear repeatedly in several rows of the table. Therefore, no attribute weight is required for the algorithm to run accurately.

There are keyword extraction methods that work good for other purposes but not for dataset discovery. This is because, in our case, we determine the keywords for each table without taking into account any other table. This is a requirement for our model because data lakes may not be static and the tables they contain may be different each time the model is run. Therefore, if we used a method that worked with all tables from the data lake together to extract the keywords for each of them, such as the ones using TF-IDF to examine the relevance of the words (Qaiser & Ali, 2018), the model would have to be executed every time the data lake changed. Moreover, the keywords for the query table could not be extracted or they could only be extracted if the algorithm was run again including both the query table and the candidate tables. Therefore, since the efficiency of the model is something we have to bear in mind in this research, these types of methods have to be discarded.

Yake uses multiple local features from the documents to extract the keywords. Hence, since it does not use any other characteristic from the data lake, it can be run for each table individually. An interesting option Yake has is that output keywords can also be composed of more than one word,

sequences that are known as n-grams. For instance, if a table contains data about countries, 'North Macedonia' could be extracted as a single keyword and it could be differentiated from, for example, other tables containing the word 'north'. Other methods are also capable of extracting keyphrases but we cannot use them because we deal with text from tables and it usually does not contain sentence structured cells but only words and concepts. Similarly to LDA, Yake outputs the importance each of the keywords has on the overall text of the table. Therefore, the output of both methods has the same structure: a set of keywords with their corresponding weight. Thus, the same similarity metric can be employed for both of them.

The similarity metric used to compare two sets of keywords is Weighted Jaccard Similarity. It, as the regular Jaccard Similarity, uses the intersection and the union of both sets but, in this case, it incorporates the weights to the equation as well. Depending on the weights of the words of the intersection, the similarity score will be smaller or larger. The model computes this value for all the candidate tables with respect to the query table, rank them and, return the top-k most similar tables.

3.1.3 Embeddings similarity

To determine the similarity between sets of keywords, we previously employed Jaccard similarity, which primarily focuses on the lexical similarity. It involves comparing sets of keywords directly without considering their underlying meanings. However, simple Jaccard similarity fails to recognize the similarity between synonyms or words that pertain to the same topic. For example, words like "dog" and "cat" are not synonymous, but they both represent domestic animals. In certain cases, a user may seek a table on animals in general when the query is specific to a particular animal. Additionally, considering that tables often contain limited textual content, it becomes crucial to identify similar tables even when the words used are not an exact match. To address these challenges, we employ semantic similarity through the utilisation of word embeddings.

Word embeddings, also known as word representations or word vector-

izations, are dense vector representations of words used in natural language processing tasks. They capture both semantic and syntactic information of words and can be used to measure word similarities (Liu et al., 2015). The main idea is that every word is represented in a multidimensional space as a vector, where semantic relationships are preserved. For instance, the vectors representing animals will all be placed together and their distance will be small. This suggests that distances and between embedded word vectors are to some degree semantically meaningful (Kusner et al., 2015). Therefore, our solution converts each keyword to a word embedding and then determines the similarity between the sets of vectors. For the vectorization of words we propose two methods: Word2Vec (Mikolov et al., 2013) and Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018).

The Word2Vec algorithm uses a neural network model to learn word associations from a large corpus of text. We used a pretrained model by Google ("Google Code Archive - Word2Vec", 2013) which includes word embeddings for a vocabulary of 3 million words that they trained on roughly 100 billion words from a Google News dataset. It has been widely used in previous research (Khatua et al., 2019; Wang et al., 2018; S. Zhang & Balog, 2021). It has been verified that word embeddings trained based on Google News data and on a table corpus lead to comparable performance (Deng et al., 2019); therefore, there is no need to train our own model. Moreover, a model trained using only table data would not yield good results because the words have no context and the model would not be able to establish relations between words. Deng et al. (2019) designed a similar model named Table2Vec that converts tables into embeddings. It could be useful for comparing tables in a more straightforward approach but not in our case because it uses table elements that often data lakes do not contain: caption and attribute names.

A limitation the pretrained Word2Vec model has is that it can only vectorize words that have an already associated vector in the model. This means that, it may happen that some keywords cannot be vectorized. This is a recurrent issue for us because tables often contain abbreviations and

other symbols that are extracted as keywords and the pretrained vectorizer model does not know about. Moreover, Google's pretrained model can only be used with single words and not with n-grams. And, as aforementioned, a feature Yake has is that it can extract n-grams as keywords. To be able to deal with this type of keywords another vectorizer we propose is a pretrained model for BERT.

BERT is an influential and extensively utilised model renowned for its ability to generate word embeddings that are rich in context and meaning. Its sophisticated architecture, built on Transformers (Wolf et al., 2020), empowers BERT to capture relationships between words, resulting in highly effective representations. We used a pretrained version of it offered by Hugging Face, "bert-base-uncased", but it does not work the same way as a pretrained model for Word2Vec does. In this case, the model is not static and words that it has never seen before can still be vectorized which is useful for both abbreviations and n-grams. This model is mainly used to capture contextual information which, when vectorizing keywords, is not available. Therefore, choosing between using Word2Vec or BERT will depend on the input tables and on the model used to extract the keywords. It is important to notice that to better compare the query table with the tables in the data lakes, all of them have to be processed using the same models.

After obtaining the embeddings, we proceed to compare the sets of keywords across different tables. This process follows a similar structure to the previously described version, where the raw sets of keywords were compared. We iterate through all the tables in the data lake, comparing their embeddings, determining their similarity, and subsequently ranking the tables. However, in this case, we utilise Weighted Cosine Similarity as our similarity metric. It calculates the regular cosine similarity between vectors and incorporates the weights generated by the keyword extraction methods. Since each table is characterised by multiple keywords, each having its own corresponding embedding, the model performs pairwise cosine similarity computations between all the embeddings of the query table and the candidate table. The results are then aggregated to derive a similarity score calculating the average of all the cosine similarity values. This value is then

compared to the similarity score of the other candidate tables in order to rank them and find the most similar ones.

3.2 Practical considerations

To complement the detailed description of our model's architecture and functionality, this subsection focuses on the practical considerations and method settings associated with its implementation. By evaluating the advantages, limitations, and potential applications of our model, we aim to provide valuable insights into its feasibility and applicability. Furthermore, we discuss specific settings and configurations that were utilised to optimise its performance. This examination of practical considerations and method settings enhances our understanding of the model and provides valuable information for its implementation in various contexts.

The first aspect to consider when executing the model is that the preprocessing of the text and the vectorization of words can be time-consuming, especially when dealing with extensive data in tables and data lakes. However, one advantageous characteristic of the proposed models is that they only need to be executed once for each table. This means that the model incurs computational expense primarily during the initial run with new data. Once the tables are converted into keywords and these keywords are transformed into embeddings, the resulting sets can be stored for future executions. Therefore, there is no need to rerun the entire model, except for processing the query table and any new tables in the data lake. After this initial step, the subsequent process of computing similarity scores and ranking them should be relatively quick and straightforward.

In scenarios where tables are small or contain limited textual data, it is important to acknowledge that our models may yield less satisfactory results. The models heavily rely on the textual content within the tables for accurate keywords, embeddings, and subsequent similarity calculations. Furthermore, it is worth noting that tables predominantly composed of numeric values pose challenges not only for our proposed models but for any model in general. This is primarily due to the limited information that can

be extracted from raw numerical data alone. The absence of explicit textual content makes it difficult for models to derive meaningful insights and relationships from such tables. However, it is possible for certain models to mitigate these challenges by leveraging additional data beyond the cell values, such as attribute names or table descriptions. By incorporating such supplementary information, these models have the potential to enhance their understanding and interpretation of the numeric-centric tables.

A parameter we have to set in our model is how many keywords have to be extracted for every table. Ideally, since each keyword is associated with its corresponding weight, the more the better in order to be able to compare accurately the different candidate tables. However, the amount of textual data a table has can affect the number of keywords that can be successfully extracted. Therefore, if we set it too high, the algorithm may not be able to extract a sufficient number of keywords from these tables. As a result, an imbalance may arise among the tables, impacting the overall performance and reliability of the algorithm. The incorporation of weights in our model serves as well to mitigate the impact of the potential imbalance among tables. We chose to initially set the parameter to extract 10 keywords. This value served as a starting point, allowing us to gather preliminary insights. However, it is important to note that this parameter has to be fine-tuned based on the observed results and the characteristics of the tables.

Finally, another aspect to consider is that depending on the specific model used, different approaches to text preprocessing may be required. It is particularly relevant when considering the extraction of n-grams as keywords (Miah et al., 2021). Techniques such as stemming and lemmatization, which aim to reduce words to their base or root form, can significantly influence the results. For instance, after lemmatization, the phrase "United States" would be transformed into "Unite State". While this transformation may seem subtle, it can potentially affect the recognition of the keyword as a country by the vectorizer algorithm. Consequently, it is essential to carefully evaluate preprocessing steps to ensure the preservation of meaningful information and accurate representation of the tables in the model.

The code for the proposed models can be found in https://github.com/victordiazdeburgos/tablediscovery.

4. Experimental evaluation

4.1 Evaluation metrics

The chosen evaluation metric for assessing the accuracy of the proposed models is Normalised Discounted Cumulative Gain (NDCG) as previously utilised by other researchers such as Trabelsi et al. (2019) and S. Zhang and Balog (2021). It measures the effectiveness of a ranking algorithm by considering both the relevance of the recommended items and their positions in the ranked list. It compares the output ranked tables of our models with what an "ideal" table search engine would return. In the context of table search by example with respect to similarity measures there is no pre-existing ground truth available against which to compare the rankings. Therefore, to establish a benchmark for evaluation, we created our own labels by annotating the similarity of the query tables with the tables in the data lake. We defined three levels of similarity: 0 for no similarity, 1 for a similar topic, 2 for the exact same topic.

In previous research, Mean Average Precision (MAP) has also been commonly used as an evaluation metric Khatiwada et al. (2023) and Nargesian et al. (2018). However, using MAP requires a labelled ground truth that represents the ranking of candidate tables, rather than just their level of similarity. Due to the characteristics of the table corpora used, defining such a ranking was not feasible. As a result, we opted to utilise Normalised Discounted Cumulative Gain (NDCG) as our evaluation metric. Following the approach of S. Zhang and Balog (2021), we focus our evaluation on comparing the top 10 tables in the ranking. Therefore, the specific evaluation metric we employ is NDCG@10 where the value "10" indicates the number of top tables in the ranking that are considered for evaluation.

A good model should not only provide accurate rankings but also oper-

ate efficiently. Hence, we consider efficiency as another important metric for evaluating the proposed models. We measure this by tracking two types of executions. The first type involves processing all the candidate tables along with the query table, which includes model-specific operations like extracting keywords or converting them into embeddings, and then finding the similarities scores and generating the ranked results. However, it's worth noting that the data lake processing, which typically consumes more time, only needs to be done once. For subsequent executions on the same data lake, the keywords and embeddings can be stored and reused. Therefore, the second execution measures the time it takes to process only the query table, determine its similarity with all candidate tables, and return the ranking, mimicking the scenario where a preprocessed data lake is used.

4.2 Overview of the results

To evaluate the proposed models we will start analysing the experimental results obtained with the table corpus from Nargesian et al. (2018). It contains tables with mostly textual data which, in theory, should be the set up that benefits the most keyword extraction algorithms and consequently, word vectorizers. The average NDCG@10 score of the four queries used can be seen in Figure 4.1. Additionally, the average execution times of all the models are shown in Figure 4.2.

Simple similarity measures work very well with these specific queries and corpus. However, it is very computationally expensive and the efficiency decreases exponentially as the data lakes get larger. As mentioned in Section 3.1.1, this model computes the Jaccard similarity for every pair of columns for every candidate table, therefore, the more columns and the more tables the more computationally expensive the model is. We can see that, with a data lake of just 23 tables, its execution time is already 8 times larger than the least efficient among the other models. Moreover, unlike the other models, this approach does not provide the option to skip any part of the process during the executions that come after the data lake has already been processed once. This is why the execution time shown in Figure 4.2 is

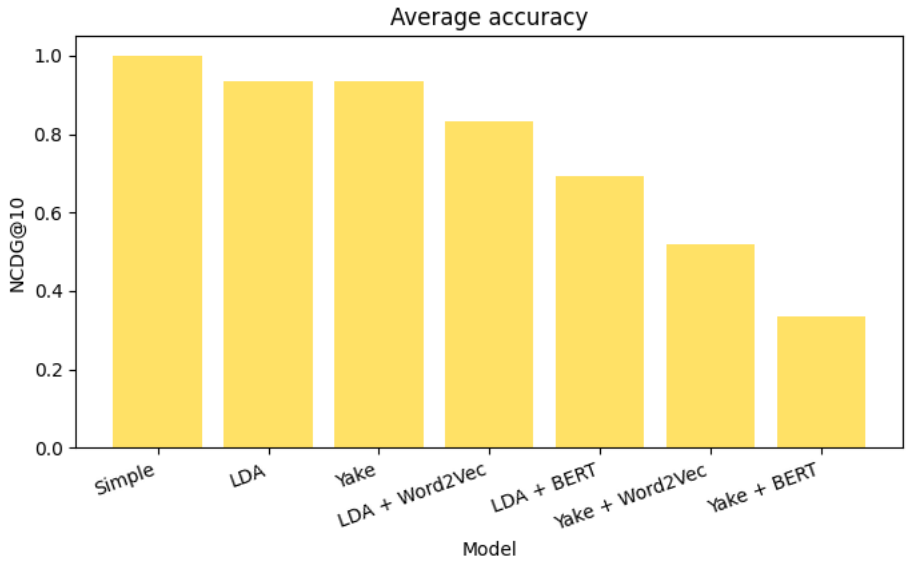

Figure 4.1: Average accuracies for 4 queries with the table corpus of Nargesian et al. (2018)

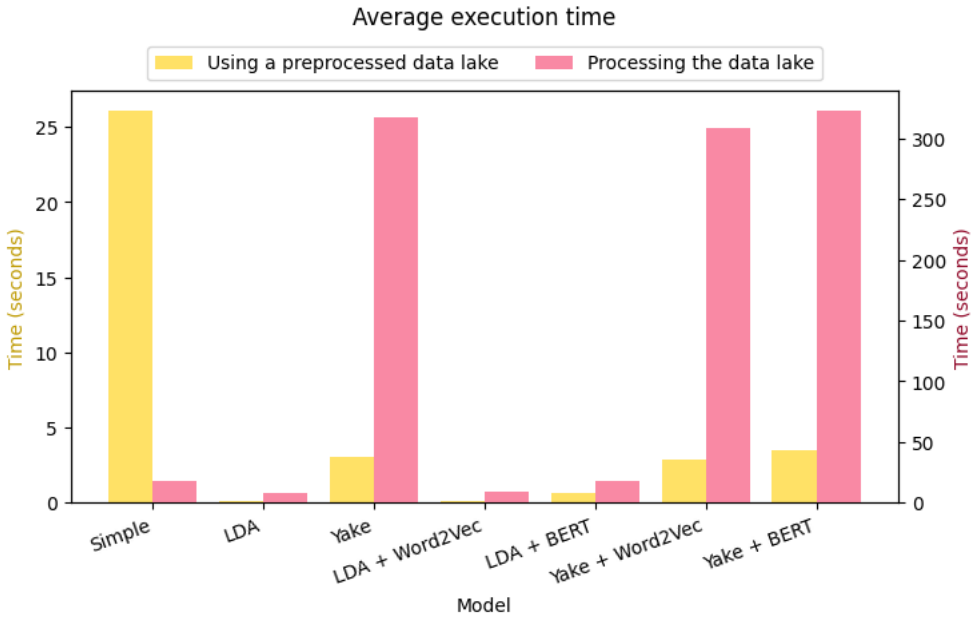

Figure 4.2: Average efficiencies for 4 queries with the table corpus of Nargesian et al. (2018)

the same for both bars regarding this model.

The average accuracy score for the two methods using keyword similarity is nearly perfect as well. LDA, however, exhibits significantly faster processing times compared to Yake. Figure 4.2 shows a substantial difference in Yake's runtime, depending whether it processes the entire data lake or it utilises a preprocessed one. The reason for that is the large amount of time Yake requires to extract the keywords for each table; consequently, this model is much more time consuming when processing the data lake compared to when it only processes the query table. This difference also demonstrates that the time to execute the weighted Jaccard similarity, used to determine the similarity between two sets of keywords, is negligible in comparison to the time spent extracting the words that best represent the tables.

The models transforming the keywords into embeddings and then determining its similarity with weighted cosine similarity work poorer than the others. It can be attributed to the table corpus we are using because it contains either very similar tables, even tables sharing the same schema, or completely different tables. Thus, the keyword similarity models work well because similar tables use the exact same words for certain attributes and Jaccard similarity is a good measure in this case. Embeddings are thought to be useful when finding tables about similar topics and categories where the analysis of the semantics of the words makes the difference in order to match related but not identical words. In terms of efficiency, BERT is slightly more time consuming than Word2Vec. However, there is almost no difference to the running time of the models using only keywords. Therefore, we can also consider the aggregated execution time by the vectorizers to be negligible.

Analysing the accuracy in Figure 4.1 we can see that, when using word embeddings, the model works better having extracted keywords with LDA than with Yake. This is because Yake has a feature, which the authors call *Word Positional*, that values the words occurring at the beginning of a document more. It is based on the assumption that relevant keywords often tend

to concentrate more at the beginning of a document (Mishra, 2022) but, in the case of documents being tables, this feature does more harm than good because tables are not ordered and first rows do not contain more information than any other row. Yake, for instance, would have been interesting to use if we had decided to include attribute names as a factor for our model. However, as mentioned earlier, we decided not to do it because data lakes may not contain this feature, and we want the models to be data-agnostic in order to be useful for any type of table corpus. Therefore, we can expect the models using Yake as a keyword extraction method to be less accurate.

Due to the limitations of Yake as a method for extracting keywords from tables, we are unable to evaluate the effectiveness of our models when using n-grams as keywords. While we could extract n-grams from tables using Yake and vectorize them with BERT, the resulting accuracy would not be a reliable indicator. It would be challenging to determine whether any inaccuracies stem from the ineffectiveness of n-grams or from Yake's inadequate extraction of keywords. However, accuracies in Figure 4.1 show that Yake yields good results when only comparing sets of keywords without considering their semantic meaning. Despite the fact that the extracted keywords may not fully represent the entire table, they prove to be sufficiently representative for the tables in the tested corpus. This is mainly attributed to the presence of highly similar tables within the corpus for which Yake can still identify sufficiently representative keywords.

The other two table corpora, GitTables and WikiTables, do not meet the desired criteria to be used in the evaluation of our models. Due to the large number of tables they contain, the randomly sampled candidate and query tables were not similar enough in order to be annotated and generate labelled ground truth data. To address this limitation, we opted to selectively analyse tables of interest. However, this approach may introduce bias into the evaluation results. As a result, instead of focusing on computing average results across multiple queries, as we did with the previous corpus, we focused on examining the performance of the proposed models on specific individual queries. This allows us to assess their effectiveness in specific scenarios while minimising the impact of potential biases.

Figure 4.3 shows the accuracy of the models for a query and candidate tables sampled from GitTables. The query has been specifically selected because it is very similar to other tables in the data lake but, as the majority of the table in this corpus, contain fewer text than the previously analysed extracted from Nargesian et al. (2018). We can see that the simple similarity model is not the best in this case. Instead, comparing the sets of keywords works better. Now that there is less text and it is more challenging to extract the meaningful keywords, we can appreciate how the model performs poorly when comparing the embeddings of the Yake's keywords. However, since the similarity between the query table and some of the candidate tables is still very high, comparing keywords using both LDA and Yake still works better than using their vectorizations. In terms of efficiency, it is worth mentioning that, as shown in Figure 4.4, processing the entire data lake still takes a noticeable amount of time, even with smaller tables. Interestingly, the model that combines the two more time-consuming methods, Yake and BERT, is as slow as the simple similarity model. Therefore, in this case, it does not only suffer from bad accuracy but also from slow processing time.

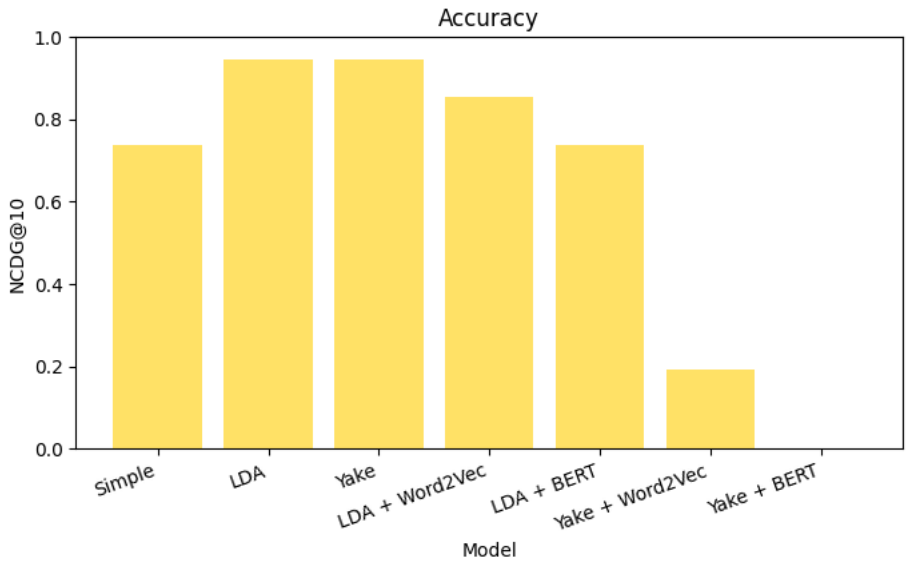

Figure 4.3: Accuracy of the models for a query with less text but still very similar to some candidate tables

Figure 4.4: Efficiency of the models for a query with less text but still very similar to some candidate tables

Finally, we examine a scenario where the query table contains only a few text entries and does not match exactly with any candidate table in the data lake. As depicted in Figure 4.5, the NDCG@10 scores for this query are notably lower due to the limited textual information in the table. Notably, in this case, the use of embeddings proves beneficial, as it outperforms the models that do not utilise them. When comparing Word2Vec and BERT, Figure 4.5 demonstrates that BERT performs better. This discrepancy can be attributed to the fact that we employ pretrained models, and tables with minimal text may contain words that are not fully recognized by these models.

A key distinction between Word2Vec and BERT lies in their vectorization capabilities. Word2Vec can only vectorize words present in a predefined dictionary, while BERT encompasses the vectorization of all words by establishing relationships with other precomputed embeddings. However, as we can see in Figure 4.6, this extra computing process takes time and it decreases the efficiency of the model. A similar challenge was faced by Nargesian et al. (2018), where the absence of a suitable pretrained model impeded the vectorization of certain words. The limited coverage of pre-

trained vectorizer methods poses a constraint, as some keywords extracted from tables may be from attributes containing codes or abbreviations not present in natural language. As a result, the selection of an appropriate method must consider both the query table's characteristics and the tables within the data lake, recognizing the potential limitations of pretrained vectorizers for specific use cases.

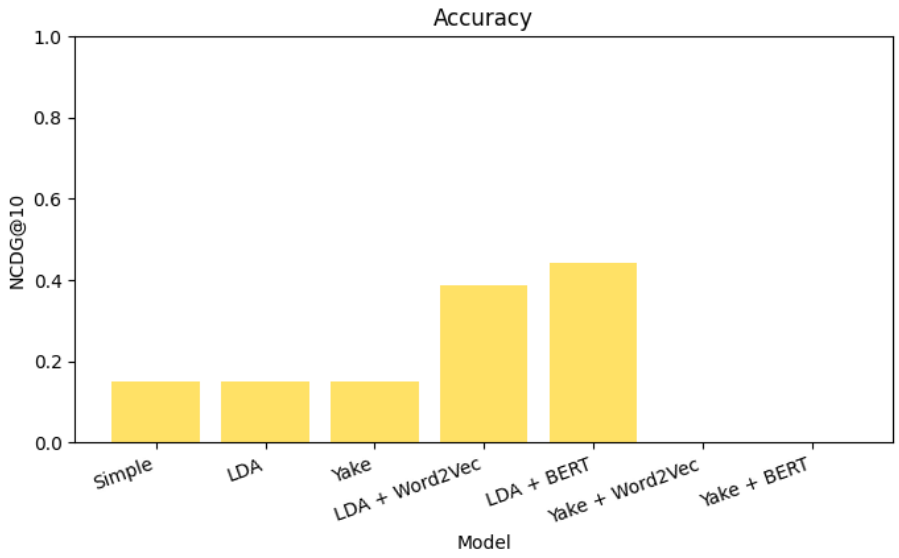

Figure 4.5: Accuracy of the models for a query with few text and not substantially similar to any candidate table

Figure 4.6: Efficiency of the models for a query with few text and not substantially similar to any candidate table

5. Conclusion

5.1 Key findings and research questions

In this project we have proposed multiple approaches to solve the problem of table discovery using similarity metrics among tables. The main idea is to find the most relevant tables in a data lake based on their similarity to a given query table. This aims to assist users to access a broader range of relevant information and enrich their dataset facilitating deeper analysis about the topic. We have proposed seven different models. The most simple one uses Jaccard similarity to compare columns or rows of the tables and determine their similarity score. It works well when there exist very similar tables in the data lake, but it is extremely time consuming. To address this challenge we proposed to extract the words that best describe the topic of the tables and compare them. We used two keyword extraction methods for that which are LDA and Yake. Both of them yield comparable results when tables in the data lake use the exact same words as the query table but they work poorer when tables share the topic but not the words.

In order to be able to match related but not exact same keywords, we suggest analysing their semantic meaning. To do so, we used word embeddings and we vectorized the words with pretrained models of Word2Vec and BERT. The former is much more efficient but, since it uses a predefined dictionary, it can only find the embeddings of the words in natural language, and not words which often tables contain such as abbreviations and codes. BERT is able to vectorize all kinds of words and, although it is not always trivial to find embedding of some concepts, it works slightly better than Word2Vec when unnatural keywords are extracted from the tables. When combining the two keyword extraction methods and the two vectorizers, the results showed that LDA is much more accurate than Yake when extracting meaningful keywords. This is because Yake is thought to

be used in texts, which do not have the same structure as tables. Therefore, from the proposed models, LDA combined with Word2Vec is what yields better results when tables contain enough natural language to be processed, while LDA combined with BERT is the appropriate choice when tables contain more abbreviations and other types of words. However, as with many other approaches from previous research, our models can only work when tables have enough textual data to be analysed and evaluated.

Both of the keyword extraction methods proposed output not only the set of keywords but also the importance they have in the table. Hence, we use these weights when determining the similarity between two tables. For the models that compare the set of keywords, we use weighted Jaccard similarity. On the other hand, to compare the embeddings, we use weighted cosine similarity. Once we have the similarity score of the query tables with all the tables in the data lake, we rank them and return the most similar ones.

Finally, to assess the performance of our models, we conducted an evaluation process that involved annotating and labelling the similarity between sample query tables and candidate tables. The chosen evaluation metric for this task was NDCG@10 (Normalised Discounted Cumulative Gain at 10). While other metrics such as MAP (Mean Average Precision) serve a similar purpose, we found that NDCG@10 was better suited for our specific scenario. This choice was primarily driven by the absence of a readily available ground truth against which we could compare the results. Given the unavailability of ground truth, NDCG@10 provided a reliable measure to evaluate the effectiveness of our models in ranking and recommending relevant tables. Additionally, we also assessed the efficiency of the models by analysing their running time. We tracked the execution time during different stages of the models, and it complemented our assessment of accuracy and provided a comprehensive understanding of the overall performance of the proposed models.

5.2 Limitations

Despite the promising results and contributions of this research, there are certain limitations that need to be acknowledged and taken into consideration when interpreting the findings. As aforementioned, the proposed models yield good results when tables contain enough textual data to be analysed. The first limitation of this study is the small size and limited content of the tables. Often tables may contain only a limited number of records or lack the diversity required to extract meaningful keywords. Additionally, a significant portion of them may consist predominantly of numeric values rather than textual data. As a result, the ability to perform comprehensive keyword extraction or analyse the textual content of the tables may be limited, thereby affecting the depth and accuracy of the analysis conducted in this study.

The second limitation pertains to the absence of an available ground truth table for evaluation. In the context of the problem addressed in this thesis, a ground truth table refers to a well-established and validated dataset against which the performance of the proposed approaches can be compared. However, due to the novelty nature of the problem, the lack of a readily available ground truth table poses challenges in evaluating the effectiveness and accuracy of the developed methodologies. Consequently, the evaluation of the proposed approach may rely on alternative means such as expert judgement on the labelling of similarities, introducing potential biases or uncertainties in the assessment process.

5.3 Future work

To mitigate the limitations and to enhance the performance of the proposed models, there are multiple potential improvements that could be explored and implemented. The first one is to try to use as much data as possible in order to better determine what the topic of a table is about. Some table corpora include attribute names and other metadata that we have not used in order to create a data-agnostic model that can work also with table

corpora that do not have this information available. However, it is unfortunate to disregard such valuable information available. Thus, it might be interesting to design a model that can analyse the schema and structure of the table and identify the relevant information that can be used. Then, the attribute name or information such as the title or a little description of the table that sometimes is available, would be also introduced in the keyword extraction methods with a higher weight in order to improve the accuracy of the models when determining the most meaningful words.

Other data that is not analysed in our models are the numerical values. M. Zhang and Chakrabarti (2013) and Trabelsi et al. (2019) propose inspecting them in order to evaluate if they are structured such as dates, times, prices or any other recognizable format. The inclusion of this feature in the models, however, would need a redesign of the whole algorithm since it cannot be easily input in the proposed text analysis. A feature that could be implemented easier could be the normalisation of non-standard words. This involves transforming symbols or abbreviations into normal natural language words. For instance, converting "€" into "euro" or "min" into "minute" would help the model to better extract relevant keywords that can be then vectorized into embeddings.

As a part of the future work, another important step would be to compare the results obtained by other researchers to determine the effectiveness of our proposed approaches relative to the current state of the art models. Benchmarking our models against existing methods can provide insights into their performance and help us understand their strengths and limitations. However, in order to facilitate a fair comparison, it is crucial to establish a proper ground truth that serves as a reliable reference for evaluating the performance of different models. This will enable a comprehensive and objective assessment of the effectiveness of our proposed approaches in the context of existing research.

Finally, the versatility of our proposed model extends beyond table search with query as an example. It can be adapted for various types of dataset discovery tasks. For instance, after transforming candidate tables into key-

words, we can search for tables using a single keyword as a query. This can be done by either transforming the keyword into an embedding or searching for an exact match within the sets of keywords of each of the tables in the data lake. Furthermore, since we treat tables as text, a similar model can be applied to discover other types of documents or schemas that can be converted into text format, not limited to tables alone. This opens up possibilities for leveraging our approach in diverse data discovery scenarios, providing a flexible and scalable solution for discovering relevant datasets across different domains.

In conclusion, this thesis has introduced an innovative approach to table discovery using similarity metrics and keyword extraction. The proposed models have demonstrated promising results in accuracy and efficiency, offering users a valuable tool to expand their data knowledge. Further research can explore enhancements such as incorporating metadata and benchmarking against existing models. Overall, this work contributes to the field of data discovery and opens up possibilities for improved data exploration and informed decision-making.

Bibliography

Bhagavatula, C., Noraset, T., & Downey, D. (2013). Methods for exploring and mining tables on Wikipedia. https://doi.org/10.1145/2501511.2501516

Bhagavatula, C., Noraset, T., & Downey, D. (2015). *TabEL: Entity Linking in Web Tables*. https://doi.org/10.1007/978-3-319-25007-6\{_}25

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022. https://doi.org/10.5555/944919.944937

Bogatu, A., Fernandes, A. A. A., Paton, N. W., & Konstantinou, N. (2020). Dataset Discovery in Data Lakes. https://doi.org/10.1109/icde48307.2020.00067

Boudin, F. (2018). Unsupervised Keyphrase Extraction with Multipartite Graphs. https://doi.org/10.18653/v1/n18-2105

Bougouin, A., Boudin, F., & Daille, B. (2013). TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction. https://hal.science/hal-00917969

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2018). *YAKE! Collection-Independent Automatic Keyword Extractor*. https://doi.org/10.1007/978-3-319-76941-7\{_}80

Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2020). YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences*, *509*, 257–289. https://doi.org/10.1016/j.ins.2019.09.013

Chen, Z., Jia, H., Heflin, J., & Davison, B. D. (2020). *Leveraging Schema Labels to Enhance Dataset Search*. Springer Science+Business Media. https://doi.org/10.1007/978-3-030-45439-5\{_}18

Chi, J., & Roberts, A. (2023). Normalized Discounted Cumulative Gain (NDCG). https://arize.com/blog-course/ndcg/

Deng, L., Zhang, S., & Balog, K. (2019). Table2Vec: Neural Word and Entity Embeddings for Table Population and Retrieval. https://doi.org/10.1145/3331184.3331333

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arXiv.1810.04805v2

Fernandez, R., Min, J., Nava, D., & Madden, S. (2019). Lazo: A Cardinality-Based Method for Coupled Estimation of Jaccard Similarity and Containment. https://doi.org/10.1109/icde.2019.00109

Gagnon, M.-P. (2007). Ontology-based integration of data sources. https://doi.org/10.1109/icif.2007.4408086

Google Code Archive - Word2Vec. (2013). https://code.google.com/archive/p/word2vec/

Guidotti, R., & Monreale, A. (2020). Data-Agnostic Local Neighborhood Generation. https://doi.org/10.1109/icdm50108.2020.00122

Hulsebos, M., Demiralp, Ç., & Groth, P. (2023). GitTables: A Large-Scale Corpus of Relational Tables. *Proc. ACM Manag. Data*, *1*(1). https://doi.org/10.1145/3588710

Hulth, A. (2004). Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction.

Ji, J., Li, J., Yan, S., Tian, Q., & Zhang, B. (2013). *Min-Max Hash for Jaccard Similarity*. https://doi.org/10.1109/icdm.2013.119

Khatiwada, A., Fan, G., Shraga, R., Chen, Z., Gatterbauer, W., Miller, R. J., & Riedewald, M. (2023). SANTOS: Relationship-based Semantic Table Union Search. *Proc. ACM Manag. Data*, *1*(1), 1–25. https://doi.org/10.1145/3588689

Khatua, A., Khatua, A., & Cambria, E. (2019). A tale of two epidemics: Contextual Word2Vec for classifying twitter streams during outbreaks. *Information Processing and Management*, *56*(1), 247–257. https://doi.org/10.1016/j.ipm.2018.10.010

Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A., & Katsifodimos, A. (2021). Valentine: Evaluating Matching Techniques for Dataset Discovery. https://doi.org/10.1109/icde51399.2021.00047

Kusner, M. J., Sun, Y., Kolkin, N. I., & Weinberger, K. Q. (2015). From Word Embeddings To Document Distances, 957–966. https://jmlr.csail.mit.edu/proceedings/papers/v37/kusnerb15.pdf

Langenecker, S., Sturm, C., Schalles, C., & Binnig, C. (2021). Towards Learned Metadata Extraction for Data Lakes. *BTW*, 325–336. https://doi.org/10.18420/btw2021-17

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of Massive Datasets*. Cambridge University Press.

Levy, O., & Goldberg, Y. (2014). Dependency-Based Word Embeddings. https://doi.org/10.3115/v1/p14-2050

Liang, O. (2021). How to measure statistical similarity on tabular data? — demonstrated using synthetic data. https://medium.com/@olivia.liang032/how-to-measure-statistical-similarity-on-tabular-data-demonstrated-using-synthetic-data-66a1aa60084d

Liu, Y., Liu, Z., Chua, T.-S., & Sun, M. (2015). Topical Word Embeddings. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, *29*(1). https://doi.org/10.1609/aaai.v29i1.9522

Miah, M. S. U., Sulaiman, J., Sarwar, T. B., Zamli, K. Z., & Jose, R. (2021). Study of Keyword Extraction Techniques for Electric Double-Layer Capacitor Domain Using Text Similarity Indexes: An Experimental Analysis. *Complexity*, *2021*, 1–12. https://doi.org/10.1155/2021/8192320

Mikolov, T., Chen, K., Corrado, G. S., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. https://arxiv.org/pdf/1301.3781

Mishra, A. (2022). Keyword Extractor YAKE! - Aditya Mishra - Medium. https://medium.com/@adityamishra.rishu/keyword-extractor-yake-35870de21a0d

Mountantonakis, M., & Tzitzikas, Y. (2020). Content-based Union and Complement Metrics for Dataset Search over RDF Knowledge Graphs. *Journal of Data and Information Quality*, *12*(2), 1–31. https://doi.org/10.1145/3372750

Nargesian, F., Zhu, E., Pu, K. Q., & Miller, R. J. (2018). Table union search on open data. *Proceedings of the VLDB Endowment*, *11*(7), 813–825. https://doi.org/10.14778/3192965.3192973

Park, J., & Lee, S.-G. (2011). Keyword search in relational databases. *Knowledge and Information Systems*, *26*(2), 175–193. https://doi.org/10.1007/s10115-010-0284-1

Putri, I., & Kusumaningrum, R. (2017). Latent Dirichlet Allocation (LDA) for Sentiment Analysis Toward Tourism Review in Indonesia. *Journal of physics*, *801*, 012073. https://doi.org/10.1088/1742-6596/801/1/012073

Qaiser, S., & Ali, R. (2018). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International journal of computer applications*, *181*(1), 25–29. https://doi.org/10.5120/ijca2018917395

Rossiello, G., Basile, P., & Semeraro, G. (2017). Centroid-based Text Summarization through Compositionality of Word Embeddings. https://doi.org/10.18653/v1/w17-1003

Santos, A., Bessa, A., Chirigati, F., Musco, C., & Freire, J. (2021). Correlation Sketches for Approximate Join-Correlation Queries. https://doi.org/10.1145/3448016.3458456

Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). Yago. https://doi.org/10.1145/1242572.1242667

Trabelsi, M., Davison, B. D., & Heflin, J. (2019). Improved Table Retrieval Using Multiple Context Embeddings for Attributes. https://doi.org/10.1109/bigdata47090.2019.9005681

Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., Kingsbury, P., & Liu, H. (2018). A comparison of word embeddings for the biomedical natural language processing. *Journal of Biomedical Informatics*, *87*, 12–20. https://doi.org/10.1016/j.jbi.2018.09.008

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., & Rush, A. M. (2020). Transformers: State-of-the-Art Natural Language Processing, 38–45. https://doi.org/10.5281/zenodo.5347031

Yang, Y., Zhang, Y., Zhang, W., & Huang, Z. (2019). GB-KMV: An Augmented KMV Sketch for Approximate Containment Similarity Search. https://doi.org/10.1109/icde.2019.00048

Zhang, K., Xu, H., Tang, J., & Li, J. (2006). *Keyword Extraction Using Support Vector Machine*. Springer Science+Business Media. https://doi.org/10.1007/11775300\{_}8

Zhang, M., & Chakrabarti, K. (2013). InfoGather+: Semantic Matching and Annotation of Numeric and Time-Varying Attributes in Web Tables. https://doi.org/10.1145/2463676.2465276

Zhang, S., & Balog, K. (2021). Semantic Table Retrieval Using Keyword and Table Queries. *ACM Trans. Web*, *15*(3), 1–33. https://doi.org/10.1145/3441690

Zhu, E., Deng, D., Nargesian, F., & Miller, R. J. (2019). JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. https://doi.org/10.1145/3299869.3300065

Zhu, E., Nargesian, F., Pu, K. Q., & Miller, R. J. (2016). LSH ensemble. *Proceedings of the VLDB Endowment*, *9*(12), 1185–1196. https://doi.org/10.14778/2994509.2994534