open proceedings

Dataset Discovery using Semantic Matching

Enas Khwaileh
Utrecht University
Utrecht, The Netherlands
e.t.k.khwaileh@uu.nl

Yannis Velegrakis
Utrecht University & University of Trento
Utrecht, The Netherlands
i.velegrakis@uu.nl

ABSTRACT

The exponential growth of data sizes and heterogeneity has made increasingly challenging to be able to identify datasets that meets specific analytical needs. Traditional keyword search methods often fail in that task since they cannot fully capture the semantics of the datasets and match them to those of the query. We introduce a novel dataset discovery method that significantly enhance both accuracy and retrieval speed. By employing advanced semantic matching at the individual field level and leveraging clustering and dimensionality reduction techniques, our method efficiently and effectively retrieves the datasets related to a query. Unlike traditional methods that focus on syntactic matches, our approach uncovers deeper semantic relationships within table data, providing more precise and relevant results. It achieves this by using transformers to generate and work with embeddings instead of the actual values. We present three different search methods that utilize these embeddings, and experimentally demonstrate the improvement that is achieved when compared to the state-of-the-art.

1 INTRODUCTION

Data federations allow organizations to access datasets located across different physically distributed data sources, without the need of materialized integration [15]. They provide the necessary accessibility and scalability, which is crucial for modern data-driven decision-making [23]. Once accessed, the datasets can be integrated, cleaned, and analyzed to generate valuable business insights [30]. A necessary first step, however, is the ability to identify the right datasets for a task at hand. To do so, a task is typically described though a query, which may be a set of keywords or seed tables [25, 53]. The query, must then be matched to the right datasets that are returned as an answer to the query. This process is referred to as *Dataset Discovery* and is known to pose significant challenges [33, 38]. Matching the query to a dataset based on semantics is of major importance, specifically in dataset federations where there is no global alignment, coordination and indexing.

There have already been a great deal of works for Dataset Discovery given a query. Some focus on matching the query to metadata or text surrounding web tables [33, 56], but these methods often fail to consider the actual content of the tables [6]. Techniques in NL2SQL pipelines, such as schema linking and schema routing, aim to map natural language queries to structured databases by identifying relevant schema components or routing queries to appropriate tables [13, 21]. They enhance database systems by allowing users to interact with structured data without requiring SQL expertise. Nevertheless, NL2SQL systems often rely heavily on predefined schema alignments and high-quality metadata, making them less effective in scenarios with incomplete or poorly maintained schema information. Furthermore, their focus on query-to-SQL translation frequently overlooks semantic relationships between datasets, limiting their applicability in federated systems where tables must be linked or integrated based on deeper contextual relevance. In the field of Information Retrieval, considerable work has been done in identifying the top-k related tables to a given keyword query [16, 47]. Unfortunately, they typically rely on syntactic matching, which often overlooks the deeper, semantic relationships within the data [55]. Some advanced techniques, such as comparing table schemas or applying local sensitivity hashing (LSH) to determine relatedness, have been proposed to improve accuracy [4, 58]. Despite these efforts, challenges remain in discovering datasets that precisely meet user specifications, especially when semantics are involved and the datasets are highly heterogeneous [22, 33]. Platforms like Google Dataset Search[1], Kaggle[2], and Datahub[3] have made it easier for users to locate publicly available datasets through keyword searches. However, these platforms face significant limitations when it comes to real-time dataset discovery. Users are often directed to multiple download links, forcing them to browse through various sources to locate the needed data, which may not always be available [48, 49]. These systems, as well, rely on basic syntactic matching techniques that, as shown in a recent study [3, 22], often fail to capture the true semantic relationships between query terms and datasets.

Another critical challenge in dataset discovery is identifying connections among diverse datasets within a federation. This issue is exacerbated by the rapid growth of heterogeneous datasets, where existing search algorithms and indexing systems designed for homogeneous datasets fall short [33]. For example, systems like Aurum attempt to address this challenge by identifying syntactic relationships between datasets and constructing enterprise knowledge graphs [16]. Other approaches focus on table unionability and joinability, exploring semantic relationships between table columns [34, 40]. While these techniques represent important progress, they still struggle with handling the full complexity of highly heterogeneous datasets and often miss subtle semantic relationships across datasets.

In this work, we propose an approach to dataset discovery that directly tackles the aforementioned challenges by leveraging advanced semantic matching techniques specifically designed for federated environments. Our methods embed tabular datasets at the cell level, capturing deeper semantic relationships across tables that traditional methods overlook. Since embeddings are not inherently reversible, our methods can be used in dataset federations where the datasets are not allowed to leave the original premises, yet they become searchable without compromising privacy or ownership. We introduce and evaluate three distinct search methods: One that is based on exhaustive search (ExS),

[1] https://datasetsearch.research.google.com/
[2] https://www.kaggle.com/
[3] https://datahub.io/search

WHO

Region	Date	Vaccine	Dosage
North America	2021-01-01	Comirnaty	First
Europe	2021-02-01	Vaxzevria	Second
Asia	2021-03-01	CoronaVac	First
Africa	2021-04-01	Covaxin	Second

CDC

State	Date	Immunogen	Manufacturer
California	2021-01-01	mRNA	Moderna
Texas	2021-02-01	Vector Virus	Janssen
Florida	2021-03-01	mRNA	Pfizer
New York	2021-04-01	Protein Subunit	Novavax

ECDC

Country	Date	Trade Name	Disease
Germany	2021-01-01	Pfizer-BioNTech	COVID-19
France	2021-02-01	AstraZeneca	COVID-19
Spain	2021-03-01	Moderna	COVID-19
Italy	2021-04-01	Pfizer-BioNTech	COVID-19

Figure 1: COVID19 Vaccine Dataset

one based on Approximate Nearest Neighbors (ANNS), and one on clustering, referred to as Clustered Targeted Search (CTS). While exhaustive search offers high accuracy by thoroughly examining all potential matches, it soon become computationally expensive. To balance accuracy with efficiency, we developed the ANNS method, which leverages vector databases to speed up the search process. It employs Product Quantization [19] to compress high-dimensional embeddings into smaller subvectors and exploits the Hierarchical Navigable Small World (HNSW) indexing method [29] for an even more efficient search. Our third proposal, the CTS, introduces a novel integration of clustering and dimensionality reduction, allowing it to achieve better search efficiency without compromising in accuracy. These methods outperform the state-of-the-art. Extensive evaluation have shown that they significantly improve the discovery of relevant datasets, especially in large-scale, heterogeneous environments.

The main contributions of this work are the following: (i) We highlight the limitations of the existing works on exploiting semantics for Dataset Discovery and providing a full fledged effective and efficient solution; (ii) we formally define the problem and introduce a novel approach that uses embeddings at the value level (instead of the tuple or table level); (iii) we propose three algorithms, that materialize a solution to the specific problem following different approaches; (iv) We experimentally evaluate the three algorithms and analyze their behavior. Although ExS faces scalability challenges, while ANNS has limitations in achieving high accuracy, CTS is free of these issues; (v) we compare our solutions (and especially the CTS) to the state-of-the-art and demonstrate that our solution outperforms them, confirming the correctness of our choices.

The remainder of this paper is organized as follows. In Section 2, we provide a motivating example to illustrate the challenges of dataset discovery. Section 3 defines formally the problem of dataset discovery, outlining the key objectives and constraints. In Section 4, we introduce the core semantic matching methods, with subsections dedicated to the Exhaustive Search (Sec. 4.1), Approximate Nearest Neighbors Search (Sec. 4.2), and Clustered Targeted Search (Sec. 4.3). The full experimental evaluation is provided in the sequel (Sec. 5), describing datasets, experiments, results and observations.

2 MOTIVATING EXAMPLE

Sarah is a data analyst at a pharmaceutical institution tasked with analyzing COVID-19 vaccine data to assess effectiveness and potential side effects. To accomplish this, she needs to retrieve datasets from various platforms. Figure 1 illustrates a sample of three relations from three such platforms, namely, the World Health Organization (WHO), the Centers for Disease Control and Prevention (CDC), and the European Center for Disease Prevention and Control (ECDC). Sarah searches for relations by posing the keyword "COVID", and clearly she gets as an answer the ECDC relation since they keyword is contained in the tuples of the Disease column. Unfortunately, she wont get the WHO or CDC tables since they contain no COVID keyword. Yet, looking at the tables, one can see that the WHO and the CDC tables are also related and should have been returned to Sarah, since they also talk about COVID. It is just that WHO refers to the COVID vaccines by their type, i.e., it uses the term 'Comirnaty' for the Pfizer-BioNTech vaccine, while CDC uses the Immunogen, i.e., mRNA or vector virus. Furthermore, even with the absence of the Disease column in ECDC, the relation would still be related to COVID since the Trade Names column contains names of COVID vaccines. Sarah's situation underlines the importance of been able to find datasets related to a keyword query even if the terms in the query do not appear in their content. In other words, there is a need for a way to match semantically the query to the datasets and calculate their relatedness.

Domain ontologies have traditionally be used in the past and could be used in Sarah's case to resolve this kind of syntactic discrepancies and link content semantically. However, they are laborious to build and not always available in each practical scenario. Ontologies can answer very precise queries, but in cases of more loose semantics, language technologies may be more effective, since they can capture more loose semantic relationships, and are better in taking context into consideration. This flexibility is of major importance in environments in which there is no global agreement, neither central coordination.

3 PROBLEM STATEMENT

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

A *tuple*, denoted by (t) is a sequence $[a_1, a_2, \ldots, a_n]$, where each $a_i \in \mathcal{A}$ for $i = 1 \ldots n$. The *cardinality* of a tuple is the number of attributes it contains. The *schema* of a tuple $[a_1, a_2, \ldots, a_n]$ is the sequence $[n_1, n_2, \ldots, n_n]$ where n_i is the name of the attribute a_i.

A *relation* is a finite set of tuples that all share the same cardinality and schema. We denote by $\mathcal{R}$ the set of all possible relations. The notion of schema extends naturally to the relation, indicating the common schema shared by all tuples within the relation. A *dataset* is a set of such relations, and a *federation* is a finite set of datasets.

A *query* is defined as a finite set of keywords $(k_1, k_2, \ldots, k_m)$, where each $k_i \in \mathcal{V}$, representing a set of concepts that are of interest to the user. Let $\mathcal{Q}$ represent the set of all possible queries. Given a query q and a federation F, our goal is to find datasets within F that are related to q. To define *relatedness*, we utilize a match function:

$$\text{match} : \mathcal{F} \times \mathcal{Q} \rightarrow \mathbb{R}$$

that returns a score based on some metric, such as cosine similarity in vector space models, used as an indication of the semantic relatedness of the query q and the contents of the dataset. A dataset is considered related if and only if $\text{match}(F, q) \geq h$, where h is a threshold. The aim of this work is to define and

optimize the function match, ensuring it effectively captures the semantic and contextual alignment between a query and datasets in a federation.

We primarily consider datasets consisting of a single relation, although the framework can be generalized to accommodate multi-relation datasets. For this reason, the terms *dataset* and *relation* will be used interchangeably throughout.

4 SEMANTIC MATCH

The main goal of this work is to develop and optimize a semantic match function match($\mathcal{R}, \mathcal{Q}$), which evaluates the degree of alignment between a query $\mathcal{Q}$ and the content of a relation $\mathcal{R}$. This function is important in identifying the most relevant datasets in federated environments, where traditional keyword searches fall short of capturing the contextual nuances embedded in the data. Our objective is to build a match function that is capable of understanding the semantic relationships between queries and datasets, ensuring that the retrieved datasets are contextually appropriate even when they don't contain explicit keyword matches.

To achieve this, we use Natural Language Processing (NLP) techniques, such as Sentence-BERT (S-BERT) [39], as a mean to compute the semantic similarity between the query $\mathcal{Q}$ and the attributes a within the relation $\mathcal{R}$. The semantic match function match($\mathcal{R}, \mathcal{Q}$) leverages S-BERT's sentence embeddings to represent the meaning of both the query and the attributes in a high-dimensional vector space. This allows us to capture the underlying meaning of the terms rather than relying on exact matches, thus enabling more robust dataset discovery.

S-BERT

produces contextualized embeddings of the input tokens that are the dataset attribute values and also the query. The individual attributes are treated as sentences and provided all together to S-BERT, which in turn generates an embedding for each attribute value. The embeddings are passed into a similarity scoring function, such as cosine similarity, which computes the degree of relatedness between the keyword query $\mathcal{Q}$ and an attribute a in a relation $\mathcal{R}$. We define the semantic representation of an attribute $a = \langle n, v \rangle$ as the pair of its attribute name alongside its embedding, i.e., the pair $\langle n, semImg(v) \rangle$, where $semImg(v)$ is the output vector generated by S-BERT. For a tuple t, we define its semantic representation (or the semantic tuple) t' as the set of $\langle n, semImg(v) \rangle \in t'$ pairs for which $\langle n, v \rangle \in t$. The semantic representation of the query $\mathcal{Q}$, denoted as $semImg(\mathcal{Q})$, is the vector representation ω of the query as generated by S-BERT. Finally, we define the semantic representation $semImg(\mathcal{R})$ of a relation $\mathcal{R}$ as the set $\{x \mid x = semImg(t), t \in \mathcal{R}\}$.

The match process involves comparing each attribute a within a relation $\mathcal{R}$ against the query $\mathcal{Q}$. For each query $\mathcal{Q}$, there may be one or more attributes a within the same relation $\mathcal{R}$, and potentially across multiple relations within a federation F. The similarity between these matches is determined by assessing the contextual meanings of a and $\mathcal{Q}$, denoted by $\mathcal{Q}' \simeq a'$. Our objective is to identify a subset $T \subseteq F$ where each relation is related to the query $\mathcal{Q}$, based on the similarity measures between $\mathcal{Q}$ and the elements of $\mathcal{R}$. By treating each attribute $a \in \mathcal{R}$ independently, our approach ensures fine-grained semantic comparisons between attributes and the query $\mathcal{Q}$. This independent evaluation allows the *match* function to capture detailed relationships between each attribute and the query, preserving the contextual meaning embedded in the data. The semantic representation of attributes ensures that individual contributions to the overall similarity score are retained, avoiding potential loss of important signals.

Having the semantic representation of the relations, the next challenge is how to measure the relatedness between a query and a relation. This is done by measuring the similarity between their respective semantic representations. For this we introduce 3 algorithms that we present next. Our overall framework is depicted graphically in Figure 2.

4.1 Exhaustive Search (ExS)

The search process for identifying relevant relations $\mathcal{R}$ within federated databases involves a thorough examination of each relation by evaluating table embeddings and associated vectors. This process begins by transforming both the keyword query Q and each attribute value v within a relation R into semantic vectors, denoted as q' and ω, respectively, using a sentence transformer such as S-BERT. These vector representations allow for a direct comparison between the query and the data. The vector representation process of the data is independent of the one for the query, meaning that the vectors for the data do not depend on the query at hand.

Once the vectors are created, the system calculates a similarity score s between the query vector q' and each attribute vector ω in the relation R. For each relation r, these individual similarity scores are averaged to produce an overall score avg_s, reflecting how well the relation as a whole matches the query. This approach ensures that the relevance of the entire relation is captured, not just individual attributes. After calculating the similarity scores for all relations, they are stored in a result list and sorted in descending order to prioritize the most relevant results.

To further refine the search results, a predefined threshold h is applied, filtering out relations that do not meet the minimum similarity requirement. This ensures that only the most relevant relations are retained. Finally, the top-k relations that exceed the threshold are returned as the most appropriate matches for the query Q. This method guarantees that no relation or attribute is overlooked, making it an essential approach for comprehensively exploring large federated datasets and laying the groundwork for more advanced search techniques.

The process for Algorithm 1 starts by embedding the keyword query Q using a sentence transformer to generate a vector representation q'. Similarly, each attribute v within the relations R in the dataset is transformed into a vector ω using the same transformer. Afterward, the system calculates the similarity score s between the query vector q' and each attribute vector ω. These scores are stored for each relation r, and the average similarity score avg_s is computed to measure the overall relevance of the relation to the query. The results are sorted in descending order, and only those exceeding the predefined threshold h are retained, ultimately returning the top-k most relevant relations.

4.2 Approximate NN Search (ANNS)

Despite the good accuracy achieved by the exhaustive search, it soon becomes inefficient as datasets get larger. To achieve the required efficiency we exploit the Approximate Nearest Neighbor (ANN) technique and a vector database. Each relation R is converted into semantic vectors using a sentence transformer, through the same process used in the exhaustive search.

Algorithm 1: Semantic matching between keyword query Q and relations R using Exhaustive Search

Input : Keyword query Q and a set of Relations R
Output: A ranked list of relations R' that are related to Q

Embed Q using the S-BERT model and obtain q';
Initialize an empty list *ResultList*;
foreach *Relation* $r \in R$ **do**
    Initialize an empty list S;
    foreach *Attribute* $v \in r$ **do**
        Embed v using a sentence transformer and obtain ω;
        Compute the similarity score s between q' and ω;
        Append s to S;
    Compute the average score of all scores in S and assign to avg_s;
    Append (r, avg_s) to *ResultList*;
Sort *ResultList* in descending order based on scores;
R' = Filter out entries from *ResultList* that have scores above a threshold h;
Return the top k entries from R';

Algorithm 2: Semantic matching between keyword query Q and relations R using Approximate Nearest Neighbors Search

Input : Keyword query Q, a set of Relations R
Output: A ranked list of relations R that are related to Q

Step 1: Populate the Vector Database
Initialize a VectorDB with an empty dataset;
foreach *relation* $r \in R$ **do**
    foreach *attribute value* $v \in r$ **do**
        $\omega \leftarrow$ Embed v using a sentence transformer;
        Save ω in VectorDB associated with its metadata (relation ID, attribute name, etc.);
Create HNSW indexing on the VectorDB contents ;

Step 2: Searching Process
$q' \leftarrow$ Embed Q using the sentence transformer;
Return the top k relations from R, ranked according to the approximate KNN and cosine similarity scores for q';

These vectors are stored in a collection having a unique identifier. Product Quantization [19] is a preprocessing technique used to compress high-dimensional embeddings into smaller subvectors. This compression technique is applied on the generated vector embeddings to significantly reduce the storage requirements and the computational costs.

The next step is to identifying the nearest points to the query point. To accelerate this process, instead of systematically checking all the points, a Hierarchical Navigable Small World (HNSW) index is generated [29] and used. The index is a multi-layer structure consisting from hierarchical set of proximity graphs (layers) for nested subsets of the stored elements. The maximum layer in which an element is present is selected randomly with an exponentially decaying probability distribution. Using this hierarchical graph structure the search space is efficiently pruned. As a distance measure between the ω sets, the cosine similarity has been selected also here, so that it is consistent with the the ExS approach, however, other metrics like dot product, or Euclidean distance could have been used. The similarity score of a dataset to the query point is the average of the similarity scores of the vectors of the relation identified by ANN. The ANN Search steps are illustrated in Algorithm 2. It can be seen that the fundamental difference from the ExS steps is that the vectors are stored in the vector database, and the HSNW index created on them, makes obsolete the computation of the similarity score steps.

4.3 Clustered Targeted Search (CTS)

Enhancing search efficiency is crucial, especially in data federation contexts. The Clustered Targeted Search (CTS) method improves upon both ExS and ANNS approaches by incorporating dimensionality reduction and advanced clustering techniques. Specifically, we utilize the Hierarchical Density-Based Clustering (HDBSCAN) algorithm [31], and the reason for using HDBSCAN is because it is well-suited for forming meaningful clusters from non-convex shapes in complex, high-dimensional datasets, such as tabular text embeddings. While HDBSCAN does not automatically provide cluster centers, we address this limitation by manually computing the clusters medoids, which act as representative points for each cluster. These medoids are used to compare the input query with the most relevant clusters.

In managing these clusters efficiently, we store each cluster in a vector database, where each collection contains unique data points, representing distinct entities within the vector space. The medoid of each cluster serves as the index for fast retrieval of search results, facilitating an optimal and focused search process.

The algorithmic steps of CTS are illustrated in Algorithm 3. It starts with transforming each data relation into semantic vectors using a sentence transformer. This step, known as table vectorization, converts the attributes of the table into numerical vectors that represent their meaning. Following this, the vectors are reduced in size using UMAP [32], a dimensionality reduction technique that ensures the vectors are more manageable while still retaining their essential semantic properties. Once the vectors are reduced, they are grouped into clusters using the HDBSCAN algorithm, which organizes the data based on their similarity in high-dimensional space. For each of these clusters, a medoid is calculated, serving as a representative point for the cluster. These medoids are then stored in a vector database to facilitate efficient retrieval during the search process. The input query is also transformed into a vector using the same sentence transformer, allowing for a direct comparison between the query and the cluster medoids. The system compares the query vector against the medoids to find the most similar clusters, focusing on the top-k clusters that are most relevant. From these, the system retrieves the top-k most relevant relations, ensuring the search is confined to the most appropriate clusters. Finally, the system returns the top-k relations based on their relevance to the query, providing a targeted and optimized search result.

The assumption in query processing is that each keyword in a query, referred to as Q, might have a single meaning in exact matching scenarios and potentially multiple meanings in both syntactic and semantic matching. Consequently, a keyword query might match with one or more attributes, labeled here as a, across one or more relations, referred to as $\mathcal{R}$. In both syntactic and semantic matching, the query Q can be linked to one or more attributes a within the same or across multiple datasets, as detailed in Algorithm 3.

Algorithm 3: Semantic matching between keyword query Q and relations R using Clustered Targeted Search (CTS)

Input: Relations R, Keyword query Q, Threshold h, Top-k parameter k
Output: Top k relations from R based on highest scores above h
foreach *relation* r *in* R **do**
  Initialize an empty list S;
  foreach *value* v *in* r **do**
    Convert v to semantic vectors ω using the sentence transformer;
    Apply dimensionality reduction techniques to ω;
    Cluster the reduced vectors using HDBSCAN to create clusters;
  foreach *cluster medoid* m **do**
    Compute the similarity score s between Q and m;
    Append (r, s) to S;
  Sort S in descending order of scores;
  Select the top k clusters from S;
  Compute the average score s of all selected clusters;
  Search inside the top-k clusters using ANNS steps in 4.2;
Sort R in descending order of similarity scores;
Filter out relations from R that have scores below threshold h;
Return the top k relations from R;

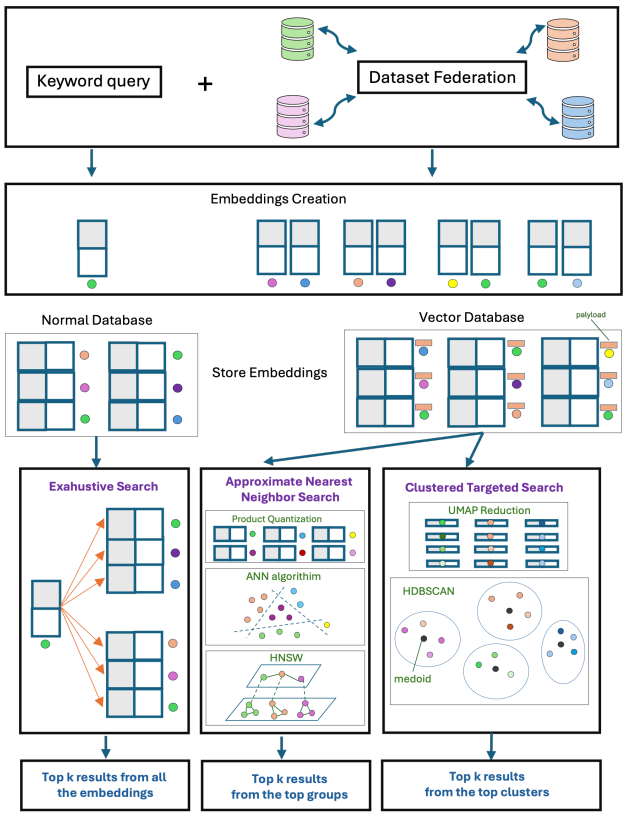

Figure 2: Semantic search dataset discovery framework

Identifying the relevant dataset for a specific query Q within a federation, denoted as F, involves analyzing potential matches for each attribute a across different relations. This analysis interprets possible similar values for each attribute, aiming to find the closest matches for each relation based on the multiple potential interpretations of Q. Previously in this work, a federation has been defined as a collection of relations, each comprising sets of tuples formed by attributes a. To thoroughly scan each relation, it is crucial to maintain the integrity of each relation and the accuracy of the attribute values.

5 EXPERIMENTS AND EVALUATION

[Datasets] The experimental evaluation utilizes two datasets from distinct domains, varying significantly in size, both previously used in state-of-the-art research.
The first dataset, WikiTables [55], comprises 1.6 million tables extracted from Wikipedia, enriched with contextual elements such as page and section titles, headings, and table captions. For this study, we focus on the table body and caption, consolidated into a single column per table, as it was done also in other works that have used that data [55][7]. The relevance of each query-table pair in the WikiTables dataset has been provided [55] alongside the data. The relevance is in a three-point scale: **0 (irrelevant)**, **1 (partially relevant)**, and **2 (fully relevant)**. The relevance data consists of 3,117 query-table pairs.
The second dataset is from the European Data Portal (EDP), which, as reported by Bernhauer [2], uses a similarity-based model for dataset discovery in addition to basic keyword search functionalities. The EDP's system suggests datasets similar to the ones found through keyword searches, using a model known as TLSH. The source code for this feature is available on GitLab [2, 10, 35]. Our test collection includes table corpora from both WikiTables and the EDP, alongside query sets for testing and development. These resources were chosen to evaluate the diversity our methods can handle regarding semantic matching search quality. The significant variance in size between the datasets—over 1.5 million entries in WikiTables compared to a much smaller corpus (around 60K) in the EDP—provides a robust test for our methods' scalability. Both datasets feature tables accompanied by rich contextual details, including titles, captions, and descriptions, essential for our testing framework.

The datasets were partitioned into three scaled-down versions for scalability testing:

- **Small Dataset (SD)**: 10% of the original data,
- **Moderate Dataset (MD)**: 50% of the original data,
- **Large Dataset (LD)**: 100% of the original data.

These partitions allow for the evaluation of the time required to retrieve set R for a series of queries Q, thus gauging system scalability and performance.

[Queries] The queries were designed to simulate realistic user needs and test the semantic retrieval capabilities of our approach. The 60 queries are divided into two subsets: Query Subset 1 (QS-1), referenced in [6], includes queries such as "Beijing Olympics" and "Phases of the Moon." These queries were collected from web users through Amazon's Mechanical Turk, where participants suggested topics or URLs for useful data tables. Query Subset 2 (QS-2), cited in [51], includes queries like "Irish counties area" and "EU countries year joined," sourced from Google Squared's query logs to reflect structured data search needs.

Each query was evaluated against the dataset using semantic matching techniques, where the table content was compared to the query semantically as used in [55]. Queries were categorized into three types based on length:

- **Short Queries (SQ)**: No more than three keywords,
- **Moderate Queries (MQ)**: Up to 30 keywords, typically forming a complete sentence,
- **Long Queries (LQ)**: Containing over 30 keywords, but no more than 300 keywords.

This categorization ensures comprehensive testing across varying levels of query complexity.

The experiment was designed using two main variables: query size and dataset size. These variables were chosen to rigorously test the **quality** of the retrieval set of tables, which vary based on the type of keyword query—ranging from a short query of one or two keywords, to a full sentence, or to a full-text query. Additionally, we assessed the **performance** based on the dataset size and combinations derived from the federation. Each dataset size category—small, moderate, and extensive—was tested with each type of query to examine different performance metrics.

As done in other dataset retrieval works [55], we divided the 3,117 query-table pairs into two sets: 1,918 pairs were used to adjust the multi-field ranking weights, and 1,199 pairs were used to evaluate the performance of our methods.

The implementation of our approach is publicly available.[4]

[Hardware Experimental setup] The experiments were performed on a 112 core CPU's, with 503 GB memory, running Ubuntu 20.04.6 LTS.

[Model Specifications] The S-BERT transformer has three different kinds of models: symmetric, asymmetric, and textual similarity with a loss function. The sentence transformer model used in this work is "all-mpnet-base-v2" [12] is an asymmetric model [5]. The reason why we use this language model to train the dataset is that it is efficient for semantic tasks, and it performs better than the other models in terms of transforming the relations' contents to the corresponding embeddings. The second reason concerns the relation where "all-mpnet-base-v2", performs the best when the keyword query and the matching attribute a are from the same length and have almost the same amount of content, which reflects the need to match the keyword query with the top-k relations.

Numerical values play an eminent role in the relations more than documents, where 26.9% of attributes in WikiTables are numerical values while a random sample from the EDP dataset is 55.3% are numeric. The "all-mpnet-base-v2" transformer model is trained on a more than 1 billion dataset that contains text and numbers and it can distinguish the numerical values according to the context mentioned inside each attribute value. This gives better results matching the query that contains numbers with the list of related tables that contain this number and refer to a piece of specific information. We set the dimensions for the word embedding to 768, and it takes the same size as the MPNet which is a masked language model that combines self-attention and recurrent neural networks.

Other software includes Docker, and Qdrant, which is one of the latest vector database technologies. Qdrant seamlessly accommodated the storage, indexing, and retrieval of high-dimensional vector data, thereby facilitating our data analysis and similarity search endeavors. Augmenting our capabilities by using the sentence transformer (S-BERT).

For the experiment, we orchestrated an environment that was the nexus of innovative technologies, thoughtful methodologies, and intricate processes.

Hierarchical Density-Based Clustering (HDBSCAN) [31] offers the advantages of DBSCAN while also providing performance and efficiency benefits, particularly in its treatment of outliers and its ability to handle non-convex vector shapes.

As our dataset comprised high-dimensional vectors, we recognized the imperative need for dimensionality reduction to enhance the results' efficiency and interpretability. We opted for UMAP (Uniform Manifold Approximation and Projection) [32] among various dimensionality reduction algorithms, including T-SNE (t-Distributed Stochastic Neighbor Embedding). Our choice of UMAP stemmed from its remarkable ability to capture global structures, scalability to accommodate large datasets, and the enhanced control it offers over the embedding process. As a preliminary step, we precomputed the k-nearest neighbors (KNN) calculations required by UMAP to optimize its runtime performance.

[Base Methods] We evaluate the performance of our proposed method by comparing it against four state-of-the-art approaches in table retrieval. Each of these methods focuses on retrieving tables based on either keyword queries or table inputs, providing a strong foundation for assessing the effectiveness of our approach.

- Table Meets LLM (TML) [45]: TML evaluates the structural understanding capabilities of large language models (LLMs) for table retrieval tasks using a benchmark called SUC. It applies advanced serialization techniques to map structured tables and keyword queries into textual input formats that LLMs can process. TML utilizes pretrained LLMs, such as GPT-4, to perform semantic matching between queries and tables. While TML demonstrates promising performance in handling smaller datasets and specific tabular tasks like cell lookup and row retrieval, its reliance on token-limited models restricts its scalability to large datasets. Furthermore, its focus on predefined benchmarks may not generalize well to real-world dataset discovery scenarios. We selected TML as a baseline because it explores the potential of LLMs for semantic matching in structured data and provides a meaningful comparison for evaluating the scalability and efficiency of our methods.
- Table Contextual Search (TCS) [55]: This work applies a learning-to-rank framework for table retrieval by mapping both queries and tables into multiple semantic spaces. It computes several similarity scores between query-table pairs and uses Random Forest regression for ranking. TCS has demonstrated state-of-the-art performance in table retrieval tasks by incorporating both traditional and semantic features. However, TCS's reliance on semantic matching can lead to challenges with ambiguous queries or overlapping terms, and its evaluation on Wikipedia tables may limit generalizability to more diverse or domain-specific datasets. We selected TCS as a baseline because it significantly improves table retrieval performance through semantic matching.
- Ad-Hoc Table Retrieval (AdH) [7]: This method leverages the BERT model for encoding table content, structure, and

[4]https://github.com/enas88/mira_gui
[5]https://huggingface.co/sentence-transformers/all-mpnet-base-v2

metadata. Various content selectors extract row, column, and cell-level information, allowing BERT to process and rank tables effectively. This approach outperforms previous methods in table retrieval by utilizing BERT's ability to capture complex syntactic and semantic relationships, making it a strong baseline for comparison with deep learning-based retrieval systems. However, a key limitation of this approach is that BERT's input length constraint can lead to the truncation of relevant table content, potentially missing important details in larger or more complex tables, especially when the input exceeds BERT's token limit.

- Multi-field Document Ranking (MDR) [36]: This work treats tables as structured documents, using a mixture of language models to rank different fields such as page titles, section titles, and table captions. Each field is scored independently, and then the scores are combined. This multi-field approach provides a comprehensive method for table retrieval, which we compare to our single-cell value representation approach. However, MDR's reliance on individual field scoring may overlook the contextual relationships between fields, leading to less effective retrieval in cases where field interdependencies are crucial.
- WebTable System (WS) [6]: This method focuses on extracting and ranking tables from the web using handcrafted features combined with linear regression models. It serves as a benchmark for traditional table retrieval systems, providing a comparison point against more recent neural-based methods. However, a limitation of this approach is that it relies heavily on manually engineered features, which may not adapt well to more complex, modern datasets or capture deeper semantic relationships in the data compared to newer neural network-based models.

5.1 Evaluation Methodology and Comparison Metrics

The evaluation metrics assesses the system's performance on two key aspects: search quality and system performance.

[Search Quality Metrics] implemented by using:

- **Mean Reciprocal Rank (MRR)**: Measures the precision of the first relevant result, emphasizing early retrieval accuracy.
- **Mean Average Precision (MAP)**: Provides an overall measure of precision across all recall levels, offering a broad view of the system's ability to retrieve relevant results.
- **Normalized Discounted Cumulative Gain (NDCG)**: Evaluates ranking quality by prioritizing higher-ranked relevant results. We calculate NDCG at different cut-off points (5, 10, 15, and 20) to assess performance at various depths.

[System Performance Metrics] Measures the time taken (in milliseconds) to retrieve results, evaluating the system's efficiency in handling queries of different lengths and dataset sizes.

These metrics, combined, offer a comprehensive evaluation of the system's retrieval capabilities, ensuring a balanced view of both ranking quality and computational efficiency.

No

Table 1: Quality of long query results

Dataset	Method	MAP	MRR	NDCG			
				5	10	15	20
LD	CTS	0.705	0.680	0.720	0.700	0.685	0.668
	ANNS	0.685	0.670	0.700	0.675	0.660	0.642
	ExS	0.670	0.655	0.690	0.670	0.650	0.635
	MDR	0.655	0.640	0.675	0.655	0.640	0.625
	WS	0.640	0.625	0.665	0.645	0.630	0.615
	TCS	0.635	0.620	0.660	0.640	0.625	0.610
	AdH	0.620	0.605	0.650	0.630	0.615	0.600
	TML	0.610	0.590	0.650	0.630	0.620	0.610
MD	CTS	0.720	0.700	0.735	0.710	0.695	0.675
	ANNS	0.705	0.690	0.720	0.700	0.680	0.665
	ExS	0.690	0.675	0.710	0.690	0.670	0.650
	MDR	0.675	0.660	0.700	0.680	0.660	0.645
	WS	0.660	0.645	0.690	0.670	0.650	0.635
	TCS	0.655	0.640	0.680	0.660	0.640	0.625
	TML	0.650	0.630	0.670	0.650	0.635	0.620
	AdH	0.640	0.625	0.675	0.655	0.635	0.620
SD	CTS	0.735	0.715	0.750	0.725	0.710	0.690
	ANNS	0.720	0.700	0.740	0.715	0.700	0.685
	ExS	0.705	0.690	0.730	0.710	0.690	0.671
	MDR	0.690	0.675	0.720	0.700	0.685	0.670
	TML	0.690	0.670	0.710	0.690	0.675	0.660
	WS	0.675	0.660	0.710	0.690	0.675	0.660
	TCS	0.670	0.655	0.705	0.685	0.670	0.655
	AdH	0.655	0.640	0.695	0.675	0.660	0.645

Table 2: Quality of moderate query results

Dataset	Method	MAP	MRR	NDCG			
				5	10	15	20
LD	CTS	0.755	0.730	0.770	0.750	0.735	0.720
	ANNS	0.735	0.715	0.760	0.740	0.725	0.710
	ExS	0.720	0.700	0.745	0.725	0.710	0.695
	MDR	0.710	0.690	0.740	0.720	0.705	0.690
	WS	0.700	0.680	0.730	0.710	0.695	0.680
	TCS	0.690	0.670	0.725	0.705	0.690	0.675
	AdH	0.675	0.655	0.710	0.690	0.675	0.660
	TML	0.620	0.600	0.650	0.630	0.620	0.610
MD	CTS	0.770	0.745	0.780	0.760	0.745	0.730
	ANNS	0.755	0.735	0.775	0.755	0.740	0.725
	ExS	0.740	0.720	0.760	0.740	0.725	0.710
	MDR	0.725	0.705	0.755	0.735	0.720	0.705
	TML	0.710	0.690	0.750	0.680	0.715	0.700
	WS	0.705	0.685	0.745	0.725	0.710	0.695
	TCS	0.690	0.670	0.735	0.715	0.700	0.685
	AdH	0.680	0.660	0.700	0.680	0.670	0.660
SD	CTS	0.785	0.765	0.795	0.780	0.765	0.750
	ANNS	0.770	0.750	0.785	0.770	0.755	0.740
	ExS	0.755	0.735	0.780	0.765	0.750	0.735
	TML	0.740	0.720	0.770	0.755	0.740	0.725
	WS	0.730	0.710	0.765	0.745	0.730	0.715
	MDR	0.720	0.700	0.750	0.730	0.720	0.710
	TCS	0.715	0.695	0.755	0.735	0.720	0.705
	AdH	0.705	0.685	0.745	0.725	0.710	0.695

5.2 Search Quality

The results of our experiments demonstrate a clear distinction in performance between our proposed methods and the base methods across different query lengths (long, moderate, and short) and dataset sizes (100%, 50%, and 10%). The evaluation of retrieval quality was carried out using metrics such as Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG) at different cut-off levels (5, 10, 15, and 20). Below, we discuss the effectiveness of each method, highlighting the strengths and weaknesses of both our proposed techniques and the baseline methods.

5.2.1 Long Queries. For long queries, the Clustered Targeted Search (CTS) consistently outperformed all other methods in terms of retrieval quality. As seen in Table 1, CTS achieved the

Table 3: Quality of short query results

Dataset	Method	MAP	MRR	NDCG			
				5	10	15	20
LD	CTS	0.810	0.790	0.825	0.810	0.800	0.785
	ANNS	0.790	0.770	0.810	0.790	0.780	0.765
	ExS	0.770	0.750	0.800	0.780	0.770	0.755
	TML	0.755	0.735	0.785	0.770	0.760	0.745
	MDR	0.740	0.720	0.780	0.760	0.750	0.735
	WS	0.725	0.705	0.775	0.755	0.745	0.730
	TCS	0.710	0.690	0.760	0.740	0.730	0.715
	AdH	0.650	0.630	0.670	0.650	0.640	0.630
MD	CTS	0.825	0.805	0.835	0.820	0.810	0.795
	ANNS	0.805	0.785	0.825	0.805	0.795	0.780
	TML	0.790	0.770	0.810	0.790	0.780	0.765
	ExS	0.775	0.755	0.805	0.785	0.775	0.760
	MDR	0.760	0.740	0.795	0.775	0.765	0.750
	WS	0.745	0.725	0.785	0.765	0.755	0.740
	TCS	0.730	0.710	0.775	0.755	0.745	0.730
	AdH	0.700	0.680	0.730	0.710	0.700	0.690
SD	CTS	0.830	0.810	0.840	0.820	0.810	0.800
	TML	0.815	0.795	0.835	0.815	0.805	0.790
	ANNS	0.800	0.780	0.825	0.805	0.795	0.780
	ExS	0.785	0.765	0.820	0.800	0.790	0.775
	MDR	0.770	0.750	0.810	0.790	0.780	0.765
	WS	0.755	0.735	0.805	0.785	0.775	0.760
	TCS	0.750	0.730	0.770	0.750	0.740	0.730
	AdH	0.740	0.720	0.795	0.775	0.765	0.750

highest MAP of 0.705 and a corresponding NDCG@20 score of 0.668 on the full dataset. This demonstrates the method's capability to retrieve the most contextually relevant tables while maintaining strong ranking performance. The ANNS method followed closely, with a MAP of 0.685 and NDCG@20 of 0.642, confirming that while ANNS is highly efficient in search, it slightly lags behind CTS in ranking precision for long queries. ExS performed reasonably well, with a MAP of 0.670 and NDCG@20 of 0.635. However, it was outperformed by both CTS and ANNS due to its reliance on brute-force search, which does not incorporate clustering for more focused retrieval.

The baseline methods exhibited varying performance. Multi-field Document Ranking (MDR) and WebTable Search (WS) dropped noticeably in performance, with MAP values of 0.655 and 0.640, respectively, due to their reliance on surface-level features that fail to capture deeper semantic relationships for long, contextually complex queries. Ad-Hoc Table Retrieval (AdH), leveraging BERT, performed slightly worse, with a MAP of 0.620 and NDCG@20 of 0.600, as its token length constraints led to truncation of relevant data, impacting precision for larger, more complex tables. Table Contextual Search (TCS) scored similarly to AdH, struggling with ambiguous queries and overlapping terms due to its reliance on semantic spaces for ranking.

The Table Meets LLM (TML) method scored the lowest among all methods for long queries on large datasets, with a MAP of 0.610 and NDCG@20 of 0.610. This can be attributed to TML's reliance on token-limited models like GPT-4, which struggle to process serialized representations of large datasets effectively. Additionally, TML's focus on benchmarks like SUC does not generalize well to real-world dataset discovery, further limiting its capability to handle complex queries involving large tables. However, as the dataset size decreased (e.g., medium dataset MD), TML showed modest improvement, achieving a MAP of 0.650 and NDCG@20 of 0.620, benefiting from reduced data complexity. On small datasets (SD), TML performed better, achieving a MAP of 0.690 and NDCG@20 of 0.660. This indicates that TML's semantic matching excels when operating within the constraints of smaller data environments, though it remains inferior to CTS and ANNS due to its token limitations and lack of optimization for large-scale table retrieval.

5.2.2 Moderate Queries. When evaluating moderate-length queries, CTS again demonstrated superior performance, as shown in Table 2. With a MAP of 0.755 and an NDCG@20 score of 0.720 on the full dataset, CTS outperformed all methods in terms of both precision and ranking quality. The ANNS method achieved a MAP of 0.735, indicating its ability to maintain a strong balance between search speed and retrieval quality.

The performance gap between CTS and ANNS for moderate queries was smaller compared to long queries. This suggests that as the query length decreases, ANNS's efficiency becomes more competitive with CTS. However, ExS trailed both CTS and ANNS, with a MAP of 0.720, reflecting its brute-force reliance, which becomes less effective as datasets increase in size.

Among the baseline methods, MDR and WS performed better with moderate queries than they did with long queries, achieving MAP values of 0.710 and 0.700, respectively. These methods rely on simpler scoring mechanisms that are better suited for moderately complex queries. Ad-Hoc Table Retrieval continued to lag with a MAP of 0.675 due to BERT's token constraints, and TCS similarly struggled with overlapping query terms.

TML displayed a noticeable improvement for moderate queries compared to long ones. On medium datasets (MD), TML achieved a MAP of 0.680 and NDCG@20 of 0.620. This reflects its ability to leverage semantic matching more effectively in scenarios where dataset complexity is reduced. On small datasets (SD), TML performed even better, with a MAP of 0.740 and NDCG@20 of 0.725, coming closer to ExS and ANNS. However, on large datasets (LD), TML remained limited, with a MAP of 0.620 and NDCG@20 of 0.610, as its reliance on token-limited LLMs restricted its ability to process large-scale serialized data effectively for moderate queries.

5.2.3 Short Queries. For short queries, CTS maintained its lead in retrieval quality, with a MAP of 0.810 and an NDCG@20 score of 0.785, as shown in Table 3. This further solidifies CTS as the top-performing method across all query lengths, benefiting from its clustering mechanism that efficiently captures contextual relevance. ANNS followed closely with a MAP of 0.790, illustrating its efficiency in handling shorter queries without compromising ranking quality.

ExS also performed well for short queries, with a MAP of 0.770 and NDCG@20 of 0.755, though it remained behind CTS and ANNS. Among the baseline methods, MDR achieved a MAP of 0.755, while WS scored 0.740, showing reasonable effectiveness for short queries but lacking the contextual depth required to match CTS and ANNS. Ad-Hoc Table Retrieval again scored the lowest among the baselines, with a MAP of 0.710, due to its token truncation issues when handling even focused, short-query datasets.

TML demonstrated its best performance for short queries, particularly on small datasets (SD), where it achieved a MAP of 0.815 and NDCG@20 of 0.790. This improvement stems from TML's semantic matching capabilities being more impactful in focused query scenarios with limited data complexity. On medium datasets (MD), TML also performed well, achieving a MAP of 0.790 and NDCG@20 of 0.765, leveraging its pre-trained LLM strengths. However, on large datasets (LD), TML scored lower, with a MAP of 0.755 and NDCG@20 of 0.745, due to token limitations and its inability to handle the scale of serialized long tables effectively, even for short queries.

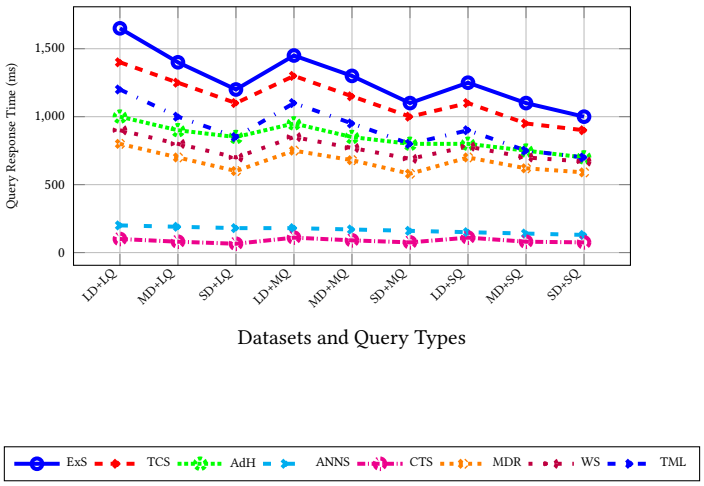

Figure 3: Query Response Time for Different Methods Across Various Dataset and Query Sizes

5.2.4 Evaluation Summary. CTS emerged as the most effective method across all query lengths and dataset sizes, achieving the highest MAP, MRR, and NDCG scores. Its clustering and semantic matching approach demonstrated unmatched robustness, making it the central contribution of this work. ANNS followed as a strong alternative, particularly excelling in shorter queries with a balance between speed and quality.

TML revealed the limitations of token-limited LLMs, struggling with large datasets and long queries but showing improvement in moderate and short queries on smaller datasets. These results highlight the potential of LLMs in constrained scenarios while emphasizing their need for optimization in large-scale retrieval.

Exhaustive Search, although reliable, lagged behind CTS and ANNS, especially for large datasets. Baseline methods like MDR and WS showed moderate success with shorter queries. AdH's consistent underperformance, underlines the superiority of CTS

5.3 CTS versus ExS versus ANNS

For the query "Climate Change Effects Europe 2020," Exhaustive Search (ExS), despite its thorough approach, did not retrieve the most relevant tables as expected. ExS searches every attribute in every table, but this exhaustive method averages similarity scores across all attributes. This leads to a dilution of relevance, as tables with general global climate change data or from different years can rank higher than tables specifically about Europe in 2020. This broad approach causes ExS to return less precise matches, missing the focus on the region and year that are central to the query.

Approximate Nearest Neighbors Search (ANNS) improves efficiency by approximating the nearest matches. While it successfully retrieved tables related to climate change and Europe, ANNS still missed the fine-grained distinction between the region (Europe) and the specific year (2020). This happens because the approximation sacrifices some accuracy, resulting in tables that may focus on Europe across multiple years or on other regions in 2020. Though ANNS provides faster results, its tendency to blend context weakens its ability to match detailed, context-specific queries like this one.

Clustered Targeted Search (CTS), on the other hand, provided better results by clustering semantically related tables. For the specific query, CTS focused on clusters containing tables specifically discussing climate change impacts in Europe during 2020. By leveraging cluster medoids to represent the most central points, CTS avoided the noise of irrelevant global data or mixed timelines. This allowed it to precisely target the key components of the query retrieving tables with detailed information on regional impacts, policy changes, and specific effects of climate change. CTS's ability to isolate relevant clusters made it both efficient and accurate, outperforming both ExS and ANNS in delivering the most relevant tables.

Table 4: Query Time (milliseconds) for CTS vs. ANNS

Dataset	Query	CTS	ANNS
100%	Long	75	100
	Moderate	85	90
	Short	110	150
50%	Long	70	75
	Moderate	80	120
	Short	80	130
10%	Long	65	95
	Moderate	75	100
	Short	75	115

5.4 Performance Evaluation

To understand how the different methods scale up, we have tested them for the different data sizes ((100%, 50%, and 10% of the original dataset) and for different query sizes (long, medium, and short). The results (both of our methods but also those we have used as base line) are illustrated in Figure 3 and Table 4.

The goal is to identify the most efficient method capable of delivering fast query results while maintaining retrieval quality. As shown in Table 4, CTS consistently outperforms other methods in terms of speed, especially with larger datasets. The combination of clustering and dimensionality reduction allows CTS to achieve the fastest response times, particularly for long queries (75 ms for the 100% size dataset) and short queries (110 ms for the 100% size dataset). ANNS demonstrates competitive performance, particularly for medium and small datasets. However, it still falls slightly behind CTS in terms of query speed. For example, ANNS took 100 ms for long queries on the 100% dataset and 115 ms for short queries on the 10% dataset, indicating a slight lag compared to CTS's performance. For moderate queries, CTS

On the other hand, ExS, while being accurate, suffers from much slower response times due to its comprehensive nature. ExS took 1,650 ms for long queries on the 100% dataset, a notable decrease in performance compared to the other methods. This inefficiency becomes more pronounced as the dataset size increases, making it impractical for real-time search in large-scale federated systems. Comparing to baseline methods, both TCS and AdH show slower response times. TCS, for instance, requires 1,400 ms for long queries on the 100% dataset, while AdH took 1,000 ms for the same task. These slower times reflect the limitations of more traditional or exhaustive matching techniques in handling large-scale data effectively. Furthermore, MDR and WS perform moderately well but cannot match the efficiency of CTS and ANNS. For example, MDR requires 800 ms for long queries on the 100% dataset, while WS took 900 ms, both slower than CTS and ANNS. These results confirm that while traditional techniques can offer reasonable performance, they fall short in terms

of both speed and scalability when compared to more advanced methods like CTS.

TML, while effective on smaller datasets, showed slower response times on larger datasets due to the token-limited nature of LLMs and the serialization overhead. It took 1,200 ms for long queries on the full dataset, improving to 1,000 ms on medium datasets and 850 ms on smaller datasets. Although TML performed better with shorter datasets, it lagged behind CTS and ANNS in handling large-scale data due to its reliance on LLMs and processing constraints. However, its performance on smaller datasets highlights its potential in constrained environments.

6 RELATED WORKS

There have been different works that have studied the Dataset Discovery challenge. Some have explored methods like exact keyword match, fuzzy matching, or semantic matching, each with its own strengths and limitations. Furthermore, approaches such as joinability and unionability matching have been proposed for identifying similar datasets in structured databases, where matching based on column compatibility or schema alignment enables more efficient integration. These methods enhance dataset discovery but require advanced techniques to address inconsistencies across heterogeneous sources.

[Exact Matching using Keyword Queries] In dataset discovery, exact keyword matching has been frequently applied, particularly in structured environments like data lakes. For instance, it has been used to efficiently retrieve tables containing specific terms in different domains [44]. Despite its usefulness, this approach faces limitations such as failing to capture contextual nuances or variations in terminology. Additionally, it often struggles with polysemy, where multiple meanings of the same keyword hinder retrieval accuracy. Zou and Yang addressed scalability by developing a search engine within the Hadoop ecosystem, but as data volumes grew, challenges related to managing increasing amounts of scientific data emerged [61]. The rigidity of exact matching was further highlighted in Zhu's work, where the dependence on metadata quality in data lakes limited search flexibility, making the approach prone to inefficiencies when metadata is poorly maintained [59]. Similarly, a survey on data lake management pointed out that over-reliance on metadata significantly diminishes the effectiveness of keyword matching, particularly in large-scale datasets where metadata quality cannot always be guaranteed [5].

[Fuzzy Matching discovery] Fuzzy matching is widely adopted method in dataset discovery, particularly in domains like finance, where exact name matches are uncommon. For example, its utility in retrieving financial tables has been well demonstrated, showing how it can handle cases of name variation and misspellings [57]. However, fuzzy matching is not without drawbacks. It can produce false positives, leading to ambiguity in results and reducing accuracy. The computational cost is also a significant challenge, as fuzzy matching algorithms often require substantial processing power, which can increase response times and operational costs. Fine-tuning these algorithms to strike the right balance between accuracy and performance is often a time-consuming process. Moreover, the method's success is heavily dependent on the quality of the data, making proper data cleaning and preparation essential.

To address some of these limitations, Lee and Lee [24] enhanced fuzzy matching with deep learning, which improved retrieval performance in large-scale industrial datasets. Another method extended the fuzzy matching approach for Industry 4.0 applications, where careful parameter tuning was required to avoid overfitting during data integration [26]. Expanding on this, Sun [46] applied fuzzy matching to multi-source heterogeneous data in the electric power sector, showcasing the broad adaptability of this method across various industries, despite its inherent complexities.

[Semantic Matching]. Semantic matching, a sophisticated approach in dataset discovery, has evolved significantly from its theoretical origins to practical implementations across various fields. Early foundational work [17] introduced semantic matching within the context of non-monotonic reasoning, which provided a conceptual basis for future developments in this domain. Later on, these ideas were expanded [11] into a comprehensive framework on ontology matching.

The notable challenges in applying semantic matching, particularly in the semantic web, were highlighted when trying to map ontologies using machine learning techniques [9]. Practical applications of this technique have been demonstrated [14], using semantic matching in the retail industry to effectively link related tables, outperforming traditional exact or fuzzy matching methods.

In the context of data lakes, Singh and Sahoo employed semantic approaches to manage metadata, focusing on issues of scalability and data quality [43]. While significant advancements have been made, such as Hassan's work on big data integration and Lin's graph-based models for semantic data lakes [18, 28], the method still faces several obstacles. These include high computational demands, a dependency on high-quality metadata, and integration challenges with existing systems, issues that have been thoroughly explored in surveys on schema matching approaches [1, 37].

The semantic matching holds great potential, and its successful application requires a robust infrastructure and well-maintained metadata to support efficient discovery and analysis in federated systems. The choice of approach will ultimately depend on the specific context and needs of the application.

[Joinability and Unionability Matching] Several studies have tackled the challenge of identifying joinable and unionable tables in data lakes, but many struggle to handle semantic matching at scale. For instance, TUS and Santos primarily focus on unionability detection but are limited by small datasets, restricting their ability to scale and generalize to larger corpora [41, 54]. Similarly, Josie used set similarity to find joinable tables through exact overlaps, but it does not address the semantic relationships between columns, which are crucial in complex datasets [52].

LSH Ensemble offers a scalable solution for estimating set containment via MinHash, yet its efficiency diminishes as the dataset size increases due to the overhead of partitioning and indexing [42]. On the other hand, DeepJoin employs deep learning models such as DistilBERT to improve semantic matching, but its performance suffers when dealing with large-scale datasets that require both exact and semantic matching [60].

Efforts like D3L attempt to handle unionability by analyzing various factors including attribute names and word embeddings, yet the computational costs are prohibitive for large datasets [27]. Similarly, InfoGather uses a graph-based approach to link joinable and unionable tables, but its high memory consumption limits its applicability to large-scale data lakes [50].

LakeBench introduces a large-scale benchmark with over 16 million tables to evaluate joinability and unionability methods.

Although it provides a comprehensive dataset for evaluation, the challenge of capturing nuanced semantic matching between query tables and large datasets remains, especially in diverse, heterogeneous data lakes [8].

[Schema Linking and NL2SQL Pipelines] Existing works on schema linking and natural language interfaces for databases have made significant progress in enhancing the performance of NL2SQL systems. However, these approaches often struggle with semantic table retrieval, particularly in large-scale and heterogeneous environments. One method introduces schema alignment techniques to improve SQL query generation but relies heavily on pre-defined schema structures, limiting its ability to generalize to diverse datasets and complex queries [21]. Another framework explores combining small and large language models to achieve zero-shot NL2SQL capabilities, yet it faces challenges in handling nuanced semantic relationships within schemas, resulting in limited effectiveness for retrieving semantically related tables at scale [13]. Deep learning-based interfaces have been proposed to link natural language queries with database schema components, improving query translation accuracy. However, their reliance on deep schema linking and dependency on the quality of metadata make them less robust in environments where metadata is incomplete or poorly maintained [20].

7 CONCLUSION

We studied the problem of finding a dataset that is related to a query. We have provided a method that uses text embeddings to accurately identify the dataset that is semantically related to a given query. We evaluated three semantic matching methods: Exhaustive Search, Approximate Nearest Neighbors Search (ANNS), and Clustered Targeted Search (CTS). Exhaustive Search, while producing highly accurate results, was slower, particularly when processing large datasets. ANN Search dramatically reduced search time while preserving a substantial degree of accuracy. Clustered Targeted Search further enhanced efficiency and reduced storage requirements by applying dimensionality reduction techniques, though it came with a slight loss in accuracy. Our experimental evaluation of these methods against multiple state-of-the-art works confirms its better efficiency and effectiveness.

REFERENCES

[1] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm (Eds.). 2011. *Schema Matching and Mapping*. Springer, .

[2] David Bernhauer, Martin Nečaskỳ, Petr Škoda, Jakub Klímek, and Tomáš Skopal. 2022. Open dataset discovery using context-enhanced similarity search. *Knowledge and Information Systems* 64, 12 (2022), 3265–3291.

[3] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2013. Methods for exploring and mining tables on Wikipedia. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, IDEA@KDD 2013, Chicago, Illinois, USA, August 11, 2013*, Duen Horng Chau, Jilles Vreeken, Matthijs van Leeuwen, and Christos Faloutsos (Eds.). ACM, 18–26.

[4] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 709–720.

[5] Vivek Borkar, Michael Carey, Chen Li, and Sharad Mehrotra. 2017. Data lake management challenges: A survey. In *Proceedings of the 5th Workshop on Managing and Mining Enriched Geo-Spatial Data*. 1–6.

[6] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. 2009. Data Integration for the Relational Web. *Proc. VLDB Endow.* 2, 1 (2009), 1090–1101. http://www.vldb.org/pvldb/vol2/vldb09-576.pdf

[7] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D. Davison. 2020. Table Search Using a Deep Contextualized Language Model. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 589–598.

[8] Yuhao Deng, Chengliang Chai, Lei Cao, et al. 2024. LakeBench: A Benchmark for Discovering Joinable and Unionable Tables in Data Lakes. *Proceedings of the VLDB Endowment* 17, 8 (2024), 1925–1938.

[9] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. 2002. Learning to map between ontologies on the semantic web. In *WWW*. ACM, 662–673.

[10] Simon Dutkowski and Andreas Schramm. 2015. Duplicate evaluation-position paper by Fraunhofer FOKUS.

[11] Jérôme Euzenat and Pavel Shvaiko. 2007. *Ontology matching*. Springer.

[12] Hugging Face. [n. d.]. Sentence-Transformers from the Hugging Face and semantic matching. ([n. d.]). https://huggingface.co/sentence-transformers/all-mpnet-base-v2.

[13] Cheng Fan, Wei Huo, Haoyan Yu, Chao Zhang, and Ce Song. 2022. Combining Small Language Models and Large Language Models for Zero-Shot NL2SQL. *Proceedings of the VLDB Endowment* 17, 11 (2022), 2750–2762. https://www.vldb.org/pvldb/vol17/p2750-fan.pdf

[14] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *Companion of the 2023 International Conference on Management of Data, SIGMOD/PODS 2023, Seattle, WA, USA, June 18-23, 2023*, Sudipto Das, Ippokratis Pandis, K. Selçuk Candan, and Sihem Amer-Yahia (Eds.). ACM, 69–75.

[15] Huang Fang. 2015. Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE, 820–824.

[16] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1001–1012.

[17] Fausto Giunchiglia and Pavel Shvaiko. 2003. Semantic matching as a non-monotonic reasoning service. In *ICSW*.

[18] Md Mokammel Hassan, Funmilade Faniyi, and Li Chen. 2018. A semantic similarity-based approach for data integration in big data environment. *Future Generation Computer Systems* 83 (2018), 413–425.

[19] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[20] Alexander Katsogiannis-Meimarakis, Aris Karagiannis, Marianthi Drosou, and Evaggelia Pitoura. 2023. Natural Language Interfaces for Databases with Deep Learning. *Proceedings of the VLDB Endowment* 16, 14 (2023), 3878–3890. https://www.vldb.org/pvldb/vol16/p3878-katsogiannis-meimarakis.pdf

[21] Byungsoo Kim, Jiwon Chung, Ji-Hoon Na, Yong-Hyun Heo, Hwanjo Lee, and Sang-Wook Cha. 2020. Natural Language to SQL: Where Are We Today? *Proceedings of the VLDB Endowment* 13, 13 (2020), 1737–1750. https://www.vldb.org/pvldb/vol13/p1737-kim.pdf

[22] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 468–479.

[23] Eddine Laouir, Ala, Imine, and Abdessamad. 2024. Private Approximate Query over Horizontal Data Federation. (2024). arXiv.

[24] Young Jae Lee and Junghwan Lee. 2019. A deep learning-based data lake search engine for information retrieval in industrial big data environments. *Journal of Industrial Information Integration* 13 (2019), 30–39.

[25] Aristotelis Leventidis, Martin Pekár Christensen, Matteo Lissandrini, Laura Di Rocco, Katja Hose, and Renée J. Miller. 2024. A Large Scale Test Corpus for Semantic Table Search. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (Eds.). ACM, 1142–1151.

[26] Jiaojiao Li, Zhikui Li, Rui Zhang, Guoliang Sun, and Xuefeng Li. 2019. A fuzzy name matching algorithm for data integration in Industry 4.0. *IEEE Access* 7 (2019), 84110–84122.

[27] Xiaodong Li et al. 2020. D3L: Deep Learning Enhanced Discovery of Unionable Tables in Data Lakes. *Proceedings of the VLDB Endowment* 13, 5 (2020), 635–647.

[28] Yuting Lin, Yan Zhao, Xiaohong Sun, Qiong Chen, and Ying Zhang. 2021. A graph-based approach for semantic data lake modeling and searching. *Journal of Systems and Software* 175 (2021), 110900.

[29] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[30] Christian Mathis. 2017. Data lakes. *Datenbank-Spektrum* 17, 3 (2017), 289–293.

[31] Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.* 2, 11 (2017), 205.

[32] Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[33] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data lake management: challenges and opportunities. *VLDB* 12, 12 (2019), 1986–1989.

[34] Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2018. Table union search on open data. *Proceedings of the VLDB Endowment* 11, 7 (2018), 813–825.

[35] Jonathan Oliver, Chun Cheng, and Yanggui Chen. 2013. TLSH–a locality sensitive hash. In *2013 Fourth Cybercrime and Trustworthy Computing Workshop*. IEEE, 7–13.

[36] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *arXiv preprint arXiv:1207.0132* (2012).

[37] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *The VLDB Journal—The International Journal on Very Large Data Bases* 10, 4 (2001), 334–350.

[38] Franck Ravat and Yan Zhao. 2019. Data Lakes: Trends and Perspectives. In *Database and Expert Systems Applications - 30th International Conference, DEXA 2019, Linz, Austria, August 26-29, 2019, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 11706)*, Sven Hartmann, Josef Küng, Sharma Chakravarthy, Gabriele Anderst-Kotsis, A Min Tjoa, and Ismail Khalil (Eds.). Springer, 304–313.

[39] Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084* (2019).

[40] Aécio S. R. Santos, Aline Bessa, Fernando Chirigati, Christopher Musco, and Juliana Freire. 2021. Correlation Sketches for Approximate Join-Correlation Queries. In *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). ACM, 1531–1544.

[41] Daniel Santos et al. 2017. Discovering Unionable Tables. *Proceedings of the VLDB Endowment* 10, 12 (2017), 1818–1829.

[42] Jianfeng Shang, Jiahui Jin, Yuliang Li, and Ji-Rong Wen. 2019. LSH Ensemble: Scaling Minhash for Finding Joinable Columns in Data Lakes. *Proceedings of the VLDB Endowment* 12, 9 (2019), 1116–1128.

[43] Pankesh Singh and Satya K Sahoo. 2017. Semantic-based metadata management in data lakes. In *Proceedings of the 13th International Conference on Semantic Systems*. 1–8.

[44] HV Sreepathy, B Dinesh Rao, J Mohan Kumar, and B Deepak Rao. 2024. Data Discovery as a Service for Data Lake. In *2024 Second International Conference on Networks, Multimedia and Information Technology (NMITCON)*. IEEE, 1–9.

[45] Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets LLM: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 645–654.

[46] Xiaohong Sun, Lianhui Zhao, Bo Zhao, and Yulong Lu. 2020. Fuzzy matching of multi-source heterogeneous big data in the electric power industry. *IEEE Access* 8 (2020), 166618–166629.

[47] Yibo Sun, Zhao Yan, Duyu Tang, Nan Duan, and Bing Qin. 2019. Content-based table retrieval for web queries. *Neurocomputing* 349 (2019), 183–189.

[48] Nasser Thabet and Tariq Rahim Soomro. 2015. Big Data Challenges. *Journal of Computer Engineering & Information Technology* 4, 3 (2015), 1–3.

[49] Alexandru Adrian Tole. 2013. Big data challenges. *Database systems journal* 4, 3 (2013).

[50] Rahul Trivedi et al. 2017. InfoGather: Augmenting Tables by Discovering Joinable and Unionable Tables. *VLDB* 11, 12 (2017), 1508–1520.

[51] Petros Venetis, Alon Y. Halevy, Jayant Madhavan, Marius Pasca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering Semantics of Tables on the Web. *Proc. VLDB Endow.* 4, 9 (2011), 528–538. http://www.vldb.org/pvldb/vol4/p528-venetis.pdf

[52] Jun Wang et al. 2018. Josie: Efficient Joinability Search Across Large Data Lakes. *Proceedings of the SIGMOD Conference* (2018), 343–355.

[53] Megan Wong, Kerry Levett, Ashlin Lee, Paul Box, Bruce Simons, Rakesh David, Andrew MacLeod, Nicolas Taylor, Derek Schneider, and Helen Thompson. 2022. Development and governance of FAIR thresholds for a data federation. *Data Science Journal* 21 (2022), 13–13.

[54] Yue Xu et al. 2020. TUS: Scalable Table Union Search via Semantic Matching. *Proceedings of the VLDB Endowment* 13, 9 (2020), 1121–1134.

[55] Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the 2018 world wide web conference*. 1553–1562.

[56] Yi Zhang and Zachary G Ives. 2020. Finding related tables in data lakes for interactive data science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. 1951–1966.

[57] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J Miller. 2019. Josie: Overlap set similarity search for finding joinable tables in data lakes. In *Proceedings of the 2019 International Conference on Management of Data*. 847–864.

[58] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *Proc. VLDB Endow.* 9, 12 (2016), 1185–1196.

[59] Hongbing Zhu, Jing Li, Jia Li, and Xiaofeng Wu. 2017. Data lake: A new paradigm for big data. *IEEE Access* 5 (2017), 23287–23294.

[60] Xiaoying Zhu et al. 2020. DeepJoin: Learning to Discover Joinable Tables in Data Lakes. *Proceedings of the VLDB Endowment* 13, 10 (2020), 1568–1581.

[61] Xingzhong Zou and Yang Yang. 2016. A dataset search engine for scientific data in Hadoop ecosystem. *Future Generation Computer Systems* 65 (2016), 91–100.