Semantic Table Retrieval Using Keyword and Table Queries

SHUO ZHANG, Bloomberg
KRISZTIAN BALOG, University of Stavanger

Tables on the Web contain a vast amount of knowledge in a structured form. To tap into this valuable resource, we address the problem of table retrieval: answering an information need with a ranked list of tables. We investigate this problem in two different variants, based on how the information need is expressed: as a keyword query or as an existing table ("query-by-table"). The main novel contribution of this work is a semantic table retrieval framework for matching information needs (keyword or table queries) against tables. Specifically, we (i) represent queries and tables in multiple semantic spaces (both discrete sparse and continuous dense vector representations) and (ii) introduce various similarity measures for matching those semantic representations. We consider all possible combinations of semantic representations and similarity measures and use these as features in a supervised learning model. Using two purpose-built test collections based on Wikipedia tables, we demonstrate significant and substantial improvements over state-of-the-art baselines.

CCS Concepts: • **Information systems → Information integration**; **Data extraction and integration**; **Retrieval models and ranking**; *Data management systems*; *Information storage systems*;

Additional Key Words and Phrases: Table search, table retrieval

ACM Reference format:
Shuo Zhang and Krisztian Balog. 2021. Semantic Table Retrieval Using Keyword and Table Queries. *ACM Trans. Web* 15, 3, Article 11 (May 2021), 33 pages.
https://doi.org/10.1145/3441690

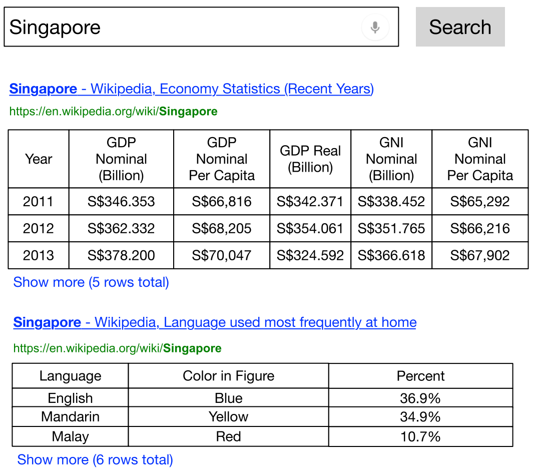
1 INTRODUCTION

Tables are a powerful, versatile, and easy-to-use tool for organizing and working with data. Because of this, a massive number of tables can be found "out there," on the Web or in Wikipedia, representing a vast and rich source of structured information. Recently, a growing body of work has begun to tap into utilizing the knowledge contained in tables. A wide and diverse range of tasks have been undertaken, including but not limited to (i) searching for tables (in response to a keyword query [4, 9, 11, 41, 46, 61] or a seed table [17]), (ii) extracting knowledge from tables (such as RDF triples [39]), and (iii) augmenting tables (with new columns [7, 9, 17, 28, 66, 70], rows [17, 66, 70], cell values [1], or links to entities [8]). Searching for tables is an important problem on its own, in addition to being a core building block in many other table-related tasks. Yet, up until recently it

This is a revised and extended version of Zhang and Balog [71]. It was done while the S. Zhang was at University of Stavanger, Norway.

Authors' addresses: S. Zhang, Bloomberg, 3 Queen Victoria St, London EC4N 4TQ, United Kindom; email: szhang611@bloomberg.net; K. Balog, Department of Electrical Engineering and Computer Science, NO-4036 Stavanger, Norway; email: krisztian.balog@uis.no.

1559-1131/2021/05-ART11 $15.00
https://doi.org/10.1145/3441690


Singapore

Search

Singapore - Wikipedia, Economy Statistics (Recent Years)
https://en.wikipedia.org/wiki/Singapore

Year	GDP Nominal (Billion)	GDP Nominal Per Capita	GDP Real (Billion)	GNI Nominal (Billion)	GNI Nominal Per Capita
2011	S$346.353	S$66,816	S$342.371	S$338.452	S$65,292
2012	S$362.332	S$68,205	S$354.061	S$351.765	S$66,216
2013	S$378.200	S$70,047	S$324.592	S$366.618	S$67,902

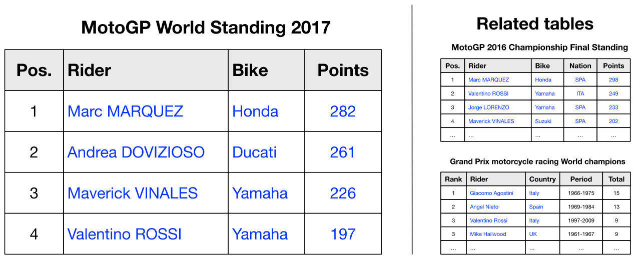
Show more (5 rows total)

Singapore - Wikipedia, Language used most frequently at home
https://en.wikipedia.org/wiki/Singapore

Language	Color in Figure	Percent
English	Blue	36.9%
Mandarin	Yellow	34.9%
Malay	Red	10.7%

Show more (6 rows total)

Fig. 1. Keyword-based table search: Given a keyword query, the system returns a ranked list of tables.


MotoGP World Standing 2017

Pos.	Rider	Bike	Points
1	Marc MARQUEZ	Honda	282
2	Andrea DOVIZIOSO	Ducati	261
3	Maverick VINALES	Yamaha	226
4	Valentino ROSSI	Yamaha	197

Related tables

MotoGP 2016 Championship Final Standing

Pos.	Rider	Bike	Nation	Points
1	Marc MARQUEZ	Honda	SPA	298
2	Valentino ROSSI	Yamaha	ITA	249
3	Jorge LORENZO	Yamaha	SPA	233
4	Maverick VINALES	Suzuki	SPA	202
...	...	...	...	...

Grand Prix motorcycle racing World champions

Rank	Rider	Country	Period	Total
1	Giacomo Agostini	Italy	1966-1975	15
2	Angel Nieto	Spain	1969-1984	13
3	Valentino Rossi	Italy	1997-2009	9
3	Mike Hailwood	UK	1961-1967	9
...	...	...	...	...

Fig. 2. Query-based table search: Given an input table, the system returns a ranked list of tables.

has not received due attention, and especially not from an information retrieval perspective. Our work, which has been published in Reference [71] and is being extended in this article, was a first attempt at aiming to fill that gap, and has spurred interest in table retrieval [3, 14, 18, 31, 47, 48, 55, 56, 57, 59, 64]. In this study, we further extend semantic table retrieval to support table-based search scenarios as well. Table-based search, despite its practical utility, has not been extensively explored to date.

We address the task of *table retrieval*, that is, the problem of generating a ranked list of tables in response to an information need, in two particular flavors: (i) *keyword-based search*, where the information need is specified as a keyword query, and (ii) *table-based search*, where an existing table is used as input. The former task corresponds to a classical ad hoc search scenario, where tables are sought for a particular purpose or need. The latter task resembles more of a recommendation problem, where the user is not required to explicitly formulate a query. Instead, we suggest tables that contain related information (e.g., additional entities and/or attributes) that could potentially complement the table the user is currently working on. This "query-by-table" paradigm could be helpful, for example, in equipping spreadsheet applications with a smart assistance feature for finding related content. Alternatively, it could be implemented as a browser plugin that can be activated upon encountering a table on a webpage to find related tables (e.g., for comparison or fact validation). See Figures 1 and 2 for an illustration.

It should be acknowledged that table retrieval is not an entirely new research problem; in fact, it has been around for a while in the database community (also known there as *relation ranking*) [7, 9, 11, 61]. However, public test collections and proper evaluation methodology are lacking, in addition to the need for better ranking techniques. Tables can be ranked much like documents, by considering the words contained in them [9, 11, 46]. Ranking may be further improved by incorporating additional signals related to table quality. Intuitively, high-quality tables are topically coherent; other indicators may be related to the pages that contain them (e.g., if they are linked by other pages [7]). However, a major limitation of prior approaches is that they only consider lexical matching between the contents of tables and queries. This gives rise to our main research objective: *Can we move beyond lexical matching and improve table retrieval performance by incorporating semantic matching?*

In this article, we introduce the **semantic table retrieval (STR)** framework to handle matching in different semantic spaces in a uniform way. It hinges on the idea of modeling both the table and the input (keyword or table) query as sets of semantic vectors. Specifically, we consider two main kinds of semantic representations: (i) sparse discrete representations based on entities, and (ii) continuous vector representations of words and of entities (i.e., word and graph embeddings). We propose two general strategies (early and late fusion), yielding four different measures for computing the similarity between queries and tables based on their semantic representations. These different similarity scores are then combined in a learning-to-rank framework, together with features aimed capturing general table characteristics.

As mentioned above, another key area where prior work has insufficiencies is evaluation. First, there is no publicly available test collection for this task. Second, evaluation has been performed using set-based metrics (counting the number of relevant tables in the top-k results), which is a very rudimentary way of measuring retrieval effectiveness. We address this by developing a purpose-built test collections, comprising of 1.6M tables from Wikipedia, and a set of queries with graded relevance judgments. For both tasks, we develop strong baselines by assembling a rich set of features from prior work and combining them in a learning-to-rank framework. While all individual features are taken from the literature, the compilation of the feature sets underlying our baselines constitutes an important contribution. We demonstrate that these strong baselines substantially outperform the best approaches known in the literature.

Concerning the effectiveness of our STR framework, our findings are as follows. For keyword-based search, we show that the semantic matching methods we propose can significantly and substantially improve retrieval performance over the strong baseline. For table-based search, our proposed approach is on par with the respective strong baseline. Importantly, this level of performance is reached without requiring the extensive feature engineering that the baseline does. We further demonstrate that retrieval performance increases as the input table grows, either horizontally or vertically, which attests to the capability of our table matching framework to effectively utilize larger inputs.

In summary, this article makes the following contributions:

- We formalize the table retrieval task in two specific flavors: keyword-based search and table-based search (Section 3).
- We develop strong baselines for both tasks by combining elements from prior studies in feature-based supervised learning approaches (Section 4).
- We present a novel semantic matching framework for table retrieval that can effectively perform matching beyond lexical similarity (Section 5).
- We develop standard test collections for both keyword-based and table-based search, which involves gathering relevance assessments (Section 6).
- We conduct an extensive experimental evaluation and demonstrate the effectiveness of our table retrieval framework (Section 7). We also carry out a thorough analysis leading into valuable insights (Section 8).

Of these, the table-based search paradigm, methods, and evaluation resources as well as the extensive analysis of both keyword-based and table-based search results are novel contributions on top of the work [71] this article extends.

The resources developed within this article will be made publicly available upon acceptance at https://github.com/iai-group/table-retrieval.

2 RELATED WORK

An increasing number of studies are addressing various table-related tasks, including table search, table mining, and table augmentation. Our work concerns table search, which is considered as a fundamental task both on its own and as a component in other tasks. Additionally, the line of research on exploiting neural embeddings for IR tasks also bears relevance to this study.

2.1 Table Search

Table search answers a query with a ranked list of tables. Based on the type of the query, table search can be divided into *keyword-based search* [4, 9, 11, 41, 46, 61] and *table-based search* [1, 17, 28, 29, 41, 66, 73].

The WebTables system by Cafarella et al. [11] pioneered keyword-based table search on top of an existing web search engine. The basic idea is to fetch the top-ranked results returned by a web search engine in response to the query, and then extract the top-k tables from those pages. Further refinements to the same idea are introduced in Reference [9]. Venetis et al. [61] leverage a database of class labels and relationships extracted from the Web, which are attached to table columns, for recovering table semantics. This information is then used to enhance table search. Using column keywords, Pimplikar and Sarawagi [46] search tables using term matches in the header, body and context of tables, as signals. An example of a keyword-based table search system interface is provided by Google Web Tables.[1] The developers of this system summarize their experiences in Reference [4]. Their query is not limited to keywords, it can also be a table.

Our recent work [71], which this article extends, formally introduces the ad hoc table retrieval task: answering a keyword query with a ranked list of tables. Semantic matching between queries and tables is proposed as a solution to this problem. As a follow-up, Deng et al. [18] train word and entity embeddings utilizing the Wikipedia table corpus and achieve comparable results. Trabelsi et al. [59] put forward a context-aware table search method based on the embeddings for attribute tokens. Different from [18], Trabelsi et al. [59] find that differentiated types of contexts such as numerical cell values are useful in constructing word embeddings. The system has up to 5% improvement in NDCG@5 over LM that uses the same context fields but treats them as the same context. The trained model can be used to predict different contexts of every table, which further improves table ranking performance. Bagheri and Al-Obeidat [3] recognize that some queries constitute tokens that are not well observed in the relevant tables, and propose a latent model to project the token-table co-occurrence matrix into latent factor matrics, which can be used for measuring similarities. This approach improves over the baselines except for STR [71], which was only significantly outperformed using the Keyword variation. In a similar setting in Reference [59], i.e., by utilizing table columns and attributes, Shraga et al. [57] propose a projection model for table retrieval using table columns as pseudo-relevance feedback. Our initial work [71] explores the use of both extrinsic similarities, such as entity results for keywords, and intrinsic similarities such as word-level similarities. Similarly, Shraga et al. [55] make a novel use of intrinsic features (passage-based) and extrinsic features (manifold-based table similarities) for table retrieval. Chen et al. [14] investigate how to encoding tabular content into BERT for generating embeddings, taking table

[1] https://research.google.com/tables.

structure and the input length limit of BERT into consideration. Taking STR Shraga et al. [55] as the baseline, the BERT-based methods report on performance improves in the range of 4–7% in terms of NDCT@20. To bridge table retrieval and end-to-end embedding learning, Shraga et al. [56] suggest MTR, which utilizes of Gated Multimodal Units to learn a joint-representation of the query and the different table modalities. Shraga et al. [56] further extend table retrieval tables from keyword queries to natural language queries. This work reports a 13% improvement over [18]. It is worth noting that these methods are not reported using the same experimental settings as the baselines.

Table-based search may be conducted for different purposes: (i) to be displayed as the answer and (ii) to serve as an intermediate step that feeds into other tasks like table mining or table augmentation. Ahmadov et al. [1] leverage table elements like entities and headings as keyword queries to retrieve a ranked list of tables. The two ranked lists of tables is merged later by performing table matching to have a more complete candidate set. Table matching is performed by dividing tables into various elements, such as table entities, headings, and columns, then computing element-level similarity. The Mannheim Search Join Engine [28] provides table search functionality with the overall aim to extend an input table with additional attributes (i.e., columns). Lehmberg et al. [28] rely mostly on table headings by comparing the heading labels between the input and candidate tables. Their method uses exact column heading matching to filter tables that share at least one heading with the input table. Then, all candidate tables are scored against the input table. Das Sarma et al. [17] find related tables for extending the seed table with extra rows or columns, referred as *entity complement* and *schema complement*, respectively. For *entity complement* tables, which aim to augment the input table with more entities as rows, they consider the relatedness between entities of the input and candidate tables. For *schema complement* tables, which seek to extend the input tables with more attributes, they take into account the coverage of entities and the benefits of adding additional attributes. The above methods perform table matching in an unsupervised manner. To enrich the diversity of search results, Nguyen et al. [41] design a goodness measure for table search and selection. They match tables by considering two tables elements: heading labels and table data. These two similarities are combined using a linear mixture. Yakout et al. [66] consider element-wise similarity across four table elements: table data, column values, page title, and column headings. These element-wise similarities are combined by training a linear regression scorer.

2.2 Table Mining

Being a rich and structured source knowledge, tables have raised great interest for various mining tasks [6, 8, 10, 11, 15, 33, 40, 53, 53, 61, 61, 67, 68, 74, 76]. Embley et al. [21] present a survey of methods for table processing and applications, like table conversion from homogeneous or heterogeneous sources. These processing steps are key building blocks in table mining. Muñoz et al. [39] aim to recover Wikipedia table semantics and store them in RDF triples. Their method utilizes DBpedia to find pre-existing relations between entities in the Wikipedia tables. It then queries the DBpedia knowledge base for existing facts that involve those entities. The prior relations contribute to extrapolate this to the rest of the table. In the end, 7.9M unique and novel RDF triples are extracted. Similar work is taken in Reference [11] based on tables extracted from a Web crawl. Instead of mining an entire corpus of tables, a single table may already store many facts, which could be answers for questions. Yin et al. [67] take a single table as a knowledge base and perform querying on it using deep neural networks, named Neural Enquirer. Neural Enquirer is fully neural system that generates distributional representations of the query and the knowledge base. It can additionally execute compositional queries as a series of operations. The training can be done in an end-to-end fashion or carried out using step-by-step supervision. The knowledge extracted from tables could be used to augment an existing knowledge base [20, 54]. For instance, Sekhavat et al. [54] design probabilistic methods to utilize table information for augmenting an existing knowledge base. They collect sentences containing pairs of entities in the same row by taking the tabular mentions as keyword queries. Patterns are extracted from these sentences by leveraging existing knowledge bases. Last, they estimate the probability of possible relations that can be added to the knowledge repository. Recently, Zhang et al. [75] propose to mine new categories based on the entity sets in tables.

Another line of work concerns table annotation and classification. By mining column content, Zwicklbauer et al. [78] propose a method to annotate table headers with semantic type information based on the column's cells. The annotation is performed in three steps. First, it uses a search-based disambiguation method to annotate cell entities. Then, it resolves entity-type by retrieving a set of types for the entity candidates. Last, the type that occurs most frequently in the set of all types of all cells is assigned to the table header. Studying a large number of tables in Reference [16], a well defined table type taxonomy is provided for classifying HTML tables. They introduce a supervised framework for classifying HTML tables into a taxonomy by examining the contents of a large number of tables. Three types of features are considered: global layout features, layout features, and content features. Global layout features include the maximum number of rows, column, and cell content length. Layout features are solely based on the size of the cells and variance, e.g., the average length of cells. Content features focus on cell content, including HTML features (e.g., the ratio of cells containing header) and lexical features (e.g., the ratio of cells where the content is a number). Apart from the above mentioned problems, table mining can also include tasks like *table interpretation* [11, 39, 61] and *table recognition* [16, 78]. In the problem space of table mining, table search is an essential component.

2.3 Table Augmentation

Table augmentation is the task of extending a table with additional elements, e.g., new columns or rows. Methods for extending a table with additional columns need to capture relevant data (i.e., existing columns), which is done with the help of table search [7, 28, 66]. For example, the Mannheim Search Join Engine [28] searches the top-k candidate tables from a corpus of web tables and picks relevant columns to merge. Extending a table with more rows also needs table retrieval [17, 66, 70, 70]. Das Sarma et al. [17] find *entity complement* tables to find the additional entities that can be put into the input table as the next rows. However, it stops at the table search step and only utilizes the tables. In Reference [70], two tasks of row population and column population are proposed to extend an entity-focused table with additional rows and columns. The authors utilize both tables and a knowledge base for row extension. Specifically, their method first finds candidates from the related tables and knowledge base. Then, it ranks the candidates based on the similarity to (i) other entities in the tables, (ii) column headings, and (iii) the caption of the table. Column extension relies only on tables, and it follows a similar approach. They find that a knowledge base and a table corpus can complement each other for table augmentation. In recent work, Deng et al. [18] used Word2vec to train embeddings for these two tasks and achieved state-of-the-art results.

Table completion is the task of filling in empty cells within a table. Ahmadov et al. [1] introduce a method to extract table values from related tables and/or to predict them using machine learning methods. In recent work, Zhang and Balog [72] present the CellAutoComplete system to address shortcomings of previous approaches. Specifically, CellAutoComplete enables a table cell to have multiple, possibly conflicting values, providing the values with evidence found from other tables or the knowledge base, or predict if a cell should be left empty. Their method leverages a corpus of Wikipedia tables and a knowledge base (DBpedia) as data sources.

2.4 Neural Embeddings for IR

Recently, unsupervised representation learning methods have been proposed for obtaining embeddings that predict a distributional context, i.e., word embeddings [36, 43] or graph embeddings [44, 51, 58]. Such vector representations have been utilized successfully in a range of IR tasks, including ad hoc retrieval [23, 38], contextual suggestion [34], cross-lingual IR [62], community question answering [77], short text similarity [27], and sponsored search [24]. For example, Ganguly et al. [23] construct a generalized language model by making use of the vector embeddings to derive the transformation probabilities between words for enhancing ad hoc retrieval effectiveness. Mitra et al. [38] propose the Dual Embedding Space Mode that uses the neural word embeddings to gauge a documents' relatedness to the query in ad hot retrieval. It exploits a novel use of both the input and output embeddings of the CBOW model to capture the topic-based semantic relationship. Manotumruksa et al. [34] exploit word embeddings to infer the vector-space representations of venues, user preferences, and users' contexture preferences for contextual suggestions. Zhou et al. [77] propose to learn continuous word embeddings with metadata of category information within community question answering to fill the lexical gap. Kenter and de Rijke [27] verify the idea of investigating only using semantic features for computing short text similarity [27] from word-level to text-level. Most recently, transformer-based models, such as BERT [19], have shown great improvements for many NLP tasks. In this work, however, we limit ourselves to traditional (non-contextual) embeddings.

3 TABLE RETRIEVAL

In this section, we formalize the table retrieval task, and explain what information is associated with a table.

3.1 Problem Statement

Table search is the task of returning a ranked list of tables from a collection of tables, in response to a query. The relevance of each returned table T is assessed independently of all other returned tables. Tables are then sorted in descending order of their scores. We consider two types of queries in this work. First, when the input is a keyword query q, we refer to this task as *keyword-based search*. The ranking of tables boils down to the problem of assigning a score to each table in the corpus, $score(q, T)$. Second, when the input is a table $\tilde{T}$, we term this task as *table-based search*. The objective is to compute the similarity between an input table $\tilde{T}$ and a candidate table T, expressed as $score(\tilde{T}, T)$.

3.2 The Anatomy of a Table

We shall assume a corpus of Web tables, where tables are embedded in webpages. This means that in addition to the core table content (caption, column headings, and table body), some contextual information is also available, such as the embedding page's title. See Table 1 for an overview, with a corresponding illustration in Figure 3.

4 BASELINE METHODS

Table retrieval can be performed by utilizing either keyword-based or feature-based methods. In this section, we discuss these two directions, and detail how existing work may be applied to our tasks, depending on the input query (keywords or table).

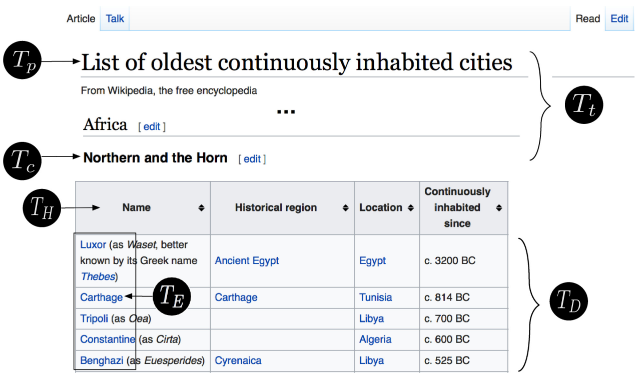
4.1 Keyword-based Methods

An easy and straightforward way to perform table retrieval is by treating the query as a set of keywords and adopting standard document ranking methods.

Table 1. Notation Used for Elements of Table T

Symbol	Table element	Explanation
T_c	Table caption	A brief explanation of a table
T_p	Page title	Title of the page where the table was extracted from
T_E	Table entities	Set of entities in the table
$T_{E'}$	Core column entities	Set of entities in the core column
T_H	Column headings	Set of table column headings
T_D	Table data	Table data, excluding column headings[†]
T_t	Table topic	The subject of a table

[†]Could also denote, depending on the algorithm, a subset of the table (i.e., selected rows or columns).



List of oldest continuously inhabited cities

From Wikipedia, the free encyclopedia

...

Africa [edit]

Northern and the Horn [edit]

Name	Historical region	Location	Continuously inhabited since
Luxor (as *Waset*, better known by its Greek name *Thebes*)	Ancient Egypt	Egypt	c. 3200 BC
Carthage	Carthage	Tunisia	c. 814 BC
Tripoli (as *Oea*)		Libya	c. 700 BC
Constantine (as *Cirta*)		Algeria	c. 600 BC
Benghazi (as *Euesperides*)	Cyrenaica	Libya	c. 525 BC

Fig. 3. Table embedded in a Wikipedia page.

In prior work, Cafarella et al. [9, 11] utilize web search engines to retrieve relevant documents; tables are then extracted from the highest-ranked documents. Rather than relying on external services, we represent tables as either single- or multi-field documents and apply standard documents retrieval techniques.

Single-field Document Representation. In the simplest case, all text associated with a given table may be used as the table's representation. This representation is then scored using existing retrieval methods, such as language models.

Multi-field Document Representation. Rather than collapsing all textual content into a single-field document, it may be organized into multiple fields, such as table caption, table headers, table body, and so on (cf. Section 3.2). For multi-field ranking, Pimplikar and Sarawagi [46] employ a *late fusion* strategy [69]. That is, each field is scored independently against the query, then a weighted sum of the field-level similarity scores is taken:

$$score(q, T) = \sum_i w_i \times score(q, f_i), \tag{1}$$

where f_i denotes the ith (document) field for table T and w_i is the corresponding field weight (such that $\sum_i w_i = 1$). Field-level similarity, $score(q, f_i)$, may be computed using any standard retrieval method. We use language models in our experiments.

Table 2. Baseline Features for Keyword-based Search (LTR-k)

Query features		**Source**	**Value**
QLEN	Number of query terms	[60]	$\{1, \ldots, n\}$
IDF_f	Sum of query IDF scores in field f	[49]	$[0, \infty)$
	Table features		
#rows	Number of rows in the table	[7, 11]	$\{1, \ldots, n\}$
#cols	Number of columns in the table	[7, 11]	$\{1, \ldots, n\}$
#NULLs	Number of empty table cells	[7, 11]	$\{0, \ldots, n\}$
PMI	ACSDb-based schema coherency score	[11]	$(-\infty, \infty)$
inLinks	In-link count of the embedding page	[7]	$\{0, \ldots, n\}$
outLinks	Out-link count of the embedding page	[7]	$\{0, \ldots, n\}$
pageViews	Page view count of the embedding page	[7]	$\{0, \ldots, n\}$
tableImportance	Inverse of number of tables on the page	[7]	$(0, 1]$
tablePageFraction	Table size to page size ratio	[7]	$(0, 1]$
Query-table features			
#hitsLC	Total query term frequency in leftmost column	[11]	$\{0, \ldots, n\}$
#hitsSLC	Total query term frequency in second-to-leftmost column	[11]	$\{0, \ldots, n\}$
#hitsB	Total query term frequency in table body	[11]	$\{0, \ldots, n\}$
qInPgTitle	Ratio of the number of query tokens found in page title to total number of tokens	[7]	$[0, 1]$
qInTableTitle	Ratio of the number of query tokens found in table title to total number of tokens	[7]	$[0, 1]$
yRank	Rank of the table's Wikipedia page in web search results for the query	[7]	$\{1, \ldots, n\}$
MLM similarity	Language modeling score between query and multi-field document representation of the table	[13]	$(-\infty,0)$

4.2 Feature-based Methods for Keyword-based Search

Another line of work uses feature-based methods. The idea is to represent both the query and the candidate table as a feature vector, and obtain a retrieval function using supervised learning [30]. In a standard document retrieval setting, features are commonly categorized into three groups: (i) document, (ii) query, and (iii) query-document features [49]. Analogously, we distinguish between three types of features: (i) table, (ii) query, and (iii) query–table features. In Table 2, we summarize the features from previous work on table search [7, 11]. We also include a number of additional features that have been used in other retrieval tasks, such as document and entity ranking.

4.2.1 Query Features. Query features have been shown to improve retrieval performance for document ranking [32]. We adopt two query features from document retrieval, namely, the number of terms in the query [60], and query IDF [49] according to: $IDF_f(q) = \sum_{t \in q} IDF_f(t)$, where $IDF_f(t)$ is the IDF score of term t in field f. This feature is computed for the following fields: page title, section title, table caption, table heading, table body, and "catch-all" (the concatenation of all textual content in the table).

4.2.2 Table Features. Table features depend only on the table itself and aim to reflect the quality of the given table (irrespective of the query). Some features are simple characteristics,

Table 3. Table Elements Used in Existing Table-based Methods

Method	T_c	T_p	T_E	T_H	T_D
Keyword-based search using T_E			√		
Keyword-based search using T_H				√	
Keyword-based search using T_c	√				
Mannheim Search Join Engine				√	
Schema complement			√	√	
Entity complement			√		
Nguyen et al.				√	√
InfoGather		√		√	√

like the number of rows, columns, and empty cells [7, 11]. One important feature is **Pointwise Mutual Information (PMI)**, which is taken from linguistics research, and expresses the coherency of a table. The correlation between two table headings cells, h_i and h_j, is given by: $PMI(h_i, h_j) = \log(P(h_i, h_j)/(P(h_i)P(h_j)))$. For example, "address" and "name" are more likely to co-occur as column headings than "address" and "wins." A table's PMI is computed by calculating the PMI values between all pairs of column headings of that table, and then taking their average. Following Reference [11], we compute PMI by obtaining frequency statistics from the **Attribute Correlation Statistics Database (ACSDb)** [12], which contains table heading information derived from millions of tables extracted from a large web crawl.

Another group of features are derived from the webpage that embeds the table, by considering its connectivity (inLinks and outLinks), popularity (pageViews), and the table's importance within the page (tableImportance and tablePageFraction).

4.2.3 Query-Table Features. Features in the last group express the degree of matching between the keyword query and a given candidate table. This matching may be based on occurrences of query terms in the page title (qInPgTitle) or in the table caption (qInTableTitle). Alternatively, it may be based on specific parts of the table, such as the leftmost column (#hitsLC), second-to-left column (#hitsSLC), or table body (#hitsB). The rank at which a table's parent page is retrieved by an external search engine is also used as a feature (yRank). (In our experiments, we use the Wikipedia search API to obtain this ranking.) Furthermore, we take the **Mixture of Language Models (MLM)** similarity score [42] as a feature, which is actually the best performing method among the four text-based baseline methods (cf. Section 7).

We utilize table features, query features, and query–table features to train a learn-to-rank scorer, to serve as a strong baseline, and label it *LTR-k*. Specifically, we manually label the candidates retrieved by the unsupervised methods introduced in Section 4.1 as training data (cf. Section 6.1), and combine different set of features for *LTR-k* to train the scorers (cf. Section 7.3). Importantly, all these features are based on lexical matching. Our goal in this article is to also enable semantic matching; this is what we shall discuss in Section 5.

4.3 Feature-based Methods for Table-based Search

For table-based search, table features can still be employed (cf. the middle block in Table 3). In fact, they may be computed for both the input and candidate tables. (When computed for the input table, these essentially become the "query features.") Query-table features, however, are different from those in keyword-based search, as we need to perform table-to-table matching as opposed to keyword-to-table matching.

We present a number of existing methods from the literature that can be used to perform table matching. On the high level, all these methods operate by (i) subdividing tables into a number of *table elements* (such as page title (T_p), table caption (T_c), table topic (T_t), column headings (T_H), table entities (T_E), and table data (T_D)), (ii) measuring the similarity between various elements of the input and candidate tables, and (iii) in case multiple elements are considered, combining these element-level similarities into a final score. Table 3 provides an overview of existing methods and the table elements they utilize.

Mannheim Search Join Engine. The Mannheim Search Join Engine [28] provides table search functionality with the overall aim to extend an input table with additional attributes (i.e., columns). First, it uses exact column heading matching to filter tables that share at least one heading with the input table: $\mathcal{T} = \{T : |\tilde{T}_H \cap T_H| > 0\}$. Then, all tables in $\mathcal{T}$ are scored against the input table using the FastJoin matcher [63]. Specifically, Lehmberg et al. [28] adapt edit distance with a threshold of δ to measure the similarity between the input and candidate tables' heading terms, $w(t_i, t_j)$, where $t_i \in \tilde{T}_H$ and $t_j \in T_H$. Terms in $\tilde{T}_H$ and T_H form a bipartite graph, with $w(t_i, t_j)$ as edge weights. Let $|\tilde{T}_H \tilde{\cap}_\delta T_H|$ denote the *maximum weighted bipartite matching score* on the graph's adjacency matrix, considering edges whose weight exceeds the edit distance threshold δ. Then, the Jaccard similarity of two tables is expressed as:

$$score(\tilde{T}, T) = \frac{|\tilde{T}_H \tilde{\cap}_\delta T_H|}{|\{t : t \in \tilde{T}_H\}| + |\{t : t \in T_H\}| - |\tilde{T}_H \tilde{\cap}_\delta T_H|},$$

where $|\{t : t \in T_H\}|$ denotes the number of unique terms in the column headings of T.

Schema Complement. Das Sarma et al. [17] search for related tables with the overall goal of extending the input table with additional attributes (referred to as *schema complement* in Reference [17]). For this task, they consider two factors: (i) the coverage of entities and (ii) the benefits of adding additional attributes. The final matching score is computed as:

$$score(\tilde{T}, T) = S_{EC}(\tilde{T}, T) \times S_{HB}(\tilde{T}, T). \tag{2}$$

The first component, **entity coverage (EC)**, computes the entity overlap between two tables:

$$S_{EC}(\tilde{T}, T) = \frac{|\tilde{T}_E \cap T_E|}{|\tilde{T}_E|}. \tag{3}$$

The second component in Equation (2) estimates the benefit of adding an additional column heading h to the input table:

$$HB(\tilde{T}_H, h) = \frac{1}{|\tilde{T}_H|} \sum_{\tilde{h} \in \tilde{T}_H} \frac{\#(\tilde{h}, h)}{\#(\tilde{h})},$$

where $\#(\tilde{h}, h)$ is number of tables containing both $\tilde{h}$ and h as column headings, and $\#(\tilde{h})$ is the number of tables containing $\tilde{h}$. The heading benefit between two tables, $S_{HB}(\tilde{T}, T)$, is computed by aggregating the benefits of adding all headings h from T to $\tilde{T}$:

$$S_{HB}(\tilde{T}, T) = aggr(HB(\tilde{T}_H, h)).$$

The aggregation function $aggr()$ can be sum, average, or max.

Entity Complement. In addition to schema complement tables, Das Sarma et al. [17] also search for *entity complement* tables, to augment the input table with additional entities (as rows). This method considers the relatedness between entities of the two tables:

$$score(\tilde{T}, T) = \frac{1}{|\tilde{T}_E||T_E|} \sum_{\tilde{e} \in \tilde{T}_E} \sum_{e \in T_E} sim(\tilde{e}, e),$$

where $sim(\tilde{e}, e)$ is a pairwise entity similarity measure. Specifically, we employ the **Wikipedia Link-based Measure (WLM)** [37], which estimates the semantic relatedness between two entities based on other entities they link to:

$$sim_{WLM}(e, \tilde{e}) = 1 - \frac{\log(\max(|\mathcal{L}_e|, |\mathcal{L}_{\tilde{e}}|)) - \log(|\mathcal{L}_e \cap \mathcal{L}_{\tilde{e}}|)}{\log(|\mathcal{E}| - \log(\min(|\mathcal{L}_e|, |\mathcal{L}_{\tilde{e}}|)))},$$

where $\mathcal{L}_e$ is the set of outgoing links of e (i.e., entities e links to) and $|\mathcal{E}|$ is the total number of entities in the knowledge base.

Nguyen et al. [41] match tables by considering both their headings and content (table data). These two similarities are combined using a linear mixture:

$$score(\tilde{T}, T) = \alpha \times sim_H(\tilde{T}, T) + (1 - \alpha) \times sim_D(\tilde{T}, T).$$

The heading similarity sim_H is computed by first creating a similarity matrix between the heading terms of $\tilde{T}_H$ and T_H, as in Section 4.3. Specifically, for two sets of headings $\tilde{T}_H$ and T_H, it first constructs a $|\tilde{T}_H| \times |T_H|$ similarity matrix $m(\tilde{T}_H, T_H)$, where $m_{ij}(\tilde{T}_H, T_H)$ denotes the degree of similarity between the heading i of $\tilde{T}_H$ and j of T_H, respectively. Next, an attribute correspondence subgraph $C \subseteq (|\tilde{T}_H| \times |T_H|)$ is obtained by solving the *maximum weighted bipartite sub-graph problem* [2]. Taking column headings as vertices of a graph and $m_{ij}(\tilde{T}_H, T_H)$ as edge weights, the task is to find the maximum weight bipartite subgraph, such that the vertices of the subgraph can be divided into two disjoint and independents set U and V, where every edge connects a vertex in V to U. Finally, heading similarity is computed as:

$$sim_H(\tilde{T}, T) = \frac{\sum_{(i,j) \in C} w_{t_i, t_j}(\tilde{T}_H, T_H)}{\max(|\tilde{T}_H|, |T_H|)}.$$

Data similarity is measured based on columns. Each table column is represented as a binary term vector, $\mathbf{T}_{D,i}$, where each element indicates the presence (1) or absence (0) of a given term in column i of table T. The similarity between two columns is measured by their cosine similarity. Table similarity considers all column combinations of $\tilde{T}$ and T. To account for the high number of possible combinations, for each table column, only the most similar column is considered from the other table:

$$sim_D(\tilde{T}, T) = \frac{1}{2}\left(\sum_i \max_j \cos(\tilde{\mathbf{T}}_{D,i}, \mathbf{T}_{D,j}) + \sum_j \max_i \cos(\tilde{\mathbf{T}}_{D,i}, \mathbf{T}_{D,j})\right).$$

InfoGather. Following Yakout et al. [66], we consider element-wise similarity across four table elements: table data, column values, page title, and column headings. Element-wise similarities are combined by training a linear regression scorer:

$$score(\tilde{T}, T) = \sum_x w_x \times sim_x(\tilde{T}, T),$$

where x is a given table element, $sim_x()$ is the element-level similarity score, and w_x is the weight (importance) of that element. Each table element is expressed as a term vector, denoted as $\tilde{\mathbf{T}}_x$ and

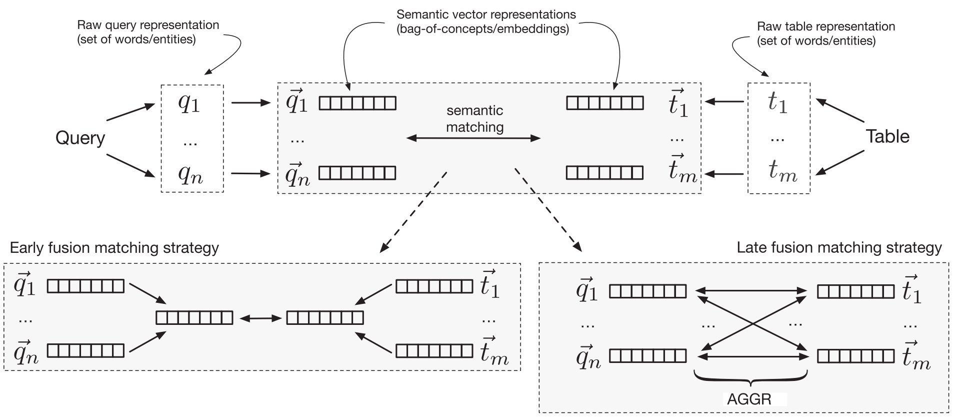

Raw query representation (set of words/entities)
Semantic vector representations (bag-of-concepts/embeddings)
Raw table representation (set of words/entities)
Query
q_1 ... q_n
$\vec{q}_1$ ... $\vec{q}_n$
semantic matching
$\vec{t}_1$ ... $\vec{t}_m$
t_1 ... t_m
Table
Early fusion matching strategy
Late fusion matching strategy
AGGR

Fig. 4. Our methods for computing query–table similarity for keyword-based search using semantic representations.

Table 4. Semantic Similarity Features

Features	Semantic repr.	Raw repr.
Entity_*	Bag-of-entities	entities
Word_*	Word embeddings	words
Graph_*	Graph embeddings	entities

Each row represents 4 features (one for each similarity matching method, cf. Table 5). All features are in [−1, 1].

$\mathbf{T}_x$ for element x of the input and candidate tables, respectively. Element-level similarity is then estimated using the cosine similarity between the two term vectors:

$$sim_x(\tilde{T}, T) = \cos(\tilde{\mathbf{T}}_x, \mathbf{T}_x) = \frac{\tilde{\mathbf{T}}_x \cdot \mathbf{T}_x}{||\tilde{\mathbf{T}}_x|| \times ||\mathbf{T}_x||}. \tag{4}$$

Specifically, following Reference [66], for the table data and page title elements we use IDF weighting, while for column heading and column values, we employ TF-IDF weighting.

Our baselines. We leverage all the table matching scores introduced in this section as features in a learning-to-rank scorer. It serves as the first strong table-based search baseline and is labeled *LTR-t1*. Additionally, we also incorporate table features (cf. Section 4.2.2), computed for both the input and candidate tables, in a second strong baseline *LTR-t2*.

5 SEMANTIC MATCHING

This section presents our main contribution, which is a framework for performing semantic matching for table retrieval. The main idea is to go beyond lexical matching by representing both queries and tables in some semantic space, and measuring the similarity of those semantic (vector) representations. Our approach consists of three main steps, which are illustrated in Figure 4. These are as follows (moving from outwards to inwards on the figure):

(1) The "raw" content of a query/table is represented as a set of terms, where terms can be either words or entities (Section 5.1).
(2) Each of the raw terms is mapped to a semantic vector representation (Section 5.2).
(3) The semantic similarity (matching score) between a query–table pair is computed based on their semantic vector representations (Section 5.3).

We compute query–table similarity using all possible combinations of semantic representations and similarity measures, and use the resulting semantic similarity scores as features in a learning-to-rank approach. Table 4 summarizes these features.

5.1 Content Extraction

We represent the "raw" content of the query/table as a set of terms, where terms can be either words (string tokens) or entities (from a knowledge base). We denote these as $\{q_1, \ldots, q_n\}$ and $\{t_1, \ldots, t_m\}$ for query q and table T, respectively.

5.1.1 Word-based. It is a natural choice to simply use word tokens to represent query/table content. That is, $\{q_1, \ldots, q_n\}$ is comprised of the unique words in the query. As for the table, we let $\{t_1, \ldots, t_m\}$ contain all unique words from the title, caption, and headings of the table. Mind that at this stage we are only considering the presence/absence of words. During the query–table similarity matching, the importance of the words will also be taken into account (Section 5.3.1).

5.1.2 Entity-based. Many tables are focused on specific entities [70]. Therefore, considering the entities contained in a table amounts to a meaningful representation of its content. We use the DBpedia knowledge base as our entity repository. Since we work with tables extracted from Wikipedia, the entity annotations are readily available (otherwise, entity annotations could be obtained automatically, see, e.g., Reference [61]). Importantly, instead of blindly including all entities mentioned in the table, we wish to focus on salient entities. Salient entities come from three sources: the core column, page title, and table caption. It has been observed in prior work [8, 61] that tables often have a *core column*, containing mostly entities, while the rest of the columns contain properties of these entities (many of which are entities themselves). We describe our core column detection method in Section 5.1.3. In addition to the entities taken directly from the body part of the table, we also include entities that are related to the page title (T_p) and to the table caption (T_c). We obtain those by using the page title and the table caption, respectively, to retrieve relevant entities from the knowledge base. We write $R_k(s)$ to denote the set of top-k entities retrieved for the query s. We detail the entity ranking method in Section 5.1.4. Finally, the table is represented as the union of three sets of entities, originating from the core column, page title, and table caption: $\{t_1, \ldots, t_m\} = T_{E'} \cup R_k(T_p) \cup R_k(T_c)$.

To get an entity-based representation for the query, we issue the query against a knowledge base to retrieve relevant entities, using the same retrieval method as above, i.e., $\{q_1, \ldots, q_n\} = R_k(q)$.

5.1.3 Core Column Detection. We introduce a simple and effective core column detection method. It is based on the notion of *column entity rate*, which is defined as the ratio of cells in a column that contain an entity. We write $cer(T_{c[j]})$ to denote the column entity rate of column j in table T. Then, the index of the core column becomes: $\arg\max_{j=1..T_{|c|}} cer(T_{c[j]})$, where $T_{|c|}$ is the number of columns in T.

5.1.4 Entity Retrieval. We employ a fielded entity representation with five fields (names, categories, attributes, similar entity names, and related entity names) and rank entities using the Mixture of Language Models approach [42]. We return the top-k entities, where k is set to 10. The field weights are set uniformly. This corresponds to the MLM-all model in Reference [26] and is shown to be a solid baseline. We acknowledge that more advanced entity retrieval methods exist [5]; however, this is outside the focus of the current work.

5.2 Semantic Representations

Next, we embed the query/table terms in a semantic space. That is, we map each table term t_i to a vector representation $\vec{t_i}$, where $\vec{t_i}[j]$ refers to the jth element of that vector. For queries,

Table 5. Similarity Measures

Measure	Equation
Early	$\cos(\vec{C}_q, \vec{C}_T)$
Late-max	$\max(\{\cos(\vec{q}_i, \vec{t}_j) : i \in [1..n], j \in [1..m]\})$
Late-sum	$\mathrm{sum}(\{\cos(\vec{q}_i, \vec{t}_j) : i \in [1..n], j \in [1..m]\})$
Late-avg	$\mathrm{avg}(\{\cos(\vec{q}_i, \vec{t}_j) : i \in [1..n], j \in [1..m]\})$

the process goes analogously. We discuss two main kinds of semantic spaces, bag-of-concepts and embeddings. The former uses sparse and discrete, while the latter employs dense and continuous-valued vectors. A particularly nice property of our semantic matching framework is that it allows us to deal with these two different types of representations in a unified way.

5.2.1 Bag-of-concepts. One alternative for moving from the lexical to the semantic space is to represent tables/queries using specific concepts. In this work, we use entities from a knowledge base. Entities have been used in the past for various retrieval tasks, in duet with the traditional bag-of-words content representation. For example, entity-based representations have been used for document retrieval [50, 65]. One important difference from previous work is that instead of representing the entire query/table using a single semantic vector, we map each individual query/table term to a separate semantic vector, thereby obtaining a richer representation.

We use the entity-based raw representation from the previous section, that is, t_i and q_j are specific entities. Below, we explain how table terms t_j are projected to $\vec{t_i}$, which is a sparse discrete vector in the entity space; for query terms it follows analogously. Each element in $\vec{t_i}$ corresponds to a unique entity. Thus, the dimensionality of $\vec{t_i}$ is the number of entities in the knowledge base (on the order of millions). $\vec{t_i}[j]$ has a value of 1 if entities i and j are related (there exists a link between them in the knowledge base), and 0 otherwise.

5.2.2 Embeddings. Embedding-based representations representations have been utilized successfully in a range of IR tasks, including ad hoc retrieval [23, 38], contextual suggestion [34], cross-lingual IR [62], community question answering [77], short text similarity [27], and sponsored search [24]. We consider both word-based and entity-based raw representations from the previous section and use the corresponding (pre-trained) embeddings. We shall use well-established embedding methods noting that these may be substituted with (more recent) alternatives [45].

Word embeddings. We map each query/table word to a word embedding. Specifically, we use word2vec [36] with 300 dimensions, trained on Google News data.[2]

Graph embeddings. We map each query/table entity to a graph embedding. In particular, we use RDF2vec [51] with 200 dimensions, trained on DBpedia 2015-10.

5.3 Similarity Measures

The final step is concerned with the computation of the similarity between a query–table pair, based on the semantic vector representations we have obtained for them. We introduce two main strategies, which yield four specific similarity measures. These are summarized in Table 5.

5.3.1 Early Fusion. The first idea is to represent the query and the table each with a single vector. Their similarity can then simply be expressed as the similarity of the corresponding vectors.

[2]It has been verified in Reference [18] that word embeddings trained based on Google News data and on a table corpus lead to comparable performance on the keyword-based table retrieval task. Therefore, in the interest of simplicity, we shall utilize word embeddings trained on the former in this work, and leave embedding learning for table retrieval for the future.

Table 6. Element-wise and Cross-element Features for Table-based Search

Element	Dimension	Element	Dimension
$\tilde{T}_H$ to T_H	$1 \times 1 \times 4 = 4$	$\tilde{T}_H$ to T_t	$2 \times 1 \times 4 = 8$
$\tilde{T}_D$ to T_D	$1 \times 3 \times 4 = 12$	$\tilde{T}_H$ to T_D	$2 \times 1 \times 4 = 8$
$\tilde{T}_E$ to T_E	$1 \times 2 \times 4 = 8$	$\tilde{T}_D$ to T_t	$2 \times 3 \times 4 = 24$
$\tilde{T}_t$ to T_t	$1 \times 3 \times 4 = 12$	$\tilde{T}_D$ to T_E	$2 \times 2 \times 4 = 16$
		$\tilde{T}_t$ to T_E	$2 \times 2 \times 4 = 16$
Total	36		72

The dimension is $r \times s \times m$, where r is reflection (1 for element-wise and 2 for cross-element), s is the number of semantic spaces, and m is the number of element-wise similarity measures.

We let $\vec{C}_q$ be the centroid of the query term vectors ($\vec{C}_q = \sum_{i=1}^{n} \vec{q}_i/n$). Similarly, $\vec{C}_T$ denotes the centroid of the table term vectors. The query–table similarity is then computed by taking the cosine similarity of the centroid vectors. Due to the compositional capabilities of embeddings, this simple centroid-based summarization of content is shown to achieve good performance [52]. When query/table content is represented in terms of words, we additionally make use of word importance by employing standard TF-IDF term weighting. Note that this only applies to word embeddings (as the other two semantic representations are based on entities). In case of word embeddings, the centroid vectors are calculated as $\vec{C}_T = \sum_{i=1}^{m} \vec{t}_i \times TFIDF(t_i)$. The computation of $\vec{C}_q$ follows analogously.

5.3.2 Late Fusion. Instead of combining all semantic vectors q_i and t_j into a single one, late fusion computes the pairwise similarity between all query and table vectors first, and then aggregates those. We let S be a set that holds all pairwise cosine similarity scores: $S = \{\cos(\vec{q}_i, \vec{t}_j) : i \in [1..n], j \in [1..m]\}$. The query–table similarity score is then computed as aggr(S), where aggr() is an aggregation function. Specifically, we use max(), sum(), and avg() as aggregators; see the last three rows in Table 5 for the equations.

5.4 Specific Instantiations

Next, we discuss specific instantiations of our framework for keyword-based and table-based search.

5.4.1 Keyword-based Search. We compute query–table similarity using all possible combinations of semantic representations (3) and similarity measures (4), and use the resulting (3 × 4) semantic similarity scores as features in a learning-to-rank approach. In addition, we also leverage the full set of features in Table 2. This specific instantiation of our framework is labeled as *STR-k*.

5.4.2 Table-based Search. Existing methods have only considered matching between elements of the same type, referred to as *element-wise* matching. Our framework also enables us to measure the similarities between elements of different types in a principled way, referred to as *cross-element* matching. Finally, as before, we can also utilize table features that characterize the input and candidate tables. Below, we detail the set of features used for measuring element-level similarity.

Element-wise similarity. We compute the similarity between elements of the same type from the input and candidate tables. Each table element may be represented in up to three semantic spaces. Then, in each of those spaces, similarity is measured using the

Table 7. Table-based Search Features Used in Various Instantiations of Our Element-wise Table Matching Framework

Method	**Table similarity features**		
	Element-wise	**Cross-element**	**Table features**
STR-t1	√		
STR-t2	√		√
STR-t3		√	√
STR-t4	√	√	√

Element-wise and cross-element features are summarized in Table 6, table feature are listed in Table 2.

four element-level similarity measures (early, late-max, late-sum, and late-avg). Element-wise features are summarized in the left half of Table 6.

Cross-element similarity. This approach compares table elements of different types in an asymmetrical way. Each pair of elements need to be represented in the same semantic space. Then, the same element-level similarity measures may be applied, as before. We list the cross-element similarity features in the right half of Table 6.

We present four specific instantiations of our table matching framework, by considering various combinations of the three main groups of features. These instantiations are labeled as *STR-t1* .. *STR-t4* and are summarized in Table 7.

6 TEST COLLECTION

We introduce our test collections, including the table corpus, test and development query sets, and the procedure used for obtaining relevance assessments.

6.1 Table Corpus

We use the WikiTables corpus [8], which comprises 1.6M tables extracted from Wikipedia (dump date: 2015 October). The following information is provided for each table: table caption, column headings, table body, (Wikipedia) page title, section title, and table statistics like number of headings rows, columns, and data rows. We further replace all links in the table body with entity identifiers from the DBpedia knowledge base (version 2015-10) as follows. For each cell that contains a hyperlink, we check if it points to an entity that is present in DBpedia. If yes, then we use the DBpedia identifier of the linked entity as the cell's content; otherwise, we replace the link with the anchor text, i.e., treat it as a string.

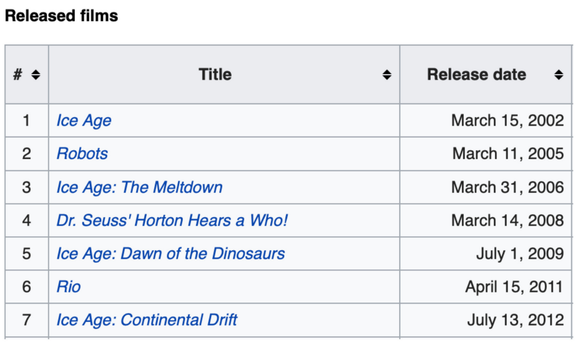
6.2 Keyword-based Search

6.2.1 Queries. We sample a total of 60 test queries from two independent sources (30 from each): (1) *Query subset 1 (QS-1)*: Cafarella et al. [9] collected 51 queries from Web users via crowdsourcing (using Amazon's Mechanical Turk platform, users were asked to suggest topics or supply URLs for a useful data table). (2) *Query subset 2 (QS-2)*: Venetis et al. [61] analyzed the query logs from Google Squared (a service in which users search for structured data) and constructed 100 queries, all of which are a combination of an instance class (e.g., "laptops") and a property (e.g., "cpu"). Following Reference [7], we concatenate the class and property fields into a single query string (e.g., "laptops cpu"). Table 8 lists some examples.

6.2.2 Relevance Assessments. We collect graded relevance assessments by employing three independent (trained) judges. For each query, we pool the top 20 results from five baseline methods (cf. Section 7.3), using default parameter settings. (Then, we train the parameters of

Table 8. Example Keyword-based Search Queries from Our Query Set

Queries from Reference [9]	Queries from Reference [61]
video games	asian coutries currency
us cities	laptops cpu
kings of africa	food calories
economy gdp	guitars manufacturer
fifa world cup winners	clothes brand


Released films

#	Title	Release date
1	*Ice Age*	March 15, 2002
2	*Robots*	March 11, 2005
3	*Ice Age: The Meltdown*	March 31, 2006
4	*Dr. Seuss' Horton Hears a Who!*	March 14, 2008
5	*Ice Age: Dawn of the Dinosaurs*	July 1, 2009
6	*Rio*	April 15, 2011
7	*Ice Age: Continental Drift*	July 13, 2012

Fig. 5. An example query for table-based search from our query set.

those methods with help of the obtained relevance labels.) Each query–table pair is judged on a three point scale: 0 (non-relevant), 1 (somewhat relevant), and 2 (highly relevant). Annotators were situated in a scenario where they need to create a table on the topic of the query, and wish to find relevant tables that can aid them in completing that task. Specifically, they were given the following labeling guidelines: (i) a table is *non-relevant* if it is unclear what it is about (e.g., misses headings or caption) or is about a different topic; (ii) a table is *relevant* if some cells or values could be used from this table; and (iii) a table is *highly relevant* if large blocks or several values could be used from it when creating a new table on the query topic. We take the majority vote as the relevance label; if no majority agreement is achieved, we take the average of the scores as the final label. To measure inter-annotator agreement, we compute the Kappa test statistics on test annotations, which is 0.47. According to Reference [22], this is considered as moderate agreement. For each input query, there are on average 7.9 relevant and 6.28 highly relevant results.

6.3 Table-based Search

6.3.1 Queries. We sample 50 Wikipedia tables from the table corpus to be used as queries. Each table is required to have at least five rows and three columns [70]. These tables cover a diverse set of topics, including sports, music, films, food, celebrities, geography, and politics. See Figure 5 as an example.

6.3.2 Relevance Assessments. Ground-truth relevance labels are obtained as follows. For each input table, three keyword queries are constructed: (i) caption, (ii) table entities (entities from table plus the entity corresponding to the Wikipedia page in which the table is embedded), and (iii) table headings. Each keyword query is used to retrieve the top 150 results, resulting in at most 450 candidate tables for each query table. All methods that are compared in the experimental section operate by reranking these candidate sets. For each method, the top 10 results are manually annotated.

Each query–table pair is judged on a three point scale: non-relevant (0), relevant (1), and highly relevant (2). A table is highly relevant if it is about the same topic as the input table, but contains additional novel content that is not present in the input table. A table is relevant if it is on-topic, but it contains limited novel content; i.e., the content largely overlaps with that of the input table.[3] Otherwise, the table is not relevant; this also includes tables without substantial content. Three colleagues were employed and trained as annotators. We take the majority vote as the relevance label; if no majority vote is achieved, the mean score is used as the final label. To measure inter-annotator agreement, we compute the Fleiss Kappa test statistics, which is 0.6703. According to [22], this is considered as substantial agreement. For each input table, there are on average 7.28 relevant and 4.18 highly relevant results.

7 EVALUATION

In this section, we list our research questions (Section 7.1), discuss our experimental setup (Section 7.2), introduce the baselines we compare against (Section 7.3), and present our results (Section 7.4).

7.1 Research Questions

The research questions we seek to answer are as follows.

- **RQ1.** Can semantic matching improve retrieval performance?
- **RQ2.** Which of the semantic representations is the most effective?
- **RQ3.** Which of the similarity measures performs better?
- **RQ4.** How much do different table elements contribute to retrieval performance?

7.2 Experimental Setup

We evaluate table retrieval performance in terms of **Normalized Discounted Cumulative Gain (NDCG)** at cut-off points 5, 10, 15, and 20. Our main evaluation metric for keyword-based search is NDCG@20, while for table-based search it is NDCG@10.[4] We also report feature importance based on Gini score, which is a measure of feature contribution. Specifically, it computes each feature's importance as the sum over the number of splits that include the feature, proportionally to the number of samples it splits [35]. Formally,

$$g(\tau) = 1 - p_1^2 - p_0^2, \tag{5}$$

where $p = \frac{n_k}{n}$ is the fraction of the n_k samples from class $k = \{0, 1\}$ of the total of n samples at node τ. To test significance, we use a two-tailed paired t-test and write †/‡ to denote significance at the 0.05 and 0.005 levels, respectively.

Our implementations are based on the Nordlys toolkit [25]. We use an inverted index as the main underlying data structure. Each table is stored in an inverted index, with title, table caption, headings, entities, and table data fields. Additionally, all table-related data is concatenated and stored in a single "catchall" field. All fields are parsed with stopword removal and stemming using Elasticsearch.

Many of our features involve external sources, which we explain below. To compute the entity-related features (i.e., features in Table 2 as well as the features based on the bag-of-entities representations in Table 4), we use entities from the DBpedia knowledge base that have an abstract

[3]We note that the novelty requirement is not something we consider in our modeling. The purpose behind the use of this particular term was to have a simple working definition of relevance that discourages finding duplicate content.

[4]This choice is determined by the depth of the pools used when performing the assessments for each task. At these cutoffs, we have complete judgments for all methods that are being compared.

Table 9. Keyword-based Search Performance

Method	NDCG@5	NDCG@10	NDCG@15	NDCG@20
Single-field document ranking	0.4315	0.4344	0.4586	0.5254
Multi-field document ranking	0.4770	0.4860	0.5170	0.5473
WebTable [11]	0.2831	0.2992	0.3311	0.3726
WikiTable [7]	0.4903	0.4766	0.5062	0.5206
LTR-k	0.5527	0.5456	0.5738	0.6031
STR-k	**0.5951**	**0.6293**†	**0.6590**‡	**0.6825**†

Significance is tested against LTR-k. Highest scores are in bold.

(4.6M in total). The table's Wikipedia rank (yRank) is obtained using Wikipedia's MediaWiki API. The PMI feature is estimated based on the ACSDb corpus [12]. For the distributed representations, we take pre-trained embedding vectors, as explained in Section 5.2.2.

7.3 Implementation of Baselines

7.3.1 Baselines for Keyword-based Search. We implement four baseline methods from the literature.

Single-field document ranking. In References [9, 11] tables are represented and ranked as ordinary documents. Specifically, we use Language Models with Dirichlet smoothing, and optimize the smoothing parameter using a parameter sweep.

Multi-field document ranking. Pimplikar and Sarawagi [46] represent each table as a fielded document, using five fields: Wikipedia page title, table section title, table caption, table body, and table headings. We use the Mixture of Language Models approach [42] for ranking. Field weights are optimized using the coordinate ascent algorithm; smoothing parameters are trained for each field individually.

WebTable. The method by Cafarella et al. [11] uses the features in Table 2 with Reference [11] as source. Following Reference [11], we train a linear regression model with fivefold cross-validation.

WikiTable. The approach by Bhagavatula et al. [7] uses the features in Table 2 with Reference [7] as source. We train a Lasso model with coordinate ascent with fivefold cross-validation.

Additionally, we present a learning-to-rank approach using a rich set of features as a strong baseline:

LTR-k. It uses the full set of features listed in Table 2. We employ pointwise regression using the Random Forest algorithm.[5] We set the number of trees to 1,000 and the maximum number of features in each tree to 3. We train the model using fivefold cross-validation (w.r.t. NDCG@20).

The baseline results are presented in Table 9. It can be seen from this table that our LTR-k baseline (row five) outperforms all existing methods from the literature; the differences are substantial and statistically significant. Therefore, in the remainder of this article, we shall compare against this strong baseline, using the same learning algorithm (Random Forests) and parameter settings. We

[5]We also experimented with Gradient Boosting regression and Support Vector Regression, and observed the same general patterns regarding feature importance. However, their overall performance was lower than that of Random Forests. We note that further improvements may be obtained by using other machine learning algorithms, e.g., LambdaMART, but this exploration is outside our scope.

note that our emphasis is on the semantic matching features and not on the supervised learning algorithm.

7.3.2 Baselines for Table-based Search. We implement eight existing methods from literature as baselines.

Keyword-based search using T_E. The candidate table's score is computed by taking the terms from $\tilde{T}_E$ as the keyword query [1]. This method queries an index of the table corpus against the table entities.

Keyword-based search using T_H. Ahmadov et al. [1] also use table headings as keyword queries. This method queries an index of the table corpus against the table headings.

Keyword-based search using T_c. Additionally, in this article we also consider using the table caption as a query. This method searches against both the caption and catchall fields.

Mannheim Search Join Engine. All candidate tables are scored against the input table using the FastJoin matcher [63]. The edit distance threshold is set to $\delta = 0.8$.

Schema complement. Das Sarma et al. [17] aggregate the benefits of adding additional attributes from candidates tables to input tables as the matching score. The heading frequency statistics is calculated based on the Wikipedia table corpora and the heading similarity is aggregated using average.

Entity complement. The aggregated scores of the benefits of adding additional entities is taken as the matching score [17]. WLM is based on entity out-links. The data similarity threshold is set the same as for string comparison, i.e., $\delta = 0.8$.

Nguyen et al. Headings and table data are represented as term vectors for table matching in Reference [41]. The smoothing parameter value is taken from [41] to be $\alpha = 0.5$.

InfoGather. Element-wise similarity across four table elements: table data, column values, page title, and column headings are combined by training a linear regression scorer [66]. InfoGather is trained using linear regression with coordinate ascent.

Additionally, we present a learning-to-rank approach in two variants, using different sets of features, as strong baselines:

LTR-t1. It uses all the table matching scores in Section 4.3. We take the same training mechanism and configuration as for LTR-k (that is, we optimize for NDCG@10).

LTR-t2. Compared to LTR-t1, it additionally considers table features, for both the query and candidate tables, and follows the same training step.

The first block in Table 10 presents the evaluation results for the eight baselines for table-based search. Among the three keyword-based search methods, which operate on a single table element (top 3 lines), the one that uses table headings as the keyword query performs the best, followed by table entities and table caption. These unsupervised methods essentially treat partial tables (omitting rows and columns) as queries. The methods in lines 4–8 consider multiple table elements; all of these outperform the best single-element method. The approach that performs best among all, by a large margin, is InfoGather, which incorporates four different table elements. Consequently, in our discussion below, we will focus exclusively on InfoGather as the best baseline from the literature.

7.4 Experimental Results

We now answer our research questions based on the results of our experiments.

RQ1. Can semantic matching improve retrieval performance?

Table 10. Table-based Search Performance

Method	**NDCG@5**	**NDCG@10**
Keyword-based search using T_E	0.2001	0.1998
Keyword-based search using T_H	0.2318	0.2527
Keyword-based search using T_c	0.1369	0.1419
Mannheim Search Join Engine [63]	0.3298	0.3131
Schema complement [17]	0.3389	0.3418
Entity complement [17]	0.2986	0.3093
Nguyen et al. [41]	0.2875	0.3007
InfoGather [66]	**0.4530**	**0.4686**
LTR-t1 (feats. from Section 4.2.3)	0.5382	0.5542
LTR-t2 (feats. from Section 4.2.3 and Table features in Tbl. 2)	**0.5895†**	**0.6050†**
STR-t1	0.5578	0.5672
STR-t2	**0.6172‡**	**0.6267‡**
STR-t3	0.5140	0.5282
STR-t4	0.5804†	0.6027†

Significance is tested against InfoGather. Highest scores are in bold.

For keyword-based search, the last line of Table 9 shows the results for our semantic table retrieval method (STR-k). It combines the baseline set of features (Table 2) with the set of novel semantic matching features (from Table 4, 12 in total). We find that these semantic features bring in substantial and statistically significant improvements over the strong LTR baseline. The relative improvements range from 7.6% to 15.3%, depending on the rank cut-off.

For table-based search, reported in Table 10, we first compare our strong baselines, LTR-*, against the best method from the literature, InfoGather. LTR-t1, which combines all table similarity features from existing approaches, achieves 18.27% improvement upon InfoGather in terms of NDCG@10, albeit the differences are not statistically significant. LTR-t2 incorporates additional table features, which leads to substantial (29.11% for NDCG@10) and significant improvements over InfoGather. Next, we consider four specific instantiations of our table matching framework (cf. Table 7), which are presented in the bottom block of Table 10. Recall that STR-t1 employs only table similarity features, thus it is to be compared against LTR-t1. STR-t2..4 additionally consider table features, which corresponds to the settings in LTR-t2. We find that STR-t1 and STR-t2 outperform the respective LTR method, while STR-t4 is on par with it. None of the differences between STR-t* and the respective LTR method are statistically significant. Taking a conservative stand, we might say that our semantic table matching methods are on par with the strong baselines (and not better than them). This is already a great result for two reasons. One is that the strong baselines rely on a large set of hand-crafted features and we can match that performance without extensive feature engineering. The other is that the strong baselines already outperform the best reported approach in the literature by a substantial margin and deliver excellent performance (LTR-* vs. InfoGather).

In summary, we answer RQ1 positively; semantic matching can improve retrieval performance substantially, both for keyword-based and for table-based search.

RQ2. Which of the semantic representations is the most effective?

For keyword-based search, Table 11 reports on all combinations of semantic representations and similarity measures. Concerning the comparison of different semantic representations, we find that bag-of-entities and word embeddings achieve significant improvements; see the rightmost column

Table 11. Comparison of Semantic Features for Keyword-based Search, Used in Combination with Baseline Features for Keyword-based Search (from Table 2), in Terms of NDCG@20

Sem. Repr.	Early	Late-max	Late-sum	Late-avg	ALL
Bag-of-entities	0.6754	0.6407	0.6697	0.6733	0.6696
Word embeddings	0.6181	0.6328	0.6371	0.6485	0.6588
Graph embeddings	0.6326	0.6142	0.6223	0.6316	0.6340
ALL	0.6736	0.6631	0.6831	0.6809	0.6825

Table 12. Comparison of Semantic Features for Table-based Search, Used in Combination with Table Features (Middle Block in Table 2), in Terms of NDCG@10

Sem. Repr.	Early	Late-max	Late-sum	Late-avg	ALL
Bag-of-entities	0.5487	0.5764	0.5420	0.5293	0.5603
Word embeddings	0.6173	0.5314	0.5512	0.5337	0.6048
Graph embeddings	0.5072	0.5676	0.5270	0.5313	0.5559
ALL	0.6157	0.6031	0.5523	0.5631	0.6267

of Table 11. It is worth pointing out that for word embeddings the three similarity measures seem to complement each other, as their combined performance is better than that of any individual method. It is not the case for bag-of-entities, where only one of the similarity measures (Late-max) is improved by the combination. Overall, we find the bag-of-entities representation to be the most effective one. The fact that this sparse representation outperforms word embeddings is regarded as a somewhat surprising finding, given that the latter has been trained on massive amounts of (external) data.

For table-based search, Table 12 displays the results for each of the three semantic representations. Among those, word-based performs the best, followed by bag-of-entities and graph embeddings. It is different from the findings on keyword-based search. The differences between bag-of-entities and word embeddings are significant ($p < 0.01$), but not between the other pairs of representations. It is worth pointing out that any of the three representations alone would deliver better performance than the best existing method in the literature, InfoGather (cf. Table 11). When combing all three semantic representations (line 4, which is the same as STR-t1 in Table 10), we obtain substantial and significant improvements (p <0.01) over each individual representation. This shows the complimentary nature of these semantic representations.

In summary, bag-of-entities representations are the most effective for keyword-based search, and word embeddings work best for table-based search. However, these representations complement each other, and thus the combination of them performs best.

RQ3. Which of the similarity measures performs better?

For keyword-based search, it is difficult to name a clear winner when a single semantic representation is used. The relative differences between similarity measures are generally small (below 5%). When all three semantic representations are used (bottom row in Table 11), we find that Late-avg and Late-sum achieve the highest overall improvement. Importantly, when using all semantic representations, all four similarity measures improve significantly and substantially over the baseline. We further note that the combination of all similarity measures do not yield further improvements

Table 13. Element-wise Similarities for Various Semantic Representations for Table-based Search

	Word embeddings				Graph embeddings				Bag-of-entities		
	T_t	T_H	T_D		T_t	T_E	T_D		T_t	T_E	T_D
$\tilde{T}_t$	**0.2814**	0.0261	0.0436	$\tilde{T}_t$	**0.2765**	0.0546	0.0430	$\tilde{T}_t$	**0.4796**	0.0808	0.0644
$\tilde{T}_H$	0.0336	**0.1694**	0.0288	$\tilde{T}_E$	**0.0700**	0.0679	0.0501	$\tilde{T}_E$	0.0705	0.0617	**0.0725**
$\tilde{T}_D$	0.0509	0.0183	**0.1276**	$\tilde{T}_D$	**0.1012**	0.0423	0.0259	$\tilde{T}_D$	**0.1052**	0.0812	0.0610

Rows and columns corresponds to elements of the input and candidate tables, respectively. The evaluation metric is NDCG@10. The best scores for each block are in bold.

over Late-sum or Late-avg. Thus, we identify the late fusion strategy with sum or avg aggregation (i.e., Late-sum or Late-avg) as the preferred similarity method.

For table-based search, Early achieves the best performance, followed by Late-max, Late-avg, and Late-sum (bottom row in Table 12).

In answer to RQ3, late fusion performs better for keyword-based search, while early fusion outperforms late fusion for table-based search. However, the differences are often small. We hypothesize that early fusion performs better when obtaining representations "symmetrically." Recall that table-based search takes entities for representing both queries and tables (or, more precisely, input and candidate tables) from the same sources symmetrically, while keyword-based search retrieves entities asymmetrically (the queries only get them from the knowledge base, while tables can also get entities from their body). However, this hypothesis would need to be carefully examined in a future study. Overall, the different similarity measures seem to complement each other, and taking their combination works well across both tasks.

RQ4. How much do different table elements contribute to retrieval performance?

Note that this research question is specific to table-based search. Based on the results in Table 10, we observe that cross-element matching is less effective than element-wise matching (STR-t3 vs. STR-t2). We also find that considering all the cross-element similarities actually hurts performance (STR-t4 vs. STR-t2). To get a better understanding of how the element-wise and cross-element matching strategies compare against each other, we break down retrieval performance for all table element pairs according to the different semantic representations in Table 13. That is, we rank tables by measuring similarity only between that pair of elements (four table similarity features in total). Here, diagonal cells correspond to element-wise matching and all other cells correspond to cross-element matching. We observe that element-wise matching works best across the board. This is in line with our earlier findings, i.e., STR-t2 vs. STR-t3 in Table 10. However, for graph embeddings and bag-of-entities representations, there are several cases where cross-element matching yields higher scores than element-wise matching. Notably, input table data ($\tilde{T}_D$) has much higher similarity against the topic of the candidate table (T_t) than against its data (T_D) element, for both graph embeddings and bag-of-entities representations. This shows that cross-element matching does have merit for certain table element pairs. We perform further analysis in Section 8.2.1.

To explore the importance of table elements, we turn to Table 13 once again. We first compare the results for element-wise similarity (i.e., the diagonals) and find that among the four table elements, table topic ($\tilde{T}_t \leftrightarrow T_t$) contributes the most and table data ($\tilde{T}_D \leftrightarrow T_D$) contributes the least. Second, our observations for cross-element matching are as follows. In terms of feature importance (measured in terms of Gini score), using word embeddings, table data ($\tilde{T}_D$) is the most important element for the input table, while for the candidate table it is table topic (T_t). Interestingly, for graph embeddings and bag-of-entities representations it is exactly the other way around: the most

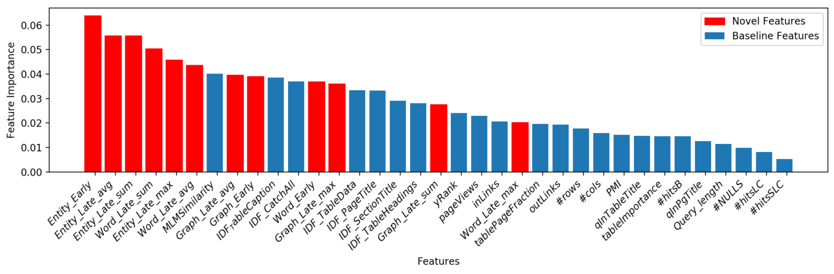


Fig. 6. Normalized feature importance (measured in terms of Gini score) of keyword-based search.

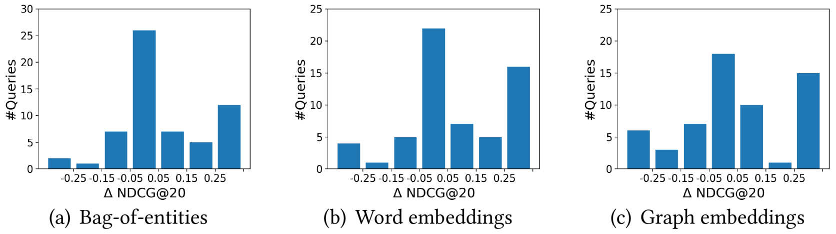


Fig. 7. Distribution of query-level differences between the LTR-k baseline and a given semantic representation for keyword-based search.

important input table element is topic ($\tilde{T}_t$), while the most important candidate table element is data (T_D).

8 ANALYSIS

We continue with further analysis of our results.

8.1 Further Analysis for Keyword-based Search

8.1.1 Features. Figure 6 shows the importance of individual features for the keyword-based search task, measured in terms of Gini importance. The novel features are distinguished by color. We observe that 8 of the top 10 features are semantic features introduced in this article. Additionally, we conduct a feature analysis based on retrieval performance. We find that most of the individual features help to improve table ranking, of which "Graph_Late_sum" improves the most in terms of NDCG@20 (0.11%) and "Word_Late_avg" in terms NDCG@5 (3.8%)."

8.1.2 Semantic Representations. To analyze how the three semantic representations affect retrieval performance on the level of individual queries, we plot the difference between the LTR-k baseline and each semantic representation in Figure 7. The histograms show the distribution of queries according to NDCG@20 score difference (Δ): The middle bar represents no change ($\Delta <$ 0.05), while the leftmost and rightmost bars represents the number of queries that were hurt and helped substantially, respectively ($\Delta >$0.25). We observe similar patterns for the bag-of-entities and word embeddings representations; the former has less queries that were significantly helped or hurt, while the overall improvement (over all topics) is larger. We further note the similarity of the shapes of the distributions for graph embeddings.

Looking at specific queries, we find one query that falls in the leftmost (i.e., largest performance drop) bucket in all the plots in Figure 7: "cereals nutritional value." This query only has a single highly relevant table according to the ground truth. Further, there are five queries on which LTR consistently outperforms all semantic representations: "Olympus digital SLRs,"

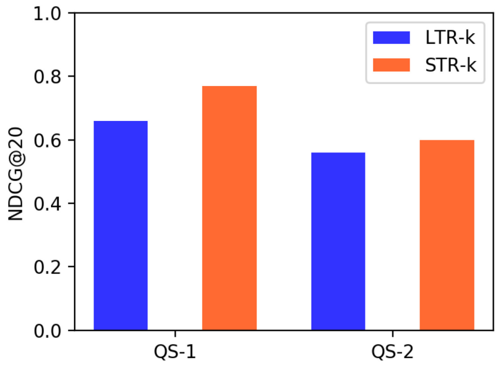
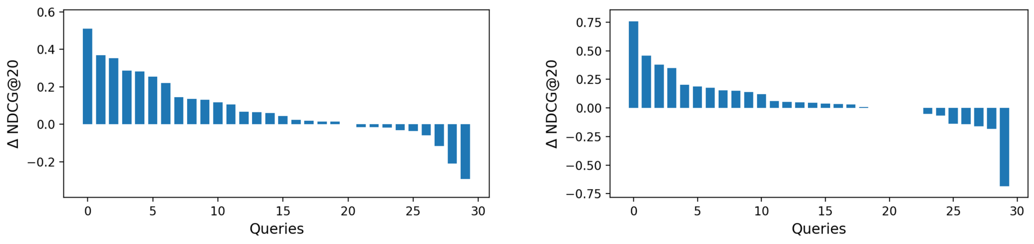


Fig. 8. Keyword-based search results, LTR-k baseline vs. STR-k, on the two query subsets in terms of NDCG@20.



(a) QS-1 (b) QS-2

Fig. 9. Query-level differences on the two query subsets between the LTR-k baseline and STR-k. Positive values indicate improvements made by the latter.

"fuel consumption," "Ibanez guitars," "science discoveries," and "lakes altitude." What is common in these queries is that they are all very precisely worded.

8.1.3 *Query Subsets.* On Figure 8, we plot the results for the LTR-k baseline and for our STR-k method according to the two query subsets, QS-1 and QS-2, in terms of NDCG@20. Generally, both methods perform better on QS-1 than on QS-2. This is mainly because QS-2 queries are more focused (each targeting a specific type of instance, with a required property), and thus are considered more difficult. Also, QS-1 has more relevant tables on average. Specifically, QS-1 has 8.4 relevant tables and 8.6 highly relevant tables, while QS-2 has 7.3 and 3.9, respectively. Importantly, STR-k achieves consistent improvements over LTR-k on both query subsets.

8.1.4 *Individual Queries.* We plot the difference between the LTR-k baseline and STR-k for the two query subsets in Figure 9. Table 14 lists the queries that we discuss below. The leftmost bar in Figure 9(a) corresponds to the query "*stocks.*" For this broad query, there are two relevant and one highly relevant tables. LTR-k does not retrieve any highly relevant tables in the top 20, while STR-k manages to return one highly relevant table in the top 10. The rightmost bar in Figure 9(a) corresponds to the query "*ibanez guitars.*" For this query, there are two relevant and one highly relevant tables. LTR-k produces an almost perfect ranking for this query, by returning the highly relevant table at the top rank, and the two relevant tables at ranks 2 and 4. STR-k returns a non-relevant table at the top rank, thereby pushing the relevant results down in the ranking by a single position, resulting in a decrease of 0.29 in NDCG@20.

Table 14. Example Keyword-based Search Queries from Our Query Set

Query	Rel	LTR-k	STR-k
QS-1-24: *stocks*			
Stocks for the Long Run/Key Data Findings: annual real returns	2	—	6
TOPIX/TOPIX New Index Series	1	9	—
Hang Seng Index/Selection criteria for the HSI constituent stocks	1	—	—
QS-1-21: *ibanez guitars*			
Ibanez/Serial numbers	2	1	2
Corey Taylor/Equipment	1	2	3
Fingerboard/Examples	1	4	5
QS-2-27: *board games number of players*			
List of Japanese board games	1	13	1
List of licensed Risk game boards/Risk Legacy	1	—	3
QS-2-21: *cereals nutritional value*			
Sesame/Sesame seed kernels, toasted	2	1	8
QS-2-20: *irish counties area*			
Counties of Ireland/List of counties	2	2	1
List of Irish counties by area/See also	2	1	2
List of flags of Ireland/Counties of Ireland Flags	2	—	3
Provinces of Ireland/Demographics and politics	1	4	4
Toponymical list of counties of the United Kingdom/Northern ...	1	—	7
Múscraige/Notes	1	—	6

Rel denotes table relevance level. LTR-k and STR-k refer to the positions on which the table is returned by the respective method.

The leftmost bar in Figure 9(b) corresponds to the query "*board games number of players*." For this query, there are only two relevant tables according to the ground truth. STR-k managed to place them in the first and third rank positions, while LTR-k returned only one of them at position 13. The rightmost bar in Figure 9(b) is the query "*cereals nutritional value*." Here, there is only one highly relevant result. LTR-k managed to place it in rank one, while it is ranked eighth by STR-k. Another interesting query is "*irish counties area*" (third bar from the left in Figure 9(b)), with three highly relevant and three relevant results according to the ground truth. LTR-k returned two highly relevant and one relevant results at ranks 1, 2, and 4. STR-k, however, placed the three highly relevant results in the top 3 positions and also returned the three relevant tables at positions 4, 6, and 7.

Observing the individual queries, we find that, thanks to the semantic representations that can help bridge the vocabulary gap, STR-k is generally able to identify more relevant results than LTR-k. It should also be noted that for queries with only a handful relevant results, the rank position of a single table can make a large difference in NDCG.

8.2 Further Analysis for Table-based Search

Next, we perform further performance analysis on individual features and on input table size for table-based search.

8.2.1 Feature Analysis. To understand the contributions of individual features, we first rank all features based on Gini importance. Then, we incrementally add features in batches of 10, and plot the corresponding retrieval performance in Figure 10. We observe that we can reach peak performance with using only the top-20 features.

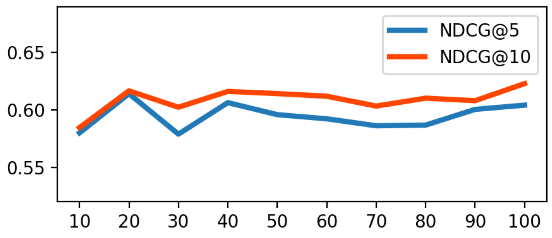


Fig. 10. Table-based search performance in terms of NDCG by incrementing the number of features used (features are ordered by Gini importance).

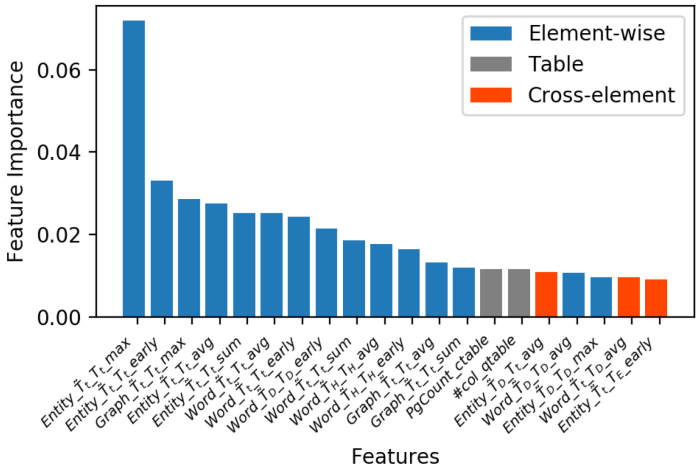


Fig. 11. Top-20 features in terms of Gini importance for table-based search.

Let us take a closer look at those top-20 features in Figure 11. We use color coding to help distinguish between the three main types of features: element-wise, cross-element, and table features. Then, based on these feature importance scores, we revisit our research questions. Concerning semantic representations, there are 8 word embedding, 7 entity embedding, and 3 graph embedding features in the top 20. Even though there are slightly more features using word embedding than entity embeddings, the latter features are much higher ranked (cf. Figure 11). Thus, the bag-of-entities semantic representation is the most effective one. Comparing matching strategies, the numbers of element-wise and crosswise features are 15 and 3, respectively. This indicates a substantial advantage of element-wise strategies. Nevertheless, it shows that incorporating the similarity between elements of different types can also be beneficial. Additionally, there are 2 table features in the top 20. As for the importance of table elements, table topic (T_t) is clearly the most important one; 8 of the top 10 features consider that element. In summary, our observations based on the top-20 features are consistent with our earlier findings.

8.2.2 Input Table Size. Next, we explore how the size of the input table affects retrieval performance. Specifically, we vary the input table size by splitting it horizontally (varying the number of rows) or vertically (varying the number of columns), and using only a portion of the table as input; see Figure 12 for an illustration. We explore four settings by setting the split rate x between 25% and 100% in steps of 25%. Figure 13 plots retrieval performance against input table size. We observe that growing the table, either horizontally or vertically, results in proportional increase in retrieval performance. This is not surprising, given that larger tables contain more information.

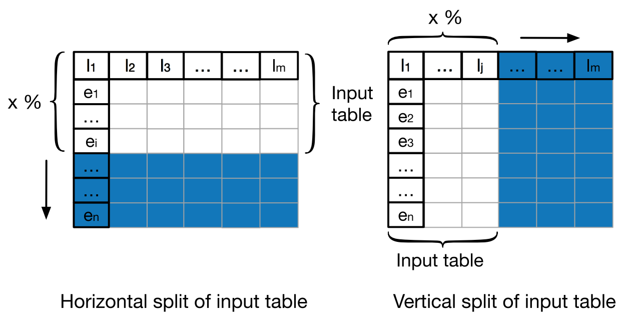


Fig. 12. Horizontal/vertically splitting of tables for performance analysis of table-based search.

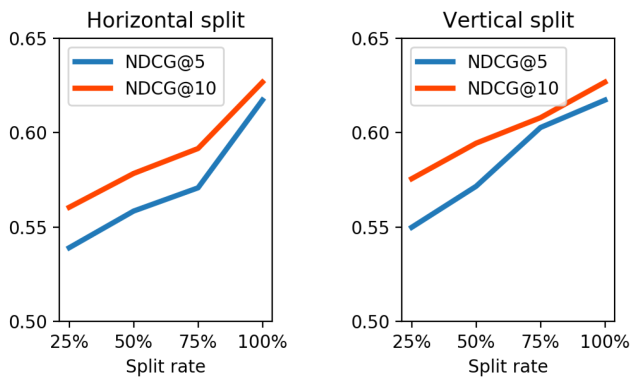


Fig. 13. Performance of STR-t2 with respect to (relative) input table size, by varying the number of rows (left) or columns (right).

Nevertheless, being able to utilize this extra information effectively is an essential characteristic of our table matching framework.

9 CONCLUSION

In this article, we have introduced and addressed the problem of *table retrieval*: answering an information need with a ranked list of tables. Specifically, we have studied this problem in two different flavors: *keyword-based search*, where the information need is specified as a keyword query, and *table-based search*, where an existing table is used as input. The main contribution of this study is a *semantic table retrieval* framework, which allows us to incorporate semantic matching into the task of table retrieval in a principled and effective way. In this framework, queries and tables can be represented using semantic concepts (bag-of-entities) as well as continuous dense vectors (word and graph embeddings) in a uniform way. We have introduced multiple similarity measures for matching those semantic representations. We have presented and experimentally compared a number of specific instantiations of the matching framework, depending on the type of the input query (keywords or table). For evaluation, we have developed two purpose-built test collections based on Wikipedia tables. We have considered a number of approaches from the literature for baseline comparison, and have also developed strong baselines for each task by combining elements from prior studies in feature-based supervised learning approaches. These strong baselines represent substantial and significant improvements over all previous methods. We have demonstrated that our semantic table retrieval approaches can either match (for table-based search) or significantly outperform (for keyword-based search) these strong baselines, while, unlike those, do not require extensive feature-engineering.

There is a number of possible avenues to be considered in future work. In this article, we have resorted ourselves to simple pre-trained embeddings. We conjecture that table retrieval would also benefit from advances in representation learning and neural language modeling, and possibly from task-specific fine-tuning. Naturally, there are also other ways for aggregating word/entity embeddings into table element level embeddings, beyond what we explored in this article. Another direction concerns the choice of the table corpus. In this work, we have worked with Wikipedia tables, which helped us to focus on modeling challenges, without being hindered by data quality issues. It remains to be seen whether our findings generalize over to a more heterogeneous table collection with varying quality and imperfect entity annotations.

REFERENCES

[1] Ahmad Ahmadov, Maik Thiele, Julian Eberius, Wolfgang Lehner, and Robert Wrembel. 2015. Towards a hybrid imputation approach using web tables. In *Proceedings of the IEEE 2nd International Symposium on Big Data Computing (BDC'15)*. 21–30.

[2] Marie Anan and Gal Avigdor. 2007. On the stable marriage of maximum weight royal couples. In *Proceedings of the IIweb'07*. 1–6.

[3] Ebrahim Bagheri and Feras Al-Obeidat. 2020. A latent model for ad hoc table retrieval. In *Advances in Information Retrieval*. 86–93.

[4] Sreeram Balakrishnan, Alon Y. Halevy, Boulos Harb, Hongrae Lee, Jayant Madhavan, Afshin Rostamizadeh, Warren Shen, Kenneth Wilder, Fei Wu, and Cong Yu. 2015. Applying WebTables in practice. In *Proceedings of the CIDR'15*.

[5] Krisztian Balog. 2018. *Entity-Oriented Search*. The Information Retrieval Series, Vol. 39. Springer.

[6] Somnath Banerjee, Soumen Chakrabarti, and Ganesh Ramakrishnan. 2009. Learning to rank for quantity consensus queries. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'09)*. 243–250.

[7] 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'13)*. 18–26.

[8] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: Entity linking in web tables. In *Proceedings of the 14th International Conference on The Semantic Web (ISWC'15)*. 425–441.

[9] Michael J. Cafarella, Alon Halevy, and Nodira Khoussainova. 2009. Data integration for the relational web. *Proc. VLDB Endow.* 2, 1 (Aug. 2009), 1090–1101.

[10] Michael J. Cafarella, Alon Halevy, and Jayant Madhavan. 2011. Structured data on the web. *Commun. ACM* 54 (2011), 72–79.

[11] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. WebTables: Exploring the power of tables on the web. *Proc. VLDB Endow.* 1, 1 (Aug. 2008), 538–549.

[12] Michael J. Cafarella, Alon Y. Halevy, Yang Zhang, Daisy Zhe Wang, and Eugene Wu 0002. 2008. Uncovering the relational web. In *Proceedings of the 11th International Workshop on the Web and Databases (WebDB'08)*.

[13] Jing Chen, Chenyan Xiong, and Jamie Callan. 2016. An empirical study of learning to rank for entity search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'16)*. 737–740.

[14] 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 International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 589–598.

[15] Fernando Chirigati, Jialu Liu, Flip Korn, You (Will) Wu, Cong Yu, and Hao Zhang. 2016. Knowledge exploration using tables on the web. *Proc. VLDB Endow.* 10, 3 (Nov. 2016), 193–204.

[16] Eric Crestan and Patrick Pantel. 2011. Web-scale table census and classification. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining (WSDM'11)*. 545–554.

[17] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding related tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. 817–828.

[18] Li Deng, Shuo Zhang, and Krisztian Balog. 2019. Table2Vec: Neural word and entity embeddings for table population and retrieval. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'19)*. 1029–1032.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* 4171–4186.

[20] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14).* 601–610.

[21] D. W. Embley, M. Hurst, D. P. Lopresti, and G. Nagy. 2006. Table-processing paradigms: A research survey. *Int. J. Doc. Anal. Recogn.* 8, 2–3 (Jun. 2006), 66–86.

[22] J. L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin* 76 (1971), 378–382.

[23] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J. F. Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15).* 795–798.

[24] Mihajlo Grbovic, Nemanja Djuric, Vladan Radosavljevic, Fabrizio Silvestri, and Narayan Bhamidipati. 2015. Context- and content-aware embeddings for query rewriting in sponsored search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15).* 383–392.

[25] Faegheh Hasibi, Krisztian Balog, Darío Garigliotti, and Shuo Zhang. 2017. Nordlys: A toolkit for entity-oriented and semantic search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17).* 1289–1292.

[26] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-entity v2: A test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17).* 1265–1268.

[27] Tom Kenter and Maarten de Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM'15).* 1411–1420.

[28] Oliver Lehmberg, Dominique Ritze, Petar Ristoski, Robert Meusel, Heiko Paulheim, and Christian Bizer. 2015. The mannheim search join engine. *Web Semant.* 35, P3 (Dec. 2015), 159–166.

[29] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types and relationships. *Proc. VLDB Endow.* 3, 1–2 (Sept. 2010), 1338–1347.

[30] Tie-Yan Liu. 2011. *Learning to Rank for Information Retrieval.* Springer, Berlin.

[31] Ying Liu, Kun Bai, Prasenjit Mitra, and C. Lee Giles. 2007. TableSeer: Automatic table metadata extraction and searching in digital libraries. In *Proceedings of the Joint Conference on Digital Libraries (JCDL'07).* 91–100.

[32] Craig Macdonald, Rodrygo L. T. Santos, and Iadh Ounis. 2012. On the usefulness of query features for learning to rank. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM'12).* 2559–2562.

[33] Jayant Madhavan, Loredana Afanasiev, Lyublena Antova, and Alon Y. Halevy. 2009. Harnessing the deep web: Present and future. *CoRR* abs/0909.1785 (2009).

[34] Jarana Manotumruksa, Craig MacDonald, and Iadh Ounis. 2016. Modelling user preferences using word embeddings for context-aware venue recommendation. *CoRR* abs/1606.07828 (2016).

[35] Bjoern H. Menze, B. Michael Kelm, Ralf Masuch, Uwe Himmelreich, Peter Bachert, Wolfgang Petrich, and Fred A. Hamprecht. 2009. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data.*BMC Bioinform.* 10 (2009).

[36] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems, Volume 2 (NIPS'13).* Curran Associates Inc., 3111–3119.

[37] David Milne and Ian H. Witten. 2008. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *Proceedings of the 1st AAAI Workshop on Wikipedia and Artifical Intellegence (WIKIAI'08).*

[38] Bhaskar Mitra, Eric T. Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *CoRR* abs/1602.01137 (2016).

[39] Emir Muñoz, Aidan Hogan, and Alessandra Mileo. 2014. Using linked data to mine RDF from Wikipedia's tables. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM'14).* 533–542.

[40] Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *CoRR* abs/1511.04834 (2015).

[41] Thanh Tam Nguyen, Quoc Viet Hung Nguyen, Weidlich Matthias, and Aberer Karl. 2015. Result selection and summarization for web table search. In *Proceedings of the 31st International Conference on Data Engineering (ISDE'15).* 231–242.

[42] Paul Ogilvie and Jamie Callan. 2003. Combining document representations for known-item search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval (SIGIR'03).* 143–150.

[43] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[44] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. 701–710.

[45] Mohammad Taher Pilehvar and Jose Camacho-Collados. 2020. *Embeddings in Natural Language Processing*. Morgan & Claypool Publishers.

[46] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *Proc. VLDB Endow.* 5 (2012), 908–919.

[47] David Pinto, Michael Branstein, Ryan Coleman, W. Bruce Croft, Matthew King, Wei Li, and Xing Wei. 2002. QuASM: A system for question answering using semi-structured data. In *Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'02)*. 46–55.

[48] P. Pyreddy and W. B. Croft. 1997. *TINTI: A System for Retrieval in Text Tables TITLE2.* Technical Report. USA.

[49] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Inf. Retr.* 13, 4 (2010), 346–374.

[50] Hadas Raviv, Oren Kurland, and David Carmel. 2016. Document retrieval using entity-based language models. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'16)*. 65–74.

[51] Petar Ristoski and Heiko Paulheim. 2016. RDF2Vec: RDF graph embeddings for data mining. In *Proceedings of the 15th International Semantic Web Conference (ISWC'16),* Lecture Notes in Computer Science, Paul T. Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil (Eds.), Vol. 9981. 498–514.

[52] Gaetano Rossiello, Pierpaolo Basile, and Giovanni Semeraro. 2017. Centroid-based text summarization through compositionality of word embeddings. In *Proceedings of the MultiLing 2017 Workshop on Summarization and Summary Evaluation across Source Types and Genres*. Association for Computational Linguistics, 12–21.

[53] Sunita Sarawagi and Soumen Chakrabarti. 2014. Open-domain quantity queries on web tables: Annotation, response, and consensus models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. 711–720.

[54] Yoones A. Sekhavat, Francesco Di Paolo, Denilson Barbosa, and Paolo Merialdo. 2014. Knowledge base augmentation using tabular data. In *Proceedings of the Conference on Linked Data on the Web (LDOW'14).*

[55] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Canim. 2020. Ad hoc table retrieval using intrinsic and extrinsic similarities. In *Proceedings of the World Wide Web Conference 2020 (WWW'20)*. 2479–2485.

[56] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Mustafa Cannim. 2020. Web table retrieval using multimodal deep learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'20)*. 1399–1408.

[57] Roee Shraga, Haggai Roitman, Guy Feigenblat, and Bar Weiner. 2020. Projection-based relevance model for table retrieval. In *Companion Proceedings of the Web Conference 2020 (WWW'20)*. 28–29.

[58] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web (WWW'15).* International World Wide Web Conferences Steering Committee, 1067–1077.

[59] M. Trabelsi, B. D. Davison, and J. Heflin. 2019. Improved table retrieval using multiple context embeddings for attributes. In *Proceedings of the 2019 IEEE International Conference on Big Data (Big Data'19)*. 1238–1244.

[60] Stephen Tyree, Kilian Q. Weinberger, Kunal Agrawal, and Jennifer Paykin. 2011. Parallel boosted regression trees for web search ranking. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11).* 387–396.

[61] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering semantics of tables on the web. *Proc. VLDB Endow.* 4, 9 (June 2011), 528–538.

[62] Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'15)*. 363–372.

[63] Jiannan Wang, Guoliang Li, and Jianhua Fe. 2011. Fast-join: An efficient method for fuzzy token matching based string similarity join. In *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering (ICDE'11).* 458–469.

[64] Xing Wei, Bruce Croft, and Andrew Mccallum. 2006. Table extraction for answer retrieval. *Inf. Retr.* 9, 5 (nov 2006), 589–611.

[65] Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. 2017. Word-entity duet representations for document ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17)*. 763–772.

[66] Mohamed Yakout, Kris Ganjam, Kaushik Chakrabarti, and Surajit Chaudhuri. 2012. InfoGather: Entity augmentation and attribute discovery by holistic matching with web tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data (SIGMOD'12)*. 97–108.

[67] Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2016. Neural enquirer: Learning to query tables in natural language. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*. 2308–2314.

[68] Meihui Zhang and Kaushik Chakrabarti. 2013. InfoGather+: Semantic matching and annotation of numeric and time-varying attributes in web tables. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD'13)*. 145–156.

[69] Shuo Zhang and Krisztian Balog. 2017. Design patterns for fusion-based object retrieval. In *Proceedings of the 39th European Conference on Advances in Information Retrieval (ECIR'17)*. 684–690.

[70] Shuo Zhang and Krisztian Balog. 2017. EntiTables: Smart assistance for entity-focused tables. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'17)*. 255–264.

[71] Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proceedings of the World Wide Web Conference 2018 (WWW'18)*. 1553–1562.

[72] Shuo Zhang and Krisztian Balog. 2019. Auto-completion for data cells in relational tables. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM'19)*. 761–770.

[73] Shuo Zhang and Krisztian Balog. 2019. Recommending related tables. arxiv:1907.03595. Retrieved from http://arxiv.org/abs/1907.03595.

[74] Shuo Zhang and Krisztian Balog. 2020. Web table extraction, retrieval, and augmentation: A survey. *ACM Trans. Intell. Syst. Technol.* 11, 2, Article Article 13 (Jan. 2020), 35 pages. DOI: https://doi.org/10.1145/3372117

[75] Shuo Zhang, Krisztian Balog, and Jamie Callan. 2020. Generating categories for sets of entities. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. Association for Computing Machinery, New York, NY, 1833–1842.

[76] Shuo Zhang, Edgar Meij, Krisztian Balog, and Ridho Reinanda. 2020. Novel entity discovery from web tables. In *Proceedings of the World Wide Web Conference 2020 (WWW'20)*. Association for Computing Machinery, New York, NY, 1298–1308. DOI: https://doi.org/10.1145/3366423.3380205

[77] Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL'15)*. 250–259.

[78] Stefan Zwicklbauer, Christoph Einsiedler, Michael Granitzer, and Christin Seifert. 2013. Towards disambiguating web tables. In *Proceedings of the 12th International Semantic Web Conference (ISWC-PD'13)*. 205–208.

Received May 2020; revised October 2020; accepted December 2020