Reproduction of the paper: Ad Hoc Table Retrieval sing Semantic Similarity

Þorunn Arna Omarsdottir
4917499
Delft, The Netherlands
Omarsdottir@student.tudelft.nl

Rukai Yin
4737371
Delft, The Netherlands
R.yin@student.tudelft.nl

Nick Boyd
4988868
Delft, The Netherlands
N.T.Boyd@student.tudelft.nl

ABSTRACT

We have decided to reproduce the paper 'Ad Hoc Table Retrieval using Semantic Matching' By Shuo Zhang and Krisztian Balog [17]. In the original paper, the problem of table retrieval answering a keyword query with a ranked list of tables is introduced and addressed. Since the scope of the paper is rather big we are not going to attempt reproducing it all. We reproduce some of the baselines they used for evaluation and focus on one of the research questions: "*Can semantic matching improves retrieval performance?*". We have employed a set of experiments and reached an agreement on that adding semantic features improves the retrieval effectiveness of the system. By obtaining the desired result we can support their finding and have more evidence to trust it.

INTRODUCTION

Tables are a good way to formalize information clearly and keep it well structured. They are therefore a powerful, versatile and easy-to-use tool. Therefore it is very important that they are simple to access and that they are part of the corpus that search engines look at when retrieving documents that match best to a given query. This requires some research regarding how it is best to search for the most relevant tables. The original paper that we are reproducing is trying to do that. The table searching task has not gotten much attention in the Information Retrieval field and the plan was to try to fix that.

The ad hoc table retrieval task is defined as follows: given a keyword query, return a ranked list of tables from a table corpus that are relevant to the query. This task is not entirely new [5, 7, 9, 10]. 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. There are some previous works on ranking tables by considering the words contained in them [9, 10, 12] and incorporating additional signals related to table quality [7]. All of the prior approaches have the same limitation that they only consider lexical matching between the contents of tables and queries. This gives rise to the main research objective in the paper: Can we move beyond lexical matching and improve table retrieval performance by incorporating semantic matching? Lexical matching mainly focuses on the the appearance of query terms in table content, while semantic matching also inspects the semantic similarity between queries and tables. In this report, we would like to answer one of the research questions in the original paper:

RQ: Can semantic matching improve retrieval performance?

The original paper considers two kinds of semantic representations and introduces a framework that handles matching in different semantic spaces in a uniform way. It also proposes four different measures for computing the similarity between queries and tables based on their semantic representations.

In this report, we reproduce a part of the work done in the original paper. We are going to show that by adding some semantic features to a ranking algorithm, the results show improvements on the retrieval performance in terms of Normalized Discounted Cumulative Gain (NDCG) as is done in the original paper. But we are also going to take a look at the Recall and Mean average precision measures.

The test collection used in the original paper as well as in this report is a corpus with 1.6M tables from Wikipedia and a set of queries with graded relevance judgments. The corpus was created in relation to the original paper since there was no corpus composed of tables available. Section 3.4.2 includes more details of the test collection.

We show some background information in Section 2 and the implementation of three baselines as well as the semantic table retrieval (STR) method in Section 3. The baselines are Single-field document ranking, Multi-field document ranking and Learning-to-rank (LTR). We describe the experiment setups, perform experiments and compare the results of all methods in Section 4. Finally, we draw some conclusions in Section 5.

The training data, outputs of this report and scripts are available here https://github.com/RUKAIYIN/IN4325.

BACKGROUND

As mentioned before table search is an important task on its own and serves as a core-building block in other table-related task. The task has not gotten very much attention but still some. We will now go over the main ideas that have been proposed related to the content of this paper.

ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

The task has gotten more attention in the database field than in the information retrieval field. Some of the theories researched there have proven very useful for us and the original paper. An example of that is the table and query features that are going to be introduced later, but most of the ideas behind them are originated from the database community[7, 10].

As has been mentioned already tables can be handled much like documents in the retrieval task. You just have to know more about where and how it is best to look for data. There are some previous works on ranking tables by considering the words contained in them [9, 10, 12] and incorporating additional signals related to table quality [7]. All of these former researches have been very useful in the writing of this paper, both for the source of knowledge and methods.

The paper [5] is simply called "Finding related tables". It appears to be one of the first papers that actually focus on the table retrieval task. Their main contribution was a framework that captures several types of relatedness and the proposed a set of algorithms for detecting related tables that can be unioned or joined to get more information.

Since the main part of the paper we are reproducing is semantic similarity it is good to mention the main research that has been published in that field that is relevant to our paper. In [16] semantic matches between columns in webtables are computed and tried to find good methods to deal with the cases when same attributes are differently represented.

APPROACH

We start by implementing three baselines, that are inspired by the baselines that were used by the original paper. We focused on the single-field document ranking baseline, multi-field document ranking and the learning-to-rank baseline (LTR). The first two baselines are kind of self-explained and they consider tables as documents. The LTR takes various features extracted from queries and tables into account. We will then reproduce a part of the semantic matching method.

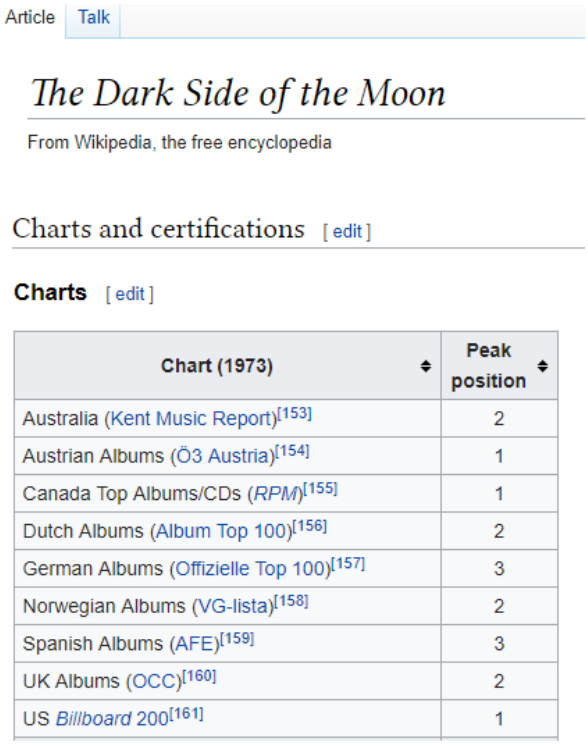
Test collection Setup

Anatomy of a Table

Before we go further it is important to explain how the tables we are going to be working with are structured. All the tables is the corpus are extracted from Wikipedia. Figure 1 shows the structure of a Wikipedia page with a table incorporated.

- *Page Title*: of the Wikipedia page the table was originally extracted from. "The dark side of the moon" in figure 1
- *Section Title*: the title of the section the table was embedded on wiki. "Charts and certifications" in figure 1
- *Table caption*: a short text providing the domain of the table. "Charts" in figure 1
- *Table heading*: the top row of the table. "Chart(1874) and Peak Position" in figure 1
- *Table body*: the rest of the table.

This structure can be assumed for the rest of the paper.


Article Talk

The Dark Side of the Moon

From Wikipedia, the free encyclopedia

Charts and certifications [edit]

Charts [edit]

Chart (1973)	Peak position
Australia (Kent Music Report)[153]	2
Austrian Albums (Ö3 Austria)[154]	1
Canada Top Albums/CDs (*RPM*)[155]	1
Dutch Albums (Album Top 100)[156]	2
German Albums (Offizielle Top 100)[157]	3
Norwegian Albums (VG-lista)[158]	2
Spanish Albums (AFE)[159]	3
UK Albums (OCC)[160]	2
US *Billboard* 200[161]	1

Figure 1. Illustrations that shows the table anatomy on a Wikipedia page

Queries

We use the same query set as was used in the original paper. That is done to be able to use the relevance judgment they gathered. The query set is combined of 60 query string, gotten from two sources.

- *Query subset 1*: From Cafarella's paper [9], which query set consist of queries from web users via crowdsourcing.
- *Query subset 2*: From Venetis's paper [11], which query set consist of queries that were analyzed from the query logs from Google Squared, which is a service where users can search for structured data.

Since the queries are all gathered from actual users they reflect real user behaviour. So they should be good to mimic real table retrieval tasks users might search for. Some example queries can be seen in table 1.

Example Queries
- world interest rates table
- running shoes
- pain medications
- football clubs city

Table 1. Example queries from our set

Table Corpus

The original paper uses WikiTables corpus [8], which comprises 1.6M tables extracted from Wikipedia (dump date: 2015 October). The authors pre-processed it and replaced all links in the table body with entity identifiers from the DBpedia

knowledge base (version 2015-10). The pre-processed version is provided on http://iai.group/downloads/smart_table/WP_tables.zip and we will use it to train all models in this report.

Relevance Judgment

In ground truth document the tables are given a relevance score in regards to the relevant query. These judgments were gathered by the researchers of the original paper[17]. They employed three independent trained judges, asked for their relevance judgment and then took the majority vote as the relevance label. If no majority agreement is achieved. they took the average of the scores as a final label. Each query-table pair was judged on a three-point scale:

- *Non-relevant: 0*, if it is unclear what the table is about or the table is about a different topic.
- *Relevant: 1*, if some cells or values could be used from this table.
- *Highly relevant: 2*, if large blocks or several values could be used from the table when creating a new table on the query topic.

The ground truth file consists of 3120 query-table pairs as ground truth, out of which 377 are labelled as highly relevant, 474 as relevant, and 2269 as non-relevant. It is available here https://github.com/iai-group/www2018-table.

Single-field document ranking

We approached this baseline by using Dirichlet language model similar to the authors of the original paper. In this case we consider a table as a normal document. Therefore, we will use an open source tool called *Terrier* to run the Dirichlet smoothing retrieval model.

The five fields: *title, secondary title, caption, table headers, table body* of our table representation were indexed as one document without consideration that they are all separate entities as data. The Dirichlet language model is using a $\mu = 2500$ due to the varying sizes of our tables in the corpus. If all the tables were similar in size, we would have approached the baseline looking to optimize this parameter.

Due to the substance of the query set and the occurrences of numbered data in the table corpus, we run the same retrieval without numbers in the documents. We predict this could improve the performance of the retrieval and could provide a better baseline for comparison. If you consider the set of queries string that are given none are searching to retrieve numbered data from the table body. Therefore, a numberless single field document could improve the performance without the need to parse useless data.

The results of each retrieval set will compared then used to prove the research question as this baseline is just lexical matching in a standard document. We will rank the tables and select top 5, 10, 15 and 20 rankings to perform evaluations in Section 4, Experiments.

Multi-field document ranking

Multi-field table representation places each of the five parts of the table: *title, secondary title, caption, table headers, table body* into their own field entities when indexing the table. If there was more than one entity per part they would be placed into separate field entities since we want to consider each piece of data separately when running the retrieval.

We approached this baseline by using BM25F, which is a per field model retrieval model. We decided not to use a language model retrieval the authors used of the original paper. This was mainly due to a lack of this feature for field models retrieval in Terrier. This retrieval model allows us to consider the occurrence of a term per field, which will provide us with the best results for the baseline.

The BM25F allows us to place weights on each of the five fields of the table representation. We did not take a similar approach when determining weights of the fields compared to the authors of the original work. They optimized the weights on a language model. It was determined to use a uniform weight per field because is it will still show the improvement over Single-Field representation. We believed that optimizing the weights would not provide results that could further prove the research question differently in this reproduction of the original paper.

The results of retrieval will be compared then used to prove the research question as this baseline is just to prove that field-based retrieval models and indices will improve over single-field table representation. They will improve due to the nature of a field based retrieval because they provide us an occurrence of a term per a field, therefore we know if specific terms are used in specific fields the table maybe ranked higher or lower. We will rank the tables and select top 5, 10, 15 and 20 rankings to perform evaluations in Section 4, Experiments.

Learning-to-rank

It uses the full set of features listed in Table 2. We employ pointwise regression using the Random Forest algorithm and train the model using 5-fold cross-validation (w.r.t. NDCG@20).

Feature extraction

Before starting with the Learning to Rank baseline, we first study the supervised ranking[23] which is the state-of-art document retrieval technique. Features may be categorized into three groups: (i) document, (ii) query, and (iii) query-document features. Analogously, we distinguish between three types of features: (i) table, (ii) query, and (iii) query- table features. In the original paper, there are 23 features extracted and used for training in the LTR baseline. We have managed to extract 14 of these features and added 2 new ones. In Table 2, we summarize all of the features we have extracted. We haven't extracted other features because due to time constraints and since we considered it more important to get to other things since we had already extracted many relevant and descriptive features of the original feature list.

Query Features		Value	Source
QLEN	length of query	{1,..,n}	[17, 13]
$IDF_{pageTitle}$	query IDF for page title	{0,..,∞}	[17, 14]
$IDF_{sectionTitle}$	query IDF for section title	{0,..,∞}	[17, 14]
$IDF_{tableCaption}$	query IDF for table caption	{0,..,∞}	[17, 14]
$IDF_{tableHeading}$	query IDF for table heading	{0,..,∞}	[17, 14]
$IDF_{tableBody}$	query IDF for table body	{0,..,∞}	[17, 14]
$IDF_{catchAll}$	query IDF for all text combined	{0,..,∞}	[17, 14]
Table Features			
#rows	number of rows in table	{1,..,n}	[17, 7, 10]
#cols	number of columns in table	{1,..,n}	[17, 7, 10]
#Nulls in paper	number of nulls in table	{0,..,n}	[17, 7, 10]
Query-table features			
#hitsLC	total hits of query terms in leftmost column	{0,..,n}	[17, 10]
#hitsSLC	total hits of query terms in second leftmost column	{0,..,n}	[17, 10]
#hitsB	total hits of query terms in whole body	{0,..,n}	[17, 10]
#hitsPT	total hits of query terms in page title	{0,..,n}	our idea
#hitsTC	total hits of query terms in table caption	{0,..,n}	our idea
yrank	rank from Wikipedia Media Search API	{1,..,n}	[17, 7]

Table 2. Baseline features used for table retrieval

Query Features.

We adopt two query features from document retrieval, namely, the number of terms in the query, and query IDF[35] 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). The original paper doesn't specify clearly how to compute the IDFs, but they refer to Tao, Liu, Xu and Li's paper[14]. By reading that paper over we found a formula for the Inverse Document Frequency for a query term. It is as follows:

$$idf(q_i) = log(\frac{|C| - df(q_i) + 0.5}{df(q_i) + 0.5})$$

Where document frequency $df(q_i)$ is the number of documents containing q_i in a stream, and $|C|$ s the total number of documents in the document collection. It is noteworthy to mention that the IDF is document-independent so all documents associated with the same query has the same IDF. Therefore this is categorized as a query feature. What this formula tells us is that we calculate the $idf(q_i)$ value for each term in a query and then sum it up for all the terms in the query to get the total value for the query.

Table Features.

Table features depend only on the table itself and aim to reflect the quality of the given table (irrespective of the query). We have extracted only the simple features such as the number of rows, columns and empty cells. The information we needed for some features that were used in the original paper are not available in the table corpus we are using for our feature extraction. Those features are the inLinks, Outlinks, pageViews and tablePageFraction. Additionally the features TableImportance and PMI have not been implemented due to time restrictions and complications in the process of getting them.

Query-Table Features.

Query-Table features express the degree of matching between the query and a given table. We have extracted 5 features that count the hits in specific parts of the table. They count the sum of occurrences for the terms in a query in the relevant table. Three of these hit features where presented in the original paper, they are hitsLC, that counts occurrences in the leftmost column in the table, hitsCLS that looks at the second leftmost column, hitsB searches the body. Then we extracted two extra that we thought would be interesting to add, the hits in the page title(hitsPT) and in the table caption(hitsTC).

Then we calculate the yrank which is the rank at which a table's parent page is retrieved by an external search engine. Like is done in the original paper we use the Wikipedia search API to obtain this ranking. Since the methods regarding that retrieval were not much discussed in the original paper, we retrieved the top 200 results for each of the 60 queries we are working with and checked if some of the retrieved results matched to the relevant table we have in our corpus.

Training

Training and testing data.

We use the same test collection in Section 3.1 to train this model. We find it unnecessary to change the quantity or reselect the queries as we would like to use the same data and hopefully to reach a similar conclusion to that in the original paper.

Methodology.

We apply the same methodology as in the paper. We employ pointwise regression using the Random Forest algorithm because its overall performance is the best among other techniques. We set the number of trees to 1000 and the maximum number of features in each tree to 3, and leave all remaining parameters as default values. We train the model using 5-fold cross-validation. Specifically, the approaches are implemented

by using the *sklearn* library. The details of the training process and *Python* scripts are available on our GitHub repository.

The output of the training is a set of scores of the relevance of the query table pairs. With a higher score, a table is more relevant to the query. We will then rank the tables and select top 5, 10, 15 and 20 rankings to perform evaluations in Section 4, Experiments.

Semantic Matching

The main idea of semantic matching 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. We reproduce the approach in three steps:

1. The "raw" content of a query/table is represented as a set of word-based terms.
2. The raw term is mapped to a semantic vector representation.
3. The semantic similarity (matching score) between a query-table pair is computed based on their semantic vector representations.

Content Extraction

We will take the raw content of queries and tables and represent it as a set of terms, where terms are words. We will denote these vectors as $\{q_1,...,q_n\}$ for query q and $\{t_1,...,t_n\}$ for table T.

More concretely, we will perform word-based extraction. It is a natural choice to simply use word tokens to represent query/table content. We let $\{q_1,...,q_n\}$ denote all unique words in the query q and $\{t_1,...,t_n\}$ denote all unique words from the title, caption and headings of the table T. That is we simply extract the words from tables/queries and insert them to the term lists respectively.

We are unable to perform the entity-based extraction because of its difficulty. The primary method to implement it is by entity retrieval, which refers to another paper. We contacted one of the authors and asked for instructions. He replied with detailed information. However, it's beyond our capability to implement entity-based representation within such short time.

Semantic Representations

Now we will embed the query/table terms that we have extracted before 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. This goes analogously for queries. In the original paper, two kinds of semantic spaces are tried out, bag-of-concepts and embeddings, each with two methods of implementation. Here we will solely focus on the Embeddings.

Embeddings

The mapping of discrete values to a vector of continuous numbers is often referred to as embedding. Recently, unsupervised representation learning methods have been proposed for obtaining embeddings that can predict context, i.e., Word Embeddings [15] and graph embeddings [6]. The word embeddings are made from pure word/term representations but the graph embeddings are made from raw entity-based representations.

Like is done in the original paper we map each query/table term to a word embedding. For that, we use the Word2vec library [15] and the Gensim python library [4]. We use a pre-trained data model that is provided by Google and is trained on Google News Data [3]. Then Word2vec is used to get a 300 dimension vector representation for each term in the extracted sets for each query/table.

Similarity Measures

The final step is to actually calculate the similarity between query-table pairs. That can be done in multiple ways and in the original paper both early fusion and Late fusion strategies where tried out. We only implement the early fusion similarity measure since the authors have introduced a specific way to measure similarity of word-based query-table pairs.

Early Fusion.

Here the main idea is to combine each query to a single vector and the same for each table. Now the content of table T is solely composed of the title, caption and headings of the table as is described in the content extraction section above. For the word embeddings these vectors are calculated by the formulas $\vec{C_T} = \sum_{i=1}^{m} \vec{t_i} \times TFIDF(t_i)$ for tables and $\vec{C_Q} = \sum_{i=1}^{m} \vec{q_i} \times TFIDF(q_i)$ for queries. In the original paper the TFIDF formula is not specified. Since the TFIDF is not always calculated in the exact same way we chose one that we thought was the most common. It is as follows:

$$TFIDF(t_i) = TF(t_i) * IDF(t_i) = \frac{f_{t_i k}}{\sum_{j=1}^{t} f_{ij}} * log(\frac{N}{n_{t_i}})$$

Where TF is the term frequency weight and IDF is the inverse document frequency of the term. Moreover $f_{t_i k}$ stands for the count of term t_i in table k, $\sum_{j=1}^{t} f_{ij}$ is the sum of all terms in the table, that is the table length. N is the number of all tables and n_{t_i} is the number of tables that contain the term t_i. It is computed the same way for the queries.

Combination.

In the original paper, the different similarity scores are combined in a learning-to-rank approach. Since we don't have more than one semantic similarity measure to combine we combine the early fusion we have with the LTR baseline discussed in Learning-to-Rank section. More details will be discussed in Section 4.

EXPERIMENTS

In this section, we discuss our experimental setup and present our results followed by further analysis.

Experimental Setup

We evaluate table retrieval performance in terms of Normalized Discounted Cumulative Gain (NDCG) at cut-off points 5, 10, 15, and 20. Specifically, we utilize trec_eval [1] to calculate these measurements. The ground truth file is fed into trec_eval along with the outcome rankings of each table retrieval methods. To be able to understand more of IR concepts, we add two more measurements: MAP and recall. We chose to

Method	NDCG@5	NDCG@10	NDCG@15	NDCG@20
Single-field document ranking	0.1860	0.2562	0.3065	0.3395
Single-field numbers removed	0.1782	0.2606	0.3029	0.3461
Multi-field	0.2949	0.3742	0.4323	0.4694
LTR baseline	0.6540	0.7941	0.8478	0.8765
STR	0.2317	0.3264	0.3901	0.4414
LTR & Semantic similarity Combined	**0.6547**	**0.7986**	**0.8542**	**0.8820**

Table 3. Table retrieval evaluation results measured by NDCG

Method	MAP@5	MAP@10	MAP@15	MAP@20
Single-field document ranking	0.1167	0.1653	0.1937	0.2178
Single-field numbers removed	0.1142	0.1682	0.1963	0.2246
Multi-field	0.1807	0.2418	0.2903	0.3260
LTR baseline	**0.5045**	0.7071	0.7901	0.8310
STR	0.1130	0.1995	0.2531	0.2987
LTR & Semantic similarity Combined	0.5039	**0.7107**	**0.7994**	**0.8376**

Table 4. Table retrieval evaluation results measure by MAP

Method	Recall@5	Recall@10	Recall@15	Recall@20
Single-field document ranking	0.1654	0.2670	0.3678	0.4257
Single-field numbers removed	0.1455	0.2749	0.3450	0.4373
Multi-field	0.1965	0.3164	0.4118	0.4808
LTR baseline	0.4997	0.7054	0.7910	0.8496
STR	0.1745	0.3181	0.4285	0.5206
LTR & Semantic similarity Combined	**0.5164**	**0.7232**	**0.8119**	**0.8501**

Table 5. Table retrieval evaluation results measured by Recall

look at those three measurements for different reasons. Recall measures a system's ability to retrieve all relevant documents and it is set based. The mean average precision (MAP) takes the order of the relevant documents into account, so that helps us to evaluate the different methods from another perspective. Then the NDCG or Normalized Discounter Cumulative Gain is the most advanced metric we use. It fits well to our setup since we have graded relevance (0, 1, 2). NDCG measures the usefulness, or gain, of a table based on its position in the result list. The gain is accumulated from the top of the result list to the bottom, with the gain of each result discounted at lower ranks [2].

Experimental Results

We put the experimental results in Table 3, 4 and 5. There we can see the results for all the 6 methods we implemented. The second-to-last row in the tables shows the result for the semantic table retrieval (STR) method. It is described in Section 3.5. In all measures, we can see that the Learning-to-Rank does very well. But when the semantic similarity from word embeddings is added as a feature to the LTR we see that it improves over the raw LTR baseline. So we can Answer our research question, can semantic matching improve retrieval performance, positively. Thereby we have gotten a result that agrees with the assumption that was made in the original paper [17] so we have more reason to believe that was correct. Even though the relative improvements have not been very high, or most often around 1%. But that is because we only had the one semantic similarity measure instead of the 4 in the original paper.

Analysis

Single-Field Document Ranking.

Let's start by looking at the single-field representation baseline and see what happened when we removed the numbers from the documents. It is interesting to see that all the measures, NDCG, MAP and Recall show worse performance for the 5 document cut-off point when the numbers have been removed. Then it improves in most other cases. Among all measures, removing numbers affects recall the most.

Multi-Field Document Ranking.

The multi-field representation baseline improves overall in all measures NDCG, MAP and Recall, but only slight improvement in all measures for the top 5 document cut-off. This could be due to lower sample size for the top document cut-off, therefore the evaluation would not show a large improvement. The experiment was able to prove that this baseline would improve over single-field representation due to the nature of a field based retrieval on a multi-field table representation.

LTR - feature importance.

In Figure 2 we show the importance of individual features for the table retrieval task using LTR, measured in terms of Gini importance. The semantic word based similarity (early-fusion) is by far the most important feature and it has the normalized value of 0.17, while the next best, number of rows, that basically just says how big the table is, has value of 0.096. *#Rows* is still the most important feature in the raw version of the LTR followed by *#yrank* and *#cols*. And we can see that all IDF related features contribute a large part to the table retrieval task.

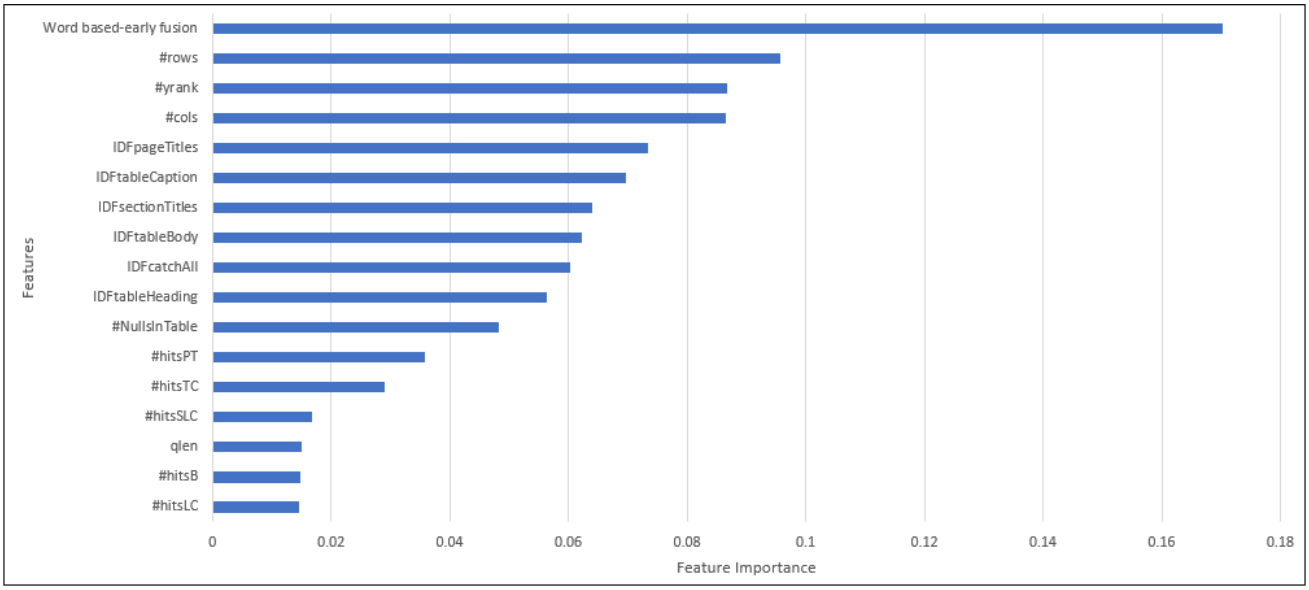


Figure 2. Normalized feature importance (measured in terms of Gini score)

Error Analysis

Here we are going to start by looking a few individual queries to see how the different retrieval methods affect them.

Lets start by seeing how different queries react to the single-field representation. I find it very interesting to analyze how the query "2008 Bejing olympics" since there is a number in the query itself. Interestingly there are more relevant tables retrieved when the numbers are excluded. The recall@20 is 0.222 when the numbers are included but 0.278 when the numbers are not part of the documents. Then there are queries like "prime ministers of england" that only has one relevant table in the corpus. It gets the score 1 for recall@20 when the numbers are included but falls down to 0 when the numbers are taken away.

Next we will be looking at the query "ipod models". According to the relevance judgment is has 2 highly relevant tables and 9 relevant tables in the corpus. The LTR managed to get all eleven of them in the first results with the highly relevant in the first to places. But when the semantic similarity had been included as well, the right eleven document where retrieved in the eleven first places in the right order.

The query "football clubs city" has very many relevant tables, or 29 highly relevant ones and 17 relevant. The combination of LTR and semantic word based similarity does not manage to do well for this query, i.e. it only manages to get MAP score of 0.4348 in the MAP@20, but that it purely because we are only looking at top 20 and 20/46 = 0.4348. So it is good to have in mind that cases like these can occur when we are looking at a specific cut-off point in the measures.

As one can notice from Table 3, 4 and 5, the STR performs much worse than the LTR and even worse than the Multi-field baseline in terms of all three measures. It performs just slightly better than the single-field baseline. We think one of the reasons lies in the limitation of the method of word embeddings. In Section 3.5.2 we used word2vec to map query/table word to a word embedding. For some of the words (terms), word2vec failed to map them and returned "not available", i.e., term "2008" in the second query "2008 beijing olympics". We then set the vector of the term to a zero vector which will then pull the centroid of the vector towards zero and leads to smaller cosine similarity of query/table vectors.

In figure 3 we randomly choose 20 queries out of the 60 in the query set. They are plotted the recall@10 value per query for both STR and LTR combined with STR. We see that the combination always acquires higher or even score in regards to STR. This tells us that The combination has a higher ability to retrieve all relevant documents.

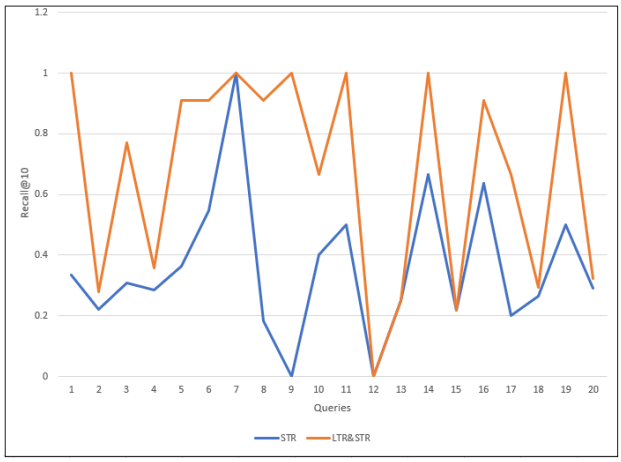


Figure 3. Recall@10 values for queries

Having discovered this, it is curious to see if there is a clear reason for the behaviour of queries 9, where there is very much difference on the performance of the two methods and 12 where both have the recall value of 0. Query number 9 is

"bittorrent clients" and 12 is "running shoes". The reason that 9 got such a low value in STR is that the word "bittorrent" was not known to the semantic vector model used to acquire the vector embeddings for the word. The reason for the zero value for 12 is simply because there was no table in the corpus that was judged as relevant for the given query by the ground truth document.

CONCLUSION

We have looked into the task of ad hoc table retrieval by reproducing part of the paper *"Ad Hoc Table Retrieval using Semantic Similarity" by Shuo Zhang and Krisztian Balog*[17]. By implementing a Learning-to-Rank baseline following the methodology that was introduced in the paper we saw that it has better results than the single-field and multi-field baselines they used from former literature. Then we agreed with their results that semantic matching can improve table retrieval. That was done by implementing one semantic similarity measure, word-based embeddings using early fusion similarity which combined with other table features achieved improvement over LTR and other baselines. We also found some inconsistency to what the original paper has concluded and analyzed reasons behind it.

Since this research is only built on Wikipedia pages the next appropriate step is to widen that out to other kinds of tables as well. We can also step away from the ranking of tables as well and focus on other types of data figures including graphs and charts based on queries in need of specific data.

REFERENCES

1. 2008. trec_eval 9.0alpha. `https://github.com/usnistgov/trec_eval`. (2008). Retrieved March 7, 2019.
2. 2014. machine learning medium. `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit`. (2014). Retrieved March 14, 2019.
3. 2016. GoogleNews-vector. `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit`. (2016). Retrieved March 9, 2019.
4. 2019. gensim 3.7.1. `https://pypi.org/project/gensim/`. (31 January 2019). Retrieved March 9, 2019.
5. Nitin Gupta Alon Halevy Hongrae Lee Fei Wu Reynold Xin Anish Das Sarma, Lu jun Fang and Cong Yu. 2012. Finding Related Tables. Proc. of SIGMOD 2012, 817–828.
6. Rami Al-Rfou Bryan Perozzi and Steven Skiena. 2014. DeepWalk: Online Learn- ing of Social Representations. Proc. of KDD '14, 701–710.
7. Thanapon Noraset Chandra Sekhar Bhagavatula and Doug Downey. 2013. Methods for Exploring and Mining Tables on Wikipedia. Proc. of IDEA 2013, 18–26.
8. Thanapon Noraset Chandra Sekhar Bhagavatula and Doug Downey. 2015. TabEL: Entity Linking in Web Tables. Proc. of ISWC '15, 425–441.
9. Alon Halevy Michael J. Cafarella and Nodira Khoussainova. 2009. Data Integration for the Relational Web. Proc. of VLDB Endow. 2, 1090–1101.
10. Daisy ZheWang EugeneWu Michael J. Cafarella, Alon Halevy and Yang Zhang. 2008. WebTables: Exploring the Power of Tables on the Web.
11. Jayant Madhavan Marius Pasca Warren Shen Fei Wu Genxin Miao Chung Wu Petris Venetis, Alon Halevy. 2011. Recovering Semantics of Tables on the Web. Proc. of VLDB Endow 4, 528–538.
12. Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering Table Queries on the Web Using Column Keywords. Proc. of VLDB Endow. 5 (2012), 908–919.
13. Kunak Agrawal Jennifer Paykin Stephen Tyree, Kilian Q Winberger. 2011. Parallell Boosted regression Trees for web search ranking. Proc. of WWW (2011), 308–396.
14. Jun Xu Tao Qin, Tie-Yan Liu and Hang Li. 2010. LETOR: A Benchmark Collection for Research on Learning to Rank for Information Retrieval. Inf. Retr. 13, 4, 1–30.
15. Kai Chen Greg Corrado and Jeffrey Dean Tomas Mikolov, Ilya Suts kever. 2013. Distributed Representations of Words and Phrases and Their Compositionality. Proc. of NIPS 2013, 3111–3119.
16. Meihui Zhang and Kaushik Chakrabarti. 2013. Semantic Matching and annotation of numeric and Time-variant attributes in web tables. Proc. of Sigmoid '13, 145–156.
17. Shuo Zhang and Krisztian Balog. 2018. Ad Hoc Table Retrieval using Semantic Similarity. International World Wide Web Conference, 1–10.