Accepted to IEEE Big Data 2019, Los Angeles, CA, 9-12 December. ©2019 IEEE

Improved Table Retrieval Using Multiple Context Embeddings for Attributes

Mohamed Trabelsi
Computer Science and Engineering
Lehigh University
Bethlehem, PA
mot218@lehigh.edu

Brian D. Davison
Computer Science and Engineering
Lehigh University
Bethlehem, PA
davison@cse.lehigh.edu

Jeff Heflin
Computer Science and Engineering
Lehigh University
Bethlehem, PA
heflin@cse.lehigh.edu

***Abstract*—Table retrieval is the task of extracting the most relevant tables to answer a user's query. Table retrieval is an important task because many domains have tables that contain useful information in a structured form. Given a user's query, the goal is to obtain a relevance ranking for query-table pairs, such that higher ranked tables should be more relevant to the query. In this paper, we present a context-aware table retrieval method that is based on a novel embedding for attribute tokens. We find that differentiated types of contexts are useful in building word embeddings. We also find that including a specialized representation of numerical cell values in our model improves table retrieval performance. We use the trained model to predict different contexts of every table. We show that the predicted contexts are useful in ranking tables against a query using a multi-field ranking approach. We evaluate our approach using public WikiTables data, and we demonstrate improvements in terms of NDCG over unsupervised baseline methods in the table retrieval task.**

***Index Terms*—Table retrieval, Table search, Word embedding, Neural embedding, Information Retrieval.**

I. Introduction

Vast amounts of information that are related to scientific, political, and cultural topics, are stored in tabular form. Many users have questions that can be resolved from this data, but these questions may go unanswered due to ignorance regarding the presence of the data, ignorance regarding where to look for the data, and inability to formulate queries using the domain-specific vocabulary of the data's owners. Our long-term goal is to develop a system that allows users to search across all tables on the Internet. This is a Big Data problem due to both volume (billions of tables) and variety (often, the schemas of tables are very different). The field of information retrieval is clearly relevant to this problem, but we argue that table retrieval presents both new challenges and opportunities. Researchers have focused on utilizing the knowledge contained in tables in multiple tasks including augmenting tables [1]–[4], extracting knowledge from tables [5], table retrieval or query answering [2], [6]–[8], and table type classification [9], [10].

Table retrieval is an important problem because tables contain valuable knowledge in various domains. A table retrieval algorithm can be used as a core component in other tasks such as table extension [11] and table mining [12]. In this paper, we address the table retrieval task by presenting a new context-aware ranking in order to extract a set of top-ranked tables for a given query.

Many methods in the literature treat tables as documents, so that document retrieval methods can be applied to table retrieval [2], [6]. The performance of table retrieval is improved by using supervised learning that includes features from tables, queries, and query-table pairs [1], [6]. In addition, semantic features are introduced to embed queries and tables into a semantic space, and then train a supervised model that captures semantic similarities in addition to traditional features [13]. However, a major drawback of prior approaches is that they are based on pretrained embeddings, and they ignore contextual information within tables in the ranking step. In order to overcome the limitations of prior methods, we propose learning a new model for word embeddings of attribute tokens that is used to predict contextual information of tables in the ranking phase. We show that the predicted contextual items can improve table retrieval performance using a collection of millions of tables from Wikipedia.

In summary, we make the following contributions:

- We build a new model for word embeddings of the tokens of table attributes using contextual information of every table. We examine multiple formulations for contexts used to create embeddings.
- We demonstrate the usefulness of an attribute's collection of values (the data tokens) in creating a meaningful semantic representation of the attribute. We show that an abstracted model of numerical values can lead to improvement on retrieval tasks without leading to a large increase in the number of tokens.
- We take an existing model for ranking document retrieval that considers partial semantic matches for tokens and adapt it for better table retrieval performance.
- We predict context of tables using the trained contextual model, and we use a mixed ranking model that incorporates the metadata of a table and the additional contexts in order to calculate the retrieval score.

II. Related work

A. Word embedding for tables

Words are embedded into low dimensional real-valued vectors based on the distributional hypothesis [14]. In many proposed models, the context is defined as the words that precede and follow a given target word in a fixed window [15]–[18]. Mikolov et al. [19] proposed the Skip-gram model which scales to a corpora with billions of words.

Recently, word embeddings are used to learn a real-valued vector representation for tables in different Natural Language Processing (NLP) tasks. Gentile et al. [20] proposed an embedding model to address the problem of entity linking within tables. They create sentences by concatenating attributes and/or values of the table, and then learn embeddings for the tokens of these sentences using the Skip-gram model. The learned embeddings are used to calculate the cosine similarity between two tables.

Nishida et al. [21] proposed a hybrid deep neural network architecture for table classification. The architecture is composed of a recurrent neural network (RNN) to encode a sequence of tokens of each cell. The next component in the architecture is the attention mechanism from Yang et al. [22] which extracts important tokens from each cell, and forms an input volume. The constructed volume is passed through a Convolutional Neural Network (CNN), and then a fully connected layer is added to compute the output layer. The input to the RNN is an embedding layer that represents each cell token as a real-valued vector. So, after finishing the training step, an embedding is obtained for every cell token.

Ghasemi-Gol and Szekely [23] introduced an unsupervised method to embed tables into a real-valued vector for the table classification task. Their proposed table embedding is based on cell tokens embedding using four contexts: text within each cell, text in adjacent cells, text in the corresponding attribute or header, and text surrounding the table in the web page. They used random indexing [24] to calculate the embeddings of cell tokens.

B. N-gram Language Models

A general model for information retrieval is the N-gram Language Model (LM), which assigns probabilities to a sequence of words. Given a query, a unigram LM is used to retrieve documents by calculating the probability that each document would generate the query [25]–[27]. For a query Q, that is composed of query terms $q_1, q_2, \ldots, q_{|Q|}$, where $|Q|$ is the length of Q, the probability of Q given a document D is given by:

$$p(Q|\theta_d) = \prod_{i=1}^{|Q|} p(q_i|\theta_d)$$

where θ_d is the LM estimated for document D and $p(q_i|\theta_d)$ is estimated using a unigram LM. In order to avoid assigning zero probability for unseen words in a document, $p(q_i|\theta_d)$ can be estimated using a linear combination of probabilities from unigram LM and Dirichlet prior smoothing [27], [28].

Linear interpolation is used to combine multiple language models in order to estimate a new language model [25], [29]. In this case, $p(w|\theta_d)$ is given by:

$$p(w|\theta_d) = \sum_{i=1}^{L} \beta_i \; p(w|\theta_{d_i}) \qquad (1)$$

where L is the number of language models, θ_{d_i} is the i^{th} representation of document D, and β_i is the weight associated with language model θ_{d_i}. The weights β_i are constrained to sum to 1, so that:

$$\sum_{i=1}^{L} \beta_i = 1, \text{ and } \beta_i \geq 0 \text{ for all } 1 \leq i \leq L$$

C. Ad hoc table retrieval

In a table retrieval task, a table can be considered as a document, and traditional document retrieval methods can be applied for table ranking. Cafarella et al. [2], [6] retrieve relevant documents using web search engines, then tables are extracted from the highest-ranking retrieved documents. The simplest approach is to represent a table by a single field containing all the text associated with the table. The retrieval score is then calculated using existing retrieval methods, such as language models or BM25 [30]. A table consists of multiple fields, and retrieval models for multi-field documents can be applied for table retrieval. Pimplikar and Sarawagi [8] proposed a late fusion [31] method for multi-field ranking. In this case, for a given query, a score is calculated independently for every field, then a linear combination of the scores is taken. The final score is given by:

$$score(Q, T) = \sum_{i} w_i \times score(Q, f_i) \qquad (2)$$

where Q is a given query, T is a table, f_i is the i^{th} field of T, and w_i is the weight associated with f_i.

For supervised ranking of documents, features are grouped into three categories: document, query, and query-document features [32]. In the case of table retrieval, multiple query, table, and query-table features are proposed in the literature [1], [6]. Zhang and Balog [13] proposed semantic matching between queries and tables using various semantic spaces which are: Word embeddings, Graph embeddings, Bag-of-entities and Bag-of-categories. The DBpedia knowledge base is used to construct a vector of zeros and ones for both bag-of-entities and bag-of-categories. The dimension of bag-of-entities is equal to the total number of entities in the knowledge base, where a value of 1 indicates that the entity is mentioned in the table. It is the same for the bag-of-categories with a dimension equal to the total number of Wikipedia categories.

The organization of the rest of the paper is as follows. Section III proposes a table retrieval method based on predicting contexts of tables using new trained word embeddings; Section IV illustrates data and query collections used in our approach, and compares baselines and our algorithm in the table retrieval task using WikiTables data.

III. Context Prediction for Table Ranking

In this section, we introduce our proposed method to enhance ranking of tables given a user's query. Given a table, we tokenize each attribute to obtain a set of attribute tokens. The proposed approach has three stages: in the first step, we build word embeddings for attribute tokens using contextual information from each table. In the second step, for a given attribute token of a table, we predict its context using the trained contextual model and augment the table with this additional, implicit, and descriptive information. In the third step, we calculate a score for a given query-table pair in order to rank and retrieve tables that are related to the query. We use a mixed ranking model that incorporates the metadata of a table and the additional predicted context in order to calculate our score.

Our work differs from the table embeddings discussed in Section II-A. Unlike Gentile et al. [20], our contexts do not depend on the arbitrary ordering of rows or columns in the dataset, and we learn a word embedding for attribute tokens by enlarging the context to cover metadata of tables. The additional contexts are useful in table retrieval as more predicted contexts are available when scoring a table against a query using multi-field ranking approaches. In Nishida et al. [21], a word embedding of every cell token is obtained after supervised training of a hybrid model. In our case, we use a different architecture with unsupervised training when learning embeddings. The frequency of tokens in tables exhibits a Zipfian distribution [33], with a long tail of tokens that only appear in one or a few tables. If the collection is divided into training and test sets, many tokens will only appear in one set or the other. Such a division will lead to many tokens that only appear in the test set, and therefore do not have embeddings. Similar to Ghasemi-Gol and Szekely [23], we identify multiple contexts from the table. However, a key difference is that our model distinguishes between the different contexts, rather than treating them uniformly, and we show that this leads to more accurate retrieval in Section IV. Also, we have different notions of context: we do not simply treat the four cells adjacent to a cell as context, and the only locality information that is used is that all cells in the header row are considered context for each other.

A. *Word embeddings for attribute tokens*

The objective is to learn an embedding for every attribute token that captures its contextual information inside a table. We use an adapted Skip-gram model [17] for tables. The training objective is to learn a real-valued representation for attribute tokens using contextual information from each table. Then, the trained model is used to predict additional tokens that are relevant to the table.

Our model differs from the original Skip-gram model in two major aspects: training context and vocabulary. In the original Skip-gram model, the context is based on the surrounding words of a given word. However, in a table, how to define context is not so obvious. Clearly, other tokens in the name of an attribute count as context, but are there other meaningful contexts? The metadata of a table clearly provides a larger context for the attributes, and an attribute is also contextualized by the other attributes that appear in the same table. We also argue that the cell values provide meaningful contextual information. For example, an attribute that consisted of 50 two-character values including 'AZ', 'KY', 'MS', 'WA' , etc. should be assigned an embedding that is similar to another attribute with the same set of values, even if the attributes shared no name tokens. Thus, the context for an attribute token is rich, but we argue that these different types of context should not all be treated uniformly. Levy and Goldberg [34] have shown that, for word embeddings, differentiating contexts can lead to embeddings that better express similarity as opposed to relatedness. Inspired by this work, we use a multi-context model, so that a token that appears in the metadata has a different impact than the same token when it appears in a cell value. Thus, for a given attribute's token, we have four different types of contexts: description, values, other tokens from same attribute, and other tokens from attributes in the same table. For simplicity, we append a distinct suffix to every context token in order to distinguish between the different contexts in the training data.

Our model uses different input and output dictionaries: the input dictionary contains tokens of attributes extracted from the collection of tables, and the output dictionary is composed of tokens from all four types of contexts. More formally, given a sequence of T training attribute tokens $a_1, a_2, a_3, \dots, a_T$, the objective of our model is to maximize the sum of log probabilities using our proposed contextual information:

$$\sum_{t=1}^{T} \Big(\sum_{w_d \in D_t} log\ p(w_d|a_t) + \sum_{w_v \in V_t} log\ p(w_v|a_t) \\ + \sum_{w_{oa} \in OA_t} log\ p(w_{oa}|a_t) + \sum_{w_{sa} \in SA_t} log\ p(w_{sa}|a_t) \Big)$$

where a_t is a given attribute's token, D_t is the set of description context tokens of a_t, V_t is the set of values context of a_t, OA_t is the set of other tokens from attributes in the same table containing a_t, and SA_t is the set of other tokens from same attribute containing a_t.

Given an output context w_c and input token a_t, the Skip-gram model defines $p(w_c|a_t)$ using the softmax function. Computing the softmax probability is expensive because it requires summing over all the words in the output dictionary. To address this problem, we estimate the softmax probability using Noise Contrastive Estimation (NCE) [35], [36]. NCE reduces the language model estimation problem to a binary logistic regression classifier that distinguishes between data and noise.

B. *Managing Numerical Cell Values*

Numerical values play a more prominent role in tables than in text documents. For example, 26.9% of cells in the WikiTables dataset are numeric values, while in a random sample of 372 tables from data.gov [33], 58.2% of the cells are numeric. An attribute's numeric values can contribute to its interpretation and thus provide useful context information: four digit numbers beginning with 19 or 20 are likely to be years, while in the United States five-digit numbers are often zip codes and sequences of the form *ddd-ddd-dddd* are often phone numbers. However, the number of unique numeric values in a dataset can be far larger than the number of unique non-numeric tokens, and adding them to the vocabulary will lead to an increase in the size of the output dictionary. For the WikiTables dataset, we obtain 2,401,425 unique tokens (including numbers) in the output vocabulary. Furthermore, context should ideally recognize that small numeric distances reflect similar contexts, and context should not be adversely impacted by noise or rounding errors. That is, the numbers 998 and 1002 are quite close, as are 3.14 and 3.14159. Likewise, two attributes with a range of values from 1960 to 2020 should be considered to have very similar values context even if less than half of the values appear in both attributes.

A typical approach to handling numbers in word embeddings is to replace each digit with a special character, such as the hash symbol ('#'). For simplicity, we tokenize using the same punctuation that we use to tokenize the rest of the text. Thus, 999-99-9999 becomes the tokens '999', '99' and '9999', and then '###','##' and '####'. This means that regardless of the number of distinct numerical values, we only add a number of tokens to the output dictionary that is less or equal to the length of the longest successive sequence of digits. As a partial solution to the problem of recognizing numbers that are close, while keeping the cardinality of numeric tokens low, we propose a new approach: keep the leading digit of a number, and only replace other digits by '#' in order to refine numerical values context. With this approach, the size of output vocabulary decreases to 2,032,424, a reduction of about 15%. However, in other datasets, such as those in data.gov, numbers are more prevalent, and the savings will be more significant.

C. *Table features*

We describe a table using three sets of features:

- original description (D_o): Set of tokens extracted from the table metadata such as its title and/or caption,
- original attributes (A_o): Set of tokens extracted from the header row of the table, and
- original values (V_o): Set of tokens extracted from the data rows of the table (i.e., all rows other than the header).

We augment these features with additional features produced by our trained model. First, for an attribute token a in the input dictionary D_I, we predict the different contexts by extracting the top k words from the output dictionary that have the highest probabilities. We denote our set of top k contexts by C_k. We divide the predicted contexts into three categories: description context D_c, values context V_c, and attributes context A_c. More formally,

$$C_k = D_c \cup V_c \cup A_c$$

Second, the hidden layer $H \in \mathbb{R}^{W_i \times h}$ of our model, where W_i is the number of attributes tokens, presents a new word embedding of dimension h for each attribute token. We use the new word embedding to extract the set of top m closest tokens, denoted by I_m, to a given attribute token a, using cosine similarity.

The final predicted context P_c is given by:

$$P_c = C_k \cup I_m = D_c \cup V_c \cup A_c \cup I_m$$

D. *Ranking methods for table retrieval*

The predicted context P_c enables us to augment every table with additional fields that capture the contextual information obtained from the original attribute tokens. The additional contexts can be combined with original fields, such as title, data, original attributes, etc., in order to improve multi-field ranking, and thus table retrieval. We propose using the multiple ranking mechanisms based on Equation (2). We use traditional ranking methods such as BM25 and TF-IDF. The third ranking method, LM-Ranking, is based on combining language models [29]. In other words, we index tables using the contents of the original fields and additional predicted contexts, then we estimate a language model for every field, and we combine the estimated language models using Equation (1).

The fourth ranking method, Late-avg, is a late fusion similarity model [13] based on Equation (2) for multi-field ranking, but with scores calculated by averaging the cosine similarity between the embeddings of all pairs of query terms and terms of field f_i. Given an embedding $E(\cdot)$ for a term t:

$$score_{la}(Q, f_i) = \frac{1}{|Q| \times m_i} \sum_{k=1}^{|Q|} \sum_{j=1}^{m_i} cosine(E(q_k), E(t_{ji})) \tag{3}$$

where m_i is the length of field f_i, q_k is the k^{th} query term of Q, and t_{ji} is the j^{th} token in f_i.

Kenter and de Rijke [37] proposed a Semantic Text Similarity (STS)-based ranking method specifically intended for short texts. This combines a traditional BM25 formula with semantic similarity computed from word embeddings. Because the semantic similarity is computed on query token/field token pairs, this has aspects of a late-fusion approach. In STS, the query is assumed to be the short text.

As a simplification of the above approach, we consider a pure late-fusion approach that only considers the semantic similarity, and ignores the BM25 aspects. In this, we select the best matching query term for each table field term and refer to it as MaxQuery:

$$score_{mq}(Q, f_i) = \sum_{j=1}^{m_i} \max_{k \in [1, \ldots, |Q|]} cosine(E(q_k), E(t_{ji}))$$

Finally, we argue that Kenter and de Rijke's assumptions about short text similarity do not apply to the table retrieval problem. In particular, rather than choosing the best query token for each field token, we choose the best field token for each query token. Typically, the set of table tokens will be much larger than the set of query tokens. A table could be a good match for a query even if only a portion of the table is relevant. Our final model, MaxTable is a late fusion similarity model, but we find the closest table term to each query term using cosine similarity, and then sum over these similarities:

$$score_{mt}(Q, f_i) = \sum_{k=1}^{|Q|} \max_{j \in [1,\ldots,m_i]} cosine(E(q_k), E(t_{ji})) \quad (4)$$

In the above ranking methods, the choice of embedding $E(\cdot)$ depends on which field is being compared to the query. Since we only apply our embedding approach to attribute tokens, other fields will contain tokens for which we do not produce embeddings. For these fields, we use pre-trained fastText [38] embeddings, which are built from character-level n-grams, allowing embeddings to be created even for terms that have not been seen before. Specifically, we use our embeddings on the original attribute field A_o, the predicted context attribute field A_c, and the closest tokens I_m. All other fields (D_o, D_c, V_o, and V_c) use fastText embeddings for computing similarity. Recall that the predicted description contexts and value contexts were produced by our embedding model, so our approach still contributes to the scores even when fastText embeddings are used for determining cosine similarity.

IV. Evaluation

A. Data and query collections

Our dataset is composed of the WikiTables corpus [39] containing over $1.6M$ tables. Each table has five indexable fields: table caption, attributes (column headings), data rows, page title, and section title. In addition, each table contains statistics which are: number of columns, number of rows, and set of numerical columns of the table. We use the same test queries that were used by Zhang and Balog [13].

We use Zhang and Balog's [13] ground truth of query-table relevance, where every query-table pair is evaluated and given one of three scores: 0 means "irrelevant", 1 means "partially relevant" and 2 means "relevant". The objective of the annotators was to use the retrieved tables to create a new table that fulfills the query. So, for a given query, they needed to find tables that are useful in forming a single table that matches the query. By using this task to evaluate a given table's relevance, if a table is not used to create the final table, it is given a relevance 0. If only some values are used from the table, it is partially relevant. Finally, if blocks of a table are used, it is considered relevant. The total number of query-table pairs is 3117. As in Zhang and Balog [13], in all reported results of our experiments, 1918 pairs are used for parameter tuning of multi-field ranking weights, and 1199 pairs are used as the test collection.

B. Baselines

We compare the performance of our proposed model with two baselines:

Single-field document ranking: A table is considered as a single field document by concatenating a subset of WikiTables fields: table caption, attributes, data rows, page title and section title (as in [2], [6]). In the result tables, we identify which fields were used. Then a language model is estimated for the formed document in order to rank a given table against the query. We can also calculate the score of a query-table pair using word embedding-based ranking methods from Section III-D.

Multi-field document ranking: In a multi-field ranking scenario, a table is defined using five fields: page title, section title, table caption, attributes and table body or values. We compare our method against two baselines in the category of multi-field document ranking using word embedding. The first multi-field ranking method, MultiField-P, is based on the pretrained fastText word embedding as in Zhang and Balog [13] when calculating cosine similarity. In the second multi-field ranking method, MultiField-G, we use a word embedding trained by the adapted Skip-gram model to the context introduced by Ghasemi-Gol and Szekely [23]. Field weights are optimized using grid search.

We conduct an ablation study that evaluates five variations of our model. The first variation is called SCON (single context), in which we treat all contexts as one context. In other words, we do not append a distinct suffix to every context token to distinguish between different contexts. The other four variations are based on different formulations of the values context V_t for a given attribute a_t. In the NOVAL variation, we ignore the values context by setting V_t=$\emptyset$. In the NONUM variation, V_t includes only string contexts. In the HASHNUM variation, we replace each digit with the hash symbol ('#'), but otherwise use all value tokens and multiple contexts. Finally, the MCON variation is our full model, where we keep the leading digit for numbers and replace other digits by '#'. In HASHNUM and MCON, V_t includes both numerical and string value contexts. In the experimental results section, we report results from all five variations.

C. Experimental Setup

We use the full set of $1,652,771$ tables to train our embedding model. For the description context D_t of a given attribute token a_t, we concatenate three fields from WikiTables: page title, section title and caption.

We set the dimension of word embeddings h to 100, and the number of labels used in NCE estimation to $10,000$. We train our model for 3 epochs with a batch size of 100. We use SGD to minimize the loss function, and update the weights of our model. We set the learning rate to 0.01. The model is implemented using TensorFlow, with Tesla T4 GPU (memory Clock Rate: 1.59 GHz). For context prediction, we set the size of C_k, k, and the size of I_m, m, to 20. In order to calculate the retrieval score for query-table pairs by combining language models, we use the implementation in Hasibi et al. [40] which is based on Elasticsearch.

D. Experimental results

1) Model statistics: We start by comparing our models in term of sizes of input dictionary, output dictionary, and training data (number of target-context pairs). The statistics are shown in Table II. For all five variations, we have the same size of input vocabulary. For output vocabulary, MCON has the largest vocabulary, as it includes multiple contexts

TABLE I: Table retrieval evaluation results using MaxTable

Method	Fields	NDCG@5	NDCG@10	NDCG@15	NDCG@20
Single-field document ranking	all	0.4715	0.4832	0.5155	0.5404
Single-field document ranking	cell values	0.3292	0.3775	0.4245	0.4657
Single-field document ranking	description	0.4632	0.4912	0.5330	0.5462
Single-field document ranking	attributes	0.3204	0.3545	0.4137	0.4584
MultiField-P	all	0.4794	0.4930	0.5298	0.5473
MultiField-G	all	0.4610	0.4818	0.5051	0.5386
SCON	all	0.4824	0.5022	0.5343	0.5494
NOVAL	all	0.4813	0.5021	0.5323	0.5491
NONUM	all	0.4862	0.5037	0.5368	0.5505
HASHNUM	all	0.4902	0.5043	0.5367	0.5505
MCON	all	**0.5088**	**0.5117**	**0.5460**	**0.5587**

and numerical context. The output dictionary size does not influence the training time because we use NCE to estimate the softmax probabilities. The number of parameters for MCON is significantly larger than NOVAL because of the difference in the size of output vocabulary. So, MCON allocates more memory than NOVAL to update model parameters. The training time is directly related to the number of target-context pairs. In our experiments using a single Tesla T4 GPU, the average training time for 4000 steps each with a batch size of 100 is 34.67 seconds. Our models are trained for three epochs. So, for MCON, we need 14.22 hours for one epoch training and the total training time is 42.66 hours. Because training depends directly on the number of target-context pairs, NOVAL is significantly less, at 13.39 hours.

TABLE II: Statistics of our models

Model	Input dict	Output dict	Target-context pairs
SCON	118,421	1,608,455	590,661,355
NOVAL	118,421	415,272	185,390,306
NONUM	118,421	1,997,633	432,705,346
HASHNUM	118,421	2,025,698	590,661,355
MCON	118,421	2,032,424	590,661,355

2) Ranking results: We evaluate the performance of our proposed method and baselines on the table retrieval task using Normalized Discounted Cumulative Gain (NDCG) [41] at cut-off thresholds 5, 10, 15, and 20. All NDCG results are reported using the TREC evaluation software, trec_eval.[1] We index tables from the WikiTables dataset using the original fields and predicted contexts. We report the retrieval results using our proposed models and multiple baselines.

In Table I, we show the NDCG results of table retrieval using the MaxTable ranking method (results for other ranking methods showed similar trends). We show that MultiField-P leads to better performance than single-field document ranking. From the results of single-field document ranking using only cell values, we observe that the cell value-based single-field document ranking is not effective in ranking query-table pairs. The additional predicted contexts using our MCON model has the best performance for all NDCG metrics.

In Table III, we summarize the results of all ranking approaches from Section III-D using our best model, MCON. Here, the MaxTable ranking method is shown to be the most effective ranking for our embedding and context prediction approach. Aggregating scores using the sum of maximum similarities is more effective than using average similarity. Late-avg is stricter as it requires a query term to have large similarity with multiple tokens of a given field in order to obtain high score. On the other hand, for MaxTable, a high similarity between a query term and one token from a given field is enough to obtain high similarity score for a given field in a table. Among the ranking methods that are not based on word embedding (LM-Ranking, BM25, and TF-IDF), LM-Ranking achieves higher performance at all NDCG cut-off thresholds. Note, that for our table features, STS actually performs worse than plain BM25, while simply using its approach to semantic similarity gets closer to BM25.

[1]https://trec.nist.gov/trec˙eval/ trec_eval.8.1.tar.gz

TABLE III: MCON table retrieval results

Method	NDCG@5	NDCG@10	NDCG@15	NDCG@20
BM25	0.4545	0.4854	0.5186	0.5449
TF-IDF	0.4316	0.4746	0.5073	0.5344
LM-Ranking	0.4755	0.4976	0.5316	0.5548
Late-avg	0.4740	0.5025	0.5241	0.5464
STS	0.4323	0.4502	0.4863	0.5158
MaxQuery	0.4642	0.4726	0.5087	0.5310
MaxTable	**0.5088**	**0.5117**	**0.5460**	**0.5587**

V. Conclusions

We have shown that using multiple, differentiated contexts can result in more useful attribute embeddings. When the MaxTable ranking method is used for the table retrieval task, our MCON system has up to 5.47% improvement in NDCG@5 over a method that uses the same context fields but treats them as the same context. Likewise, we have shown that the data values of an attribute provide useful context information: our full system always performs better than the version that does not include values as context by as much as 5.71% in NDCG@5. Finally, we have shown that our simple treatment of numeric values also leads to better embeddings: when numeric values are dropped, NDCG@5 scores drop by up to 4.44%.

This work leaves many avenues for further investigation. Our evaluation method focused on the indirect application of attribute embeddings to the table retrieval problem. So, it is possible to examine the quality of embedding in the more direct task of determining attribute similarity. However, doing so will require the creation of a ground truth dataset. Finally, some researchers have focused on using learning-to-rank methods for document and table retrieval [13], [42]–[44], and it would be interesting to see how such methods would benefit from our embedding approach.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1816325.

References

[1] C. S. Bhagavatula, T. Noraset, and D. Downey, "Methods for exploring and mining tables on wikipedia," in *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics, IDEA '13*. ACM, 2013, pp. 18–26.

[2] M. J. Cafarella, A. Halevy, and N. Khoussainova, "Data integration for the relational web," *Proc. VLDB Endow.*, vol. 2, no. 1, pp. 1090–1101, Aug. 2009.

[3] A. Das Sarma, L. Fang, N. Gupta, A. Halevy, H. Lee, F. Wu, R. Xin, and C. Yu, "Finding related tables," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012, pp. 817–828.

[4] O. Lehmberg, D. Ritze, P. Ristoski, R. Meusel, H. Paulheim, and C. Bizer, "The mannheim search join engine," *Web Semant.*, vol. 35, no. P3, pp. 159–166, 2015.

[5] E. Muñoz, A. Hogan, and A. Mileo, "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*, 2014, pp. 533–542.

[6] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: Exploring the power of tables on the web," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 538–549, Aug. 2008.

[7] T. Nguyen, Q. Viet Hung Nguyen, M. Weidlich, and K. Aberer, "Result selection and summarization for web table search," *Proceedings - International Conference on Data Engineering*, vol. 2015, pp. 231–242, 05 2015.

[8] R. Pimplikar and S. Sarawagi, "Answering table queries on the web using column keywords," *Proc. VLDB Endow.*, vol. 5, no. 10, pp. 908–919, Jun. 2012.

[9] E. Crestan and P. Pantel, "A fine-grained taxonomy of tables on the web," in *CIKM*, 2010.

[10] P. Pantel and E. Crestan, "Web-scale table census and classification," in *WSDM*, 2011.

[11] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: entity augmentation and attribute discovery by holistic matching with web tables," in *SIGMOD Conference*, 2012.

[12] Y. A. Sekhavat, F. D. Paolo, D. Barbosa, and P. Merialdo, "Knowledge base augmentation using tabular data," in *LDOW*, 2014.

[13] S. Zhang and K. Balog, "Ad hoc table retrieval using semantic similarity," in *WWW*, 2018.

[14] Z. S. Harris, "Distributional structure," *WORD*, vol. 10, no. 2-3, pp. 146–162, 1954.

[15] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003.

[16] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proceedings of the 24th International Conference on Machine Learning, ICML '07*. ACM, 2007, pp. 641–648.

[17] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[18] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[19] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, 2013, pp. 3111–3119.

[20] A. L. Gentile, P. Ristoski, S. Eckel, D. Ritze, and H. Paulheim, "Entity matching on web tables: a table embeddings approach for blocking," in *EDBT*, 2017.

[21] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo, "Understanding the semantic structures of tables with a hybrid deep neural network architecture," in *AAAI*, 2017.

[22] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, "Hierarchical attention networks for document classification," in *HLT-NAACL*, 2016.

[23] M. Ghasemi-Gol and P. A. Szekely, "Tabvec: Table vectors for classification of web tables," *CoRR*, vol. abs/1802.06290, 2018.

[24] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive Computation*, vol. 1, pp. 139–159, 2009.

[25] K. Collins-thompson, P. Ogilvie, Y. Zhang, and J. Callan, "Information filtering, novelty detection, and named-page finding," in *Proceedings of the 11th Text Retrieval Conference*, 2002.

[26] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '98*. ACM, 1998, pp. 275–281.

[27] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to ad hoc information retrieval," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '01*. ACM, 2001, pp. 334–342.

[28] C. Zhai, C. Zhai, and J. Lafferty, "Two-stage language models for information retrieval," in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02*. ACM, 2002, pp. 49–56.

[29] P. Ogilvie, J. Callan, and J. Callan, "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*. ACM, 2003, pp. 143–150.

[30] S. E. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford, "Okapi at trec-3," in *TREC*, 1994.

[31] S. Zhang and K. Balog, "Design patterns for fusion-based object retrieval," in *Advances in Information Retrieval - 39th European Conference on IR Research (ECIR)*, Apr. 2017, pp. 684–690.

[32] T. Qin, T. M. Liu, J. Xu, and H. Li, "Letor: A benchmark collection for research on learning to rank for information retrieval," *Information Retrieval*, vol. 13, pp. 346–374, 2009.

[33] Z. Chen, H. Jia, J. Heflin, and B. D. Davison, "Generating schema labels through dataset content analysis," in *Companion Proceedings of the The Web Conference 2018, WWW '18*, 2018, pp. 1515–1522.

[34] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *52nd ACL*, 2014, pp. 302–308.

[35] M. U. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 307–361, Feb. 2012.

[36] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *ICML*, 2012.

[37] T. Kenter and M. de Rijke, "Short text similarity with word embeddings," in *CIKM*, 2015.

[38] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[39] C. Bhagavatula, T. Noraset, and D. Downey, "Tabel: Entity linking in web tables," in *International Semantic Web Conference*, 2015.

[40] F. Hasibi, K. Balog, D. Garigliotti, and S. Zhang, "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*. ACM, 2017, pp. 1289–1292.

[41] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.

[42] B. Wang and D. Klabjan, "An attention-based deep net for learning to rank," *ArXiv*, vol. abs/1702.06106, 2017.

[43] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, "End-to-end neural ad-hoc ranking with kernel pooling," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '17*. ACM, 2017, pp. 55–64.

[44] B. Mitra, F. Díaz, and N. Craswell, "Learning to match using local and distributed representations of text for web search," in *WWW*, 2016.