Accepted to IEEE International Conference on Big Data (IEEE BigData), December 2020.

A Hybrid Deep Model for Learning to Rank Data Tables

Mohamed Trabelsi, Zhiyu Chen, Brian D. Davison, Jeff Heflin
Computer Science and Engineering
Lehigh University, Bethlehem, PA, USA
{mot218,zhc415,davison,heflin}@cse.lehigh.edu

***Abstract*—We address the problem of ad hoc table retrieval via a new neural architecture that incorporates both semantic and relevance matching. Understanding the connection between the structured form of a table and query tokens is an important yet neglected problem in information retrieval. We use a learning-to-rank approach to train a system to capture semantic and relevance signals within interactions between the structured form of candidate tables and query tokens. Convolutional filters that extract contextual features from query/table interactions are combined with a feature vector based on the distributions of term similarity between queries and tables. We propose using row and column summaries to incorporate table content into our new neural model. We evaluate our approach using two datasets, and we demonstrate substantial improvements in terms of retrieval metrics over state-of-the-art methods in table retrieval and document retrieval, and neural architectures from sentence, document, and table type classification adapted to the table retrieval task. Our ablation study supports the importance of both semantic and relevance matching in the table retrieval.**

***Index Terms*—Table retrieval, Table search, Neural networks, Learning to rank.**

I. Introduction

In this era of Big Data, datasets are publicly available for users to explore a vast amount of information across a variety of topics. Among all types of publicly available datasets, the data table represents the most prevalent form of data. 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 creators. Ad hoc table retrieval is a problem that has many similarities to document retrieval, but one that also presents both new challenges and opportunities. Researchers have focused on utilizing the knowledge contained in tables in multiple tasks including augmenting tables [1], [2], extracting knowledge from tables [3], table retrieval [2], [4]–[7], and semantic labeling [8], [9]. A table retrieval algorithm can be used as a core component in other tasks such as table extension [10] and table mining [11]. In this paper, we address the table retrieval task by presenting a new deep semantic and relevance matching model in order to extract a set of top-ranked tables for a given query.

Supervised learning, based on features from tables, queries, and query-table pairs [1], [4], has resulted in the best performing table retrieval systems. Building on this, Zhang and Balog [12] introduced semantic features to embed queries and tables into a semantic space, and then train a supervised model using the semantic and traditional features. However, there are major drawbacks of prior approaches for ad hoc table retrieval. First, they are based on hand-crafted features, and that limits the ability to capture multiple levels of similarity between query and table. Second, they ignore query relevance matching. Third, they assume equal contribution of each query token to the final relevance score when ranking web tables against a given query.

Several methods in the literature for document retrieval focus on modeling contextual information in search queries and documents. The contextual information captured by classical models such as TF-IDF and BM25 are usually coarse-grained and rely on matching the surface forms of query terms. To address this, deep learning-based methods have been proposed to achieve semantic matching [13]–[17].

In order to overcome the limitations of prior methods in table retrieval, we propose a new model that combines deep contextual features with features based on term similarity distributions. Our algorithm learns convolutional filters that extract contextual features from query/table interactions (semantic matching). This is combined with a feature vector based on the distributions of term similarity between queries and tables (relevance matching). Additionally, we incorporate table values into our model using row and column summaries. Finally, we learn the contribution of each query token to the final relevance score. These models are trained using a learning-to-rank approach with a listwise loss function. We show that our new method can improve table retrieval performance using a collection of tables from Wikipedia [12] and Web tables from a Microsoft dataset [18].

In summary, we make the following contributions:

- We propose a new semantic similarity model that is able to capture multiple levels of semantic signals between query and table. In order to capture contextual information, we apply various-sized convolutional filters to an interaction matrix built from the embeddings of query and table tokens, and then apply a second layer of convolutional filters to extract higher level features. Our representation of the table includes summary vectors about the contents of the table, both in terms of values in each column and values in selected rows.
- We demonstrate the usefulness of query relevance-specific components for the table retrieval task. Using

kernel pooling, we learn a feature vector based on the probability distribution of the similarity of each document token to each query token, and we learn the contribution of each token to the final relevance score using a Term Gating Network. Each of these components lead to improvement on retrieval tasks without leading to a large increase in the number of parameters of the model.
- We compare our proposed method not only against methods from table retrieval and document retrieval, but we also adapt architectures from multiple domains to the table retrieval task. We show that ad hoc table retrieval benefits from table-specific architectures; that is, straightforward application of leading document retrieval approaches results in reduced performance.

II. Background

A. Learning to rank for document retrieval

Learning to rank (LTR) for document retrieval relies on a training data that is composed of query-document relevance pairs to train a model to predict rankings [19]. LTR models represent a rankable query-document pair as a feature vector $F(q, d)$, where q is a query and d is a document. In traditional machine learning models, the function F represents hand-crafted features. A ranking model M is trained to map the feature vector $F(q, d)$ to a real-valued score such that more relevant documents to a given query are scored higher in order to maximize a rank-based metric. Liu [19] categorized the LTR methods into three categories based on their training objectives. The pointwise category associates each query-document pair with a relevance score, and the objective is to predict the exact relevance score using classification or regression models. The pairwise category does not try to accurately predict the exact relevance score of a query-document pair; instead, it focuses on the relative order between two documents. Finally, the listwise category is directly related to the ranking task where the input to a listwise based algorithm is the entire set of documents associated with a query in the training data.

In recently proposed LTR models, deep architectures are used to learn both features and models simultaneously. Huang et al. [13] proposed the first Deep Neural Network architecture for web search using query-title pairs. The proposed model is based on the Siamese architecture [20], in which a neural network model independently maps the query q and the title of a given document d into feature vectors.

Existing deep matching models for the ad-hoc retrieval task can be categorized into two types. The first type, the representation-focused model, tries to extract a good feature representation for a sequence of tokens using a deep neural network. For example, Shen et al. [14] proposed a Convolutional Deep Structured Semantic Model (C-DSSM) in which a convolutional neural network (CNN) is used in a Siamese architecture. So the feature extractor F is a CNN that is applied to letter-tri-gram input representation, while M is the cosine similarity function. In ARC-I [15], F is a CNN, and M is a multi-layer perceptron (MLP). Models in this first group defer the interaction between two sentences until learning individual representations, so that there is a risk of losing important details for the matching task.

The second group, the interaction-based models, starts by building local interactions between two texts and then trains a deep model to capture the important interaction patterns for matching. For example, in ARC-II [15], F maps each text to a sequence of word embeddings, while M is a CNN over the interaction matrix between the two texts. In DUET [16], an interaction-based network and a representation-based network are combined in a single architecture. Jaech et al. [21] proposed a deep network architecture called Match-Tensor that uses multiple representations for the interaction matrix in order to capture rich signals to calculate query-document relevance. Two bi-directional LSTM models are used to encode the query and document word-embedding sequences into LSTM states. Guo et al. [17] developed a deep relevance matching model (DRMM) to perform term matching using histogram-based features. The authors argued that the interaction-based models that apply convolutional filters on the interaction matrix are too dependent on relative positions of tokens, and do not sufficiently distinguish between similarity matching and exact matching signals. A Term Gating Network (TGN) controls the contribution of each query token to the final score. Dai et al. [22] proposed a Convolutional Kernel-based Neural Ranking Model (Conv-KNRM) for ad-hoc search. Conv-KNRM uses a CNN to embed n-grams of the query and document into a unified embedding space, and computes the similarity between each pair of n-gram embeddings. These similarities are compared to a set of K kernels, where each kernel is a normal distribution with a given mean and standard deviation. Then kernel-pooling [23] is used to summarize the similarities into a soft-matching feature vector of dimension K.

B. 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 to table ranking. Cafarella et al. [2], [4] retrieve relevant documents using web search engines, and 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 [24]. However, a table has multiple components of varying importance, which means that retrieval models for multi-field documents are often more appropriate [5]. For supervised ranking of documents, features are grouped into three categories: document, query, and query-document features [25]. In the case of table retrieval, multiple query, table, and query-table features are proposed in the literature [1], [4]. Zhang and Balog [12] proposed extending these features with semantic matching between queries and tables using various semantic spaces.

Recent work has used embedding techniques to learn a low dimensional representation for table tokens. Deng et al. [26] proposed a natural language modeling based approach to embed table tokens into low dimensional vectors. The

trained embedding is then used with entity similarity from a knowledge base to rank tables. Trabelsi et al. [27] proposed a new word embedding model for the tokens of table attributes using contextual information of every table. The model is used to predict additional contexts of each table that are incorporated into a mixed ranking model to compute relevance score. Using matrix factorization, Chen et al. [6] generated additional headers that are used in ranking table-query pairs.

III. Proposed Method

In this section, we introduce our proposed deep relevance model to enhance ranking of tables given a user's query. In training, a set of queries $Q = \{q^1, q^2, \ldots, q^m\}$ is given. Each query q^i is associated with a list of tables $t^i = (t^i_1, t^i_2, \ldots, t^i_{n^i})$, where t^i_j denotes the j^{th} table and n^i is the size of t^i. Each list of tables t^i is associated with a list of relevance scores $y^i = (y^i_1, y^i_2, \ldots, y^i_{n^i})$ where y^i_j denotes the relevance score of table t^i_j with respect to query q^i. We propose f_w, a new deep relevance model with parameters w, that is used to predict the relevance score of a given query-table pair (q^i, t^i_j). Our proposed model $f_w = M \circ F$ contains a feature extractor function F and a ranking model M. A feature vector $x^i_j = F(q^i, t^i_j)$ is created from each query-table pair (q^i, t^i_j). Then a ranking function M is used to predict a relevance score $M(x^i_j)$. So for a given query q^i and a list of tables t^i associated with the query, the objective is to obtain a list of scores $z^i = (M(x^i_1), M(x^i_2), \ldots, M(x^i_{n^i}))$. The predicted relevance scores are used to rank query-table pairs so that higher ranked tables should be more relevant to the query.

A. Listwise loss function for table retrieval

We propose using a listwise based loss function for table retrieval rather than relying on pointwise and pairwise approaches for two reasons. First, we are interested in training our model to generate a ranked list of tables for a given query without requiring the predictions of our model to match the ground truth relevance scores. Second, although negative sampling can be used in the pairwise approach to avoid the quadratic increase of query-table pairs, the pairwise strategy can increase data imbalance when there is a dominating class [19]. So the loss function L is given by:

$$L(w) = \sum_{i=1}^{m} l(y^i, z^i) \tag{1}$$

where l is a listwise loss function. We adapt the loss function proposed by Cao et al. [28] to the table retrieval task. Given a query q^i associated with a list of tables t^i and relevance scores y^i, the feature extractor F extracts features from a given query-table pair, then a ranking model M generates a score list z^i. As in Cao et al. [28], the ground truth relevance scores and the predicted scores are converted into probabilities $P_{y^i}(q^i, t^i_j)$ and $P_{f_w}(q^i, t^i_j)$, respectively, using the softmax function. With cross entropy loss, for a query q^i, $l(y^i, z^i)$ equals to:

$$l(y^i, z^i(w)) = -\sum_{j=1}^{n^i} P_{y^i}(q^i, t^i_j) \times log(P_{f_w}(q^i, t^i_j)) \tag{2}$$

The gradient of $l(y^i, z^i(w))$ with respect to model parameters w is detailed in Cao et al. [28].

B. Deep relevance model architecture

We propose a new interaction-based deep semantic and relevance matching model (DSRMM) for table retrieval. There are two classes of neural architectures for ad hoc retrieval. Semantic similarity architectures treat the query and target as equals, and try to match them. Query relevance architectures exploit characteristics of the ad hoc retrieval task. Our hybrid model combines both concepts into one architecture. We extract semantic and relevance feature vectors from the deep semantic similarity model and query relevance matching network, respectively. The two features are then concatenated and passed through a fully connected layer to predict a retrieval score under the semantic and relevance settings. For a given query q^i and table t^i_j, the final relevance score is given by

$$f_w(q^i, t^i_j) = NN_c([SS(q^i, t^i_j); QR(q^i, t^i_j)]) \tag{3}$$

where SS is the semantic similarity neural network, QR is the query relevance neural network, and NN_c is a neural network used to predict the relevance score from a vector concatenating the outputs of the semantic and relevance networks.

1) Inputs to Networks: The input to our architecture is a query-table pair. A given table t^i_j contains description, cell values, and attributes or headers as shown in Figure 1. The description denotes the metadata of the table such as page title, section title, table caption, etc. The input representation of a table, denoted by T^i_j, contains the word embeddings of description and attributes. Cell values contain rich information that can be used to match query and tables. Some queries depend on the presence of specific columns, others depend on the presence of specific rows. In order to incorporate row and column representations into T^i_j, we present a row/column summarizer component that compresses each row and each column into a fixed length feature vector. In particular, given the k-th row r_k and l-th column c_l from t^i_j, the outputs of the summarizer component $S(r_k)$ and $S(c_l)$ are given by:

$$S(r_k) = \frac{1}{|r_k|} \sum_{w \in r_k} v_w \quad \text{and} \quad S(c_l) = \frac{1}{|c_l|} \sum_{w \in c_l} v_w$$

where v_w is the word embedding of token w, and $|r_k|$ and $|c_l|$ are the number of tokens in the k-th row and l-th column of t^i_j, respectively. A table t^i_j with n_r rows and n_c columns results in $n_r + n_c$ additional feature vectors that are concatenated to T^i_j to form the $|t^i_j| \times k$ table representation, where k is the dimensionality of word embeddings. The query representation, denoted by Q^i, of q^i is calculated using the word embedding of each token in the query. So the dimensionality of Q^i is $|q^i| \times k$, where $|q^i|$ is the length of the query. For the rest of the paper, we assume that $|t^i_j|$ is equal to m for all tables, and $|q^i|$ is equal to n for all queries.

2) Semantic similarity component: Semantic similarity extracts contextual features from query-table interactions to infer the semantic meaning and relation between query and table. The top portion of Figure 1 illustrates the semantic similarity

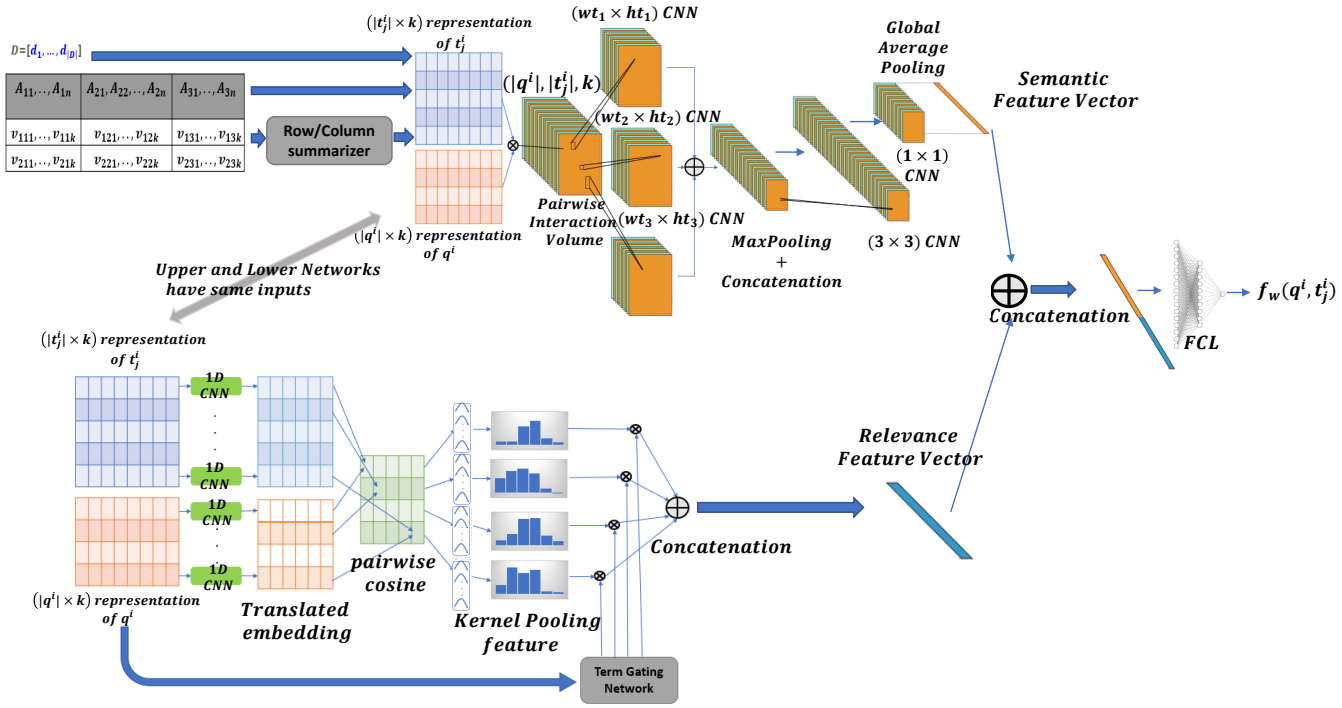

Fig. 1: Architecture of Deep Semantic and Relevance Matching Model (DSRMM) for table retrieval. $\oplus$ denotes the concatenation operator, and $\otimes$ is the pointwise multiplication operator. The semantic and relevance networks extract semantic and relevance feature vectors respectively. The concatenated vector of semantic and relevance features is passed through a fully connected network to predict the final relevance score between query and table.

component, SM, in detail. To capture semantic similarity, we build the interaction matrix X between query and table using the pointwise multiplication between pairwise rows from query representation Q^i and table representation T^i_j. In tables, token order only matters locally; the rows and columns could be arbitrarily ordered without changing the meaning of the table. Thus, there is less utility in encoding sequences with the bi-LSTM models as is done in Match-Tensor [21].

To capture multiple levels of similarity between the query and table, we propose using multiple convolutional filters with different width and height. The width indicates the number of query tokens that are used in the convolution. The set of width values is given by $\{wt_1, wt_2, wt_3\}$. The height value indicates the number of tokens from the table that are used in the convolution. The set of height values is given by $\{ht_1, ht_2, ht_3\}$. Each table is represented as a matrix $T^i_j \in \mathbb{R}^{m \times k}$, and each query is represented as a matrix $Q^i \in \mathbb{R}^{n \times k}$. After pointwise multiplication of each query-table pair of tokens, we obtain the interaction tensor X with dimension $n \times m \times k$. We pass X through k_1 filters of size $(wt_1 \times ht_1)$, k_2 filters of size $(wt_2 \times ht_2)$, and k_3 filters of size $(wt_3 \times ht_3)$ to obtain $X_{wt_1 \times ht_1}$, $X_{wt_2 \times ht_2}$, and $X_{wt_3 \times ht_3}$ feature maps respectively. We apply a max pooling operation to each feature map, and we concatenate the resulting tensors into one tensor with size $(n/2 \times m/2 \times (k_1 + k_2 + k_3))$. In order to extract high level semantic interactions between the table and query, we apply k_4 convolutional filters of size (3×3); then we reduce the number of depth channels using s filters with size (1×1). We summarize the information obtained in each channel using Global Average Pooling (GAP) to obtain our semantic similarity feature vector $SS(q^i, t^i_j)$.

3) Query relevance component: Guo et al. [17] demonstrated that many neural models for document retrieval depended on semantic similarity, but this is an inappropriate measure for the task. They argued that ad hoc retrieval depends on exact matching and query term importance, and that sometimes queries only need to match part of the document, not the document as a whole. Semantic similarity on the other hand assumes that the items being matched are roughly equivalent in scope, that their meanings are composed from their parts, and that items must be matched in their entirety. We believe that this argument for a specialized architecture applies equally to table retrieval. However, because semantic similarity still provides a useful signal based on context, we design a model that includes the best features of both approaches.

Our query relevance component adapts Guo et al.'s matching histogram mapping [17] to the table retrieval task. The matching histogram mapping is based on a hard assignment of matching similarities between a given query token and the table tokens. This histogram-based feature counts the number of table tokens whose similarity to the query token is within the bin's range. However, this representation is not differentiable and not computationally efficient. Therefore, we adapt kernel-pooling [23] for soft-match signals to the table

retrieval task. The objective of using kernel pooling is to extract a soft matching histogram between a given query token and table tokens. Given a query token q_l^i and table t_j^i, we use $r1$ 1-D convolutions to translate each token. Then we calculate cosine similarity between the translated tokens. There are two advantages of the convolution. First, it allows us to learn similarities that are present in our query/table collection but that were not captured by the Glove [29] corpus. Second, instead of updating the word embedding ($|V| \times k$ parameters, where V is the vocabulary), we update the convolutional filters ($r1 \times k$ parameters), so that we decrease the complexity of the model (as $r1 << |V|$).

q_l^i is embedded to $\mathbf{q_l^i}$, then translated to $\mathbf{v_l^i}$. The cosine similarity between $\mathbf{v_l^i}$ and the s^{th} token of t_j^i is given by C_{ls}. Suppose that we have K kernels for soft matching, with mean $\mu = \{\mu_1, \mu_2, \ldots, \mu_K\}$, and standard deviation $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$; the soft matching assignment of query token q_l^i to k^{th} kernel is given by

$$K_k(q_l^i, t_j^i) = \sum_{s=1}^{m} exp(-\frac{(C_{ls} - \mu_k)^2}{2\sigma_k^2}) \quad (4)$$

K_k calculates the soft matching similarities around μ_k with a variance σ_k^2. The closer C_{lm} is to μ_k, the higher is K_k. The kernel pooling feature vector of query token q_l^i and table t_j^i is given by:

$$KP(q_l^i, t_j^i) = [K_1(q_l^i, t_j^i); K_2(q_l^i, t_j^i), \ldots, K_K(q_l^i, t_j^i)] \quad (5)$$

Since exact matching is an important signal in any retrieval task, we reserve the first kernel K_1 for soft exact matching. So, we set μ_1 to 1, and σ_1 to 0.001.

Query tokens are not equally important for relevance matching. In order to model each query token importance, we use a Term Gating Network (TGN) [17] to control the contribution of each query token to the final relevance score. For a given query q^i, the gating function is given by:

$$g_j = \frac{exp(\mathbf{w_g q_j^i})}{\sum_{l=1}^{n} exp(\mathbf{w_g q_l^i})} \quad (6)$$

where $\mathbf{w_g}$ is the weight vector of the term gating network, and $\mathbf{q_l^i}$ is the embedding of query token q_l^i, The final feature vector using relevance matching is given by:

$$QR(q^i, t_j^i) = [h_1^{ij}; h_2^{ij}, \ldots, h_{|q^i|}^{ij}] \quad (7)$$

with

$$h_l^{ij} = g_l \times KP(q_l^i, t_j^i) \quad (8)$$

Our final model captures both semantic similarity and query relevance matching which both play an important role in ad hoc table retrieval.

IV. Evaluation

We evaluate our approach using two different data collections and compare it against a number of baselines.

A. Data and query collections

1) WikiTables: This dataset is composed of the WikiTables corpus [30] containing over $1.6M$ tables. Each table has five indexable fields: table caption, attributes (column headings), data rows, page title, and section title. Additional LTR features [12] are provided for each table. We use the same test queries that were used by Zhang and Balog [12]. For the ground-truth relevance scores, every query-table pair is evaluated using three numbers: 0 means "irrelevant", 1 means "partially relevant" and 2 means "relevant". The total number of queries in the WikiTables collection is 60. We report results of the mean of five runs of five-fold cross-validation of the entire query-table pairs collection for our method and baselines.

2) WebQueryTable: Additionally, we use the WebQueryTable [18] collection, which also contains query-table pairs. Each query-table pair has a binary relevance value, and only one table is relevant to a given query. The tables were selected from the top-ranked web page, and the annotators were asked to label whether the extracted table is relevant to the query or not. The number of tables is 148,851. Each table has four indexable fields: table caption, table sub-caption, attributes, and data rows. The total number of queries is 21,142. The dataset is pre-split, where we use 252,420 query-table pairs to train our model, and 71,479 pairs for testing.

B. Baselines

We compare the performance of our proposed model against several baselines from multiple fields.

1) Unsupervised table retrieval: We compare the performance of our proposed method against MCON [27] which is based on new word embeddings for attribute tokens of tables. When calculating the retrieval score of MCON, we use the MaxTable ranking method which was shown to be the best ranking method for unsupervised table retrieval [27].

2) Ad-hoc table retrieval: We compare DSRMM against state-of-the-art methods for table retrieval: LTR and STR [12]. Table2Vec [26] is another system that solves the same task, but the reported performance is lower than that of STR. Therefore, we do not include Table2Vec in our evaluation.

3) Unsupervised document ranking approaches: Unsupervised document ranking approaches can be applied to table retrieval if a linearization is applied to the table to create a sequence of terms. Depending on the structure of the table, we obtain two categories of baselines:

Single-field document ranking: A table is considered as a single field document by concatenating indexable fields. We compare our approach against two baselines in the category of single-field document ranking. The first ranking method, SingleField-BM25, is based on BM25 to calculate a retrieval score. In the second ranking method, called SingleField-P, we calculate the score of a query-table pair using word embedding-based ranking method MaxTable [27].

Multi-field document ranking: In a multi-field ranking scenario, a table is defined using multiple fields. For example, in WikiTables data, the fields are: page title, section title, table caption, attributes and table body or values. We compare our

TABLE I: Table retrieval evaluation results for WikiTables dataset

Category	Method	NDCG@5	MRR	MAP
Unsupervised table retrieval	MCON [27]	0.515±0.018	0.532±0.019	0.519±0.021
Ad-hoc table retrieval	LTR [12]	0.514±0.039	0.570±0.038	0.522±0.035
	STR [12]	0.582±0.037	0.636±0.037	0.591±0.035
Unsupervised document ranking	SingleField-BM25	0.451±0.032	0.516±0.038	0.477±0.030
	SingleField-P	0.471±0.023	0.501±0.027	0.482±0.031
	MultiField-P	0.499±0.026	0.523±0.027	0.492±0.029
LTR for document retrieval	C-DSSM [14]	0.510±0.027	0.548±0.026	0.521±0.026
	ARC-I [15]	0.553±0.033	0.607±0.032	0.553±0.029
	ARC-II [15]	0.567±0.029	0.613±0.029	0.562±0.029
	DUET [16]	0.524±0.037	0.579±0.041	0.528±0.035
	Match-Tensor [21]	0.569±0.030	0.613±0.034	0.565±0.031
	DRMM [17]	0.482±0.023	0.522±0.028	0.491±0.025
	Conv-KNRM [22]	0.595±0.033	0.638±0.031	0.608±0.032
Sentence classification	Kim [31]	0.566±0.034	0.617±0.035	0.564±0.037
Document classification	HAN [32]	0.567±0.034	0.614±0.037	0.565±0.033
Table type classification	TabNet [33]	0.570±0.030	0.616±0.029	0.568±0.029
Ad-hoc table retrieval	Our proposed model (DSRMM)	**0.640±0.029**	**0.680±0.028**	**0.642±0.029**

method against MultiField-P which is based on the pretrained Glove word embedding when calculating cosine similarity. MaxTable [27] similarity measure is used to calculate the score between query tokens and a given field in a table.

4) Learning to rank for document retrieval methods: We compare our proposed method against C-DSSM [14], ARC-I [15], ARC-II [15], DUET [16], Match-Tensor [21], DRMM [17], and Conv-KNRM [22]. A given document is the concatenation of the description and attributes of a table.

5) Sentence classification: Kim [31] proposed a CNN for sentence classification. A 1-D convolutional layer with multiple filter widths is applied to the concatenation of word embeddings of sequence tokens. In our table retrieval task, a given sentence is the concatenation of tokens in description, query, and attributes. The final output is a relevance score instead of classification score.

6) Document classification: HAN [32] is a hierarchical attention network for document classification. It is composed of four parts: a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer. A GRU-based [34] sequence encoder is used to encode each token in a given sentence into a hidden state. In a table retrieval scenario, a document contains sentences from the query, table description, and attributes.

7) Table type classification: Nishida et al. [33] proposed a hybrid model, called TabNet, for table classification. The model is composed of a recurrent neural network (RNN) to encode a sequence of tokens of each cell. Then, the attention mechanism from Yang et al. [32] is applied to extract important tokens from each cell, and form an input volume. To use TabNet for table retrieval, we add a cell to the table that contains the query. So, in addition to encoding the original cells of a given table using RNN, we also encode query tokens.

C. Experimental Setup

In DSRMM, we use a pretrained neural word embedding from Glove with $k = 300$. We choose not to update the word embedding when minimizing the loss function for two reasons: first, we have far fewer labeled query-table pairs than examples from the unlabeled text corpus used to train Glove. Second, by freezing word embeddings, we reduce model complexity, and focus the efforts of training on extracting semantic and relevance matching. We train our model for 30 epochs, and each batch contains only tables that are candidates of a given query in order to calculate the listwise loss. We use the Adam optimizer for gradient descent. We set the number of query tokens n to 6, and the number of table tokens m to 100. The m table tokens contain the first 50 tokens from description, first 30 tokens from attributes, and 20 rows and columns. We start by including column summaries, and then row summaries because in many cases, tables contain more rows than columns. We set the number of CNN filters of the first layer $k1$, $k2$, and $k3$ to 20. The set of width values $\{wt_1, wt_2, wt_3\}$ is equal to $\{3, 5, 7\}$, and the set of height values is equal to $\{3, 3, 3\}$. $k4$, which is the number of CNN filters in the second layer, is equal to 200. We set the dimension s of the semantic matching component to 100. We set the number of kernels in the relevance matching component to 5. So, given that the cosine similarity is between $[-1, 1]$, the means of the kernels are $\mu = [1, 0.75, 0.25, -0.25, -0.75]$. The standard deviations of the kernels are $\sigma = [0.001, 0.1, 0.1, 0.1, 0.1]$. We reserve the first kernel ($\mu_1 = 1$ and $\sigma_1 = 0.001$) for the exact match. We release our code on GitHub[1].

D. Experimental results

We evaluate the performance of our proposed method and baselines on the table retrieval task using Normalized Discounted Cumulative Gain (NDCG), Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP).

1) Ranking results: Table I shows the performance of different approaches on the WikiTables collection. We show that our proposed method DSRMM outperforms the baselines for all evaluation metrics. Consistent with what has been shown in ad hoc document retrieval, supervised approaches perform better on ad hoc table retrieval than unsupervised approaches. Among unsupervised table and document retrieval, MCON

[1]https://github.com/medtray/IEEEBigData2020-DSRMM-Table-Retrieval

achieves higher performance for all evaluation metrics. This can be explained by the use of a mixed ranking model that incorporates the metadata of a table and the additional contexts in order to calculate the retrieval score.

TABLE II: Table retrieval results for WebQueryTable.

Method	NDCG@5	MRR/MAP
Match-Tensor [21]	0.3232	0.3256
Conv-KNRM [22]	0.6052	0.5978
Kim [31]	0.3078	0.3097
HAN [32]	0.4620	0.4384
TabNet [33]	0.4876	0.4597
DSRMM	**0.6516**	**0.6345**

Although Kim [31], HAN, and TabNet are not designed for table retrieval, we show that these models have competitive results compared to many of the other methods. Among these methods, TabNet has the best performance, and this can be explained by the fact that TabNet is designed for table type classification, so the input of TabNet is similar to table retrieval based methods. On the other hand, the primary input of HAN and Kim [31] are documents and sentences, respectively.

The deep semantic matching component of DSRMM extracts interactions between query tokens and table tokens using convolutional filters. We explain the improvement in performance of our model compared to baselines by three facts. First, the different low-level filters capture multiple levels of similarity which are important in a retrieval task. Second, some patterns are hard to capture using only one layer of convolutional filters, so in DSRMM, the second set of convolutional filters identify high level interactions. Third, the semantic component is a position dependent component, and treats all query tokens equally. To solve that, our proposed relevance matching component provides position free and strength-preserving histograms that are weighted by the importance of each query using a TGN. To test significance, we use a two-tailed paired t-test between DSRMM and the best baseline reported in Table I which is Conv-KNRM. We found a t-test significance at level 0.05 for all evaluation metrics.

For the WebQueryTable dataset, we compare the performance of our method against the top two approaches in LTR for document retrieval category from Table I: Conv-KNRM and Match-Tensor, and neural models from sentence, document, and table type classification (Kim [31], HAN, and TabNet). We note that we do not compare to LTR/STR because these methods require a wide range of features that are not provided in the dataset. For WebQueryTable dataset, there is only one relevant table per query, so MRR is always equivalent to MAP (and thus only one column is used to report results on both metrics in the Table II). As shown in Table II, our method outperforms Match-Tensor, Kim [31], HAN, and TabNet by a large margin. As with WikiTables, Conv-KNRM is the closest competitor. Conv-KNRM captures semantic matching of unigrams, bigrams, and trigrams between query and table tokens, but the query tokens are treated uniformly.

2) Analysis of alternative design choices: In order to justify the importance of each component in our proposed method, we present an ablation study of our hybrid model using the WikiTables dataset. We train two versions of our model: the first version is only the semantic similarity component, and the second version is only the query relevance component. Our study shows that the semantic similarity network has better performance than the query relevance network. The full architecture outperforms both isolated components, and adding query relevance to semantic similarity increases NDCG@5 from 0.601 to 0.640. Furthermore, removing the term gating network drops the performance of the query relevance component from 0.542 to 0.532, and this supports the idea of having different contributions to relevance score for each query term.

We study the effect of removing the Row/Column summarizer from DSRMM as shown in the last row of the ablation analysis. So we define a baseline in which we randomly select 50 string values from each table, and append the values tokens to description and attributes tokens. Adding all values is computationally expensive and consumes a significant amount of memory. Removing the Row/Column summarizer lowers scores for all evaluation metrics. Thus, each component of our system has a positive effect on the final results. Table III shows a decrease in retrieval metrics for the query relevance component when using 2 kernels in kernel pooling as opposed to 5 kernels in the original DSRMM. So, two kernels are not enough to extract fine-grained relevance matching signals.

For system variations analysis, Table III shows that the listwise based approach leads to better retrieval results than the pointwise loss function which is consistent with what has been shown in document retrieval results. With listwise loss, DSRMM focuses on ranking the tables, rather than predicting the exact relevance score. We study a second system variation that consists of adding STR features to the DSRMM model. STR represents the set of features for query, table, and query-table pairs and semantic features from various spaces. We use precalculated STR features from [12]. We append STR features to our proposed semantic and relevance features, and train our model. Table III shows that adding STR features leads to a slight improvement over vanilla DSRMM for NDCG@5. However, the DSRMM model trained with description, attributes, and row and column summaries has the best performance for MRR and MAP, by using only word embedding space for semantic and relevance matching. So extracting STR features is no longer required to achieve the best performance in table retrieval. This is especially important since features like bag-of-entities and bag-of-categories [12] are not always available.

V. Conclusion

We have shown that a hybrid deep model that combines a semantic similarity component and a query relevance component outperforms the best previously published results in table retrieval (STR) [12], achieving up to 9.96% improvement in NDCG@5 score. Furthermore, we have demonstrated how our approach can be used on tables for which fewer metadata features are available than those required by STR. We have shown that adding the LTR features to our system helps less

TABLE III: Analysis of alternative design choices using WikiTables dataset

Analysis	Method	NDCG@5	MRR	MAP
Ablation	Semantic similarity component only	0.601±0.031	0.640±0.033	0.596±0.031
	Query relevance component only	0.542±0.036	0.594±0.030	0.549±0.027
	Query relevance component only without TGN	0.532±0.034	0.583±0.030	0.537±0.032
	Query relevance only with 2 kernels	0.502±0.034	0.555±0.036	0.511±0.034
	DSRMM without Row/Column Summarizer	0.627±0.026	0.668±0.028	0.629±0.028
System variations	DSRMM+Pointwise loss	0.617±0.026	0.655±0.028	0.612±0.028
	DSRMM+STR	**0.642±0.021**	0.679±0.023	0.641±0.021
Proposed method	DSRMM	0.640±0.029	**0.680±0.028**	**0.642±0.029**

than adding information about the data values in the table using row and column summaries. This suggests that the specialized LTR [1], [4] and STR [12] features are not useful once one has developed a high quality model for table retrieval.

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 *Proc. of ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, 2013, pp. 18–26.

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

[3] 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*, 2014, pp. 533–542.

[4] 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.

[5] 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.

[6] Z. Chen, H. Jia, J. Heflin, and B. D. Davison, "Leveraging schema labels to enhance dataset search," in *Proc. European Conference on Information Retrieval (ECIR)*. Springer, 2020, pp. 267–280.

[7] Z. Chen, M. Trabelsi, J. Heflin, Y. Xu, and B. D. Davison, "Table search using a deep contextualized language model," in *Proc. of the 43rd Int'l ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020, pp. 589–598.

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

[9] M. Trabelsi, J. Cao, and J. Heflin, "Semantic labeling using a deep contextualized language model," *CoRR*, vol. abs/2010.16037, 2020.

[10] M. Yakout, K. Ganjam, K. Chakrabarti, and S. Chaudhuri, "Infogather: entity augmentation and attribute discovery by holistic matching with web tables," in *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data*, 2012, pp. 97–108.

[11] Y. A. Sekhavat, F. D. Paolo, D. Barbosa, and P. Merialdo, "Knowledge base augmentation using tabular data," in *Proceedings of the Workshop on Linked Data on the Web*, ser. CEUR Workshop Proceedings, 2014, co-located with the 23rd International World Wide Web Conference.

[12] S. Zhang and K. Balog, "Ad hoc table retrieval using semantic similarity," in *Proc. World Wide Web Conference (WWW)*, 2018, pp. 1553–1562.

[13] P. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. P. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *Proc. of the 22nd ACM Int'l Conf. on Information and Knowledge Management (CIKM)*, 2013, pp. 2333–2338.

[14] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "Learning semantic representations using convolutional neural networks for web search," in *23rd International World Wide Web Conference, WWW, Companion Volume*, 2014, pp. 373–374.

[15] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2042–2050.

[16] B. Mitra, F. Diaz, and N. Craswell, "Learning to match using local and distributed representations of text for web search," in *Proceedings of the 26th International Conference on World Wide Web, WWW*, 2017, pp. 1291–1299.

[17] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proc. of the 25th ACM Int'l Conf. on Information and Knowledge Management (CIKM)*, 2016, pp. 55–64.

[18] Z. Yan, D. Tang, N. Duan, J.-W. Bao, Y. Lv, M. Zhou, and Z. Li, "Content-based table retrieval for web queries," *Neurocomputing*, vol. 349, pp. 183–189, 2017.

[19] T.-Y. Liu, "Learning to rank for information retrieval," *Found. Trends Inf. Retr.*, vol. 3, no. 3, pp. 225–331, Mar. 2009.

[20] Y. Lecun and Y. Bengio, *Convolutional networks for images, speech, and time-series*. MIT Press, 1995.

[21] A. Jaech, H. Kamisetty, E. K. Ringger, and C. Clarke, "Match-tensor: a deep relevance model for search," *ArXiv*, vol. abs/1701.07795, 2017.

[22] Z. Dai, C. Xiong, J. Callan, and Z. Liu, "Convolutional neural networks for soft-matching n-grams in ad-hoc search," in *Proc. of the 11th ACM Int'l Conf. on Web Search and Data Mining (WSDM)*, 2018, p. 126–134.

[23] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power, "End-to-end neural ad-hoc ranking with kernel pooling," in *Proc. 40th Int'l ACM SIGIR Conf. on Research and Development in Information Retr.*, 2017, pp. 55–64.

[24] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *Overview of the Third Text REtrieval Conference (TREC-3)*, January 1995, pp. 109–126.

[25] 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.

[26] L. Zhang, S. Zhang, and K. Balog, "Table2vec: Neural word and entity embeddings for table population and retrieval," in *Proc. of the 42nd Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval*, 2019, p. 1029–1032.

[27] M. Trabelsi, B. D. Davison, and J. Heflin, "Improved table retrieval using multiple context embeddings for attributes," in *2019 IEEE International Conference on Big Data (BigData)*, 2019, pp. 1238–1244.

[28] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine Learning*, 2007, pp. 129–136.

[29] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Proc. of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[30] C. Bhagavatula, T. Noraset, and D. Downey, "Tabel: Entity linking in web tables," in *International Semantic Web Conference*, 2015, pp. 425–441.

[31] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

[32] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, "Hierarchical attention networks for document classification," in *Proc. of Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.

[33] K. Nishida, K. Sadamitsu, R. Higashinaka, and Y. Matsuo, "Understanding the semantic structures of tables with a hybrid deep neural network architecture," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 168–174.

[34] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc, of the Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Oct. 2014, pp. 1724–1734.