StruBERT: Structure-aware BERT for Table Search and Matching

Mohamed Trabelsi  
Lehigh University  
Bethlehem, PA, USA

Zhiyu Chen  
Lehigh University  
Bethlehem, PA, USA

Shuo Zhang  
Bloomberg  
United Kingdom

Brian D. Davison  
davison@cse.lehigh.edu  
Lehigh University  
Bethlehem, PA, USA

Jeff Heflin  
heflin@cse.lehigh.edu  
Lehigh University  
Bethlehem, PA, USA

ABSTRACT
A large amount of information is stored in data tables. Users can search for data tables using a keyword-based query. A table is composed primarily of data values that are organized in rows and columns providing implicit structural information. A table is usually accompanied by secondary information such as the caption, page title, etc., that form the textual information. Understanding the connection between the textual and structural information is an important yet neglected aspect in table retrieval as previous methods treat each source of information independently. In addition, users can search for data tables that are similar to an existing table, and this setting can be seen as a content-based table retrieval. In this paper, we propose StruBERT, a structure-aware BERT model that fuses the textual and structural information of a data table to produce context-aware representations for both textual and tabular content of a data table. StruBERT features are integrated in a new end-to-end neural ranking model to solve three table-related downstream tasks: keyword- and content-based table retrieval, and table similarity. We evaluate our approach using three datasets, and we demonstrate substantial improvements in terms of retrieval and classification metrics over state-of-the-art methods.

CCS CONCEPTS
• Information systems → Retrieval models and ranking: Structured text search.

KEYWORDS
table matching, table search, table similarity

1 INTRODUCTION
Researchers have focused on utilizing the knowledge contained in tables in multiple tasks including augmenting tables [2, 6, 49, 54, 56], extracting knowledge from tables [26], table retrieval [4, 5, 7, 10, 27, 33, 36], and table type classification [11, 12]. Users can search for data tables using a keyword-based query as in document retrieval. Additionally, users can look for similar data tables on the web to an existing table. This can be seen as a query by example scenario or content-based table retrieval. Content-based table retrieval requires a table matching phase to predict the semantic similarity between the query table and queried table. Another table-related task, that requires a table matching phase, is table similarity [19] in which the objective is to predict a binary semantic similarity between two tables. A table similarity algorithm can be used as a core component in multiple tasks such as table classification and clustering [22, 52], table fusion [17] and finding related tables [15]. We consider content-based table retrieval and table similarity as two instances of table matching. Prior methods [8, 55] ignore the structural information as data values are linearized, and treated as a single text. In table matching, Habibi et al. [19] decouples the textual information from the structural information of a data table.

A table is composed of structural information defined by rows and columns, so that we estimate the table matching score based on the semantic matching between rows and columns from both tables. Row-based matching selects mainly candidate tables that can be considered as additional records to the query table. On the other hand, column-based matching identifies tables that can potentially be used for table augmentation by join operations. In both cases, we have a structured query that is matched against structured tables. Users can also search for tables that match a sequence of keywords composed of several values, attributes, metadata, etc. Figure 1 depicts both row/column-based matching between tables and row/column-based queries. In the former case, Figure 1 shows how columns and rows are matched to capture the semantic similarity between tables. In the latter case, Figure 1 shows two examples of keyword-based table retrieval where the query is a simple and unstructured natural language sequence. The row-based query is relevant to multiple rows of a table, and the column-based query contains keywords related to a subset of attributes from a table.

In order to overcome the limitations of prior methods in table retrieval and similarity, we propose a new model, called Structure-aware BERT (StruBERT), that fuses the textual and structural information of a data table to produce a context-aware representation.
Row-based query: information about Ronaldo and Messi

Column-based query: List of countries of players

Figure 1: Row-based matching (in orange) and column-based matching (in green) can happen between a keyword-based query and a table or between two tables.

for both textual and tabular content of a table. In general, a table can be viewed as a row- and column-based structure, and rows and columns should correspond to both (1) the relevance score in table matching where rows and columns of a table pair are matched, and (2) the retrieval score in keyword-based table retrieval where table content is considered as a relevant field to the keywords of a query. Based on the observations from matching cases in Figure 1, we propose a unified model that produces both row- and column-based features to predict the semantic matching between structured/unstructured query and structured table. Inspired by TaBERT [51], which computes a joint representation for table columns and utterance tokens using vertical self-attention, we propose a horizontal self-attention that produces a joint representation for table rows and query tokens. Our proposed model produces four feature vectors that correspond to the joint representations of the structural and textual information of a table. Two fine-grained features represent the context-aware embeddings of rows and columns, where both horizontal and vertical attentions are applied over the column- and row-based sequences, respectively. Two coarse-grained features capture the textual information from both row- and column-based views of a data table. These features are incorporated into a new end-to-end ranking model, called miniBERT, that is formed of one layer of Transformer [44] blocks, and operates directly on the embedding-level sequences formed from StruBERT features to capture the cross-modal signals of rows and columns.

In summary, we make the following contributions: (1) We propose a new structure-aware BERT model, called StruBERT, that introduces the idea of horizontal self-attention and fuses the structural and textual information of a data table to produce four context-aware features: two fine-grained structure- and context-aware representations for rows and columns, and two coarse-grained representations for row- and column-guided [CLS] embedding. (2) We propose a new ranking model, called miniBERT, that operates directly on the embedding-level sequences formed from StruBERT features to solve three table-related downstream tasks: keyword- and content-based table retrieval, and table similarity. (3) We evaluate over three datasets, and demonstrate that our new method outperforms the state-of-the-art baselines, and generalizes to multiple table-related downstream tasks.

2 RELATED WORK

For supervised ranking of tables, multiple query, table, and query-table features are proposed in the literature [2, 5]. Zhang and Balog [55] proposed extending these features with semantic matching between queries and tables using semantic spaces. Recent works have used embedding techniques to learn a low dimensional representation for table tokens. Deng et al. [53] proposed a natural language modeling-based approach to create embeddings for table tokens. The trained embedding is then used with entity similarity from a knowledge base to rank tables. Trabelsi et al. [43] proposed a new word embedding model for the tokens of table attributes using the contextual information of every table.

Deep contextualized language models, like BERT [16] and Roberta [25], have been recently proposed to solve multiple tasks [13, 23, 29, 30, 35, 38, 42, 45, 48, 50]. Building on BERT, Chen et al. [8] proposed a BERT-based ranking model to capture the matching signals between the query and the table fields using the sentence pair setting. They first select the most salient items of a table to construct the BERT representation, where different types of table items and salient signals are tested. Trabelsi et al. [40] have shown that neural ranking models can be used in table retrieval by proposing a deep semantic and relevance matching model (DSRMM). Shraga et al. [37] use neural networks to learn unimodal features of a table which are combined into a multimodal representation. Tables can also be represented as graphs to solve table retrieval [9, 41, 46].

Table similarity consists of predicting the semantic similarity between tables and then classifying a table pair as similar or dissimilar. Das Sarma et al. [15] proposed a table similarity method that is based on entity consistency and expansion, and schema similarity, and is used to find related tables in a large corpus of heterogeneous data. Deep learning models have been leveraged to predict the similarity score between tables. TabSim [19] treats the data table fields independently in which one bi-LSTM model is used to map the caption of a data table to an embedding vector, and a second attention-based model is used to compute the embeddings of the columns of a data table.

3 PROBLEM STATEMENT

We formally define the three table-related downstream tasks that we address in this paper.

3.1 Keyword-based Table Retrieval

Given a keyword-based query \( q = q_1 q_2 \ldots q_m \) where \( m \) is the length of the query and \( q_i \) is the \( i \)-th token of \( q \), the objective is to find a relevant set of tables from a table corpus \( \mathcal{C} = \{ T_1, T_2, \ldots, T_l \} \), with \( l \) is the total number of data tables. Our model takes as input a
### Table Views

The primary input to our model is a table \( T_j \) that has \( s \) rows and \( l \) columns. Each table has two forms of information. The first form is the structural information which is composed of headers and data values. A table can be seen as a 2D matrix of cells, and for the purpose of explanation, we assume that the first row corresponds to the headers \( c_1, c_2, \ldots, c_l \), and the remaining \( s - 1 \) rows are data values. The \( i \)-th column of \( T_j \) has the values \( v_{1i}, v_{2i}, \ldots, v_{(s-1)i} \). The second form of information is the textual information which corresponds to the context fields of a table. Several text fields can be used to describe the table such as the caption, the title of the page and section that contain the table, etc. We denote these context fields by the metadata which forms the textual information of a table. In the case of keyword-based table retrieval, the query is considered as an additional form of textual information because the final representations of StruBERT should capture early interactions between the table and query as in interaction-based retrieval models [18, 20, 31] that have achieved better results than the representation-based models [28]. By learning early interactions between the table and keyword-based query, StruBERT produces structure- and context-aware features, where the query is part of the context.

As shown in Figure 2, we propose forming two sets of sequences, denoted by column- and row-based sequences, that are formed on the basis of column- and row-based views, respectively, of a given data table. Yin et al. [51] proposed a row linearization to form sequences from a data table in order to solve the semantic parsing over tables task. Inspired by that, we incorporate row linearization to form row-based sequences, and we propose a column linearization to form column-based sequences.

Given that \( T_j \) has \( l \) columns, we form \( l \) column-based sequences. The \( i \)-th column-based sequence is given by:

\[
\hat{c}_i = c_{i1} \text{[SEP]} c_{i2} \text{[SEP]} \ldots \text{[SEP]} c_{i(s-1)} \text{[SEP]}
\]

where \( t_i \in \{\text{real, text} \} \) is the type of \( c_i \).

The first column in the data table shown in Figure 2 has a type \text{text}, and the
third column has a type real. We use the first column of the table in Figure 2 to illustrate an example of a column-based sequence:

player text Ronaldo [SEP] player text Messi [SEP] ...

We denote the set of column-based sequences by \( \mathcal{C} = \{c_1, c_2, \ldots, c_l\} \). Similarly, we form \( s = \) 1 row-based sequences. The \( i \)-th row-based sequence is given by:

\[
\tilde{r}_i = c_i \langle \text{SEP} \rangle v_{i1} \langle \text{SEP} \rangle c_{i2} v_{i2} \langle \text{SEP} \rangle \ldots \langle \text{SEP} \rangle c_{i\ell} v_{i\ell} \langle \text{SEP} \rangle
\]

We use the first row of the data table in Figure 2 to illustrate an example of a row-based sequence:

player text Ronaldo [SEP] team text Manchester United [SEP] ...

We denote the set of row-based sequences by \( \tilde{R} = \{\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_{(s-1)}\} \). \( \mathcal{C} \) and \( \tilde{R} \) capture only the structural information of \( T_j \). To incorporate the textual information into the structure-based sequences, we concatenate the textual information with each sequence from the structure-based sequences \( \mathcal{C} \cup \tilde{R} \) using the [CLS] and [SEP] tokens of BERT. Given that the textual information \( T_{e_j} \) of \( T_j \) is formed of \( p \) fields \( f_1, f_2, \ldots, f_p \), the new structure- and context-aware sequences are given by:

\[
\tilde{c}_i = [\text{CLS}] T_{e_j} [\text{SEP}] c_i [\text{SEP}]
\]

\[
\tilde{r}_i = [\text{CLS}] T_{e_j} [\text{SEP}] \tilde{r}_i [\text{SEP}]
\]

where:

\[
\tilde{T}_{e_j} = f_1 [\text{SEP}] f_2 [\text{SEP}] \ldots [\text{SEP}] f_p
\]

We denote the column- and row-based structure-aware sequences by \( \mathcal{C} = \{\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_l\} \), and \( \tilde{R} = \{\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_{(s-1)}\} \), respectively.

4.2 StruBERT Model

Figure 2 presents StruBERT which is composed of two phases: sequence encoding and self-attention over encoded sequences.

4.2.1 Sequence Encoding. To capture the dependencies between the textual information and data values in each sequence from \( \mathcal{C} \cup \tilde{R} \), BERT is used as a sequence encoder which produces contextualized embeddings for each token in the tokenized sequence using BERT tokenizer. BERT is preferred over a recurrent architecture because BERT is composed of Transformer blocks that capture long-range dependencies with self-attention better than recurrent architectures [44], and is pretrained on large textual data.

After row (column) linearization and BERT tokenization, each cell has multiple tokens. To compute a single embedding for each cell, we incorporate cell-wise average pooling [51] after the BERT encoding step to pool over the contextualized tokens for each cell defined by [header_name type cell_content]. BERT is composed of \( L \) layers of Transformer blocks. The cell-wise average pooling is applied on the contextualized embedding that is obtained from the last layer. The contextualized embedding of the column-based sequence \( \tilde{c}_i \) is given by:

\[
\tilde{c}_i = [\text{CLS}] T_{e_j} [\text{SEP}] v_{i1} [\text{SEP}] \ldots [\text{SEP}] v_{i(s-1)} [\text{SEP}]
\]

where:

\[
v_{ki} = \frac{\sum_{w \in \text{BertTok}(c_i, t_{ik}, \mathbf{l})} h_{w}^{L}}{\left| \text{BertTok}(c_i, t_{ik}, \mathbf{l}) \right|}, \quad k = 1, 2, \ldots, s - 1
\]

\( \text{BertTok}(c_i, t_{ik}, \mathbf{l}) \) represents the tokens that are obtained after tokenizing the sequence \( c_i t_{ik} \) using BERT tokenizer, and \( h_{w}^{L} \in \mathbb{R}^d \) is the contextualized embedding of dimension \( d \) from the \( L \)-th layer of BERT for the token \( w \in \text{BertTok}(c_i t_{ik}, \mathbf{l}) \). Similarly, the cell-wise average pooling is used to compute the contextualized embedding for the row-based sequence \( \tilde{r}_i \) denoted by \( \tilde{r}_i \).

We denote the column- and row-based contextualized embeddings that are obtained after BERT and cell-wise average pooling by \( \mathcal{C} = \{\tilde{c}_1, \tilde{c}_2, \ldots, \tilde{c}_l\} \) and \( \tilde{R} = \{\tilde{r}_1, \tilde{r}_2, \ldots, \tilde{r}_{(s-1)}\} \), respectively.

4.2.2 Horizontal and Vertical self-attention. Self-attention is incorporated into StruBERT for two reasons. First, the contextualized embeddings in \( \mathcal{C} \) capture independent column-level structural and textual information, and ignore the row-level dependency as a result of tabular structure. The same conclusion applies for \( \tilde{R} \) where column-level dependency is not captured in the row-level embeddings. Second, cell values are not equally important for the representations of rows and columns. We incorporate vertical self-attention [51] to operate over row-based embeddings to produce column embeddings, and we propose a horizontal self-attention that operates over column-based embeddings to form row embeddings. Both attentions are similar to the Transformer [44], and the naming of horizontal and vertical attention comes from the orientation of input sequences to attention blocks.

**Horizontal self-attention:** To capture the row-level dependency between the column-based contextualized embeddings of \( \mathcal{C} \), we propose a multi-head horizontal self-attention that operates on horizontally aligned tokens from the column-based embeddings as shown in Figure 2. The horizontal self-attention is formed of \( H \) layers of Transformers, and we use the output of the last layer as the row-level self-attention representation. We produce two types of features from the horizontal self-attention step after applying row-level average pooling. First, we obtain \( s - 1 \) row embeddings which can be seen as fine-grained structure- and context-aware features. Second, by averaging the [CLS] embedding from each column, we produce a row-guided [CLS] which represents a coarse-grained structure and context-aware feature. In conclusion, the horizontal self-attention features are based on interpreting the data table as a column-based structure, followed by row-level dependency.

**Vertical self-attention:** Similarly, a data table can be interpreted as a row-based structure, followed by column-level dependency. In this case, \( V \) layers of vertical self-attention [51] operate on the row-based contextualized embeddings of \( \tilde{R} \). We also obtain two types of features from the vertical self-attention. First, we obtain \( l \) fine-grained column embeddings by averaging the last output of the vertical self-attention over the vertically aligned tokens from the row-based embeddings. Second, we obtain a coarse-grained column-guided [CLS] embedding that interprets the data table as a row-based structure, followed by column-level dependency.

In conclusion, StruBERT generates four structure- and context-aware features: two fine-grained features which are the contextualized row and column embeddings, denoted by \( E_r \in \mathbb{R}^{(s-1) \times d} \) and \( E_c \in \mathbb{R}^{l \times d} \), respectively, and two coarse-grained features which are the row- and column-guided [CLS] embeddings, denoted by \( \mathcal{C}_{r} \in \mathbb{R}^{d} \) and \( \mathcal{C}_{c} \in \mathbb{R}^{d} \), respectively.

5 STRUBERT IN DOWNSTREAM TASKS

We integrate StruBERT as a feature extractor \( F \) into end-to-end architectures to solve table-related downstream tasks. In this section,
we address the tasks of table search and table matching, and we show how to map StruBERT features to a classification or retrieval score depending on the task.

5.1 Table Matching

In table matching tasks, both the query and the queried object are data tables. The neural ranking model should capture the semantic similarity between the structural and textual information of a table pair in order to predict the relevance score. To this end, we propose a Siamese [3]-based model that predicts the relevance score of a table pair \((T_i, T_j)\). In table matching, the textual information of each table contains only the metadata because the keyword-based query is absent. Structure- and context-aware features are extracted from each table using StruBERT:

\[
F(T_i, T_j) = (\text{StruBERT}(T_i), \text{StruBERT}(T_j))
\]

\[
F(T_i, T_j) = ((E_i^l, E_i^c, [CLS]_i^l, [CLS]_i^c), (E_j^l, E_j^c, [CLS]_j^l, [CLS]_j^c))
\]

After extracting features from each table using StruBERT, we obtain coarse- and fine-grained features for each table. We propose a ranking model that captures the semantic similarities within the fine-grained features \((E_i^l, E_j^l)\) and coarse-grained features \(([CLS]_i^l, [CLS]_j^l)\) and \(([CLS]_i^c, [CLS]_j^c)\).

5.1.1 Cross-matching of Fine-grained Features: To capture cross-matching signals of row and column embeddings for table pairs, we propose a model, called miniBERT, that operates directly on the embedding-level sequences of fine-grained features of StruBERT. miniBERT is composed of three trainable vectors \([\text{REP}]_c^i \in \mathbb{R}^d\), \([\text{REP}]_r^i \in \mathbb{R}^d\), and \([\text{SEP}] \in \mathbb{R}^d\), and 1 layer of Transformer blocks with 4 attention heads.\(^1\) The input to miniBERT for the column-based cross-matching of a table pair \((T_i, T_j)\) is shown in Figure 3. \([\text{REP}]_c^i\) is introduced to aggregate the matching signals between \(E_i^l\) and \(E_j^l\). We form the embedding-level sequence for the column embeddings of a table pair \((T_i, T_j):\)

\[
M_{c,c_j} = [\text{REP}]_c^i \oplus E_i^c \oplus [\text{SEP}] \oplus E_j^c^l
\]

where \([\text{SEP}]\) is used to separate \(E_i^c\) and \(E_j^c\). As in BERT, we sum three different embeddings to obtain the input embeddings to miniBERT. As shown in Figure 3, in addition to the column embeddings, the segment embeddings are used to indicate the column embeddings that belong to \(T_i\) and \(T_j\), and the position embeddings are used to encode the position of each vector in \(M_{c,c_j}\). The position embedding of \([\text{REP}]_c^i\) is in particular useful to indicate that the first position in the pair of the first column aggregates the embedding-level sequence \(M_{c,c_j}\). So, miniBERT takes the embedding-level sequence, that is formed by summing the column, segment and position embeddings, as input, then miniBERT outputs the hidden state of \([\text{REP}]_c^i\) from the Transformer block, denoted by miniBERT([\text{REP}]_c^i), that captures the bidirectional cross-attention between \(E_i^l\) and \(E_j^l\).

Similarly, we use miniBERT to compute the hidden state of \([\text{REP}]_r^i\), denoted by miniBERT([\text{REP}]_r^i), from the embedding-level

\(^1\)We tried to increase the number of layers and attention heads, but we did not notice an improvement in the reported evaluation metrics.

![Figure 3: Embedding-level sequence input of miniBERT for cross-matching of columns. The input to miniBERT is the sum of column, segment, and position embeddings. In this example, \(E_c^i \in \mathbb{R}^{3 \times d}\) is composed of \(c^i_k \in \mathbb{R}^d, k \in \{1, 2, 3\}\), and \(E_c^j \in \mathbb{R}^{4 \times d}\) is composed of \(c^j_k \in \mathbb{R}^d, k \in \{1, 2, 3, 4\}\).](image-url)
data table. We use the same notations of the table matching case, and we denote the outputs of StruBERT for a given query-table pair \((q, T_i)\) by: \(E_q^r, E_q^c, [CLS]^r_q, [CLS]^c_q\). We apply miniBERT to the single embedding-level sequences defined by:

\[
\begin{align*}
M_{r,q} &= [REP]^r_q \odot E_q^r(q) \odot [SEP] \\
M_{c,q} &= [REP]^c_q \odot E_q^c(q) \odot [SEP]
\end{align*}
\]  

(12)

where \(E_q^r\) and \(E_q^c\) are functions of \(q\) because \(q \in T_{eq}\) in the case of keyword-based table retrieval. We use the final hidden states of \([REP]^r\) and \([REP]^c\), that are obtained from miniBERT as the row- and column-based aggregate for the query-table pair \((q, T_i)\), respectively. A query-table pair \((q, T_i)\) is represented using four feature vectors: row and column outputs from miniBERT and row- and column-guided [CLS] embeddings \([CLS]^r_q, [CLS]^c_q\). We concatenate these features to obtain the final representation for \((q, T_i)\), which is used as input to a linear layer to predict the relevance score of the query-table pair \((q, T_i)\).

6 Evaluation

6.1 Data Collections

6.1.1 WikiTables. This dataset is composed of the WikiTables corpus [1] containing over 1.6M tables. Each table has five indexable fields: table caption, attributes, data rows, page title, and section title. We use the same test queries that were used by Zhang and Balog [55]. For the ground truth relevance scores of query-table pairs, every pair is evaluated using three numbers: 0 means irrelevant, 1 means partially relevant, and 2 means relevant. There are 60 queries in the WikiTables collection, and 3117 query-table pairs.

In addition to the keyword-based table retrieval, we adapt WikiTables for the table similarity. As in TabSim [16], we iterate over all the queries of WikiTables, and if two tables are relevant to a query, the table pair is given a label 1. On the other hand, an irrelevant table to a query is considered not similar to all tables that are relevant to the query, and therefore the table pair is given a label 0.

6.1.2 PMC. Habibi et al. [19] proposed a table corpus that is formed from PubMed Central (PMC) Open Access subset, and used for evaluation on the table similarity task. This collection is related to biomedicine and life sciences. Each table contains a caption and data values. The table pairs are annotated for binary classification by comparing the caption and data values of each table. A table pair is given a label dissimilar if both the caption and data values are labeled dissimilar, otherwise the table pair is given the label similar. In the PMC table corpus, there are 1391 table pairs, where 542 pairs are similar and 849 pairs are dissimilar.

6.1.3 Query by Example Data. Zhang and Balog [57] proposed a query by table dataset that is composed of 50 Wikipedia tables used as input queries. The query tables are related to multiple topics, and each table has at least five rows and three columns. For the ground truth relevance scores of table pairs, each pair is evaluated using three numbers: 2 means highly relevant and it indicates that the queried table is about the same topic of the query table with additional content, 1 means relevant and it indicates that the queried table contains a content that largely overlaps with the query table, and 0 means irrelevant. The total number of table pairs is 2850.

6.2 Baselines

6.2.1 Keyword-based Table Retrieval. For evaluation on the keyword-based table retrieval, we compare against the following baselines:

**MultiField-BM25:** In a multi-field ranking scenario, a table is defined using multiple fields. MultiField-BM25 combines BM25 [34] scores for multi-field tables.

**STR** [55]: Multiple embedding-based features are computed for a table and query, then different matching strategies are used to generate the ranking features from the embeddings. A random forest is used to predict the relevance score of a query-table pair.

**BERT-Row-Max** [8]: The [CLS] embedding of the sequence formed from the query and table is used to predict the relevance score of a query-table pair.

**DSRM** [40]: A joint model that captures both semantic and relevance matching signals from a query-table pair to predict a real-valued relevance score.

**TabBERT** [51]: A model that is originally proposed for semantic parsing over tables. We use the embedding of the [CLS] token from the last layer of the vertical self-attention as input to a MLP layer.

Due to limited computational resources, we use the BERT-base-cased for our method StruBERT, and for BERT-based baselines which are BERT-Row-Max and TabBERT. We note that the original paper of BERT-Row-Max uses the BERT-large-cased.

6.2.2 Table Matching. For evaluation in table matching, we compare against the following baselines:

**Embedding+MLP:** A table is flattened and concatenated with the metadata to form a single document for each table. Then, the mean of word embeddings using Glove [32] is calculated for each table. The final ranking feature is computed using pointwise multiplication between the embeddings of tables, then forwarded to a MLP layer to predict the relevance score.

**TF-IDF+MLP:** TableRank [24] computes Term Frequency-Inverse Document Frequency (TF-IDF) for tables. The TF-IDF score is computed using the metadata and values of a given table, instead of the document that contains the table. A MLP layer is used instead of the cosine similarity to predict the semantic matching score.

**TabSim** [16]: Two separate neural network models are introduced to form the representations of a table: one model extracts the embedding from the caption, and a second model extracts column embeddings from the data values.

**TaBERT** [51]: A TaBERT-based Siamese model is used to evaluate the semantic similarity between tables. For a given table, we extract the [CLS] embedding obtained from applying the vertical self-attention over the row-level sequences of the table. Then, point-wise multiplication is applied between the [CLS] embeddings of both tables, and the resulting vector is forwarded to a MLP layer to predict the table matching score.

**StruBERT (KP):** This baseline is a variation of our method that uses a kernel pooling (KP) [47]-based ranking model on top of StruBERT features. KP is the main component of strong ranking models [14, 47], and we adapt KP for cross-matching of fine-grained features. We construct the interaction matrices \(I = \{I_{r,i}, I_{c,j}\}\), where \(I_{r,i}\) and \(I_{c,j}\) denote the interactions of \(E_q^r - E_q^r\) and \(E_q^c - E_q^c\), respectively, and the elements of each matrix are computed using cosine similarity between the embeddings of the corresponding rows and columns. To summarize each interaction matrix into a
fixed-length feature vector, we use KP to extract soft-match signals between different fields of $T_i$ and $T_j$.

**StruBERT (CNN)**: This baseline is a variation of our method that uses Convolutional Neural Networks (CNN) on top of StruBERT features. This baseline is based on the interaction tensor, denoted by $S$, which is computed using pointwise multiplication between pairwise column (row) embeddings of a table pair. Inspired by DeepRank [31], we use one layer of CNN filters with all possible combinations of widths and heights that are applied to $S$:

$$h^{|x|}_{i,j} = \sum_{s=1}^{y} \left( \sum_{l=1}^{d} w_{s,l}^{(i,|x|)} S^{(l)}_{i+s,j+l} + b_{s,l}^{(i,|x|)} \right), \ k = 1, \ldots, K$$  \hspace{1cm} (13)

where $y$ is the maximum size of a CNN filter, $S^{(l)}_{i+s,j+l}$ is a $s \times t$ matrix from the $l$-th channel of $S$ starting from $i$-th row and $j$-th column, $K$ is the total number of CNN filters, and $w_{s,l}^{(i,|x|)}$ and $b_{s,l}^{(i,|x|)}$ are parameters of CNN. Then, we keep only the most significant matching signal from each feature map to form a single vector.

### 6.3 Experimental Setup

Our model is implemented using PyTorch, with two Nvidia GeForce GTX 1080. For keyword- and content-based table retrieval, the parameters of our model are updated using a mean square error pointwise multiplication between predicted and groundtruth relevance scores, and for table similarity, we use the cross-entropy loss function. The dimension $d$ is equal to 768. The number of Transformer layers $H$ and $V$ in the horizontal and vertical self-attention, respectively, are equal to 3. In StruBERT, BERT-base-cased and the vertical self-attention are initialized using TaBERT$_{Base}$ ($K = 3$)\(^2\) which is pretrained using content snapshots with 3 rows. Such pretraining requires high-memory GPUs that are not currently possessed by our team; therefore we randomly initialize the horizontal self-attention so that the row-based dependencies are only captured during fine-tuning on the target dataset. We expect an increase in the results with pretraining the horizontal self-attention on similar task to the Masked Column Prediction (MCP) from TaBERT$^{[51]}$ (in our case, the pretraining task should be a Masked Row Prediction). We leave pretraining the horizontal self-attention as a future direction.

\(^2\)https://github.com/facebookresearch/TaBERT

### 6.4 Experimental Results

We report results using five-fold cross validation. For keyword-based table retrieval, we use the same splits as Chen et al. [8] to report results on five-fold cross validation for our method and baselines. We evaluate the performance of our proposed method and baselines on the keyword- and content-based table retrieval tasks using Normalized Discounted Cumulative Gain (NDCG) [21], Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP). We evaluate the performance of our method and baselines on the table similarity task using macro-averaged precision (P), recall (R) and F-score, and accuracy of predictions on the testing set. To test significance, we use the paired Student’s t-test and write $\dagger$ to denote significance at the 0.05 level over all other methods.

#### 6.4.1 Table Similarity Results

Table 1(a) shows the performance of different approaches on the PMC collection. Given that table similarity is an instance of table matching, StruBERT features are used based on the steps that are described in Section 5.1. We show that our proposed method StruBERT outperforms the baselines for all evaluation metrics. By incorporating the structural and textual features into a cross-matching based ranking model, we were able to capture the semantic similarity between tables both in term of tabular content and metadata, and this leads to an increase in evaluation metrics compared to baselines that either ignore the structural

#### Table 1: Table similarity results using a classification threshold equal to 0.5.

| Method Name       | Macro-P | Macro-R | Macro-F | Accur. |
|-------------------|---------|---------|---------|--------|
| Tfidf+MLP         | 0.7834  | 0.6735  | 0.6529  | 0.6951 |
| Embedding+MLP     | 0.8496  | 0.7710  | 0.7736  | 0.7931 |
| Tfidf+Embedding+MLP | 0.8736  | 0.8381  | 0.8447  | 0.8506 |
| TaSim [19]        | 0.8865  | 0.8545  | 0.8613  | 0.8705 |
| TaBERT [51]       | 0.9109  | 0.9024  | 0.9055  | 0.9067 |
| StruBERT (fine)   | 0.9208  | 0.9058  | 0.9104  | 0.9124 |
| StruBERT (coarse) | 0.9276  | 0.9154  | 0.9194  | 0.9210 |
| StruBERT (KP)     | 0.9148  | 0.9060  | 0.9091  | 0.9109 |
| StruBERT (CNN)    | 0.9293  | 0.9164  | 0.9205  | 0.9224 |
| StruBERT          | 0.9321$\dagger$ | 0.9284$\dagger$ | 0.9300$\dagger$ | 0.9310$\dagger$ |

(a) PMC

| Method Name       | Macro-P | Macro-R | Macro-F | Accur. |
|-------------------|---------|---------|---------|--------|
| Tfidf+MLP         | 0.6256  | 0.5022  | 0.3559  | 0.5378 |
| Embedding+MLP     | 0.8429  | 0.8419  | 0.8423  | 0.8433 |
| Tfidf+Embedding+MLP | 0.8632  | 0.8554  | 0.8574  | 0.8594 |
| TaSim [19]        | 0.8480  | 0.8458  | 0.8466  | 0.8478 |
| TaBERT [51]       | 0.9696  | 0.9626  | 0.9649  | 0.9653 |
| StruBERT (fine)   | 0.9850  | 0.9852  | 0.9851  | 0.9852 |
| StruBERT (coarse) | 0.9838  | 0.9816  | 0.9825  | 0.9826 |
| StruBERT (KP)     | 0.9733  | 0.9713  | 0.9722  | 0.9724 |
| StruBERT (CNN)    | 0.9782  | 0.9737  | 0.9753  | 0.9756 |
| StruBERT          | 0.9945$\dagger$ | 0.9938$\dagger$ | 0.9941$\dagger$ | 0.9942$\dagger$ |

(b) PMC

#### Table 2: Content-based table retrieval results on the query by example dataset [57].

| Model         | NDCG@5 | MRR   | MAP   |
|---------------|--------|-------|-------|
| BM25          | 0.5369 | 0.5832| 0.5417|
| DSRMM [40]    | 0.5768 | 0.6193| 0.5914|
| TabSim [19]   | 0.5739 | 0.6056| 0.5932|
| TaBERT [51]   | 0.5877 | 0.6120| 0.5942|
| StruBERT (fine)| 0.6015| 0.6419| 0.6091|
| StruBERT (coarse)| 0.6140| 0.6478| 0.6142|
| StruBERT (KP) | 0.5990 | 0.6200| 0.5959|
| StruBERT (CNN)| 0.6177 | 0.6378| 0.6179|
| StruBERT      | 0.6345$\dagger$ | 0.6601$\dagger$ | 0.6297$\dagger$ |
information or treat the textual and structural information separately. Considering the table as a single document in TF-IDF and embedding baselines lead to the lowest results, which indicates that the structural similarity between tables is an important factor in table similarity. The results on this dataset show a clear advantage from using embedding-based features (traditional or contextualized) compared to term frequency features that are based on exact matching. StruBERT (fine) and StruBERT (coarse) show ablation study results for predicting semantic similarity using only fine- and coarse-grained features, respectively. By combining both categories of features, we achieve higher evaluation metric results.

Table 1(b) shows the performance of the different approaches on the WikiTables. Consistent with PMC, our results on the WikiTables show the importance of the structure- and context-aware features in improving the table similarity prediction. Table similarity results on WikiTables and PMC show that StruBERT achieves significant improvements in the evaluation metrics of two data table collections from different domains, which supports the generalization characteristic of our proposed method.

6.4.2 Content-based Table Retrieval Results. Table 2 shows the content-based table retrieval results. Given that content-based table retrieval is an instance of table matching, StruBERT features are used based on the steps that are described in Section 5.1. The StruBERT model that combines fine- and coarse-grained features achieves a 7.96% improvement in terms of NDCG@5 upon TaBERT that only uses the [CLS] embedding obtained from the vertical self-attention. We also report ablation study results where we predict the semantic similarity between tables using only fine- or coarse-grained features. Both categories of features lead to better retrieval results than the baselines, and by combining both the fine- and coarse-grained features, we capture the textual and structural similarities between tables. An important step in table similarity assessment is the order-invariant cross-matching between columns (rows) of tables which is satisfied using miniBERT as a ranking model on top of StruBERT features.

Our approach uses a novel miniBERT model to map StruBERT features to a relevance score. We investigate the performance of alternative ranking models when given StruBERT features. Table 2 shows the results of comparing miniBERT against StruBERT (KP) and StruBERT (CNN) in the case of content-based table retrieval. miniBERT outperforms both baselines in all evaluation metrics. Kernel pooling summarizes the one-to-one matching signals computed in the interaction matrix to a single feature vector. So, StruBERT (KP) does not solve the case of many-to-many matching of rows or columns. On the other hand, by applying CNN to the interaction tensor, we capture the semantic similarity between multiple columns (rows) from a table pair, so StruBERT (CNN) captures the many-to-many matching signals. However, the convolution operation captures local interactions, and is not permutation invariant in the sense that rows and columns could be arbitrarily ordered without changing the meaning of the table. miniBERT deals with both many-to-many matching and the permutation invariant property by taking advantage of self-attention heads.

6.4.3 Keyword-based Table Retrieval Results. Table 3 shows the performance of different approaches on the WikiTables collection. In keyword-based table retrieval, StruBERT features are used based on the steps that are described in Section 5.2. We show that our proposed method StruBERT outperforms the baselines for all evaluation metrics. By adding the query to the textual information of a given table, we obtain fine- and coarse-grained features that capture early interactions between the query, and the structural and textual information of a table.

For the ablation study of the keyword-based table retrieval, we notice that summarizing the table and query using the [CLS] token in BERT-Row-Max, TaBERT, and StruBERT (coarse) leads to better results than fine-grained features of StruBERT (fine). This means that there are more natural language queries in the keyword-based table retrieval for WikiTables collection that are relevant to a high level summary of the textual and structural information than the specific details captured by rows and columns. After combining the fine- and coarse-grained features for all query-table pairs, StruBERT captures the semantic similarity between the query and the textual information, and the query and the structural information defined by rows and columns, and this leads to the best retrieval metrics.

7 CONCLUSIONS

We have shown that a structure-aware model should augment the vertical self-attention with our novel horizontal self-attention to more accurately capture the structural information of a table, and that when we combine this with textual information using a BERT-based model, the resulting StruBERT system outperforms the state-of-the-art results in three table-related tasks: keyword- and content-based table retrieval, and table similarity. StruBERT embeddings are integrated into our miniBERT ranking model to predict the relevance score between a keyword-based query and a table, or between a table pair. Despite being a general model, StruBERT outperforms all baselines on three different tasks, achieving a near perfect accuracy of 0.9942 on table similarity for WikiTables, and improvement in table search for both keyword- and table-based queries with up to 7.96% increase in NDCG@5 score for content-based table retrieval. An ablation study shows that using both fine- and coarse-grained features achieves better results than either set alone. We also demonstrate that using miniBERT as a ranking model for StruBERT features outperforms other common ranking models.

Acknowledgments

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

miniBERT is composed of one layer of Transformer blocks with four attention heads. To better understand how miniBERT works, we show the attention heads that correspond to a table similarity case. The first table is composed of the headers Club and City/Town, and the second table is composed of the headers Team, Location, Stadium, and Coach. Figure 4 illustrates the four attention heads of miniBERT. Figure 4(a) indicates that the 1st attention head focuses on the header Location from the second table which attends mainly to the header City/Town from the first table and contributes significantly to the embedding \([\{REP\}_c]\). The second attention head, illustrated in Figure 4(b), is more general as it indicates multiple cross matching signals between columns of both tables. The third attention head in Figure 4(c) is similar to the 1st attention head with more focus on the header Stadium from the second table. This can be explained by the co-occurrence of the header Stadium with headers Club and City/Town. We also observe similar patterns in the 4th attention head that focuses mainly on the header Coach. The analysis of the attention heads shows the advantage of using the Transformers blocks to capture the many-to-many relationships between columns of tables by aggregating information both within and across table columns.

[49] Yang Yi, Zhiya Chen, Jeff Heflin, and Brian D Davison. 2018. Recognizing quantity names for tabular data. In Proceedings of the International Workshop on Web Document Analysis (WDA 2001). 31–34.

[50] Zeynep Akkalyoncu Yilmaz, Shengjin Wang, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Applying BERT to Document Retrieval with Birch. In Proceedings of the International Workshop on Web Document Analysis (WDA 2001). 31–34.

[51] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TabBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, 8413–8426.

[52] Manoru Yoshida and Kentaro Torisawa. 2001. A method to integrate tables of the World Wide Web. In Proceedings of the International Workshop on Web Document Analysis (WDA 2001). 31–34.