MGNETS: Multi-Graph Neural Networks for Table Search

Zhiyu Chen
zhc415@lehigh.edu
Lehigh University
Bethlehem, PA, USA

Mohamed Trabelsi
mot218@lehigh.edu
Lehigh University
Bethlehem, PA, USA

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

Dawei Yin
yindawei@acm.org
Baidu Inc.
Beijing, China

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

ABSTRACT

Table search aims to retrieve a list of tables given a user's query. Previous methods only consider the textual information of tables and the structural information is rarely used. In this paper, we propose to model the complex relations in the table corpus as one or more graphs and then utilize graph neural networks to learn representations of queries and tables. We show that the text-based table retrieval methods can be further improved by graph-based predictions which fuse multiple field-level information.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; **Structured text search**.

KEYWORDS

table search; neural networks; information retrieval

ACM Reference Format:
Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Dawei Yin, and Brian D. Davison. 2021. MGNETS: Multi-Graph Neural Networks for Table Search. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482140

1 INTRODUCTION

A massive number of tables extracted from the Web have been used in various research tasks such as question answering [4], entity linking [3, 23] and table augmentation [2, 5, 10, 21]. Previous table search methods [6–8, 16, 17, 22] either treat a table as a regular or multifield document. The structure of a table is underutilized. Chen et al. [7] slice a table into smaller pieces and select only the most salient ones for final ranking. However, the structure information across different tables is missed. As a structured document, a table itself can be naturally viewed as a graph. The words appearing in the same column or row usually have certain relationships. Such relationships can hardly be captured by models which treat the table as flat texts. As shown in Figure 1, we know that "Euro" and "United States Dollars" are two different types of currencies since they appear in the same column "Currency". If the table is flattened into a sequence, the existing methods [7, 14, 22] for text retrieval will neglect the semantic relationship between table attributes/headers and table values.

Page Title: List of circulating currencies

Section Title: List of circulating currencies by state or territory

Caption: The list denotes the circulating currencies by all countries and territories across the world.

Table T_i:

State or territory	Currency	ISO code
Austria	Euro	EUR
Germany	Euro	EUR
United States	United States dollar	USD
...	...	...

Figure 1: An example of a Web table with page title and section title as context fields.

In this work, we propose a graph neural network-based method for ad hoc table retrieval. By explicitly modeling the table corpus as one or more graphs, we directly encode the structural information of the table which is often ignored by previous methods. Our principal contributions are: We propose a novel multi-graph neural network method for ad hoc table retrieval. We propose table-based pointwise mutual information (TPMI) to calculate the semantic correlation of terms in the table corpus. The experimental results demonstrate that our proposed method outperforms baselines and can improve the performance of previous sequence-based methods.

2 METHOD

In this section, we introduce the details of our proposed ***M**ulti-**G**raph **NE**ural networks for **T**able **S**earch* (MGNETS for short).

2.1 Problem Statement

In the task of table search, given a query q usually consisting of several keywords $q = \{k_1, k_2, ..., k_l\}$, our goal is to rank a set of tables $D = \{T_1, T_2, ..., T_n\}$ in descending order of their relevance scores with respect to q. Each table T_i has corresponding context fields $\{p_{i1}, ..., p_{ik}\}$. A data table is a set of cells arranged in rows

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11…$15.00
https://doi.org/10.1145/3459637.3482140

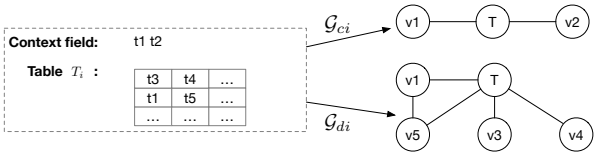

Figure 2: An example of constructed graphs from a table.

and columns like a matrix. Each cell could be a single word, a real number, a phrase or even sentences. The first row of a table is the header row and consists of header cells. The context fields associated with a table instance depend on the source of the dataset. For example, a table from Wikipedia usually has caption, page title, and section title as context fields as shown in Figure 1.

2.2 Graph Construction

To construct the graph $\mathcal{G}$ for table search, we first need to define the nodes set $\mathcal{V}$ and edges set $\mathcal{E}$ of the graph. We have two types of nodes in the graph. Every unique term that appears in the data tables or context fields is represented as a **term node** $v_t \in \mathcal{V}$. Every data table has a corresponding **table node** $v_T \in \mathcal{V}$. We add an edge between two nodes if they have a co-occurrence relationship. We list the possible types of edges below.

Table-Term Edges. A term node can be from either a table or a context field of a table. A table-term edge $(v_T, v_t) \in \mathcal{E}$ is constructed if the term t occurs in the table T or any context fields of table T. We can also treat queries in the training set as another context field.

Term-Term Edges. In previous work of applying graph neural networks for text classification [13, 20], a fixed size sliding window is applied on all documents in the corpus and the pointwise mutual information (PMI) between two terms is calculated to determine the corpus-level co-occurrence. Unlike a text document, a data table has a non-linear structure so that defining the co-occurrence relationship using a fixed size sliding window and calculating the PMI is not directly applicable. Instead of using a fixed size sliding window, we treat every column or every row as the sliding window. For a data table with r_i rows and c_i columns, there are $r_i + c_i$ context windows. Now we define the **table-based pointwise mutual information (TPMI)** score for term pair t_i and t_j:

$$TPMI(t_i, t_j) = log\frac{P(t_i, t_j)}{P(t_i)P(t_j)} = log\frac{C(t_i, t_j) \times C}{C(t_i)C(t_j)} \tag{1}$$

where $C = \sum_{i=1}^{n}(r_i + c_i)$ corresponding to the total number of sliding windows in all data tables, $C(t_i)$ is the number of rows and columns containing t_i and $C(t_i, t_j)$ is the number of rows and columns containing both t_i and t_j. A positive TPMI score indicates two terms are semantically correlated. We add (t_i, t_j) into $\mathcal{E}$ if two table terms t_i and t_j have a positive TPMI score.

Multi-graph Construction. We can construct one heterogeneous graph which contains all types of edges mentioned above. However, the semantic relation of a term pair in one field does not imply the relation in another field and constructing a graph with all possible edges could result in ambiguous semantic meanings. Therefore, we construct multiple subgraphs and each subgraph captures certain semantic relations among nodes. In this paper, we construct two subgraphs $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$ and $\mathcal{G}_c = (\mathcal{V}_c, \mathcal{E}_c)$:

$$\mathcal{V}_d = \{\mathcal{V}_T \cup \mathcal{V}_{dt}\} \qquad \mathcal{V}_c = \{\mathcal{V}_T \cup \mathcal{V}_{ct}\}$$
$$\mathcal{E}_d = \{(v_i, v_j) | (v_i, v_j) \in \mathcal{E}, v_i, v_j \in \mathcal{V}_d\}$$
$$\mathcal{E}_c = \{(v_i, v_j) | (v_i, v_j) \in \mathcal{E}, v_i, v_j \in \mathcal{V}_c\}$$

where $\mathcal{V}_T$ is the set of nodes representing data tables, $\mathcal{V}_{dt}$ represents all term nodes in data tables, and $\mathcal{V}_{ct}$ represents the nodes constructed from context fields. The first subgraph contains term nodes from data tables while the second subgraph contains term nodes from context fields. An example of constructed graphs $\mathcal{G}_{ci}$ and $\mathcal{G}_{di}$ from table T_i is shown in Figure 2. $\mathcal{G}_d$ can be constructed by merging all $\mathcal{G}_{di}$ for $T_i \in D$. $\mathcal{G}_c$ can be constructed in a similar way. Note that table nodes are implicitly connected by term nodes.

2.3 Multi-graph Encoder

After obtaining the constructed graphs $\mathcal{G}_d$ and $\mathcal{G}_c$, we can employ graph neural networks to learn the embeddings of nodes. The representation learning process of GNN models can be divided into two steps: neighborhood aggregation and combination [19]. The k-th layer of a GNN model is:

$$h_v^{(k)} = \phi^{(k)}(h_v^{(k-1)}, \psi^{(k)}(\{h_u^{\{k-1\}} : u \in \mathcal{N}(v)\})) \tag{2}$$

where $h_v^{(k)}$ denotes the feature for node $v \in \mathcal{V}$ at k-th layer, $\mathcal{N}(v)$ denotes the neighborhood nodes of $v \in \mathcal{V}$. $\psi^{(k)}$ is the aggregation function at k-th layer which aggregates the representations of v's neighbors. $\phi^{(k)}$ generates the node representation of v at the k-th layer by combining its representation from the previous layer and aggregated neighbor representations. For different GNN models, the aggregate/combine functions are different [12, 15, 18]. In this paper, we use Graph Isomorphism Network (GIN) [19] as the multi-graph encoder to learn node embeddings in $\mathcal{G}_d$ and $\mathcal{G}_c$:

$$h_d^{(k)} = MLP_d^{(k)}((1 + \epsilon_d^{(k)})h_d^{(k-1)} + \sum_{u \in \mathcal{N}(v_d)} h_u^{(k-1)}) \tag{3}$$

$$h_c^{(k)} = MLP_c^{(k)}((1 + \epsilon_c^{(k)})h_c^{(k-1)} + \sum_{u \in \mathcal{N}(v_c)} h_u^{(k-1)}) \tag{4}$$

where $\epsilon_d^{(k)}$ and $\epsilon_c^{(k)}$ are learned parameters. Multi-layer perceptrons (MLPs) are used to obtain features at k-th layer from aggregated node embeddings. We initialize the node embeddings $h_d^{(0)} \in \mathbb{R}^{|\mathcal{V}_d|}$ and $h_c^{(0)} \in \mathbb{R}^{|\mathcal{V}_c|}$ with one-hot encoding features for $v_d \in \mathcal{V}_d$ and $v_c \in \mathcal{V}_c$ respectively. After stacking l layers of GNN models, we obtain $l + 1$ embeddings for each node in $\mathcal{G}_d$ and $\mathcal{G}_c$.

2.4 Pooling Layer

The community has been studying how to design readout functions at the node level for node classification and graph level for graph classification [12, 15, 18, 19]. In this section, we propose to learn query and table representations from node embeddings obtained from Section 2.3.

Given a query table pair (q, T_i) and $\mathcal{G}_d$, we can find the set of nodes $\mathcal{V}_q \subseteq \mathcal{V}_d$ and $\mathcal{V}_{ti} \subseteq \mathcal{V}_d$ constructed from query $q \in Q$ and table T_i respectively. To learn the representation of q and T_i in $\mathcal{G}_d$, we can use the graph-level readout function which has been previously used in graph classification. However, to learn the

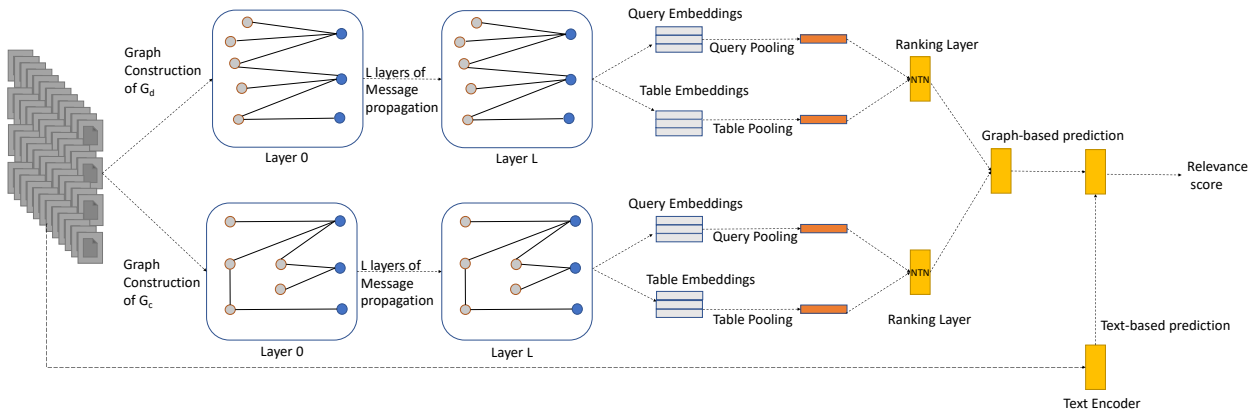

Figure 3: An illustration of the overall framework. From the corpus we construct two different graphs. After the message propagation through the multi-graph encoder, we learn query and table representations from each graph. We obtain the final ranking score from graph-based prediction and also the text-based prediction from a text encoder.

representation of a query or a table at k-th layer, we operate on the subsets of nodes instead of all the nodes:

$$h_{qd}^{(k)} = Sum(\{h_v^{(k)} : v \in \mathcal{V}_q\}), \quad h_{ti}^{(k)} = Sum(\{h_v^{(k)} : v \in \mathcal{V}_{ti}\}) \quad (5)$$

In experiments not shown here, we find that summation of node embeddings is more effective than mean pooling or max pooling. Equation (5) does not require additional parameters which makes it computationally efficient. Therefore, we use it for the initial node embeddings which are one-hot encoding features. Inspired by SimGNN [1], we use the following attention-based pooling function to learn query/table representations for other layers ($k > 0$):

$$h_{qd}^{(k)} = H_{qd}^{(k)\top} \cdot \sigma(H_{qd}^{(k)} \cdot Mean(H_{qd}^{(k)} W_1^{(k)})) \quad (6)$$

$$h_{ti}^{(k)} = H_{ti}^{(k)\top} \cdot \sigma(H_{ti}^{(k)} \cdot Mean(H_{ti}^{(k)} W_1^{(k)})) \quad (7)$$

where $H_{qd}^{(k)} \in \mathbb{R}^{|\mathcal{V}_q| \times d}, H_{ti}^{(k)} \in \mathbb{R}^{|\mathcal{V}_{ti}| \times d}$ denotes node embeddings of the query and the table respectively; $Mean(\cdot)$ calculates the average embedding after the node embeddings are transformed by a trainable weight $W_1^{(k)} \in \mathbb{R}^{d \times d}$. Note that $W_1^{(k)}$ is a shared parameter. $\sigma(\cdot)$ is the sigmoid function and its output can be considered as the weights for query/table nodes so that the final representation is the weighted sum of all node embeddings.

Given the context fields of T_i and $q \in Q$, we obtain the query representation $h_{qc}^{(k)}$ and context field representation $h_{ci}^{(k)}$ from $\mathcal{G}_c$ in a similar way based on Equations (5 - 7).

2.5 Ranking Layer

In Section 2.4, we generate the query representation ($h_{qd}^{(k)}$ and $h_{qc}^{(k)}$), table representation ($h_{ti}^{(k)}$) and context field representation ($h_{ci}^{(k)}$) from each layer of the multi-graph encoder with a pooling layer. In order to predict the final relevance score of a given query table pair (q, T_i), we first generate the single-graph predictions where each uses different graph information. In the following, we show how to generate the prediction based on features learned from $\mathcal{G}_d$. First, the query representation and table representation are fed into two separate multi-layer perceptrons (MLP):

$$h_{qd}^{\prime(k)} = MLP_{qd}^{(k)}(h_{qd}^{(k)}), \quad h_{ti}^{\prime(k)} = MLP_{ti}^{(k)}(h_{ti}^{(k)}) \quad (8)$$

Then a neural tensor network (NTN) is used to generate the prediction at k-th layer (Equation (9)).

$$y_d^{(k)} = h_{qd}^{\prime\mathbf{T}(k)} W_2^{(k)} h_{ti}^{\prime(k)} + W_3^{(k)} \begin{bmatrix} h_{qd}^{\prime(k)} \\ h_{ti}^{\prime(k)} \end{bmatrix} + b_1^{(k)} \quad (9)$$

We use a linear layer to combine the predictions from all layers.

$$y_d = [y_d^{(0)}; ...; y_d^{(l)}] W_4 + b_2 \quad (10)$$

The prediction y_c based on the embeddings of $\mathcal{G}_c$ is similar to the steps in Equations (8-10). Another linear layer is used to combine the predictions from different graphs: $y_g = [y_d; y_c] W_5 + b_3$. Our method can be easily extended to have more than two graphs. For example, we can build separate graphs for each context field and combine predictions for each graph. For the purpose of explanation, we construct only one graph for all the context fields.

We can also predict the relevance score between q and T_i only based on text information: $y_t = TextEncoder(q, text_i)$ where $TextEncoder(\cdot)$ can be any previous table retrieval method which treats tables as text documents and $text_i$ is the text representation of a table(e.g. the concatenation of terms in T_i and its context fields). We obtain the final relevance score y by combining the graph-based and text-based predictions with a linear transformation: $y = [y_t; y_g] W_6 + b_4$.

2.6 Model Training

To learn the parameters, we optimize the model with pointwise mean square loss as in [7, 22]: $\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y_i})^2 + \beta \cdot ||\Theta||^2$ where y_i is the prediction from our model and $\hat{y_i}$ is the ground truth for i-th training sample. Θ denotes all trainable parameters and β controls the L2 normalization which can affect overfitting.

3 RESULTS & ANALYSIS

3.1 Experimental Settings

Dataset. To evaluate the performance of MGNETS, we utilize the WikiTables benchmark [22] which includes 1.6M tables extracted from Wikipedia articles and each table has three context fields: page title, section title and caption. We also treat table headers as an additional context field. We evaluate the results using NDCG@5,

Table 1: Performance comparison with baselines. The superscript † denotes statistically significant improvements over all other methods.

Model	MAP	P@5	NDCG@5	NDCG@10
Multi-field BM25	0.4596	0.3273	0.4365	0.5049
STR	0.5711	0.3927	0.5762	0.6048
ConvKNRM	0.5561	0.3800	0.5556	0.5901
MGNETS-Conv	0.5912 (+6.3%)	0.3907 (+2.8%)	0.5910 (+6.4%)	0.6168 (+4.5%)
BERT-Row-Max	0.6146	0.4080	0.6167	0.6322
MGNETS-BERT	**0.6339** (+3.1%)	**0.4180**† (+2.5%)	**0.6373**† (+3.3%)	**0.6490**† (+2.7%)

NDCG@10, P@5 and MAP. All results are tested for statistical significance using the paired Student's t-test at 95% confidence.

Parameter Settings. We implement our models using Pytorch and DGL[1]. The node embedding size is fixed to 50. For all multi-layer perceptrons, we use two layers. The Adam optimizer [11] is used to optimize all models. Five-fold cross validation is used to obtain the final evaluation metrics and we use the same splits as [7] for fair comparison. All the models are trained with 100 epochs. In terms of other hyperparameters, we apply a grid search method here: learning rate is searched in $[1e^{-6}, 3e^{-6}, 1e^{-4}, 3e^{-4}, 1e^{-2}, 3e^{-2}]$, and L2 normalization coefficient is tuned in $[5e^{-6}, 5e^{-5}, 5e^{-4}, 5e^{-3}]$. We only use one layer of GIN in the multi-graph encoder.

Baselines. We compare to the following: **Multi-field BM25** [14], which is an unsupervised method that treats tables as multifield documents and ranks tables with combined BM25 scores from each field; **STR** [22], which proposes multiple embedding-based features and different strategies to generate ranking features from those embeddings. A random forest fits the ranking features in a point-wise manner; **Conv-KNRM** [9] where convolutional neural networks are used to learn n-gram soft matching signals between queries and documents; and, **BERT-ROW-MAX** [7], the previous state-of-the-art method using BERT as the backbone. It selects the most important rows with max salience selector and concatenates them with context fields as BERT input. Due to limited computational resources, we use the BERT-base-cased[2] instead of BERT-large-cased as in the original paper to initialize the BERT component. We use Conv-KNRM or BERT-ROW-MAX as the text encoder in MGNETS, named as MGNETS-Conv and MGNETS-BERT respectively.

3.2 Results and Discussion

Overall Performance. We start by comparing the performance of MGNETS with all other baselines, as reported in Table 1. We can observe that our proposed **MGNETS-BERT** achieves the best performance across all evaluation metrics. As a strong interaction-based neural IR model, ConvKNRM underperforms the feature-based STR model, which indicates that the table retrieval task should not be treated as a traditional document retrieval task. Combining both text-based predictions and graph-based predictions, MGNETS-BERT outperforms BERT-ROW-MAX by 2.5 − 3.3% and MGNETS-Conv outperforms ConvKNRM by 2.8 − 6.4%. The results verify the effectiveness of our designed framework where the multi-graph information can benefit methods where only text information is used. However, we do not claim our method to be the new state-of-the-art method, since our goal is to study whether the learned graph features can provide additional signals. It is possible that combining other designed features can further improve the performance such as in Chen et al. [7], which is beyond the scope of this paper.

[1]https://github.com/dmlc/dgl
[2]https://huggingface.co/bert-base-cased

Table 2: Ablation study of our framework.

Model	MAP	P@5	NDCG@5	NDCG@10
MGNETS-BERT	0.6339	0.4180	0.6373	0.6490
MGNETS (graph only)	**0.5812**	**0.3913**	**0.5823**	**0.6072**
GNETS ($\mathcal{G}$)	0.5753	**0.3913**	0.5748	0.6065
GNETS ($\mathcal{G}_c$)	0.5634	0.3820	0.5657	0.6003
GNETS ($\mathcal{G}_d$)	0.5701	0.3860	0.5678	0.5975
BERT-ROW-MAX (table)	0.5859	0.3933	0.5784	0.6061
ConvKNRM (table)	0.5403	0.3787	0.5382	0.5746

Ablation Study. To evaluate the effectiveness of several key components in MGNETS, we performed ablation studies as shown in Table 2. The 2nd line in Table 2 shows the result when the text encoder was removed. Encouragingly, MGNETS still performs better than all baselines in Table 1 except BERT-ROW-MAX. Without ConvKNRM as the text encoder, the performance of MGNETS-Conv does not decrease too much compared with MGNETS (graph only).

We also compare the performance of the model when using a single graph instead of multiple graphs. GNETS is the model that only operates on one graph and outputs a single graph-based prediction. GNETS ($\mathcal{G}_d$) achieves better results than GNETS ($\mathcal{G}_c$) on all metrics except NDCG@10. This indicates that the features in data tables could be more effective than features in context fields. We can see that even with all types of edges, GNETS ($\mathcal{G}$) underforms MGNETS, which suggests it is more difficult for graph neural networks to extract features from one single heterogeneous graph than from separate graphs. One possible reason is that edges constructed from different fields may have different semantic meanings and constructing a single heterogeneous graph could result in an ambiguous semantic space.

We further show the results of baselines (last 2 lines in Table 2) when only using data tables as input (i.e., no context fields). The NDCG@5 scores of BERT-ROW-MAX and Conv-KNRM decrease by 6.2% and 3.7% respectively, while GNETS ($\mathcal{G}_d$) has similar performance with GNETS ($\mathcal{G}$). It verifies our observation that the text-based methods are less effective to extract features from flattened tables where structure information is underutilized. The results also indicate that our proposed methods are more robust when context information is missing.

4 CONCLUSIONS

In this paper, we propose a graph-based solution MGNETS to address the task of table retrieval. We first study how to model the table corpus as one or more heterogeneous graphs and then utilize graph neural networks to learn the representations of queries and tables from complex structures. Experimental results demonstrate the features learned from multiple graphs can improve the text-based neural IR models. Future work might investigate more sophisticated relations in the table corpus and achieve a better understanding of semantic meanings in different fields.

Acknowledgments

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

REFERENCES

[1] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. 2019. SimGNN: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. 384–392.

[2] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2013. Methods for exploring and mining tables on wikipedia. In *Proceedings of the ACM SIGKDD workshop on interactive data exploration and analytics*. 18–26.

[3] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. TabEL: entity linking in web tables. In *Proc. Int'l Semantic Web Conf. (ISWC)*. 425–441.

[4] Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data. In *Findings of EMNLP 2020*.

[5] Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D Davison. 2018. Generating schema labels through dataset content analysis. In *Companion Proceedings of the The Web Conference 2018*. 1515–1522.

[6] Zhiyu Chen, Haiyan Jia, Jeff Heflin, and Brian D Davison. 2020. Leveraging schema labels to enhance dataset search. *Advances in Information Retrieval* 12035 (2020), 267.

[7] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D. Davison. 2020. Table Search Using a Deep Contextualized Language Model. In *Proceedings of 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. 589–598. https://doi.org/10.1145/3397271.3401044

[8] Zhiyu Chen, Shuo Zhang, and Brian D. Davison. 2021. WTR: A Test Collection for Web Table Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, Canada) *(SIGIR '21)*. 2514–2520.

[9] Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. 2018. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*. 126–134.

[10] Anish Das Sarma, Lujun Fang, Nitin Gupta, Alon Halevy, Hongrae Lee, Fei Wu, Reynold Xin, and Cong Yu. 2012. Finding Related Tables. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (Scottsdale, Arizona, USA) *(SIGMOD '12)*. Association for Computing Machinery, New York, NY, USA, 817–828. https://doi.org/10.1145/2213836.2213962

[11] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

[12] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[13] Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 8409–8416.

[14] Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering Table Queries on the Web using Column Keywords. *Proc. VLDB Endow.* 5 (2012), 908–919.

[15] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*. Springer, 593–607.

[16] Mohamed Trabelsi, Zhiyu Chen, Brian D Davison, and Jeff Heflin. 2020. A hybrid deep model for learning to rank data tables. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 979–986.

[17] Mohamed Trabelsi, Brian D Davison, and Jeff Heflin. 2019. Improved table retrieval using multiple context embeddings for attributes. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 1238–1244.

[18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ

[19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[20] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 7370–7377.

[21] Yang Yi, Zhiyu Chen, Jeff Heflin, and Brian D Davison. 2018. Recognizing quantity names for tabular data. In *ProfS/KG4IR/Data: Search@ SIGIR*.

[22] Shuo Zhang and Krisztian Balog. 2018. Ad hoc table retrieval using semantic similarity. In *Proc. World Wide Web Conference (TheWebConf)*. 1553–1562.

[23] Shuo Zhang, Edgar Meij, Krisztian Balog, and Ridho Reinanda. 2020. Novel entity discovery from web tables. In *Proceedings of The Web Conference 2020*. 1298–1308.