Relational Graph Embeddings for Table Retrieval

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*—Ad hoc table retrieval is the problem of identifying the most relevant datasets to a user's query. We present an approach to the problem that builds a knowledge graph by combining information about the collection of tables with external sources such as WordNet and pretrained Glove embeddings. We apply multi-relational graph convolutional networks to learn embeddings for the knowledge graph nodes and utilize three different methods to create vectors representing the tables and queries from these embeddings. We create a novel learning-to-rank neural architecture that incorporates the multiple embeddings in order to improve table retrieval results. We evaluate our approach using two large collections of tables from public WikiTables and Web tables data, demonstrating substantial improvements over state-of-the-art methods in table retrieval.**

***Index Terms*—Knowledge Graph, Table retrieval, Graph Convolution Networks, Neural networks, Learning to rank.**

I. Introduction

Knowledge graphs represent human knowledge in structured form using fact triples *(subject, predicate, object)* indicating the relations between entities. Many approaches have been proposed to learn representations for entities and relations in knowledge graphs [1]. Recently, researchers have focused on a new direction called graph neural networks [2] to solve multiple tasks [3]. Graph neural networks capture rich relational structures and encode the global structure of a graph in low dimensional feature vectors known as graph embeddings. The Graph Convolutional Network (GCN) [4], which is a simple and effective graph neural network, can capture high order neighborhood information to learn representations of nodes in a graph. Schlichtkrull et al. [5] show that an extension of the GCN framework, known as R-GCN, can be applied on knowledge graphs to learn representations for graph nodes and relations. In link prediction, R-GCN can be considered as an autoencoder consisting of an encoder that computes node representations, and a decoder function that predicts the scores of edges, such that known facts can be recalled and previously unknown facts can be inferred.

Ad hoc table retrieval, i.e., finding tables relevant to a search query, is an important problem because there are a vast array of tabular datasets available online, but it is difficult for users to identify tables that best meet their needs. Researchers have focused on utilizing the knowledge contained in tables in multiple tasks including augmenting tables [6], [7], extracting knowledge from tables [8], table retrieval [7], [9]–[12], and semantic labeling [13], [14]. Tables can be considered as documents, so that document retrieval methods can be applied to table retrieval [7], [9]. Prior research has shown that classical bag-of-words based models are not effective in capturing fine-grained contextual structures for information retrieval, and the same is true for table retrieval. Supervised learning, based on hand-crafted features from tables, queries, and query-table pairs [6], [9], has resulted in the best performing table retrieval systems. Building on this, Zhang and Balog [15] introduced semantic features to embed queries and tables into a semantic space, and then trained a supervised model using the semantic and traditional features.

Inspired by recent progress of transfer learning on graph neural networks, we propose a two-phased table retrieval method which uses graph embeddings pretrained on a large table corpus, denoted as Multiple Embeddings R-GCN (MultiEm-RGCN). In phase I, we construct a knowledge graph containing two types of knowledge: dataset-dependent knowledge and dataset-agnostic knowledge. The graph contains words and tables from a collection of tables as nodes. To incorporate dataset-dependent knowledge, point-wise mutual information (PMI) between word nodes, and term frequency-inverse document frequency (TF-IDF) between table and word nodes are computed based on the collection of tables. To avoid overfitting, we incorporate dataset-agnostic knowledge via external resources containing semantic knowledge from pretrained word embeddings, and lexical knowledge from WordNet. This incorporates a subgraph from WordNet into our knowledge graph. We model the graph with R-GCN which is used to learn multiple types of embeddings simultaneously. In phase II, we solve the table retrieval task by incorporating the R-GCN heterogeneous embeddings from phase I into a new learning-to-rank (LTR) architecture that combines multiple embedding spaces into one joint model. Results on the WikiTables [16] dataset demonstrate that our method outperforms state-of-the-art table retrieval methods.

In summary, we make the following contributions:

- We propose the Multiple Embeddings R-GCN (MultiEm-RGCN) method to solve table retrieval. MultiEm-RGCN has two phases. In phase I, we propose a new knowledge graph that incorporates both dataset-dependent and dataset-agnostic knowledge from table corpus. External semantic and lexical resources are used for edges and nodes leading to an heterogeneous graph.
- Multiple types of embeddings are learned simultaneously from our proposed knowledge graph using R-GCN with

*Equal contribution

the link prediction pre-training task. This leads to an embedding for each node in the knowledge graph (word, WordNet, and table nodes). In phase II, R-GCN embeddings are incorporated into an LTR architecture that combines multiple embeddings from our heterogeneous graph to solve the table retrieval task.

Experimental results on two public datasets (WikiTables [16] and WebQueryTable [17]) demonstrate that our knowledge graph based method outperforms state-of-the-art baselines.

II. Related work

A. Knowledge graph embeddings

Various methods have been proposed for representation learning of knowledge graphs, which aims to project entities and relations into a continuous space. TransE [18], inspired by Word2Vec [19], is the most representative translation-based model, which considers the translation operation between head and tail entities for relations. The variants of TransE, such as TransH [20] and TransR [21], follow a similar principle but use different scoring functions to learn the embeddings. Socher et al. [22] apply neural tensor networks to learn knowledge graph embeddings. Dettmers et al. [23] propose a convolutional neural network approach to learn knowledge graph embeddings and use them to perform link prediction. RDF2Vec [24] adapts the Word2Vec [19] approach to RDF graphs in order to learn embeddings for entities in RDF graphs.

The recent success of graph neural networks has boosted research on various tasks. R-GCN [5] pioneered the use of graph convolutional networks to model relations in knowledge graphs. The embeddings learned by R-GCN have shown to be effective for downstream tasks such as entity classification and link prediction. More recently, Xu et al. [25] first construct a product knowledge graph and then propose a self-attention-enhanced distributed representation learning method with an efficient multi-task training schema to learn the graph embeddings, which can improve the performance of downstream tasks such as search ranking and recommendation. The motivation behind our work is similar, where we first construct a knowledge graph from table collections and then learn the graph embeddings in order to perform table retrieval.

B. Word embedding for tables

Words are embedded into low dimensional real-valued vectors based on the distributional hypothesis. In many proposed models, the context is defined as the words that precede and follow a given target word in a fixed window [26], [27]. Mikolov et al. [19] proposed the Skip-gram model which scales to corpora with billions of words. Recent work has used embedding techniques to learn a low dimensional representation for table tokens in multiple tasks related to tables. Ghasemi-Gol and Szekely [28] defined a new unsupervised embedding for tables to perform table classification. They define four distinct contexts for each cell value: text within each cell, text in the corresponding attribute or header, text in adjacent cells, and text surrounding the table in the web page. Trabelsi et al. [29] proposed a word embedding of table attributes tokens using contextual information of every table. Chen et al. [11] proposed a method to generate table headers which can be used as additional features for table search.

C. Ad hoc table retrieval

In a table retrieval task, a table can be considered as a document, and traditional document retrieval methods can be applied for table ranking. Cafarella et al. [7], [9] 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 [30].

In supervised table retrieval, multiple query, table, and query-table features are proposed in the literature [6], [9]. Zhang and Balog [15] proposed extending these features with semantic matching between queries and tables using various semantic spaces: Word embeddings, Graph embeddings, Bag-of-entities and Bag-of-categories. The DBpedia knowledge base is used to construct a Boolean vector for both bag-of-entities and bag-of-categories. The dimension of bag-of-entities is equal to the total number of entities in the knowledge base, where a value of 1 indicates that the entity is mentioned in the table. The same applies to bag-of-categories with a dimension equal to the total number of Wikipedia categories. One drawback to this approach is that it is only effective when the primary subjects of the tables appear in DBpedia. Since DBpedia is sourced from Wikipedia, this means it will have many relevant entities for famous people, places, and artistic works, but will have sparser coverage on more specialized topics, such as various proteins, aircraft parts, etc.

III. Knowledge Graph Construction and Embedding Learning

We introduce the following notation for the rest of the paper. We denote a knowledge graph by $G = (V, T, R)$, with a set of nodes V, a set of relation types R, and a set of directed edges $(v_i, r, v_j) \in T$, where v_i,$v_j \in V$ and $r \in R$. T can be seen as an RDF collection that contains (s, p, o) triples representing the knowledge graph. In this section, we first give a brief overview of R-GCN. Then we introduce how to construct a knowledge graph (KG) based on the table corpus given a set of predefined relations, which incorporate dataset-dependent knowledge and dataset-agnostic knowledge. After that, we describe how to learn high-quality node embeddings based on the constructed KG with link prediction as the pretraining task under the R-GCN framework. Ultimately, we present a Multiple Embeddings R-GCN (MultiEm-RGCN) model with two phases: phase I consists of training unsupervised embedding using R-GCN, and phase II consists of incorporating the multiple embeddings into a new LTR model.

A. Relational graph convolutional networks (R-GCN)

R-GCN [5] can be seen as an extension of GCN [4], [31] for relational data that operates on a local graph neighborhood

using a message-passing framework [32]. The R-GCN model updates the hidden representation of node v_i in the relational graph as given by:

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right) \quad (1)$$

where $h_i^{(l)} \in R^{d(l)}$ is the l-th layer hidden state of node v_i in the neural network, $d(l)$ is the dimension of the embedding in the l-th layer, and $\sigma(.)$ is a nonlinear activation function. $\mathcal{N}_i^r$ denotes the set of r-neighbors, where $r \in \mathcal{R}$. $c_{i,r}$ is a normalization constant that is equal to $|N_i^r|$. Unlike the linear transformation in GCN that can be applied to any node in a given layer, R-GCN has a relation specific linear transformation, denoted by $W_r^{(l)}$, that depends both on the type and direction of the edge in a directed and labeled graph. $W_0^{(l)}$ is a trainable matrix that incorporates the l-th layer representation into the $l+1$-th layer of the neural network. R-GCN is formed of multiple stacked layers with non-linear activation functions to capture complex patterns in the graph that are not only related to direct neighbors. Updating the nodes in R-GCN for a given layer is done in parallel to reduce computation time.

B. Heterogeneous Knowledge Graph Construction

In this subsection, we describe how to build a meaningful knowledge graph for a large collection of tables that captures both general knowledge and dataset specific knowledge. Our graph contains word nodes and table nodes. The word nodes are constructed from the table collection and external resources. For a given word in the table collection, we also use its synonyms defined in WordNet and the hypernyms of these synonyms, which also have corresponding nodes in our constructed knowledge graph. In other words, our final graph includes a subset of WordNet relevant to the table collection.

We construct edges that encode two types of knowledge: ***dataset-dependent knowledge*** and ***dataset-agnostic knowledge***. For dataset-dependent knowledge, we build table-word edges and word-word edges from the table collection. However, only pretraining node embeddings from such a graph could overfit the dataset collection and harm the generalization ability of learned node embeddings, especially given a small training set. Therefore, we also propose to encode dataset-agnostic knowledge from external resources such as other pretrained word embeddings and WordNet. By constructing edges that encode both dataset-dependent knowledge and dataset-agnostic knowledge, we assume the learned node embeddings can capture both dataset specific information and open world knowledge. The overview of our proposed knowledge graph is shown in Figure 1. We build our graph G using RDF triples T. Initially, T is empty, and in this section we show how to build T.

1) Dataset-agnostic knowledge: We consider two types of dataset-agnostic knowledge. The first is ***semantic knowledge*** from word embeddings, such as Glove [27], pretrained on a large corpus. The cosine similarity of a word pair can be treated as prior information for two word nodes in our graph.

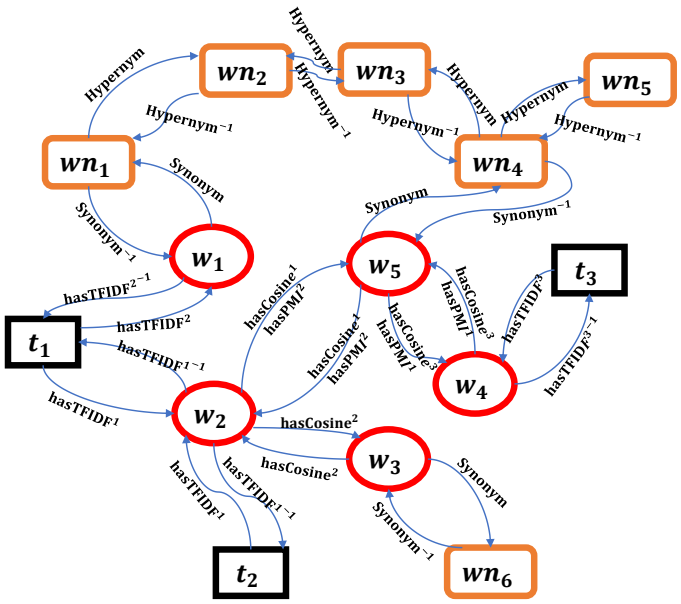

Fig. 1: Overview of phase I of the proposed method MultiEm-RGCN. We use the same edge for $hasCosine$ and $hasPMI$ to avoid clutter in the graph. We can notice that the words w_2 and w_3 have only the $hasCosine^2$ relation because w_2 and w_3 do not co-occur in the tables collection.

Let $d_{semantic}(w_i, w_j)$ denote the cosine similarity between two words w_i and w_j, which is a real value in the interval $[-1, 1]$. When building the knowledge graph, we do not keep the exact value of $d_{semantic}(w_i, w_j)$. Instead, we define several relations that represent different levels of similarities. This idea is inspired by the histogram matching method by Guo et al. [33] used for query document matching. We discretize the interval into a set of ordered bins and each bin has a corresponding edge type. Suppose that W is the set of word tokens in the training collection. We build the triples with p_{cosine} as the predicate only for those word pairs whose cosine similarity is larger than a threshold M_{cos}, since we care more about the semantic similarities rather than the dissimilarities, and the inferred dissimilarities from word embeddings could be inaccurate and trivial (two words randomly selected are likely to be dissimilar). In order to reduce the effect of extreme values which may result in bins with few data points, we calculate the mean m_{cos} and standard deviation std_{cos} of the set of all valid cosine similarities. Then we set the interval as $[min_{cos}, max_{cos}]$ where min_{cos} is the smallest cosine similarity that is larger than $m_{cos} - 2 \times std_{cos}$, and max_{cos} is the largest cosine similarity that is smaller than $m_{cos} + 2 \times std_{cos}$. We linearly divide $[min_{cos}, max_{cos}]$ into n_{cos} intervals and the k-th interval has a corresponding predicate $hasCosine^k$. For example, if $d_{semantic}(w_i, w_j)$ belongs to the k-th interval, then we add $(w_i, hasCosine^k, w_j)$ to T. Note that there are word pairs that have cosine similarity smaller than min_{cos}: we

assign them to the 1st bin, and those have cosine similarity larger than max_{cos} are assigned to the last bin. Here, we define the set of semantic relation triples as

$$SemT = \{(w_i, hasCosine^k, w_j) | w_i, w_j \in W \text{ and } d_{semantic}(w_i, w_j) > M_{cos}\} \quad (2)$$

where $d_{semantic}(w_i, w_j)$ belongs to the k-th interval. We add $SemT$ to T.

The second type of dataset-agnostic knowledge incorporated into our graph is ***lexical knowledge*** from WordNet [34]. Specifically, we define two additional relations in $\mathcal{R}$. The first relation corresponds to the edges between a word and its synonyms (also called synsets) defined in WordNet. In particular, given a word w_i, we extract its synonyms, denoted by Syn_i. We define the set of synonym relation triples $SynT_i$ associated with w_i and its synonyms Syn_i as

$$SynT_i = \{(w_i, Synonym, w_n) | w_n \in Syn_i\} \quad (3)$$

with $Synonym \in \mathcal{R}$. We add $SynT_i$ for every word $w_i \in W$ to T.

The second relation corresponds to the edges between synonyms in the graph. Given a node w_{ni} which is a synonym of w_i, we extract the hypernyms of w_{ni}, denoted by Hyp_i. Then, we define the hypernym relation triples $HypT_i$ associated with w_{ni} and its hypernyms as

$$HypT_i = \{(w_{ni}, Hypernym, w_{nj}) | w_{nj} \in Hyp_i\} \quad (4)$$

with $Hypernym \in \mathcal{R}$. T is expanded by adding all triples from $HypT_i$.

To have a complete directed graph, for all triples (s, p, o), we add the triples (o, p^{-1}, s) to T, with p^{-1} is the inverse of p. We have already added all cosine similarity edges because $hasCosine$ and $hasCosine^{-1}$ are identical. We calculate the triples from the inverse of $Synonym$ and $Hypernym$, denoted by $Synonym^{-1}$ and $Hypernym^{-1}$, respectively, and we add the calculated triples to T. We choose not to treat $Synonym$ and $Synonym^{-1}$ as identical relations because the domain and range have different types (word-WordNet entity edge).

2) Dataset-dependent knowledge: In order to incorporate dataset specific information, we connect table nodes with word nodes using TF-IDF relations. In particular, we calculate the TF-IDF value for a word w_i in table $t \in C$, denoted as TF-IDF(t, w_i). We follow the same binning approach that we used for discretizing semantic similarities in order to obtain a triple from TF-IDF(t, w_i). For example, given n_{tfidf} different intervals, if TF-IDF(t, w_i) belongs to the k-th interval, we obtain the triple $(t, hasTFIDF^k, w_i)$ which is added to T. In this case, we expand $\mathcal{R}$ by adding n_{tfidf} relations that are related to TF-IDF.

The cosine similarity between two words from a pretrained embedding encodes the co-occurrence information in the large pretraining corpus. By encoding the local co-occurrence information in our table collection, the constructed knowledge graph can retain dataset-specific relations. If we take the headers of tables as an example, in multiple tables, we could frequently find this sequence of headers (with different orders): Birth date, Birth place, Death date, Death place, etc. So the co-occurrence of tokens in this sequence of headers should be high. We utilize PMI to describe local context information using a sliding window strategy. The edge weight of each pair of words is calculated by:

$$PMI(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)\, p(w_j)} \quad (5)$$

where $p(w_i, w_j)$ is the probability of co-occurrence of words w_i and w_j in the same sliding window of length s_w, which is estimated by:

$$p(w_i, w_j) = \frac{\#N_{\text{co-occurrence}}(w_i, w_j)}{\#N_{\text{windows}}} \quad (6)$$

with $\#N_{\text{co-occurrence}}(w_i, w_j)$ is the number of times the pair (w_i, w_j) co-occurs in the same sliding windows over the whole table collection, and $\#N_{\text{windows}}$ is the total number of sliding windows of size s_w over the whole collection of tables.

Binning is used again for $PMI(w_i, w_j)$. Given n_{pmi} intervals for the set of PMI values, if $PMI(w_i, w_j)$ belongs to the k-th interval, we obtain the triple $(w_i, hasPMI^k, w_j)$. Given that the PMI calculation is symmetric, the triple $(w_j, hasPMI^k, w_i)$ is also valid, and we add both triples to T. After adding the triples, we expand $\mathcal{R}$ by adding n_{pmi} relations that are related to PMI.

We add the inverse relations in order to finish constructing the graph. Like the $hasCosine$ relation, $hasPMI$ and $hasPMI^{-1}$ are identical. For TF-IDF, we define a new relation TF-IDF$^{-1} \in \mathcal{R}$ in order to compute the directed edges from words to table nodes.

C. Knowledge Graph Embedding Learning

We use link prediction as the pretraining task to learn node embeddings of the constructed knowledge graph in section III-B. The objective of link prediction is to predict new facts given by (s, p, o) triples. So, the directed labeled graph G contains only a subset of possible edges, and the objective is to predict the score $f(s, p, o)$ of possible edge (s, p, o) to determine the validity of the triple. We use the graph auto-encoder model introduced by Schlichtkrull et al. [5] that consists of a node encoder and scoring function for decoder. The role of the encoder is to compute the embedding $e_i \in \mathbb{R}^d$ of node v_i in the graph. So, the encoder is the R-GCN model, and the node embedding is obtained by setting e_i to h_i^L, where L is the number of layers in R-GCN and h_i^L is the hidden representation of node v_i from the last layer. The role of the decoder is to reconstruct edges of the graph using node embeddings. This means that the decoder maps (s, p, o) triples into a real valued score. The advantage of the encoder-decoder architecture is the end-to-end training of both the embedding and the scoring function.

The decoder function is based on DistMult factorization [35] that has shown good results in link prediction despite its simple expression. Every relation $p \in \mathcal{R}$ is associated with

a diagonal matrix $R_p \in \mathbb{R}^{d\times d}$, and the score $f(v_i, p, v_j)$ of triple (v_i, p, v_j) is given by:

$$f(v_i, p, v_j) = e_{v_i}^T R_p e_{v_j} \tag{7}$$

where e_{v_i} and e_{v_j} are the embeddings of nodes v_i and v_j respectively. The graph auto-encoder is trained with negative sampling as in [5], [35], [36]. In particular, we treat the triples in T as positive triples. For each $t_p \in T$, we sample w negative samples by either corrupting the object or subject of t_p. We optimize the graph auto-encoder parameters via link prediction by minimizing the cross-entropy loss:

$$\mathcal{L} = -\frac{1}{(1+\omega)|E|} \sum_{(s,p,o,y)\in\mathcal{T}} y \log \sigma(f(s,p,o)) + (1-y)\log(1-\sigma(f(s,p,o))) \tag{8}$$

where $\mathcal{T}$ represents the set of positive and corrupted triples, y is the label of triple which is set to 1 for positive triples, and 0 for corrupted triples, and σ is the logistic sigmoid function. Minimizing the cross-entropy loss leads to having a higher $f(s, p, o)$ score for positive triples than the corrupted ones.

IV. KG Embedding for Table Retrieval

Our proposed heterogeneous graph includes information from both the table collection and external resources, where various relations among nodes are encoded. As described in section III-C, after training the graph auto-encoder on the link prediction task, the encoder provides an embedding for each node in the graph that captures the graph structure and the information that is passed from node to node using the edges labeled by relations from $\mathcal{R}$.

We show the quality of the trained embedding by incorporating node representations into a learning-to-rank (LTR) model designed for table retrieval. Our trained graph auto-encoder simultaneously provides embeddings for different types of nodes (tables, words, synsets) in the knowledge graph so that we can tackle the table retrieval problem by using each embedding type independently or by having a joint model that combines the multiple embeddings. In this section, we discuss multiple LTR models that take advantage of our proposed graph embeddings in order to improve the results of ad hoc table retrieval. In training, a set of queries $Q = \{q_1, q_2, \ldots, q_s\}$, and a table corpus $\mathcal{C} = \{t_1, t_2, \ldots, t_l\}$, are given, where s and l are the total number of queries and data tables, respectively. We denote the number of tokens per query $q \in Q$ by n, and the number of tokens per table $t_j \in \mathcal{C}$ by m. Each table t_j has a relevance score, denoted by y_j, to a given query q. We propose f_w, a new joint embedding LTR model with parameters w, which incorporates multiple embedding spaces to predict the relevance score of a given query-table pair (q, t_j), such that higher ranked tables should be more relevant to the query.

1) Graph word embedding: The first type of embedding used in our LTR model is the word embedding obtained from word nodes. After using the encoder to calculate the embedding of each node, we collect word nodes to form a word vocabulary for the tables collection, which is used to compute the word embeddings of queries and tables. For a given query $q = q_1, q_2, \ldots, q_m$ where m is the length of the query and q_l is the l-th token of q, the R-GCN word representation is given by

$$\boldsymbol{q} = \boldsymbol{q_1} \oplus \boldsymbol{q_2} \oplus \boldsymbol{q_3} \oplus \cdots \oplus \boldsymbol{q_m} \tag{9}$$

where $\boldsymbol{q_k} \in \mathbb{R}^d$ is a d-dimensional R-GCN word embedding of token q_k and $\oplus$ is the concatenation operator to build the matrix $\boldsymbol{q} \in \mathbb{R}^{m\times d}$. A given table t_j is linearized by concatenating metadata, such as table caption, page title, headers, and data values. Then, R-GCN word embeddings are used to map t_j to an embedding matrix $\boldsymbol{t_j} \in \mathbb{R}^{n\times d}$.

For a given query-table pair (q, t_j), the inputs to the word embedding-based LTR model are $\boldsymbol{q}$ and $\boldsymbol{t_j}$. We choose the Convolutional Kernel-based Neural Ranking Model (Conv-KNRM), proposed by Dai et al. [37], as our word embedding-based LTR architecture.

Conv-KNRM uses a Convolutional Neural Network (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 [38] is used to summarize the similarities into a soft-matching feature vector of dimension K; intuitively, this vector represents the probabilities that the similarities come from the distribution specified by each kernel. The soft-matching feature vector is computed for different n-grams of query and document, and then they are concatenated into a single feature vector. The extracted feature is then passed through a learning-to-rank layer to predict a relevance score. We choose Conv-KNRM as the main component in our LTR model because it shows good performance in multiple benchmarks for document retrieval. Moreover, the CNN approach of modeling n-grams makes cross-matching between query and document tokens feasible, effective, and efficient. The input embedding layer to conv-KNRM is initialized using our word embeddings.

2) Graph embeddings for WordNet entities: Given a query $q = q_1, q_2, \ldots, q_m$ where m is the length of the query and q_l is the l-th token of q, we translate each token q_l into a set of synonyms and hypernyms, denoted by $Trans(q_l)$, using WordNet. First, we extract the set of synonyms from WordNet and we append it to $Trans(q_l)$. Then, we use a stack to extract the hypernyms of synonyms, and then the transitive closure, with a maximum of 20 hops, over hypernyms. So, $Trans(q_l)$ forms a sequence of synonyms and hypernyms from WordNet. Finally, the translated query, $Trans(q)$, is given by:

$$Trans(q) = [Trans(q_1); Trans(q_2); \ldots; Trans(q_m)] \tag{10}$$

We apply the same idea to obtain the translation $Trans(t_j)$ of a given table t_j. After the translation step, our queries and tables are represented as a sequence of WordNet entities. Given that our graph auto-encoder produces embeddings for WordNet

entities (synonyms and hypernyms), we compute WordNet embeddings for the translated query and table:

$$\boldsymbol{Trans}(\boldsymbol{q}) = \bigoplus_{w_n \in Trans(q)} \boldsymbol{w_n}; \boldsymbol{Trans}(\boldsymbol{t_j}) = \bigoplus_{w_n \in Trans(t_j)} \boldsymbol{w_n}$$

where $\boldsymbol{w_n} \in \mathbb{R}^d$ is a d-dimensional R-GCN WordNet embedding of w_n, $\boldsymbol{Trans}(\boldsymbol{q}) \in \mathbb{R}^{|trans(q)| \times d}$ and $\boldsymbol{Trans}(\boldsymbol{t_j}) \in \mathbb{R}^{|trans(t_j)| \times d}$ are the R-GCN embedding matrices of $Trans(q)$ and $Trans(t_j)$, respectively. We train a Conv-KNRM model on table-query pairs of translated sequences using $\boldsymbol{Trans}(\boldsymbol{q})$ and $\boldsymbol{Trans}(\boldsymbol{t_j})$ as inputs, and with an embedding layer that is initialized using the R-GCN embedding for WordNet nodes.

3) Graph table embedding: The embeddings for both word and WordNet nodes can be used directly for tables and query tokens. In contrast, embeddings of table nodes only provide representations for tables. For a given query-table pair (q, t_j), in order to compute an embedding for queries using table node embeddings, we propose an approach inspired by pseudo relevance feedback, where we generate a pseudo-query by aggregating the top-J tables returned by BM25. In particular, given a query q and a collection of tables $\mathcal{C}$, the pseudo-query q' is the sequence of closest J tables, $t_1, t_2, \ldots, t_J$, to q using BM25. The R-GCN table embeddings are used to compute the embedding matrix $\boldsymbol{q'} \in \mathbb{R}^{J \times d}$ which is given by

$$\boldsymbol{q'} = \overline{\boldsymbol{t_1}} \oplus \overline{\boldsymbol{t_2}} \oplus \overline{\boldsymbol{t_3}} \oplus \cdots \oplus \overline{\boldsymbol{t_J}} \tag{11}$$

where $\overline{\boldsymbol{t_i}}$ is the table embedding of t_i which is computed using R-GCN for table nodes in the knowledge graph.

Then, we aggregate $\boldsymbol{q'}$ using a simple aggregation function for neural networks, in order to compute the query embedding. Our neural aggregation function is based on ARC-I [39] which summarizes the meaning of a sequence through layers of convolution and pooling, and produces a fixed length feature vector $\boldsymbol{q_{agg}}$. In our case, the input sequence to ARC-I is $\boldsymbol{q'}$. The last layer of the feature extractor of the table embedding model consists of a pointwise multiplication layer between the table embedding $\overline{\boldsymbol{t_j}}$ of t_j and $\boldsymbol{q_{agg}}$. The resulting feature vector is passed through a multilayer perceptron (MLP) to predict the relevance score of (q, t_j).

4) Joint embedding: We describe phase II of our proposed MultiEm-RGCN which combines all three types of embeddings into one LTR model. As shown in Figure 2, the phase I embeddings are used to compute the query representation $\hat{\boldsymbol{q}}$ and table representation $\hat{\boldsymbol{t_j}}$:

$$\hat{\boldsymbol{q}} = \boldsymbol{q} \oplus \boldsymbol{Trans}(\boldsymbol{q}) \oplus \boldsymbol{q'}; \quad \hat{\boldsymbol{q}} \in \mathbb{R}^{(n+|Trans(q)|+J) \times d}$$
$$\hat{\boldsymbol{t_j}} = \boldsymbol{t_j} \oplus \boldsymbol{Trans}(\boldsymbol{t_j}) \oplus \overline{\boldsymbol{t_j}}; \quad \hat{\boldsymbol{t_j}} \in \mathbb{R}^{(m+|Trans(t_j)|+1) \times d}$$

Then, we pass $\hat{\boldsymbol{q}}$ and $\hat{\boldsymbol{t_j}}$ through a Conv-KNRM model to predict the final relevance score of (q, t_j). The model is trained to minimize the listwise loss function ListNet [40], and generate a ranked list of tables for each query that matches the ranking using the ground truth relevance scores. We choose not to update phase I embeddings when minimizing the listwise loss function to reduce model complexity, and focus the efforts of training on learning the CNN filters and MLP weights of Conv-KNRM.

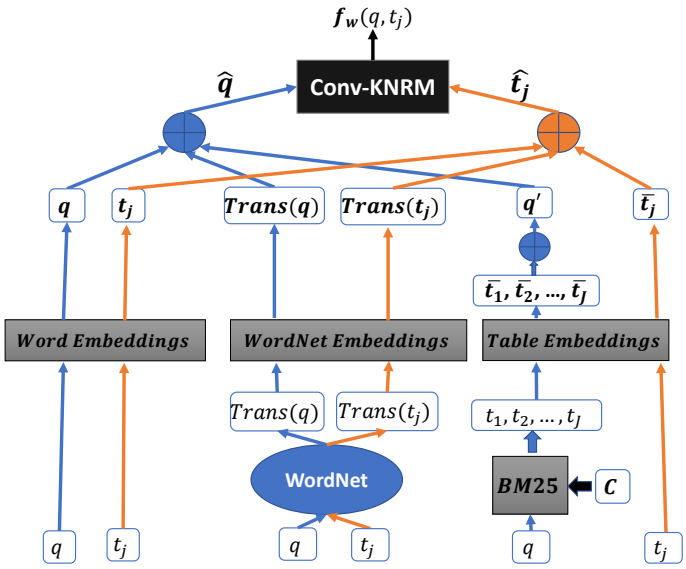

Fig. 2: Overview of phrase II of the proposed method MultiEm-RGCN. The blue and orange edges represent the data flow for a given query q and table t_j, respectively. $\oplus$ denotes the concatenation operator used to form the table embedding of pseudo-query $\boldsymbol{q'}$, the final query representation $\hat{\boldsymbol{q}}$, and the final table representation $\hat{\boldsymbol{t_j}}$. The Conv-KNRM bloc takes $\hat{\boldsymbol{q}}$ and $\hat{\boldsymbol{t_j}}$ as inputs to predict the final relevance score $f_w(q, t_j)$.

V. Evaluation

A. Data and query collections

The first dataset is the WikiTables [1] corpus [16] containing over $1.6M$ tables. Each table has five indexable fields: table caption, attributes (column headings), data rows, page title, and section title. In addition, each table contains statistics: number of columns, number of rows, and set of numerical columns of the table. We use the same queries that were used by Zhang and Balog [15] where every query-table pair is evaluated using three numbers: 0 means "irrelevant", 1 means "partially relevant" and 2 means "relevant". This collection has 60 queries with ground-truth relevance judgments. To evaluate table retrieval, we report results of five-fold cross validation of the entire query-table pairs collection for our proposed method and baselines.

The second dataset is the WebQueryTable[2] collection that is introduced by Yan et al. [17]. Unlike the WikiTables collection that contains tables only from Wikipedia, WebQueryTable is composed of more various tables collected from web pages. The total number of tables in WebQueryTable is $297,884$. Each table has four indexable fields: table caption, table subcaption, attributes (column headings), and data rows. In addition, WebQueryTable [17] contains $21,142$ queries. Each

[1] http://websail-fe.cs.northwestern.edu/TabEL/

[2] https://github.com/tangduyu/Table-Intelligence/tree/master/table-search

TABLE I: Parameters values used in our proposed model

Model phases	Parameters	Values
Phase I	number of layers L	2
	M_{cos} threshold	0.5
	n_{cos},n_{tfidf},n_{pmi} and intervals	3
	PMI sliding window size s_w	20
	Table selection probability p	0.0001
	Size of subgraph Sz	50000
	Dimension of embedding d	100
	Negative samples w	10
	optimizer	Adam optimizer with $l_r = 0.001$
Phase II	Number of kernels K	5
	number of extracted tables J	100
	number of layers in ARC-I	2 with 50 CNN each
	n-grams in Conv-KNRM	unigram, bigram, and trigram
	number of CNN filters per n-gram in Conv-KNRM	128
	length of query n	6
	number of tokens m per table	80
	optimizer	Adam optimizer with $l_r = 0.001$

query-table has a binary relevance value, and only one table is relevant to a given query. The total number of query-table pairs is $1,051,075$, where $839,239$ pairs are used to train our LTR model of MultiEm-RGCN, and $211,836$ pairs are used for testing.

B. Baselines

1) Unsupervised ranking approaches: A table is considered as a single field document by concatenating indexable fields. For example, in the WikiTables collection, we concatenate table caption, attributes, data rows, page title and section title. We compare our approach against a single-field ranking method which is based on BM25 to calculate a retrieval score. On the other hand, a data table can be considered as multi-field document, so we compare against a multi-field ranking method that is based on the pretrained Glove word embedding when calculating cosine similarity. MaxTable [29] similarity measure is used to calculate the score between query tokens and a given field in a table.

2) Supervised ranking approaches: We compare our method against state-of-the-art approaches for table retrieval: LTR and STR [15]. We also compare against various embeddings that are used as input to Conv-KNRM. The first set of embeddings are pretrained on large text corpuses and are Word2Vec [19], Glove [27] and fastText [41]. The second set of embeddings are pretrained on WikiTables which are TabVec [28], and MCON [29].

C. Experimental Setup

In all reported results, we choose not to update the embeddings when minimizing the loss function for table retrieval for two reasons: first we would like to directly compare the quality of embeddings that we obtain from multiple methods. Second, by freezing word embeddings, we reduce model complexity, and focus the efforts of training on only updating the parameters of LTR model.

Table I summarizes the parameters that are used in MultiEm-RGCN. In each training step of R-GCN, we randomly construct a connected subgraph of size S_z to make computations feasible. Inverse relations enable constructing a subgraph with multiple types of nodes. For example, without inverse of TF-IDF relation, it is not possible to add a table node to the subgraph when the current node is of type word. Given that our graph contains more edges connecting words and WordNet nodes, we force including table nodes in the subgraph with probability p instead of picking a random node.

We evaluate the performance of our proposed method and baselines on the table retrieval task using Normalized Discounted Cumulative Gain (NDCG) [42], Mean Reciprocal Rank (MRR), and Mean Average Precision (MAP).

D. Evaluation using the Wikitables corpus

1) Ranking results: Table II shows the performance of different approaches on the WikiTables collection. We show that our proposed method MultiEm-RGCN outperforms the baselines for all evaluation metrics.

Among R-GCN embeddings, the word-based embedding has better retrieval results than WordNet and table embeddings as shown in Table II. This can be explained by the fact that the graph contains many edges that have word nodes as the subject or object. So, updating word nodes is more frequent than updating WordNet and table nodes. In addition to that, unlike table nodes that have only input edges from word nodes using $hasTFIDF^{-1}$ relation, and WordNet nodes that have only inputs from other word and WordNet nodes, word nodes receive input messages from all three types of nodes in the graph using multiple relations.

Table II shows that using only R-GCN word embedding leads to better retrieval results than the baselines, but it is not the same case for WordNet and table embeddings which are more useful when used in the joint model MultiEm-RGCN. Most of the Conv-KNRM based baselines have better results than STR, the state-of-the-art method for table retrieval. Conv-KNRM+MCON performs worse, likely because it only computes word embeddings for attributes. Among the baselines, Conv-KNRM combined with fastText achieves higher performance for all evaluation metrics. The use of character-level n-grams in fastText allows word embeddings

TABLE II: Table retrieval evaluation results using our proposed embedding and baselines for WikiTables dataset

Category	Method	NDCG@5	MAP	MRR
Unsupervised ranking	Single-field ranking	0.4511±0.0320	0.4773±0.0301	0.5162±0.0381
	Multi-field ranking	0.4990±0.0263	0.4922±0.0291	0.5235±0.0274
Supervised ranking	LTR [6], [9]	0.5142±0.0391	0.5224±0.0354	0.5704±0.0193
	STR [15]	0.5823±0.0376	0.5910±0.0375	0.6360±0.0372
	Conv-KNRM+Glove [27]	0.5950±0.0332	0.5981±0.0323	0.6291±0.0312
	Conv-KNRM+fastText [41]	0.6012±0.0325	0.6011±0.0342	0.6366±0.0300
	Conv-KNRM+Word2vec [19]	0.6003±0.0313	0.6009±0.0252	0.6313±0.0266
	Conv-KNRM+TabVec [28]	0.5951±0.0366	0.6022±0.0349	0.6352±0.0368
	Conv-KNRM+MCON [29]	0.5820±0.0345	0.5838±0.0335	0.6248±0.0365
MultiEm-RGCN	R-GCN Word embedding	0.6130±0.0338	0.6079±0.0327	0.6414±0.0354
	R-GCN WordNet embedding	0.5264±0.0920	0.5356±0.0721	0.5752±0.0845
	R-GCN Table embedding	0.4744±0.0441	0.4869±0.0412	0.5354±0.0454
	MultiEm-RGCN without $\boldsymbol{q}'$	0.6201±0.0282	0.6197±0.0283	0.6511±0.0283
	MultiEm-RGCN	**0.6246±0.0277**	**0.6242±0.0267**	**0.6565±0.0241**

to be created even for terms that have not been seen before, and reduce the negative effect of out of vocabulary tokens on calculating the final relevance score of a query-table pair.

We explain the improvement in performance of our model compared to baselines by two facts. First, our proposed graph combines rich semantic and lexical general knowledge from Glove and WordNet with data specific knowledge. This leads R-GCN to learn nodes embeddings with a balance between general knowledge and tables collection characteristics. Second, our heterogeneous graph provides multiple embeddings that can be incorporated into a single LTR architecture in order to aggregate matching signals between query and table in multiple spaces. This leads to more accurate calculation of relevance score of a query-table pair.

2) Adding more features: We examine the effect of adding data values and STR features to the MultiEm-RGCN model. Table III shows table retrieval results using MultiEm-RGCN with different combinations of features. Since it can be computationally expensive to process all values from a table, we randomly select 50 string values from each table, and append the values tokens to description and attributes tokens in phase II of MultiEm-RGCN. As shown in Table III, we obtain slight improvements in retrieval results when adding random values to description and attributes.

TABLE III: Table retrieval performance using MultiEm-RGCN with different features for WikiTables dataset

Method	NDCG@5	MAP	MRR
Description+ attributes	0.6246±0.0277	0.6242±0.0267	0.6565±0.0241
Description+ attributes+values	0.6263±0.0252	0.6256±0.0287	0.6574±0.0339
Description+STR+ attributes+values	**0.6272±0.0225**	**0.6285±0.0235**	**0.6595±0.0258**

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 [15]. We append STR features to word, WordNet, and tables feature vectors, and then train end-to-end the full system. Table III shows that adding the large number of STR features only leads to a slight improvement over using only table content and metadata. Thus, not only does MultiEm-RGCN unified knowledge graph exceed the performance of specialized LTR [6], [9] and STR [15] features, but it also almost entirely captures any useful signal present in those features. R-GCN word, WordNet, and table embeddings are directly used in a joint LTR architecture, and this leads to significant improvement over the state-of-the-art STR table retrieval method.

3) Embeddings visualization: We visualize the embeddings learned by MultiEm-RGCN. Figure 3 shows the t-SNE visualization of nodes embeddings from the second layer in R-GCN using the WikiTables collection. Figure 3 shows three different spaces from embeddings which are: word (red dots), WordNet (green dots), and table (blue dots). For each embedding space, we randomly zoom a region to show the embeddings of nodes of our proposed graph. For word embeddings, we can see that the words *mobile*, *telephone*, *phone*, *online*, *internet*, *web*, *website*, etc., are close to each other. The same interpretation is valid for WordNet embeddings where synsets *bridge.v.03*, *bridge.v.04*, *crossing.n.05*, *lake.n.03*, *bridge.v.01*, *metro.v.01*, *train.v.10*, etc., are mapped to the same region in the WordNet embedding space. Finally, for table embedding space, a selected region has the tables *table-1064-451*, *table-1064-381*, *table-1064-384*, *table-1064-402*, *table-1047-153*, *table-1047-143*, *table-0938-612*, etc., in the zoomed region after t-SNE visualization. All these tables are related to the 2012 Summer Olympics, and they show the list of world records in Olympics for multiple sport events.

E. Evaluation using the WebQueryTable corpus

TABLE IV: Table retrieval results for WebQueryTable.

Method	NDCG@5	MRR/MAP
Single-field ranking	0.5560	0.5362
Multi-field ranking	0.5849	0.5631
Conv-KNRM+Glove [27]	0.6004	0.5825
Conv-KNRM+fastText [41]	0.6097	0.5878
Conv-KNRM+Word2vec [19]	0.6072	0.5859
Conv-KNRM+TabVec [28]	0.6059	0.5856
Conv-KNRM+MCON [29]	0.5996	0.5798
MultiEm-RGCN	**0.6438**	**0.6200**

We also conduct experiments on WebQueryTable [17]. We compare the performance of our method against unsuper-

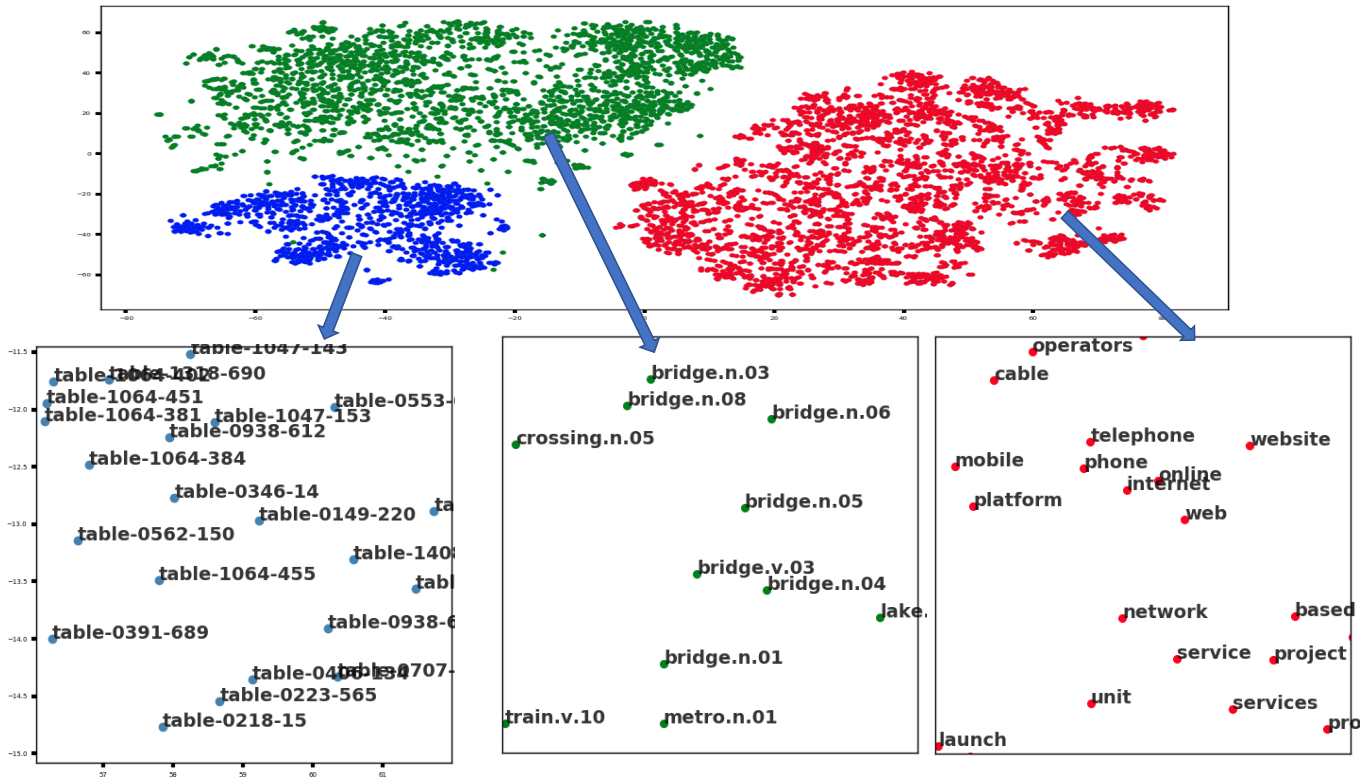

Fig. 3: The t-SNE visualization of MultiEm-RGCN embeddings. There are three spaces of embeddings: word (red dots), WordNet (green dots), and table (blue dots). For each type of embedding, we zoom a region to show the embeddings of nodes of our proposed graph.

vised and supervised baselines, except for LTR/STR because these methods require a wide range of features that are not provided in the dataset. Similar to WikiTables, we obtain three spaces of embeddings, which supports the hypothesis that MultiEm-RGCN simultaneously learns multiple types of embeddings from our heterogeneous graph. For the WebQueryTable dataset, there is only one relevant table per query, so MRR is always equivalent to MAP (and thus MRR and MAP are shown in the same column in Table IV). Consistent with WikiTables, our results on WebQueryTable show that incorporating multiple embeddings from MultiEm-RGCN into Conv-KNRM improves the evaluation metrics of table retrieval. This supports the hypothesis that MultiEm-RGCN captures rich semantic and lexical general knowledge from Glove and WordNet with data-specific knowledge when learning the embeddings of nodes. Then, as in WikiTables, our LTR model in MultiEm-RGCN combines matching signals from word, WordNet, and table nodes, which gives the possibility for query and table to be matched in multiple spaces.

VI. Conclusion

In this study, we propose a novel table retrieval method denoted by MultiEm-RGCN. We have shown that a relational graph convolution network that incorporates both dataset-dependent knowledge and dataset-agnostic knowledge outperforms the best previously published results in ad hoc table retrieval (STR) [15]. MultiEm-RGCN has two phases. The first phase consists of building a knowledge graph for a table corpus that contains multiple types of nodes and edges. This heterogeneous graph aims to capture data-agnostic knowledge that is semantic and lexical, and dataset-dependent knowledge that is derived from contextual information and term frequencies. A simple graph encoder with two R-GCN layers, and the DistMult decoder function are used to learn node embeddings by minimizing a link prediction loss function. The second phase consists of using R-GCN embeddings for the table retrieval task. This is achieved by building a new LTR architecture that combines word, WordNet, and table embeddings into one joint model that improves retrieval metrics.

Future work includes investigating what additional knowledge can make the graph more effective. For example, including information such as what terms appear in column heading or more techniques of value selection so that data values can be best used to improve retrieval results without substantially increasing training time. In addition, an interesting future direction consists of testing the transfer learning ability of MultiEm-RGCN embeddings using table collections from multiple domains and tasks. Finally, we would like to

explore generalizing the approach to non-tabular datasets, such as RDF dumps or XML files.

Acknowledgment

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

REFERENCES

[1] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 09 2017.

[2] H. Cai, V. Zheng, and K. Chang, "A comprehensive survey of graph embedding: Problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, 09 2017.

[3] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: A review of methods and applications," *ArXiv*, vol. abs/1812.08434, 2018.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[5] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," *Lecture Notes in Computer Science*, p. 593–607, 2018.

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

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

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

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

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

[17] 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.

[18] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, 2013, pp. 2787–2795.

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

[20] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Twenty-Eighth AAAI conference on artificial intelligence*, 2014, pp. 1112–1119.

[21] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015, pp. 2181–2187.

[22] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems*, 2013, pp. 926–934.

[23] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *32nd AAAI Conference on Artificial Intelligence*, 2018, pp. 1811–1818.

[24] P. Ristoski and H. Paulheim, "Rdf2vec: Rdf graph embeddings for data mining," in *International Semantic Web Conference*. Springer, 2016, pp. 498–514.

[25] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Product knowledge graph embedding for e-commerce," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 672–680.

[26] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Workshop Track Proceedings*, 2013.

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

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

[29] 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 (Big Data)*. IEEE, 2019, pp. 1238–1244.

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

[31] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 2224–2232.

[32] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. JMLR.org, 2017, p. 1263–1272.

[33] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, 2016, pp. 55–64.

[34] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, p. 39–41, Nov. 1995.

[35] B. Yang, S. W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, May 2015.

[36] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. JMLR.org, 2016, p. 2071–2080.

[37] Z. Dai, C. Xiong, J. Callan, and Z. Liu, "Convolutional neural networks for soft-matching n-grams in ad-hoc search," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 2018, p. 126–134.

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

[39] 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: Annual Conference on Neural Information Processing Systems*, 2014, pp. 2042–2050.

[40] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007)*, ser. ACM International Conference Proceeding Series, vol. 227. ACM, 2007, pp. 129–136.

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

[42] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems (TOIS)*, vol. 20, no. 4, pp. 422–446, 2002.