A Neural Bag-of-Words Modelling Framework for Link Prediction in Knowledge Bases with Sparse Connectivity

Fanshuang Kong
BDBC and SKLSDE, School of Computer Science and Engineering, Beihang University
Beijing, China

Richong Zhang*
BDBC and SKLSDE, School of Computer Science and Engineering, Beihang University
Beijing, China

Hongyu Guo
Institute for Information Technology of the National Research Council Canada
Ottawa, Ontario, Canada

Samuel Mensah
BDBC and SKLSDE, School of Computer Science and Engineering, Beihang University
Beijing, China

Zhiyuan Hu
College of Information Science and Technology, Beijing University of Chemical Technology
Beijing, China

Yongyi Mao
School of Electrical Engineering and Computer Science, University of Ottawa
Ottawa, Ontario, Canada

ABSTRACT

Knowledge graphs such as DBPedia and Freebase contain sparse linkage connectivity, which poses severe challenge to link prediction between entities. In addressing this sparsity problem, our studies indicate that one needs to leverage model with low complexity to avoid overfitting the weak structural information in the graphs, requiring the simple models which can efficiently encode the entities and their description information and then effectively decode their relationships. In this paper, we present a simple and efficient model that can attain these two goals. Specifically, we use a bag-of-words model, where relevant words are aggregated using average pooling or a basic Graph Convolutional Network to encode entities into distributed embeddings. A factorization machine is then used to score the relationships between those embeddings to generate linkage predictions. Empirical studies on two real datasets confirms the efficiency of our proposed model and shows superior predictive performance over state-of-the-art approaches.

CCS CONCEPTS

• **Information systems** → **Data extraction and integration**; *Information extraction*; • **Computing methodologies** → Semantic networks.

KEYWORDS

Knowledge base; Link prediction

ACM Reference Format:
Fanshuang Kong, Richong Zhang, Hongyu Guo, Samuel Mensah, Zhiyuan Hu, and Yongyi Mao. 2019. A Neural Bag-of-Words Modelling Framework for Link Prediction in Knowledge Bases with Sparse Connectivity. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313550

1 INTRODUCTION

Knowledge Bases (KBs) such as DBPedia [8], Yago [18] and Freebase [1] contain precious large collaborative intelligence, on which many state-of-the-art NLP applications heavily rely, including question -answering [5], entity disambiguation [3], and information retrieval [4], amongst many others [7, 22, 23]. Knowledge within such graphs is typically represented as a triple (h, r, t), which indicates that there is a relation r from head entity h to tail entity t.

Unfortunately, the complexity of the real-world makes these knowledge graphs far from complete. In particular, the structural information provided in the existing knowledge triples is typically very sparse, which poses a severe challenge for link prediction in such graphs. For example, in the DBPedia knowledge graph, there are 99.41% entities (42966066 of entities in total) with at most 43 relations. Similarly, the Yago graph contains 95.17% of entities(38734252 of entities in total) but with at most 39 linkages [25]. To cope with such data scarcity, it is desirable to exploit additional information contained in the text descriptions of the entities and relations. To date, the distributed representations of the entities and relations are primarily constructed via encoding their corresponding text information. However, since an entity is only observed to participate in very few relations, and in more extreme cases, to participate in no relations at all (which gives rise to the OOKB, namely "out-of-knowledge-base" link prediction problem), each training example (an entity-relation-entity triple) only contains very weak supervisory signal to train the text encoder. Furthermore, this problem can be aggravated when the number of entities and the vocabulary size of the text are large, which is the case for current popular knowledge graphs. In such a scenario, one is naturally motivated to use a simple text encoder as possible so as to maintain a low model complexity and avoid over-fitting resulting from data sparsity.

[*] Corresponding author: zhangrc@act.buaa.edu.cn

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313550

In this paper, instead of promoting the use of sophisticated text encoders such as convolutional neural networks (CNNs), we advocate the philosophy of returning to the more basic bag-of-words models. Specifically, we propose a neural bag-of-word (NBOW) modelling framework, which contains three simple modules: selecting keywords to form bags of words, encoding entities from the bags of words, and computing the score of a triple using a factorization machine. To this end, we present two concrete examples in this framework, in which the entity encoding is carried out by simply averaging the word vectors in its bag of words or further modelled using a graph convolutional network (GCN) respectively. Our experimental results on two datasets, both old and new, demonstrate that a bag-of-words model is more effective than a CNN method in terms of entity encoding for link prediction in sparse knowledge graphs. In particular, we empirically show that, such simple models outperform significantly more sophisticated models, including the current art method ConMask in link prediction over sparse knowledge bases, for out-of-knowledge-base (or "open-world") link prediction. Our simple models have established new state-of-the-art results for the two benchmark datasets.

The rest of this paper is organized as follows: In Section 2, we present prior works on link prediction in knowledge base completion. In Section 3, we describe our model for entity encoding and relation scoring for knowledge base instances. In Section 4, we perform an experimental comparison of current methods against our model. We perform these experiments on the datasets DBPedia50K[17] and FB15K-237* which we generate from FB15K-237. We show that our method for link prediction is scalable and outperforms 'state-of-the-art' method, ConMask [17] in link prediction performance for the open-world assumption. We give concluding remarks in Section 5.

2 RELATED WORKS

Knowledge Base Completion (KBC) is one of the most important tasks in NLP. Many efforts have been invested in the link prediction task of KBC. For a typical link prediction model, we have an encoder and a decoder. An encoder learns feature representations of entities or relations and the decoder exploits these representations to predict the truth value of a given triple.

One of the first works of KBC falls under translation-based models [2, 9, 20]. These models generally adopt the same principle, $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, if the triple (h, r, t) is indeed a fact. More specifically, TransE [2] assumes a score funtion

$$f_r(h, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2^2 \tag{1}$$

which captures one-one relations, while other expressive translation-based models [9, 20] are capable of capturing other relations such as one-many, many-one and many-many by introducing a relationship-dependent parameter for the translation of head and tail entities.

Aside translational models, we have models such as ANALOGY [10] and its sub-models HolE [11], ComplexE [19], DistMult [24] addressing the problem of link prediction. ANALOGY addresses the problem by imposing analogical properties for the multi-relational encoding of entities and relations. ANALOGY decodes by using a relation-specific normal matrix W_r as a composition operator on the head and tail entity embeddings. The goal of ANALOGY is to receive a high score for a factual triple (h, r, t), where the score is formulated as

$$f_r(h, t) = \mathbf{h}^T W_r \mathbf{t} \tag{2}$$

ProjE[16] views the link prediction task as a ranking problem. ProjE is a neural model with a combination layer and a projection layer. Given a tail entity, the combination layer combines the tail entity embedding with a relation embedding. The resulting embedding vector is projected to all candidate head entities to compute a score for the head entity. The goal is to ensure that, top-ranked entities are the correct entities in the relation. The score for a candidate head h given a tail entity t and relation r is formulated as

$$f_h(t, r) = g(\mathbf{h} f(\mathbf{t} \oplus \mathbf{r}) + b_p) \tag{3}$$

where f and g are activation functions and $\oplus$ is a learnable combination operator.

Despite the promising successes in link prediction, the above model architectures restrict representation learning on the KB's structure. This constrains learning for only entities and relations in the KB. It also makes it non-trivial to leverage information captured in descriptions of entities which are usually accompanied with entities. Some recent works have addressed these issues and have achieved a boost in link prediction performance [17, 21].

DKRL [21] takes into account the textual information accompanied with entities. DKRL proposed a continuous bag-of-words model (CBOW) and a convolutional neural network (CNN) to build representations for word-based entity descriptions. Representations for RDF triples are learned from the structure of the KB. With the representations for both RDF triples and entity descriptions, they employ a translation-based model for decoding in the link prediction problem. ConMask [17] on the other hand uses a relationship-content mask, semantic averaging and a CNN to learn relationship-dependent embeddings for word-based entities. ConMask outperforms DKRL in link prediction at the expense of a high order of magnitude in parameters. In both models, they employ a CNN to assist in extracting representations for word-based entities. However, we show that a simpler framework, specifically, a neural bag-of-words model can be used to learn representations for word-based entities more efficiently.

In the present work, we develop a neural bag-of-words model, a simple model for link prediction in knowledge bases. We model the entity encoding for link prediction by constructing a bipartite graph of word-entity connectives, where entities are represented by their names or part of its descriptions (see Figure 1). Motivated by some recent works [15, 26] which employs a GCN to deal with highly multi-relational data such as knowledge graphs, we develop the encoder that operates directly on the bipartite graph and further employ a factorization machine for the link prediction task.

3 NEURAL BAG-OF-WORDS MODEL

3.1 Problem Definition

A Knowledge Base is a graph $\mathcal{G} = \{E, R, T\}$, where E is a set of entities-nodes, R is a set of relations-edges and T is a set of factual triples. $T = \{(h, r, t), h, t \in E, r \in R\}$. An entity of a triple may have a relevant textual description. We call this a snippet. We respectively denote h_s, t_s as the snippets of h, t. We denote (h', r, t') as a negative triple, i.e. a triple which does not exist in a knowledge base.

Similarly, snippets of entities of a negative triple is also denoted as h'_s and t'_s.

Our NBOW approach aims to find missing factual triples

$$T' = \{(h, r, t) | (h, r, t) \notin T, h, t \in E', r \in R\},$$

where E' is a set of entities such that $E \subset E'$. That is, OOKB entities belong to the set $E' \backslash E$.

For the link prediction task, the NBOW method consists of two steps: 1) encoding an entity's name, along with its snippet, and 2) predicting the possible linkage between encoded entities. We will discuss the two steps in the next two sections.

3.2 Entity Encoding

The entity encoding process aims to first extract keywords from an entity's name or description, resulting in a bag of words for each entity. The information in the bag of words is aggregated to form the distributed representation for the entity's name or its snippet. The aggregation can be efficiently achieved by either simply averaging the word vectors in its bag or leveraging a graph convolutional network to better capture the structural information of the words in the bag.

3.2.1 Content Extraction. We first extract relevant words from the entity's description. To this end, we use TF-IDF to highlight the important words (keywords) in the entity's description. Other works, such as ConMask [17] use a relation-dependent mask to extract relevant text given a relation. Unlike their model, the extracted content using TF-IDF can simultaneously involve several relations. We address this challenge by using a relation-attentive factorization model (we will discuss this in Section 3.3.1).

After having all distinct words from entity names and their snippets, we then construct a word embedding matrix $V \in \mathbb{R}^{l^w \times d^w}$ where l^w is the size of the vocabulary and d^w is the word embedding dimension. For a given triple (h, r, t), we build entity name representations $\mathbf{h}$ and $\mathbf{t}$ for both h and t. Each draws values from $\mathbb{R}^{d^w}$. We do this by aggregating the information in the bag for the entity, either by simply averaging the embeddings in the bag or leveraging a graph convolutional network to better infuse the knowledge in the bag. Similarly, we build entity snippet representations $\mathbf{h_s}$ and $\mathbf{t_s}$ for both h_s and t_s.

3.2.2 Aggregate Bag-of-Words with Average Pooling. The aggregation process (ψ) is defined as a function $\psi : S \rightarrow \mathbb{R}^{d^w}$, mapping a set of vectors drawn from $\mathbb{R}^{d^w}$ to the vector space $\mathbb{R}^{d^w}$. For n vectors, S can be represented as the set $S = \{v_1, v_2, \ldots, v_n\}$. The encoder $\psi(S)$ can take the following forms:

$$\psi(S) = \sum_{i=1} v_i \quad \text{(sum pool)} \tag{4}$$

$$\psi(S) = \frac{1}{n} \sum_{i=1} v_i \quad \text{(average pool)} \tag{5}$$

$$\psi(S) = \max(S) \quad \text{(max pool)} \tag{6}$$

where the function max(·) is an elementwise max function [6]. In this paper, the average pool function is chosen due to its superior performance.

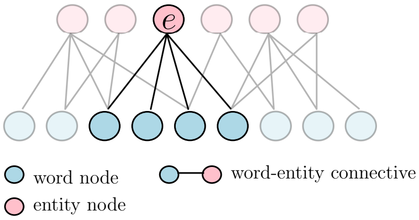

Figure 1: A bipartite graph induced from the KB. A word node is given by its word embeddings. An entity node is a word-based entity given by a feature vector resulting from the aggregation of word embeddings of its neighbors. An entity is connected to a word if that entity contains that word. As an example, the highlighted neighbors of entity e is aggregated using a continuous-bag-of-words encoder to form a feature representation for e.

3.2.3 Aggregating Bag-of-Words with GCN. Although one can leverage the simple average pool for aggregating information out of the bag-of-words model, other simple strategies can be used to further improve the information aggregation. To this end, we devise a basic graph convolutional network (GCN) model for this purpose.

We make use of word-based entities to induce two bipartite graphs $\mathcal{G}'_1$ and $\mathcal{G}'_2$ from $\mathcal{G}$ (see Figure 1). $\mathcal{G}'_1$ is a bipartite graph decomposed into word nodes and nodes for entity names, given by real-valued vectors. Similarly, $\mathcal{G}'_2$ is also decomposed into word nodes and nodes for relevant entity descriptions, given by real-valued vectors. Due to the similarity of the structure of $\mathcal{G}'_1$ and $\mathcal{G}'_2$, we denote either one of the bipartite graphs as $\mathcal{G}'_e$. Nodes are connected in $\mathcal{G}'_e$ if they share common words. Here and throughout the present work, we represent a word-based entity (entity name or entity snippet) as the italic word *entity* for simplicity and clarity.

The input to the GCN model is the bipartite graph $\mathcal{G}'_e$, where word nodes are initialized with pretrained word embeddings and *entity* nodes are initialized by the aggregation of the feature vectors of its neighbors. The GCN model proposed here makes efficient use of edges of subgraphs of $\mathcal{G}'_e$ to propagate and share information between nodes. More precisely, edges in $\mathcal{G}'_e$ serves as a transmission medium which has the sole purpose of propagating information to and from word nodes to *entity* nodes for the update of their embeddings. The GCN model is equipped with a CBOW encoder which extract *entity* embeddings for *entity* nodes by aggregating the word embeddings of its neighbors. The neighborhood of any node in $\mathcal{G}'_e$ is a bipartite subgraph which can be interpreted as an information propagation neural network. These subgraphs share parameters defining how information is transformed and propagated within the graph. The GCN encoder model learns convolution operators on these subgraphs for the transformation of information and the propagation across their edges to update embeddings of nodes.

When the GCN model operates on $\mathcal{G}'_e$, taking into account only the first-order neighborhood of nodes (word, *entity*), we learn parameters for a single update of the node embeddings. At each convolutional layer, convolution operations are imposed on the graph to update word nodes and *entity* nodes. To learn more informative representations, we successively apply convolutions on $\mathcal{G}'_e$

across the k-th order neighborhood. The GCN model architecture is illustrated in Fig. 2.

A single update for a word node takes the following form:

$$h_w^{k+1} = \phi\left(\sum_{j \in \mathcal{N}_w} \frac{1}{c_w} w_j^k h_j^k,\ w_w^k h_w^k\right) \tag{7}$$

Similarly, an update for an *entity* node takes the following form:

$$h_e^{k+1} = \phi\left(\sum_{j \in \mathcal{N}_e} \frac{1}{c_e} w_j^k h_j^k,\ w_e^k \psi(\mathcal{N}_e)\right) \tag{8}$$

where $(\,\cdot\,,\,\cdot\,)$ is an element-wise product, $h_e^k, h_w^k \in \mathbb{R}^{d^w}$ are the hidden state representation of *entity* e and word w respectively for the k-th convolutional layer of the neural network. $\mathcal{N}_w, \mathcal{N}_e$ correspond to the neighborhood of word w and *entity* e respectively. $\frac{1}{c_e}$, $\frac{1}{c_w}$ are normalization constants, where $c_e = |\mathcal{N}_e|$ and $c_w = |\mathcal{N}_w|$. $\psi(\cdot)$ is the bag-of-words model which we discussed in the previous section. $\phi(\cdot)$ is a non-linear activation function. For our model, we choose tanh function as the activation function $\phi(\cdot)$. w_j^k is a trainable parameter on layer k for the node j.

In short, for a given triple (h, r, t) with corresponding entity snippets h_s and t_s, the GCN model outputs real-valued representations $\mathbf{h}, \mathbf{t}, \mathbf{h_s}$ and $\mathbf{t_s}$.

3.3 Relation Scoring

To score the linkage predictions, we leverage a simple factorization machine which aggregates scores for different pairs of representations of a given triple. Traditional methods for interaction pair selection, specifically, [2, 20] measure scores of a candidate triple (h, r, t) by calculating the distance between the embedding $\mathbf{h}+\mathbf{r}$ and $\mathbf{t}$. Other expressive relation-dependent operators have successfully measured the score on candidate triples [11, 19].

The main assumption in these methods is that, there is a relation specific interaction which captures the relation r between h and t of any given triple. However, the presence of additional features h_s and t_s, the relevant descriptions of h and t offers new opportunities. ConMask incorporates some of these ideas by fusing different pair of variables in their model and achieve state-of-the-art results. In our approach, we postulate that, the pairs $h{-}r$, $t{-}r$, $h{-}t$, $h{-}t_s$, $h_s{-}t_s$ and $h_s - t$ captures informative interactions which can improve the link prediction task.

3.3.1 Relation-Attentive FM. We present a relation-attentive factorization model, a decoder which measure scores for different pair of representations. The decoder is likened to a factorization machine [13, 14] which highlights strong relationships between a pair of variables using a relation interaction parameter.

To capture interactions of pairs, we recall a training instance as a triple $(h, r, t) \in T$ with additional features h_s and t_s denoting snippets of the head entity h and tail entity t. Let $\mathcal{X}$ denote a training instance consisting of the set of variables $\{h, r, t, h_s, t_s\}$. We model the link prediction problem by measuring the overall scores of $h - r$ interactions, $t - r$ interactions, $h - t$ interactions, $h - t_s$ interactions, $h_s{-}t_s$ interactions and $h_s{-}t$ interactions for a training instance. Here, we extract *entity* embeddings from the encoder and initialize the relation r with a random vector. We define an interaction set I consisting of the 6 pair of interaction variables. For a given unordered pair $\{e_1, e_2\} \in I$, where $e_1, e_2 \in \mathcal{X}$, we formulate a relation interaction parameter γ(different from the embedding of relation r) which focuses on relation-specific parts of the interacting pair to compute a response. We define a score for the interaction of a pair as

$$g_\gamma(e_1, e_2) = \langle \mathbf{e_1} \odot \mathbf{e_2}, \gamma \rangle \tag{9}$$

The score assigns a scalar value to show the strength of interaction of a pair. This architecture enables the attention on latent information of a pair of variables which is closely related to the relation itself. This process is similar to that of ConMask, where a relationship-dependent content mask highlights relevant text at an early stage. We aggregate scores across all pairs in I. The total score for a training instance takes the form

$$f(\mathcal{X}) = \sum_{\{i,j\} \in I} g_\gamma(i, j) \tag{10}$$

The goal is to learn the relation interaction parameter γ such that, the model score high values for factual triples and low values for negative triples. See figure 3 for an illustration of this model.

3.3.2 Optimization Objective. For the model to distinguish factual triples from negative triples, we evaluate the model by generating a set of negative triples for each factual triple and add to the training set T. The optimization function we wish to minimize is

$$\mathcal{L} = -\sum_{\mathcal{X}, y} y \log \sigma(f(\mathcal{X})) - (1 - y) \log(1 - \sigma(f(\mathcal{X}))) \tag{11}$$

where $\sigma(\cdot)$ is a softmax activation function, y is a binary value set to $+1$ if we have a factual training instance and 0 if we have a negative training instance.

4 EXPERIMENTS

In this section, we evaluate our NBOW model. The two versions of our model are denoted as AFM, where entity embeddings are extracted by aggregating relevant word embeddings using a CBOW encoder, and GCN+AFM, where the aggregated word embeddings are further enhanced using a GCN. In both models, we employ a factorization machine for the link prediction task in the open-world assumption. We propose both models to observe the boost in performance when we employ a GCN to model the noisy dependency of the entity embeddings. We compare our simple methods to the current state-of-the-art strategy ConMask and two more competitive baselines. In our evaluation, we report measures for Filter Mean Rank (FMR), Filtered Hit@10 (FHIT), Filtered Mean Reciprocal Rank (FMRR).

4.1 Datasets

We perform experiments on an extracted DBPedia dump; DBPedia50K introduced in [17]. DBPedia50K originally contains 32K entities. There are 28K redundant entities, thus, these entities have no factual triples in the dataset. Hence, they do not influence results for any model but just lead to a statistic for the number of entities as shown in Table 1. DBPedia50K is very different from some standard dataset like FB15K and FB15K-237. DBPedia50K has a sparse KB structure and a dense GCN graph structure. But FB15K and FB15K-237 have a dense KB structure and sparse GCN graph structure. To validate the efficiency of our model on the open-world

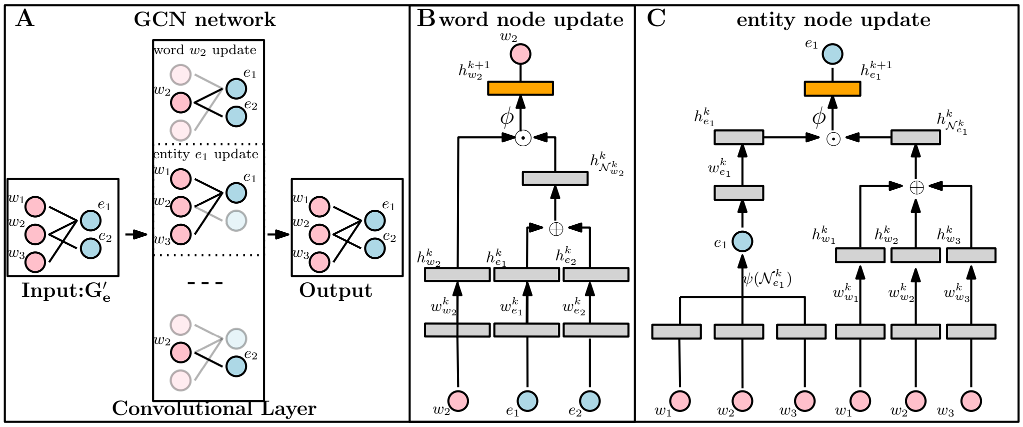

Figure 2: (A) Overview of a single layer GCN model architecture with first-order filters for word-based entity encoding. (B) Shown is a first-order filter imposed on the neighborhood of the word node (pink) to transform and propagate information from *entity* nodes (blue) to learn a representation for the word node. (C) Similar to that of B for information transformation, shown here is the propagation of information from the neighborhood of *entity* node to update its representation. Before update, a CBOW encoder ϕ aggregates the neighbors of the *entity* node resulting to an initial *entity* representation for the learning process. $\oplus$ is a vector summation operator and $\odot$ is an element-wise operator (B and C is based on the example given in A and follows Eq. 7 and 8 respectively)

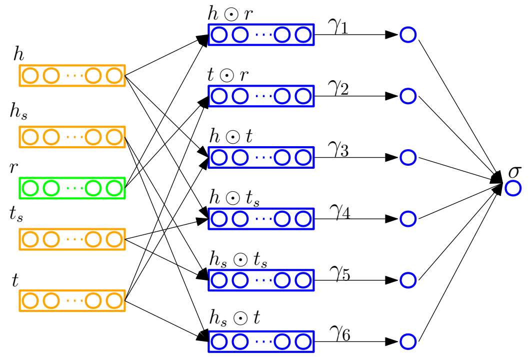

Figure 3: Relation-attention FM architecture

assumption for link prediction, we introduce a new dataset based on FB15K-237. We refer to this new dataset as FB15K-237*. FB15K however has many reciprocal triples so we don't generate an open world version of it.

FB15K-237* is generated by randomly selecting 10% of entities in the KB. These selected entities are classified as open-entities. Triples in the KB which contain these open-entities are held out for test and validation. All other triples are used for training. We define the validation set on 10% of the held out triples, the remaining 90% is used for the test set. The statistics of all datasets are shown in Table 1.

4.2 Parameter Settings

For a comparable study, we use similar parameters used in ConMask. We set the word embedding size $k = 200$ and draw embeddings from 200-dimensional trained GloVe vectors which have been made publicly available [12]. Maximum length for entity name $k_n = 32$, maximum length for entity snippet $k_c = 128$. Given a training instance $\mathcal{X} = \{h, t, h_s, t_s, r\}$, the encoder extracts *entity* embeddings for h, t, h_s, t_s and initializes a random vector for r. The relation-attentive model learns interactions from 6 unordered 2-tuples in the set $\{\{h, r\}, \{t, r\}, \{h, t\}, \{h, t_s\}, \{h_s, t_s\}, \{h_s, t\}\}$. During training, our neural network is composed of 1 convolutional layer as seen in (A) of Fig. 2. The batch size we use is $k_b = 200$ and we choose Adam as our optimizer with learning rate $lr = 0.01$. For a factual triple, we have 10 negative triples constructed by randomly corrupting either the head or tail entity. We train our model for 200 epochs.

4.3 Open-World Link Prediction

We follow evaluation protocols used in ConMask [17] for a comparable analysis. For the open-world link prediction task, we aim to predict missing entities h or t for a given factual triple (h, r, t). Given a partial triple $(?, r, t)$ or $(h, r, ?)$, this task requires that we rank a set of candidate entities for the position of missing entity. We refer to this missing entity as the target entity e. At the testing phase, for each partial test triple $(?, r, t)$ or $(h, r, ?)$, we apply a filtering method to rank specific entities. The filter skips all target entity candidates which have never been connected to the relation r in the training set. As noted by [17], it avoids the ranking of relation instances which might be nonsensical. For every target entity selected, we compute a similarity score according to our proposed score function f on the training instance $\mathcal{X}$ for the candidate triple (e, r, t) or (h, r, e). We use these scores to rank entities e in descending order and measure the performance of the model with the following evaluation metrics: (1) the average rank of correct entities (denoted as Filtered Mean Rank), (2) the proportion of correct entities in the top 10 of all entities (denoted as Filtered Hits@10), and (3) a filtered mean reciprocal rank. Note that, an effective model scores a *low* filtered mean rank (FMR), a *high* filtered mean reciprocal rank (FMRR) with a *high* Flitered Hits@10 (FHIT).

Table 1: Dataset Statistics: #Closed Ent. is the number of entities in the KB, #Open Ent. is the number of OOKB entities & #Words is the number of words we learn embeddings for the word-based entity embedding extraction task

Dataset	#Ent.	#Rel.	#Train	#Valid	#Test	#Closed Ent.	#Open Ent.	#Words.
FB15k-237*	14,438	237	241,962	6270	56,431	12,907	1,531	42,780
DBPedia50K	30,449	365	32,388	399	10,969	24,624	5,825	48,185

Table 2: Open-World Performance on DBPedia50K and FB15K-237*

Model	DBPedia50K						FB15K-237*					
	Head			Tail			Head			Tail		
	FMR	FHIT	FMRR	FMR	FHIT	FMRR	FMR	FHIT	FMRR	FMR	FHIT	FMRR
Target Filtering Baseline	573	0.08	0.04	104	0.23	0.11	474	0.1	0.05	292	0.22	0.12
DKRL	490	0.09	0.08	70	0.40	0.23	-	-	-	-	-	-
ConMask(Re-Run)	95	0.39	0.35	16	0.81	0.61	146	0.37	0.2	88	0.53	0.34
AFM	94	0.52	0.35	13	**0.83**	**0.64**	**122**	0.44	0.28	93	0.52	0.34
GCN+AFM	**89**	**0.56**	**0.39**	**12**	**0.83**	0.63	125	**0.47**	**0.31**	**85**	**0.54**	**0.35**

We evaluate AFM and GCN+AFM on the link prediction task in open-world KBC for the datasets summarized in table 1. We report results for models AFM, GCN+AFM, ConMask and other baseline methods implemented in the works of [17]. [17] re-implemented the method for DKRL [21] to handle the OOKB link prediction task. The overall performance of the models on DBPedia50K and FB15K-237* are shown in Table 2. The best performance are bold-faced.

As we can observe, AFM generally achieves better performance than the baseline methods on both datasets. More importantly, it shows evidence of superior performance to the state-of-the-art method ConMask in both DBPedia50K and FB15K-237*. To compare the detail performance metrics, we observe that AFM managed to outperform ConMask entirely on DBPedia50K, showing a significant boost for FMR, FHIT and FMRR in both head and tail target entity prediction. On the FB15K-237* dataset, the results are competitive for both ConMask and AFM. Here, we observe that AFM impressively outperforms ConMask when predicting head target entities. On the other hand, for the tail target entity prediction, AFM has a slight degradation in performance (lower by 0.01) in terms of FHIT, a higher FMR (higher by 5), and equal FMRR when comparing to ConMask. Promisingly, when we augment AFM with the GCN, namely the GCN+AFM model, it would be reasonable to expect a boost in predictive performance since we expect the GCN to enhance the entity embeddings. We clearly observe that this is the case, GCN+AFM outperforms all baselines including ConMask in every metric on the two datasets.

4.3.1 Scalability. In our efficiency analysis, we calculate the computational time for training an epoch (measured in seconds) on the DBPedia50K dataset. All training is done on a GeForce Titan Xp system with 2×12GB memory. We compare training times of our models and that of ConMask in particular. This result is significant because knowledge bases are known to be usually large in size and will therefore have a large number of training instances. For example, DBPedia500K [17] has $3,102,677$ training instances. With this large number of instances, state-of-the-art models like ConMask with complex architectures requiring the tuning of hundreds of hyperparameters will take a longer time to train in 1 epoch. Models for KBC tasks must therefore be as simple as possible but efficient to learn large KBs. From the analysis, we observed that ConMask used $80s$ to complete an epoch training on DBPedia50K, while GCN+AFM required only $50s$ and AFM executed much faster as expected. Our proposed methods reduces about 40% of execution time when compared to that of ConMask. These results suggests that our simple methods have better scalability to large knowledge graphs. The results also informs us that our model makes better use of its parameters, and benefits from simplicity and efficiency compared to ConMask which has a significant model complexity.

5 CONCLUSION

In this study, we advocate the philosophy of exploiting simple models for link prediction in large knowledge graphs due to their sparse linkage connectivity between entities. We propose a simple neural bag-of-words model for link prediction in large knowledge graphs, named GCN+AFM, which accounts for both entity representation and link prediction. Specifically, our proposed model utilizes TF-IDF and average pooling and then enhances entity representations via a GCN. Experimental results show that the two versions of our model, namely AFM and GCN+AFM, achieve superior performance and reduce an order of magnitude in the number of parameters needed and about 40% execution time when compared to the current art model. We also demonstrate that the simple version of our proposed model, named AFM, achieves better performance to state-of-the-art methods. These facts make our proposed strategy suitable for learning from large real-world knowledge graphs with sparse entity connectivity.

ACKNOWLEDGMENTS

This work was supported in part by the China 973 Program under Grant 2015CB358700, in part by the National Natural Science Foundation of China under Grant 61772059 and Grant 61421003, in part by the Beijing Advanced Innovation Center for Big Data and Brain Computing, and in part by the State Key Laboratory of Software Development Environment.

REFERENCES

[1] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. 1247–1250.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[3] Silviu Cucerzan. 2007. Large-Scale Named Entity Disambiguation Based on Wikipedia Data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 708–716.

[4] Jeffrey Dalton, Laura Dietz, and James Allan. 2014. Entity query feature expansion using knowledge base links. In *Proceedings of the 37th international ACM SIGIR conference on Research and development in information retrieval*. 365–374.

[5] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. 2010. Building Watson: An Overview of the DeepQA Project. *Ai Magazine* 31, 3 (2010), 59–79.

[6] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge Transfer for Out-of-Knowledge-Base Entities : A Graph Neural Network Approach. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*. 1802–1808.

[7] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum. 2008. NAGA: Searching and Ranking Knowledge. In *2008 IEEE 24th International Conference on Data Engineering*. 953–962.

[8] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and others. 2015. DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* 6, 2 (2015), 167–195.

[9] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2181–2187.

[10] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-relational Embeddings. *international conference on machine learning* (2017), 2168–2178.

[11] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic embeddings of knowledge graphs. *national conference on artificial intelligence* (2016), 1955–1961.

[12] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543.

[13] Steffen Rendle. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*.

[14] Steffen Rendle. 2012. Factorization Machines with libFM. *ACM Transactions on Intelligent Systems and Technology* 3, 3 (2012), 57.

[15] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, Ivan Titov, Max Welling, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. *extended semantic web conference* (2018), 593–607.

[16] Baoxu Shi and Tim Weninger. 2016. ProjE: Embedding Projection for Knowledge Graph Completion. *arXiv preprint arXiv:1611.05425* 1 (2016).

[17] Baoxu Shi and Tim Weninger. 2018. Open-World Knowledge Graph Completion. *national conference on artificial intelligence* (2018).

[18] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*. 697–706.

[19] ThÃĽo Trouillon, Johannes Welbl, Sebastian Riedel, ÃĽric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. (2016).

[20] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI'14 Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 1112–1119.

[21] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2659–2665.

[22] Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit Semantic Ranking for Academic Search via Knowledge Graph Embedding. In *WWW '17 Proceedings of the 26th International Conference on World Wide Web*. 1271–1279.

[23] Bishan Yang and Tom M. Mitchell. 2017. Leveraging Knowledge Bases in LSTMs for Improving Machine Reading.. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1436–1446.

[24] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *international conference on learning representations* (2015).

[25] Xiaowang Zhang, Mingyue Zhang, Peng Peng, Jiaming Song, Zhiyong Feng, and Lei Zou. 2018. gSMat: A Scalable Sparse Matrix-based Join for SPARQL Query Processing. *arXiv preprint arXiv:1807.07691* (2018).

[26] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. 2018. Modeling Polypharmacy Side Effects with Graph Convolutional Networks. *intelligent systems in molecular biology* 34, 13 (2018), 258814.