Knowledge Graph Embedding using Graph Convolutional Networks with Relation-Aware Attention

Nasrullah Sheikh[*]
IBM Research - Almaden
San Jose, CA
nasrullah.sheikh@ibm.com

Xiao Qin[*]
IBM Research - Almaden
San Jose, CA
xiao.qin@ibm.com

Berthold Reinwald
IBM Research - Almaden
San Jose, CA
reinwald@us.ibm.com

Christoph Miksovic
IBM Research - Zurich
Switzerland
cmi@zurich.ibm.com

Thomas Gschwind
IBM Research - Zurich
Switzerland
thg@zurich.ibm.com

Paolo Scotton
IBM Research - Zurich
Switzerland
psc@zurich.ibm.com

ABSTRACT

Knowledge graph embedding methods learn embeddings of entities and relations in a low dimensional space which can be used for various downstream machine learning tasks such as link prediction and entity matching. Various graph convolutional network methods have been proposed which use different types of information to learn the features of entities and relations. However, these methods assign the same weight (importance) to the neighbors when aggregating the information, ignoring the role of different relations with the neighboring entities. To this end, we propose a relation-aware graph attention model that leverages relation information to compute different weights to the neighboring nodes for learning an embeddings of entities and relations. We evaluate our proposed approach on link prediction and entity matching tasks. Our experimental results on link prediction on three datasets (one proprietary and two public) and results on unsupervised entity matching on one proprietary dataset demonstrate the effectiveness of the relation-aware attention.

CCS CONCEPTS

• **Computing methodologies** → **Knowledge representation and reasoning**; **Learning latent representations**; **Unsupervised learning**.

KEYWORDS

Knowledge Graphs, Embedding, Graph Attention

ACM Reference Format:
Nasrullah Sheikh, Xiao Qin, Berthold Reinwald, Christoph Miksovic, Thomas Gschwind, and Paolo Scotton. 2020. Knowledge Graph Embedding using Graph Convolutional Networks with Relation-Aware Attention. In *San Diego '20: 2nd International Workshop on Deep Learning on Graphs: Methods and Applications, August 23–27, 2020, San Diego, CA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

[*]Both authors contributed equally

1 INTRODUCTION

Knowledge Graphs (KG) represent facts in the form of entities and relations between them. A fact is represented by a triplet (h, r, t) where h, t represent the head and tail entities respectively, and r represents the relation between h and t. Furthermore, entities and relations may have some additional information such as attributes associated with them. Data from different domains such as enterprises, gene ontology, etc. can be modeled as KGs which is useful in different applications. KGs are critical to enterprises as they enable an organization to view, analyze, derive inferences, and build up knowledge for competitive advantage; for example, discovering new links between entities is useful in many scenarios such as discovering new side effects of a drug, or establishing new corporate relationships. One of the biggest challenges is to extract the data from various structured and unstructured sources and build it in a KG such that it can be used effectively in various tasks such as search and answering, entity matching, and link prediction. An example of an enterprise knowledge graph is shown in Figure 1. The graph shows companies, subsidiaries, products, industry types, and product types represented as entities; *produces*, *in_industry*, *subsidiary_of*, *acquired*, and *is_a* represent the relationships between the entities.

Often these KGs are sparse and have missing information. For example, in Figure 1, the relation between *Tesla Inc* and *Automotive* is missing (<*Tesla Inc, ?, Automotive*>). In general, the missing information in KGs can be of the form $(h, r, ?)$, $(?, r, t)$, and $(h, ?, t)$. Towards this end, various methods have been proposed such as [4, 6, 11, 14, 21] which learn embeddings of entities and relations, and use a scoring function to determine if a triplet (h, r, t) is valid or not. Models such as DistMult [21] process each triplet independent of other triplets, and hence do not exploit the neighborhood information in learning. Graph convolution based methods overcome this problem by aggregating the features from the neighboring entities and applying a transformation function to compute the new features. But these graph-based methods give equal weights to each of the neighboring entities, ignoring that the neighbors have different significance in computing new features [15]. This attention mechanism considers edges having the same type, thus,

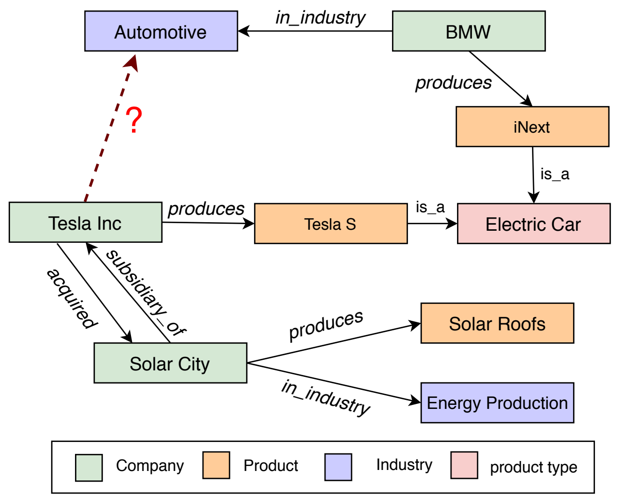

Figure 1: An example of an enterprise Knowledge Graph having 3 entity types and 5 link types. The red dotted arrow is the missing link information.

it cannot be directly extended to knowledge graphs which have multiple relation types between entities.

In a knowledge graph, relation types between entities determine the semantics of an edge. This semantic information is crucial in various downstream tasks such as link prediction and entity matching. Therefore, the relationship types cannot be ignored in computing the importance of neighbors. Towards this end, we introduce RelAtt– a relation aware masked attention mechanism in knowledge graphs which includes the features of relation for computing the attention. This attention is applied to the messages from neighbors during the propagation phase of graph neural networks to learn the embedding of entities and relations. The learning is optimized through a scoring function which scores a valid triplet higher than an invalid triplet based on the representation of entities and relations. Moreover, our proposed model is inductive, i.e., the learned model can be used to infer embeddings of unseen nodes. We evaluate our proposed approach on link prediction using two public datasets and one proprietary dataset against the state-of-the-art baselines. We also evaluate the embeddings on unsupervised entity linking task on the proprietary dataset.

The rest of the paper is organized as follows: Section 2 presents the related work, and Section 3 provides a brief description of Graph Neural Networks. In Section 4, we describe our proposed model, Section 5 describes the procedure to train the model. In Section 6, we describe the evaluation on link prediction task (Subsection 6.1) and entity matching task (Subsection 6.2) which includes details of datasets, experimental settings and results respectively. Finally, we conclude in Section 7.

2 RELATED WORK

Knowledge graph embedding methods have drawn a lot of attention due to recent advancements in representation learning. KG embedding methods learn a representation of entities and relations, and these representations are used for various downstream machine learning tasks such as KG Completion [18] and entity classification [11]. Knowledge graph embedding methods for link prediction can be classified into two groups: *translational and semantic models* and *neural network models*. Moreover, these methods either use only the observed facts or exploit additional information to learn better embeddings such as entity types [19] and logical rules [17].

Translational and Semantic Models. use a different parameterization of entities or relations, and scoring functions to determine the plausibility of a fact in a knowledge graph. Given a triplet (h, r, t), translation-based methods [1, 8, 18] use relation as a translational vector and apply a translation by relation on h. For a fact to hold, the embedding of translated entity h should be close to t i.e, the distance between these two should be minimum. TransE [1] represents entities and relations in same embedding space and uses a margin-based distance scoring function. TransE cannot handle one-to-many and many-to-many relations. To overcome this, various other translation-based methods have been proposed such as [8, 18]. TransR [8] represents each relation in a different relational space and both entities (h, t) are first translated into the relation space and then distance function is applied to check the validity of a triplet. On the other hand, semantic-based models [10, 14, 21] classify triplets based on a similarity function. These models fail to capture the complex relationships and are also limited to learn expressive features due to their shallow structure.

Neural Models. - use neural network models [3, 4, 6, 9, 11] to learn better embeddings of entities and relations which are then subsequently used in downstream tasks. Dong et al. [6] proposed a multi-layer perceptron based approach where embeddings of h, r, t are concatenated at the input layer and non-linear transformation is applied to classify the triplets. Convolutional Neural Network-based approaches have been proposed such as ConvKB [9] and ConvE [4]. The entities and relations in ConvKB and ConvE are represented in 1D and 2D respectively, and both employ a convolutional operation to learn embeddings of entities and relations, which are used by a scoring function for classification of triplets. Schlichtkrull et al. [11] proposed RGCN - a relational graph convolutional neural network model which uses a message-passing approach to aggregate the features from neighboring entities to learn the embedding of entities and relations to use in entity classification and link prediction tasks. These models give equal weights to the neighbors of a node which limits the learning of quality embeddings because the neighborhood nodes have different relationships and significance.

3 BACKGROUND

For the sake of completeness, in this section we provide a brief description of message passing based Graph Neural Networks (GNNs). For each node, a message passing GNN [20] iteratively aggregates representation from its neighbors. Each iteration defines a layer of GNN, and l iterations (layers) encode the structural information of the graph within its l-hop neighborhood. The l-th layer of a GNN is described as:

$$a_{v_i}^{(l+1)} = Agg\big(\{h_u^{(l)} | u \in \mathcal{N}(v_i)\}\big), \tag{1}$$

where $h_{v_i}^{(l+1)}$ is the vector representation of the node v_i at the l-th iteration. The $\mathcal{N}(.)$ function returns neighbors of a node, and Agg

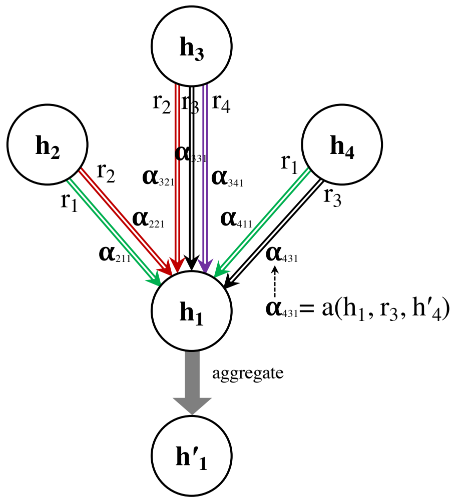

Figure 2: The attention mechanism of RelAtt.

is the aggregation function which is defined as per the modelling approaches of different methods [7, 11].

4 METHODOLOGY

In this section, we describe RelAtt (Relation-aware Attention), model that exploits the relation between two entities to learn the importance of neighboring nodes by using the idea of [15], and then recursively propagates node features in the graph. Our model has two components: *embedding layer*, and *knowledge graph convolutional layer with relation-aware attention*

4.1 Embedding Layer

The additional information such as attributes associated with entities in the KG contain semantic information and thus, can be leveraged in learning. For each entity, we concatenate the various textual attributes and use a pre-trained BERT [5] model to obtain their embeddings. These embeddings form an initial feature vector of entities to be used in the training. In case of datasets which do not have attributes, the embedding layer is initialized randomly.

4.2 Knowledge Graph Convolutional Layer with Relation-Aware Attention

This layer is defined as a single neural network layer which performs *relation-aware attention, feature propagation and aggregation.* The input to this layer is set of N node features from embedding layer, $\mathbf{h} = \{h_1, h_2, \cdots, h_N\}$ where $h_i \in \mathbb{R}^d$ represents the d-dimensional features of i^{th} node; a set of relation types $R = \{r_1, r_2, \cdots, r_k\}$; and a set of relation features $\mathbf{m} = \{m_1, m_2, \cdots m_k\}$, where $m_r \in \mathbb{R}^d$ is the feature vector of r^{th}-relation type of dimension d.

4.2.1 Relation-Aware Attention. In a Knowledge Graph, nodes have different types of relationships, thus the importance of neighbors is not only dependent on their features but also on the features of relationships. To this end, we follow [15] and apply a shared linear transformation on triplets (h, r, t), parameterized by a weight matrix $\mathbf{W}$. Then, we perform *self-attention* with respect to a shared relation between entities to compute attention coefficients: $a : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. The attention mechanism is shown in Figure 2. The attention coefficient of a triplet (h, r, t) is computed as:

$$e_{(h,r,t)} = a(\mathbf{W}h_h, \mathbf{W}m_r, \mathbf{W}h_t) \tag{2}$$

The attention mechanism a is a trainable function parameterized by a weight vector $\mathbf{a} \in \mathbb{R}^{3d}$, which is given as:

$$e_{(h,r,t)} = \mathbf{a}^T[\mathbf{W}h_h||\mathbf{W}m_r||\mathbf{W}h_t] \tag{3}$$

where $.^T$ and $||$ are transpose and concatenation operations respectively. Moreover, the attention is masked i.e, the attention is computed for directly connected neighbors only given by $\mathcal{N}_h$. We apply softmax to make attention coefficients comparable across the neighborhood as given in Eq. 4

$$\alpha_{(h,r,t)} = softmax(e_{(h,r,t)}) = \frac{exp(e_{(h,r,t)})}{\sum_{(r',t')\in\mathcal{N}_h} exp(e_{(h,r',t')})} \tag{4}$$

4.2.2 Feature Propagation and Aggregation. Knowledge graph convolutional network architectures such as RGCN [11] consider the heterogeneity of the edges and use a message passing framework to compute a new representation of a head node by applying some relation-specific transformation on representation of neighbors before aggregating at the head node. Following Equation 1, a generalized framework for knowledge graphs can be expressed as:

$$h'_h = \sigma\left(Agg_{(r,t)\in\mathcal{N}_h} f(h_h, r, h_t)\right) \tag{5}$$

where f is a relation-specific transformation on representation of immediate neighborhood nodes given by $\mathcal{N}_h$, Agg is an aggregator function such as *SUM, MEAN* that combines these transformed messages from neighbors before passing it to an activation function (σ), and h'_h is the new hidden features of entity h.

Combining Equation 5 and Equation 4 describes a single neural layer for knowledge graph convolution with relation-aware attention.

$$h'_h = \sigma\left(Agg_{(r,t)\in\mathcal{N}_h} \alpha_{(h,r,t)} f(h_h, r, h_t)\right) \tag{6}$$

The Equation 6 is agnostic to the underlying knowledge graph convolution message passing paradigm. However, in this work we use RGCN [11] as the underlying convolutional message passing method. Following RGCN which uses *SUM* as aggregation function, we can extend Equation 6 to L-layers as:

$$h_h^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{t\in\mathcal{N}_h^r} \alpha_{(h,r,t)} \frac{1}{\mathcal{N}_h^r} \mathbf{W}_r^{(l)} h_t^{(l)} + \mathbf{W}_0^{(l)} h_h^{(l)}\right) \tag{7}$$

where $\mathbf{W}_r^{(l)}$ is the weight matrix corresponding to relation r in l^{th}-layer, and $\mathcal{N}_h^r$ gives the set of neighbors which share relation r with entity h.

5 TRAINING

The objective of the knowledge graph based embedding methods is to learn embeddings of entities and relations which are fed into a scalar output producing scoring function(g) which scores true triplets much higher than false triplets. Various methods such as DistMult [21], NTN [12] have proposed different scoring functions.

Our proposed method given above is limited to use only scoring functions which describe relation in $\mathbb{R}^d$ space. Therefore, we use DistMult [21] scoring function given as:

$$g(h, r, t) = h_h^T \mathbf{M}_r h_t \tag{8}$$

We train the model using a negative sampling [11, 21] approach. For each positive triplet $\tau \in T^+$, we generate a set of negative samples by either corrupting h or t which produces a set of negative triplets T^-. Given the set of positive and negative triplets $T = T^+ \bigcup T^-$, we optimize the model on cross entropy loss so as to learn entity and relation embeddings.

$$\mathcal{L} = \frac{1}{|T|} \sum_{\tau \in T} y \ log\, l\big(g(\tau)\big) + (1-y) log\big(1 - l(g(\tau))\big) \tag{9}$$

where τ is training example (h, r, t); l is logistic *sigmoid* function; y is 1 or 0 for positive or negative triplet respectively.

6 EXPERIMENTAL SETUP AND EVALUATION

In this section, we provide the details of datasets, baseline methods for comparison, and training settings and evaluation protocol for link prediction and entity matching tasks.

6.1 Supervised Task: Link Prediction

First, we evaluate RelAtt with the link prediction task in a supervised learning setting. That is, the RelAtt is trained using a large portion of the original knowledge graph, and the goal is to predict the missing h or t in the omitted triplets. Link prediction is a common task for evaluating a knowledge graph embedding method and is typically measured by Mean Reciprocal Rank (MRR) and Hits@K. For each test triplet, we obtain a set of all possible triplets and score them through the model. The triplets are ranked on the score, and the position of true test triplet in a sorted list is its rank(c), and $1/c$ is reciprocal rank. The mean of reciprocal ranks of all true test triplets is called Mean Reciprocal Rank. Hits@K gives the number of times the test triplets occur in top k rankings in the ranked list. The higher values of MRR and Hits@K indicate the better performance of the model.

6.1.1 Datasets. We evaluate our proposed approach on three datasets - two widely used public datasets (FB15k-237 and WN18) and one proprietary dataset (Comp). FB15k-237 [13] is obtained from FB15k by removing inverse relations; which is a subset of relational database FreeBase, containing general facts. WN18 [2] is a subset of WordNet and contains lexical relations between words. It mostly contains *hyponym* and *hypernym* relations. Comp is a proprietary dataset and is extracted from the relational database of companies which include different branches, subsidiaries, headquarters, and products manufactured/produced by companies located in two countries. The companies, products and cities are the entities, and there are 6 different relationships between these entities[1]. The train, test and validation splits are from DistMult [21] and in case of Comp dataset, we used 80% triplets for training, and 10% of triplets each for validation and test. The statistics of the datasets are given in Table 1.

[1] *manufactures, category_of, parent_category, owns, in_city, in_country*

Table 1: Dataset statistics.

Dataset	WN18	FB15k-237	Comp
# Entities	40943	14,541	11,585
# Relations	18	237	6
# Features	-	-	768
# Train Triplets	141,442	272,115	60,177
# Valid Edges	5000	17,535	7,522
# Test Edges	5000	20,466	7,522

6.1.2 Baselines. We compared our approach with two methods: DistMult [21] and RGCN [11]. DistMult is a factorization based method which represents entities in a d dimensional vector space and each relation is represented by a diagonal matrix . RGCN is a graph convolutional based approach to learn embeddings of entities and relationships. For experimental evaluation we used DGL[16] of RGCN, and HNE[22] implementation of DistMult. For the dataset with attributes, we used a variant RGCN that includes an embedding layer for attributes. Since, DistMult does not use attribute information, therefore we omitted the attributes for learning on the Comp dataset .

6.1.3 Experimental Setup and Results. We selected hyperparameters of our model and baselines on their respective validation sets by grid search and early stopping on the filtered Mean Reciprocal Rank (MRR) metric using full-batch optimization. For fairness of comparison, we used the same hyperparameter search space for our model and the baselines, and trained models for a minimum of 6000 epochs. The hyperparameter space is given as - learning rate {0.01, 0.001}, number of hidden layers {1,2}, hidden layer dropout {0.0,0.1,0.2, 0.3}, attention dropout {0.0,0.1,0.3,0.6}, embedding size {100, 200,400}, number of bases {2,3,5,10,50,100}, and negative samples {10}. We used *basis decomposition* [11] for regularization and optimized the loss function using Adam optimizer.

After training the models, we follow the evaluation protocol of Yang et.al [21] for link prediction and report results on two widely used and standard metrics: *Mean Reciprocal Rate* and *Hits@k* (1,3,10). Both metrics are obtained using filtered approach i.e, for each test triplet, we generate all potential triplets but ensuring that none of the generated triplet appears in training, validation or test triplets, and rank the test triplets to obtain MRR and Hits@K.

The results of link prediction on all datasets are shown in Table 2. Our model RelAtt consistently shows performance improvements in all three datasets across all baselines. Our model (RelAtt) shows an average improvement of 2.8% in *MRR* in all three datasets. We attribute this to the attention mechanism, that helps in learning weights for the messages from the neighbors. This suggests that all neighbors are not equal, and have different roles.

6.2 Unsupervised Task: Entity Matching

Next, we demonstrate the effectiveness of RelAtt on learning a useful graph representation in an unsupervised setting. We evaluate RelAtt with an entity matching task which aims to find data instances that refer to the same real-world entity. In particular, our defined task is to match entities across two knowledge graphs

Table 2: Results of link prediction on FB15k-237, WN18 and Comp dataset.

Dataset	Method	DistMult	RGCN	RelAtt
FB15k-237	MRR	0.201	0.226	**0.234**
	Hits@1	0.130	0.126	**0.139**
	Hits@3	0.219	0.252	**0.258**
	Hits@10	0.363	0.421	**0.428**
WN18	MRR	0.745	0.750	**0.767**
	Hits@1	0.592	0.612	**0.639**
	Hits@3	**0.892**	0.882	0.889
	Hits@10	0.934	0.939	**0.940**
Comp	MRR	0.081	0.110	**0.113**
	Hits@1	0.046	**0.073**	**0.073**
	Hits@3	0.082	0.107	**0.113**
	Hits@10	0.142	0.173	**0.178**

Table 3: Statistics of the query graphs

Property	**Measurement**
# of graphs	1,426
avg nodes	20.90
avg edges	39.76
# of node types	4
# of relations	3
# of features	768
# of labels	1,426
avg node degree of the matching entity in R	19.88
avg node degree of the matching entity in Qs	12.89

at a time. One graph is referred as a *reference* graph which usually captures the full knowledge of a database. Given an entity in another graph referred as a query graph, the task is to find the corresponding entity in the reference graph.

We approach the above task in the following way: (1) we train RelAtt on the full reference graph with the optimization goal to minimize the link prediction error; (2) we obtain the embeddings of the query entities by applying the trained RelAtt on the query graphs; (3) we then perform matching, i.e. finding the most similar entities to the given entities in the query graphs by comparing the cosine similarity.

6.2.1 Dataset. The dataset consists of a reference graph and a set of query graphs. We use the full Comp as the reference graph to train RelAtt. The query graphs are extracted from a proprietary relational database and the ground truth of the correct matching are manually labeled. The statistics of the dataset is reported in Table 3. From the query graph, we obtained multiple variations of query graphs by sampling neighbors of a matching entity based on their average node degree. We used a threshold (th) to filter out the neighbors. For example, at $th = 0.2$ we select 20% of neighbors and the average node degrees of query graphs reduce to 9.8. These variations also limit the contextual information. A lower value of threshold produces query graphs with lesser contextual information.

Table 4: Results of entity matching on Comp dataset.

th	**Model**	**Hits@1**	**Hits@5**	**Hits@10**	**Hits@30**
-	BERT	0.7373	0.8200	0.8440	0.8759
0.2	RGCN	0.6248	0.7061	0.7446	0.7845
	RelAtt	**0.7380**	**0.8425**	**0.8766**	**0.9231**
0.3	RGCN	0.6270	0.7170	0.7685	0.8324
	RelAtt	**0.7540**	**0.8549**	**0.8846**	**0.9253**
0.4	RGCN	0.6567	0.7888	0.8440	0.8977
	RelAtt	**0.7627**	**0.8621**	**0.8875**	**0.9209**
0.5	RGCN	0.7250	0.8520	0.8861	0.9202
	RelAtt	**0.7736**	**0.8665**	**0.8919**	**0.9245**

The idea is to test the model behaviour when query graphs have less contextual information present.

6.2.2 Baseline. We evaluate our model against two methods. The first baseline uses only textual attributes present in the nodes of the query graphs. These textual features are fed into pre-trained BERT [5] model to obtain the embeddings of entities. Second, baseline RGCN ingests both attributes and query graph structure to obtain the embeddings.

6.2.3 Experimental Setup and Results. We followed the same training protocol described in subsection 6.1.3 to train RelAtt and RGCN models. We use this trained model to infer the embeddings of entities in query graphs. Now having the embeddings of entities in knowledge graph and inferred embedding of query node, we use cosine similarity metric to find a similar entity to the query node in the knowledge graph, and report the results on *Hits@K* in Table 4. The results show that using only textual attributes is not sufficient for entity matching as it does not use any contextual information. It can be seen in the case of RGCN, as value of th increases from 0.2 to 0.5, the performance of RGCN improves because of the increased amount of context information. At lower values of contextual information (0.2), RGCN performance gets worse as compared to BERT, possibly because RGCN does not differentiate between neighbors. This disadvantage is overcome by RelAtt as it employs an attention mechanism that helps in learning better representations. RelAtt performs better even when less contextual information is present.

7 CONCLUSION

In this work, we proposed a relation-aware masked attention mechanism that leverages the relation and neighborhood information to compute the importance of neighbors. Using this attention, the features are propagated from the neighbors of an entity to update its embedding. We evaluated the proposed model on the link prediction task on three datasets and entity matching task on one dataset which showed that the attention mechanism helps in learning a better representation.

This work uses the KG structure and attributes present on the entities, and treats the entities as homogeneous types. This opens up the future direction to exploit the heterogeneity of entities in learning. Another exciting direction is to explore the sampling strategies on KGs such that the computational costs of training on large graphs can be reduced without losing the quality of learned embeddings.

REFERENCES

[1] 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.

[2] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI'11)*. AAAI Press, 301–306.

[3] Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial Learning for Knowledge Graph Embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 1470–1480.

[4] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2017. Convolutional 2D Knowledge Graph Embeddings. *CoRR* abs/1707.01476 (2017).

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[6] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 601–610.

[7] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *NIPS*.

[8] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 2181–2187.

[9] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Q. Phung. 2017. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. *CoRR* abs/1712.02121 (2017).

[10] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, Madison, WI, USA, 809–816.

[11] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Springer.

[12] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'13)*. 926–934.

[13] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*. Association for Computational Linguistics, Beijing, China.

[14] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16)*. JMLR.org, 2071–2080.

[15] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations* (2018).

[16] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, Ziyue Huang, Qipeng Guo, Hao Zhang, Haibin Lin, Junbo Zhao, Jinyang Li, Alexander J Smola, and Zheng Zhang. 2019. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. *ICLR Workshop on Representation Learning on Graphs and Manifolds* (2019). https://arxiv.org/abs/1909.01315

[17] Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge Base Completion Using Embeddings and Rules. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press, 1859–1865.

[18] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14)*. AAAI Press, 1112–1119.

[19] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 2965–2971.

[20] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *Proceedings of the ICLR 2019, May 6-9, 2019, New Orleans, LA, USA*.

[21] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *3rd International Conference on Learning Representations, ICLR San Diego, CA, USA, May 7-9,*.

[22] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. *arXiv preprint arXiv:2004.00216* (2020).