Neighbor-T: Neighborhood transformer aggregation for enhancing representation of out-of-knowledge-base entities

Jing Xie[a], Jingchi Jiang[a], Jinghui Xiao[b], Yi Guan*[a]

[a]School of Computer Science, Harbin Institute of Technology, Harbin, China; [b]Noah's Ark Lab, Huawei Technologies, Beijing, China

ABSTRACT

Knowledge representation learning (KRL) aims to obtain the embedding of entities and relations from the information of knowledge graph (KG). Most existing methods can only model the entities in the training data, while failing to generalize to out-of-knowledge-base (OOKB) entities which only appear in the testing. To solve this issue, one common approach is to train an aggregator by leveraging the auxiliary knowledge such as neighbor information and entity descriptions. In this work, we propose a novel aggregation model called neighborhood transformer (Neighbor-T) to enhance the representations of OOKB entities. Compared with previous methods, Neighbor-T shows effectiveness on neighbor information aggregation because of self-attention mechanism. Experiments demonstrate that our enhanced representation outperforms the state-of-the-art on two knowledge graph completion tasks under OOKB setting: triple classification and entity prediction.

Keywords: Knowledge base representation, out-of-knowledge-base entity, transformer

1. INTRODUCTION

Large-scale knowledge graphs (KGs) such as WordNet[1], Wikidata[2] and Freebase[3] collect a mass of real-world facts in the form of triples. The nodes in KGs represent entities and edges represent relations. A piece of knowledge is often formalized as (h, r, t) where h and t are entities and r is relation. For example,

$$(Columbia\ University, education\_student, Barack\ Obama)$$

represents the fact that $Barack\ Obama$ is the student of $Columbia\ University$. In order to efficiently support downstream tasks such as question answering, machine reading comprehension and link prediction, various KG embedding models[4-7] have been proposed which projects symbolic entities and relations into continuous vector spaces. It learns the embedding for each entity and relation according to the triples in KGs.

Although the scale of current KGs is large, they all suffer from the incompleteness problem. The "close world assumption" assumes that all entities are present in the training data. And they directly use their embeddings in the downstream task. However, new entities keep appearing in the real world, and the previous works cannot provide an efficient solution except retraining the whole model. These new entities are called out-of-knowledge-base (OOKB) entities[8], which should be represented in a more effective way.

Two triple classification examples under OOKB setting. Red circles are OOKB entities and blue circles are existing entities in KG. Triples in the left box are auxiliary triples. Our goal is to classify whether $location$ is the true relation between the OOKB head entity and the existing tail entity in KG.

Our experience shows that new entities do not appear individually. They always come with either the triples in contexts or the descriptions of new entities. Thus, we can use this information to construct the representation of OOKB entities. This paper focuses on making use of triple information instead of the description because not every KG's entity has a text description but each entity should at least appear in a triple. These triples contain both the new entity and the existing entity of KG are called auxiliary triples. Recent works[8-10] point out that is feasible to train a neighbor aggregator to represent OOKB entities by using auxiliary triples. When a new entity occurs, they first find its all relevant auxiliary triples, then use the aggregator to incorporate all information of these triples to obtain the entity's embedding. Although

*guanyi@hit.edu.cn

International Conference on Computer Application and Information Security (ICCAIS 2021), edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12260, 1226002 · © The Authors.
 · doi: 10.1117/12.2637402

these works achieve good results in OOKB entity representation, their embeddings can be further enhanced without extra information.

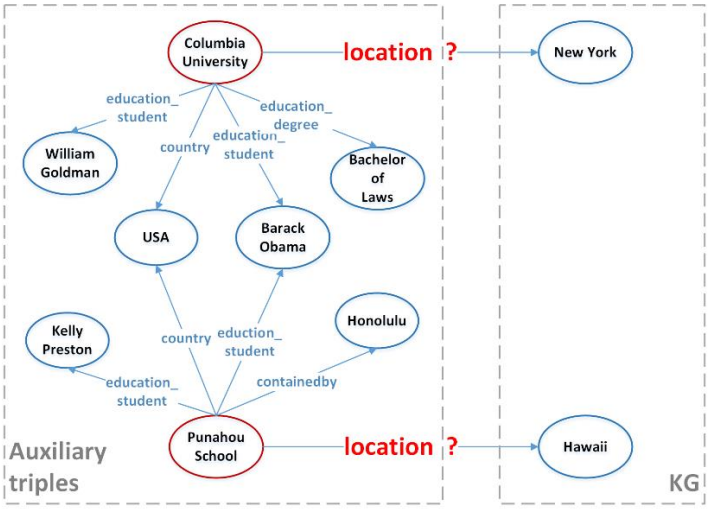

Figure 1. Two triple classification examples under OOKB setting. Red circles are OOKB entities and blue circles are existing entities in KG. Triples in the left box are auxiliary triples. Our goal is to classify whether location is the true relation between the OOKB head entity and the existing tail entity in KG.

This paper concentrates on using graph contextual information[11] to enhance representations of OOKB entities. Each entity in the KG has different contextual information. Figure 1 illustrates two triple classification examples of "when giving an OOKB head entity and an existing tail entity in KG, does the shown relation correctly hold between them". For the entity $Columbia\ University$, its contextual information are four neighbors, while for the entity $Barack\ Obama$ when being the neighbor of $Columbia\ University$, its contextual information is the other three neighbors of $Columbia\ University$. To the best of our knowledge, the current state-of-the-art (SOTA) aggregator of Logic Attention based Neighborhood aggregation (LAN)[9] only uses the contextual information of the OOKB entities, while neglects the contextual information of neighbors. In our example, we can find $(education\_student, Barack\ Obama)$ is the neighbor of both $Columbia\ University$ and $Punahou\ School$. However, $Barack\ Obama$ has different alumnus when he studies in different school. Thus, this neighbor should propagate different information to its head entities. In this work, we utilize Transformer to capture contextual information of neighbor. The new aggregator is named Neighbor-T, and we used it after LAN to enhance representations of OOKB entities. Our main contributions are three-folds:

- We propose a novel aggregator Neighbor-T to capture contextual information of neighbors of the OOKB entity.
- We propose a three-stage training method to learn parameters when we sum the outputs of Neighbor-T and LAN to construct enhanced representations.
- We formulate two tasks of knowledge base completion (KBC) tasks under OOKB settings, and our method outperforms the SOTA.

2. RELATED WORK

Here we survey two topics related to this work: OOKB entity representation and using Transformer to capture contextual information.

2.1 OOKB entity representation

The methods of OOKB entity representation fall into two categories: (1) using the text description of the OOKB entity. The typical work are references[12-15]. (2) using the neighbors of the OOKB entity from auxiliary triples. The typical work are references[8-10]. The text descriptions provide rich information for the entity, however, this constraint is too strong that not all KGs contain the information of entity description. On the other hand, we can always extract triples from the text as auxiliary triples to represent the OOKB entity. Thus, our work focuses on the second category. Reference[8] is the first work to use graph neural network (GNN) to aggregate neighbor information which proves GNN based aggregators

perform better than the simple approximation like TransE. The drawback of this work is that treats all neighbor equally. To address that issue, Wang et al.[9] and Zhao et al.[10] both consider the difference among neighbors. The former work considers in more respects in which the aggregator is aware of redundancy and query relation. It reduces the impact of similar neighbor information and measure the importance of neighbors by the query relation. For example, if asking someone's $nationality$, the aggregator need to concern more on neighbors which contain $birthplace$ and $live\_in$ other than $gender$. Besides, reference[16] propose a model that only uses graph structure to predict relations between nodes. However, their work only considers the relations information and can not distinguish entities.

2.2 Using transformer to capture contextual information

The Transformer model has been widely used in recent researches such as pre-trained language mode[11], KG embedding[17-18] and machine translation[19]. The core of Transformer is self-attention mechanism[20], that each token in the input sequence can gather information from other tokens. Reference[17] uses a pre-trained language model Bert to get KG representations. They concatenate the head name, relation name and tail name in one triple as input sequence, then fine tune the parameters of the model on the triple classification task. Each word in the sequence can get information from the other words. Reference[18] only inputs one triple with entity and relation IDs into Transformer per time, and both entity and relation can obtain information from the opposite side. Unlike the traditional training objective of KG-embedding models, Transformer based models use a special task of "masked token prediction" to learn parameters. In the training stage, a head or tail is masked and the model need to predict the true entity. However, the above two mentioned works are not under the OOKB setting. Moreover, they both model the single triple. In this work, we study a more complex scenario which deals with several triples at one time.

3. PRELIMINARIES

In this section, we formally define the OOKB entity problem in knowledge graph completion (KBC) and detail the OOKB entity representation.

3.1 OOKB entity problem definition

A knowledge graph G consists of an entity set $\mathcal{E}$, a relation set $\mathcal{R}$ and a collection of true triples $\{(h,r,t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. For each triple (h,r,t), we define the reverse relation r^{-1} and add (h,r^{-1},t) to G. Triple classification and entity prediction are two important tasks in KBC. Unlike the conventional settings of both tasks where all entities in the test procedure have been trained, this work considers a challenging scenario that unseen entities are given in the test procedure. We define the unseen entity as $e, e \notin \mathcal{E}$ and the unseen entity set as $\mathcal{E}_{\text{OOKB}}$. The new triple classification and entity prediction tasks are defined as follows.

Task 1 Triple classification with OOKB entity

Given a triple $(h,r,t), r \in \mathcal{R}$, where $h \in \mathcal{E}_{\text{OOKB}}$ or $t \in \mathcal{E}_{\text{OOKB}}$, the task is to train a model to classify whether the triple is true or false.

Task 2 Entity prediction with OOKB entity

Given a head entity $h, h \in \mathcal{E}_{\text{OOKB}}$ and a relation $r, r \in \mathcal{R}$, or given a tail entity $t, t \in \mathcal{E}_{\text{OOKB}}$ and a relation $r, r \in \mathcal{R}$, the task is to train a model which ranks all tail candidate entities $t, t \in \mathcal{E}$ or ranks all head candidate entities $h, h \in \mathcal{E}$.

The main problem of the two tasks is how to represent the OOKB entity. For both tasks, we import the auxiliary triples to construct the representation of OOKB entities, which defined as $G_{aux} = \{(h,r,t) | r \in \mathcal{R}, h \in \mathcal{E}_{\text{OOKB}}, t \in \mathcal{E}\} \cap \{(h,r,t) | r \in \mathcal{R}, h \in \mathcal{E}, t \in \mathcal{E}_{\text{OOKB}}\}$. Here, we also add (h,r^{-1},t) to G_{aux}. For each OOKB entity e, we can now use the auxiliary triples which contains e to represent it.

3.2 OOKB entity representation

To obtain the OOKB entity representation, we need to construct an aggregation to integrate all neighbor information of this OOKB entity which provided by the auxiliary triples. We define $N_t(e)$ as neighbor triples, $N_t(e) = \{(e,r,e') | r \in \mathcal{R}, e' \in \mathcal{E}\}$. We also define $\mathbf{v}_e \in \mathbb{R}^d$ as a d-dimensional vector to representation entity e, and define $\mathbf{v}_r \in \mathbb{R}^d$ to represent relation r. According to the exist method[21-22], the aggregation consists of transition function $T(\mathbf{v}_e)$ and pooling function $P(\mathbf{v}_e)$. These two components are described below in detail.

Transition Function. For each OOKB entity e, its neighbor triples may contain different kinds of relation and entities. The transition function $T(\boldsymbol{v}_e)$ aims to apply the influence of neighbor relation to the representation of the neighbor entity, which can propagate the relation-special information to the OOKB entity. According to the way of relation projection, transition function contains three categories:

Non-relation projection. For an OOKB entity e, its neighbor triple $(e, r, e') \in N_t(e)$ propagates the whole information of the entity e' to the e:

$$T(\mathbf{v}_e) = \mathbf{v}_e{}' \tag{1}$$

Distance-based relation projection. For an OOKB entity e, its neighbor triple information depends on the distance between e and e', the typical examples inspired by TransE and TransH are listed:

$$T(\mathbf{v}_e) = \mathbf{v}_e{}' - \mathbf{v}_r \tag{2}$$

$$T(\mathbf{v}_e) = \mathbf{v}_e{}' - \mathbf{w}_r^{\top}\mathbf{v}_e{}'\mathbf{w}_r \tag{3}$$

where $\mathbf{w}_r$ and $\mathbf{w}_r^{\top}$ construct a relation-related hyperplane[23].

NN-based relation projection. For an OOKB entity e, we use a relation-specific matrix $\mathbf{M}_r$ to reflect the influence of relation r on entity e', the examples of the transition function are listed:

$$T(\mathbf{v}_e) = \tanh(\mathbf{M}_r\mathbf{v}_e{}') \tag{4}$$

$$T(\mathbf{v}_e) = \mathrm{ReLU}(\mathbf{M}_r\mathbf{v}_e{}') \tag{5}$$

Pooling Function. For each OOKB entity e, it often has many neighbor triples. Thus, pooling function is used to summarize all these neighbor information. Typical pooling function $\mathrm{P}(\mathbf{v}_e)$ contains sum pooling, mean pooling and max pooling, which all satisfy permutation invariance. That means all the input of neighbors are unordered. These three pooling functions are defined by:

$$P(\mathbf{v}_e) = \sum_{(e,r,e\prime)\in N_t(e)} \mathbf{v}_e{}' \tag{6}$$

$$P(\mathbf{v}_e) = \frac{1}{N}\sum_{(e,r,e\prime)\in N_t(e)} \mathbf{v}_e{}' \tag{7}$$

$$P(\mathbf{v}_e) = \max\left(\{\mathbf{v}_e{}'\}_{(e,r,e\prime)\in N_t(e)}\right) \tag{8}$$

where N is the total number of $N_t(e)$ and max is the element-wise max function.

Unlike the above pooling functions that treat all neighbor equally, attention-based pooling assumes the different neighbor contributes the different information. For each neighbor, pooling function assigns a weight to enlarge or reduce its information, and then sum all changed information together.

4. PROPOSED MODEL

For a triple (e, q, e'), q represents the query relation in the triple[9], and r represents the neighbor relation of the entity e. We aim to train an aggregator that represents e by its neighbor information. In this section, we present a comprehensive introduction of the aggregator. Since we build our model based on LAN, we will first introduce it and then show the details of Neighbor-T. We also present the details of training procedure.

4.1 LAN

The core of LAN is the attention mechanism. In the aggregator, neighbors should contribute differently to the $\mathbf{v}_e$ according to its importance in representing e. LAN donates two different kinds of attention mechanisms: the first one is the logic rule mechanism which captures the attention between the neighbor relation r and the query relation q; the second is the neural network mechanism which captures the attention between the neighbor entity $e_i, e_i \in N_t(e)$ and the query relation q. LAN implements the confidence of logic rule $r_1 \Rightarrow r_2$ as follows:

$$\mathcal{P}(r_1 \Rightarrow r_2) = \frac{\sum_{e\in\mathcal{E}} \mathbb{I}(r_1\in N_t(e) \wedge r_2 \in N_t(e))}{\sum_{e\in\mathcal{E}} \mathbb{I}(r_1 \in N_t(e))} \tag{9}$$

The function $\mathbb{I}(\mathrm{x})$ here is a logic function, which equals 1 when x is true and 0 otherwise. According to this equation, we easily find that the confidence is larger if relation r_1 and r_2 often appear as the entity's neighbor relation at the same

time. Thus, for the entity e, LAN calculates all confidences of neighbor relation r and query relation q to measure which neighbor is more important. However, the entity may have many neighbors and some of them provide similar information, which will cause redundancy. The logic rule mechanism is defined as follows:

$$\alpha_{j|i,q}^{Logic} = \frac{\mathcal{P}(r \Rightarrow q)}{\max(\{\mathcal{P}(r' \Rightarrow r) | r' \in N_t(e) \wedge r' \notin r\})} \tag{10}$$

where j is the j-th neighbor of entity e_i. We can find that if the neighbor relation r can be implied by another neighbor relation r', its contribution to represent entity e_i will be decreased. This logic rule attention uses the statistical relevance of relations to distinguish the neighbor information.

The neural network mechanism adopts an attention network to measure the relevant of neighbor entity and query relation[24], which is defined as:

$$\alpha_{j|i,q} = \mathbf{u}_a^{\top} \cdot \tanh\left(\mathbf{W}_a \cdot \left[\mathbf{z}_q; T(\mathbf{v}_e)\right]\right) \tag{11}$$

where $\mathbf{u}_a$ and $\mathbf{W}_a \in \mathbb{R}^{d \times 2d}$. $\mathbf{z}_q \in \mathbb{R}^d$ is a relation-aware embedding representation of the query relation. $T(\mathbf{v}_e)$ here uses equation (3). To normalize all neighbor attention weights, the softmax function has been used:

$$\alpha_{j|i,q}^{NN} = \mathrm{softmax}\left(\alpha_{j|i,q}\right) = \frac{\exp(\alpha_{j|i,q})}{\sum_{e_j \in N_t(e_i)} \exp(\alpha_{j|i,q})} \tag{12}$$

Finally, LAN incorporates two attention mechanism together to gather all neighbor information:

$$\mathbf{v}_{e_i}^{LAN} = \sum_{e_j \in N_t(e_i)} \left(\alpha_{j|i,q}^{Logic} + \alpha_{j|i,q}^{NN}\right) T(\mathbf{v}_e) \tag{13}$$

4.2 Neighbor-T

As we want to train an aggregation to represent an entity by its neighbor information, we need to take advantages of the graph contextual information. The concept of contextual information first comes from language model. For a word in the text, its contextual information is often a sequence of words before or after it, which implies rich semantic patterns. For an entity in the KG, it also has contextual information. We define its neighbor entity and neighbor relation as graph contextual information. According to this definition, we could find some special graph semantic patterns and use them to construct the entity's embedding representation.

Transformer model is successfully used to capture contextual information in the language model by its self-attention mechanism. In this work, we conduct Transformer in the KG to capture graph contextual information and aggregate them to represent the entity. We name this aggregation as Neighbor-T. The architecture of Neighbor-T is shown in Figure 2. Firstly, for each entity e_i, we extract its neighbor triple set $N_t(e_i)$. The input sequence consists of two parts: neighbor entity sequence and neighbor relation sequence. In transition layer, equation (3) is used to conduct the influence of the relation on the corresponding entity, and we can get the transitional embedding $T\left(\mathbf{v}_{e_j}\right)$ of each neighbor. Secondly, we feed transitional embeddings of all neighbors to a Transformer to exchange information by self-attention mechanism. At this stage, each neighbor has a transformed embedding $T'\left(\mathbf{v}_{e_j}\right)$, which has incorporated the contextual information from the other neighbors. Lastly, we sum all transformed embeddings to obtain the output embedding:

$$\mathbf{v}_{e_i}^{Neighbor-T} = \sum_{e_j \in N_t(e_i)} \alpha_{j|i,q}^{Logic} T'\left(\mathbf{v}_{e_j}\right) \tag{14}$$

Here we also use the prior statistical attention weights $\alpha_{j|i,q}^{Logic}$ to fill the information gap between neighbor relation and query relation.

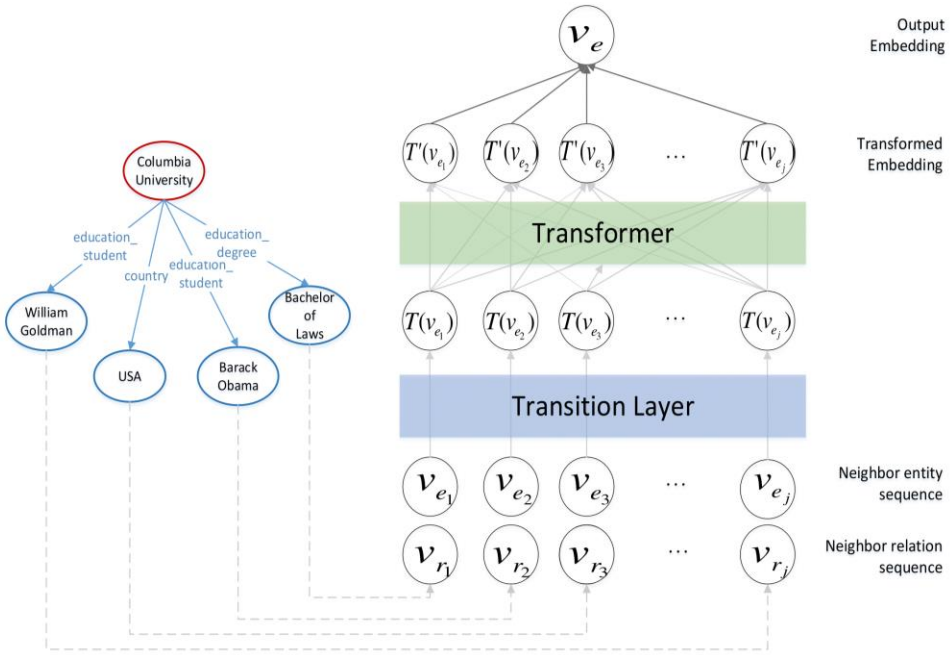

Figure 2. The architecture of Neighbor-T for entity representation.

4.3 Objective and model training

In the aggregation training stage, we can only get all training triples, which means there is no OOKB entity. Thus, for each training triple, we treat its head entity and tail entity as OOKB entities in this sample and extract subgraphs of the head entity and tail entity to construct the model input.

In this work, we propose a three-stage training method to learning parameters since we use Neighbor-T after LAN to construct enhanced representations of OOKB entities. Firstly, we get the pre-trained embeddings of all existing entities by training parameters of LAN. Since the amount of training triple is limited, we need to give Transformer a better initialization. The frequently-used score function is to evaluate the possibility of a triple(h, q, t). The score is larger when the triple is likely to hold. Here, we use TransE to calculate the triple score:

$$f(h,q,t) = -\left|\mathbf{v}_h + \mathbf{v}_q - \mathbf{v}_t\right|_{L1} \tag{15}$$

where $L1$ is the L1 norm. We treat all triples in KG as positive sample and randomly corrupt the head or tail entity by another entity to construct negative samples. Then we use margin-based loss function as our objective:

$$\mathcal{L}_1 = \sum_{i=1}^{N}[\gamma - f(h_i,q_i,t_i) + f(h_i',q_i,t_i')]_+ \tag{16}$$

where $[x]_+ = max(0,x)$, N is total number of positive and negative samples, and γ is the max margin between positive and negative samples.

In the second stage, we train the parameters in Neighbor-T. It is notice that we fix all entity embeddings because they have been trained in LAN. The prediction objective used here is to label the masked entity, which is to predict the true entities of head and tail. For a masked entity e_i, we can obtain its output embedding $\mathbf{v}_{e_i}^{Neighbor-T}$ by Neighbor-T. Then we calculate the probability of each candidate entity and use cross-entropy as the training loss:

$$\mathcal{L}_2 = -\sum_{i=1}^{N}\mathbf{y}_i \log \mathbf{p}_i \tag{17}$$

$$\mathbf{p}_i = \text{softmax}\left(\mathbf{v}_{e_i}^{Neighbor-T} \cdot \mathbf{E}\right) \tag{18}$$

where $\mathbf{E} \in \mathbb{R}^{n\times d}$ is the matrix of the entity embeddings, $\mathbf{y}_i$ is the weight vector of the true label distribution. Here we use soft label strategy instead of one-hot label. N is double amount of positive triples because we predict both head and tail entities. This stage is used to make the model learn how to capture contextual information of neighbors.

In the third stage, we sum the output embeddings of LAN and Neighbor-T to get a stronger representation of entities:

$$\mathbf{v}_{e_i}^{O} = \mathbf{v}_{e_i}^{LAN} + \mathbf{v}_{e_i}^{Neighbor-T} \tag{19}$$

We train all parameters in LAN and Neighbor-T by Equation 16 in this stage. This is for finetuning the parameters in the model to adapt two KBC tasks.

5. EXPERIMENTS

In this section, we demonstrate the effectiveness of Neighbor-T in two important KBC tasks under OOKB settings: triple classification and entity prediction.

5.1 Experimental design

Datasets. Our experiments are all in OOKB settings, that is there are some OOKB entities in the testing. The previous work[8] has constructed nine datasets from WordNet11 (WN11)[25]: Head, Tail, Both-1000, 3000, 5000. Head, Tail. Both are the position of OOKB entity, and 1000, 3000, 5000 is the amount of the testing triples which contain OOKB entities. For example, Both-1000 has extracted 1000 triples from the original testing dataset of WN11, and each head and tail entity of the triple are treated as OOKB entities. The original training dataset is split into the new training dataset and auxiliary dataset. The original triples which do not contain OOKB entities are placed in the new training dataset and the original triples which contain one OOKB entity are placed in the auxiliary dataset. The triples with two OOKB entities are discarded. For the validation set, all triples which contains OOKB entities are also discarded to avoid the data leakage. The above nine datasets are used for triple classification task. Wang et al.[9] construct other ten datasets from Freebase15K (FB15K)[4] for entity prediction task in the same way: Head, Tail-5, 10, 15, 20, 25, where the number is the percentage of the extracted testing triples. In this task, there is no Both setting because we cannot predict the missing entity by another missing entity and the given relation. We directly use these two groups of datasets and the statistics for them are in Tables 1 and 2.

Table 1. Statistics of constructed WN11 dataset.

	Head			**Tail**			**Both**		
	1000	3000	5000	1000	3000	5000	1000	3000	5000
Training triple	108197	99963	92309	96968	78763	67774	93364	71097	57601
Auxiliary triple	4325	12376	19625	15277	31770	40584	18638	38285	48425
OOKB entities	348	1034	1744	942	2627	4011	1238	3319	4963
Average neighbor number	5.8	5.6	5.4	5.5	5.1	4.9	5.4	4.9	4.5

Table 2. Statistics of constructed FB15K dataset.

	Head					**Tail**				
	5	10	15	20	25	5	10	15	20	25
Training triple	188238	108854	71407	49456	37986	170672	99783	67651	46982	34126
Auxiliary triple	235746	249798	228484	205242	179656	254454	261341	243316	222200	195627
OOKB entities	1460	2082	2342	2544	2666	1330	1934	2207	2351	2415
Average neighbor number	41.5	31.6	25.5	21.1	17.7	39.4	30.9	25.4	21.3	17.9

Implementation Details. In triple classification task, we can calculate all scores by equation (15). For each relation r, we select a threshold σ_r to classify the true triples if $f(h,r,t >= \sigma_q)$, and false triple in the other condition. In this task, we use the classification accuracy as the evaluation metrics. The best σ_r is optimized by maximizing classification accuracy on the validation triples. We adopt the same parameter configuration for all nine datasets: the learning rate is 0.001, embedding dimension is 100, batch size is 512, margin is 300, and 64 neighbors are randomly selected for each entity,

which are same as reference[9]. Because the scale of our training data is not as large as language model corpus, here we only use one head and one-layer Transformer. We also use dropout strategy to avoid over-fitting the training data. The dropout is set to 0.1.

In entity prediction task, we need to calculate the scores of all candidate entities to determine which is more likely to be the tail entity when giving the head entity and relation, or to be the head entity when giving the tail entity and relation. Here the candidate entities are all existing entities in the training data. After achieving the scores of candidate entities, we rank them in descending order. The evaluation metrics here are mean rank (MR), mean reciprocal rank (MRR) and the proportion of ranks no larger than n (Hits@n, n=1,3,10). All results are under *filtered setting*[4], that means any candidate triple already exists in the train, or validation data needs to be removed before ranking. To align with LAN, we also conduct experiments on Head-10 and Tail-10. The LAN parameters are same as them in triple classification task except the margin changes to 1.0. The dropout here is 0.8 because these two datasets are easily over-fitting. We will give the additional analysis about this point in the following.

Our training process is divided into three stages, we train the LAN with 500 epochs and train Neighbor-T with 1000 epochs. In triple classification tasks, we train the LAN+Neighbor-T with 100 epochs; and in entity prediction task, we train it with 500 epochs. The reported results are the testing results performer best in validation data.

6. RESULTS AND DISCUSSION

6.1 Triple classification with OOKB entity

Table 3 shows the model comparison on nine triple classification datasets. We use three previous models as our baselines. MEAN is the results of reference[8], we directly list their results from the original paper. Results of LSTM and LAN come from reference[9]. To our knowledge, LAN performs best in all nine datasets before our work is proposed. Thus, we retrain the LAN with their released source code and the same parameter settings at least 5 times and list the best results in LAN(ours). On the basis of LAN(ours), we train Neighbor-T and combine LAN and Neighbor-T to represent entities. The table shows LAN+Neighbor-T achieves the best results in all Tail datasets and Both datasets, which show our model are effective to capture contextual information of both OOKB entities and their neighbors. We find results of Head datasets show a little low results than LAN(proposed), we conjecture it is because the head entities in WN11 are naturally more coarse-grained, which need more information to describe them.

Table 3. Evaluation accuracy on triple classification.

Model	Head			Tail			Both		
	1000	3000	5000	1000	3000	5000	1000	3000	5000
Mean	0.873	0.843	0.833	0.840	0.752	0.692	0.830	0.733	0.682
LSTM	0.870	0.835	0.818	0.829	0.714	0.631	0.785	0.716	0.658
LAN(propose)	**0.888**	**0.852**	**0.842**	0.847	0.788	0.743	0.833	0.769	0.706
LAN(ours)	0.872	0.849	0.823	0.847	0.787	0.743	0.835	0.752	0.691
LAN+Neighbor-T	0.867	0.847	0.830	**0.849**	**0.799**	**0.765**	**0.846**	**0.775**	**0.751**

The results in the first three rows come from the original paper and the results in the last two rows are our implementations.

6.2 Entity prediction with OOKB entity

Table 4 shows the model comparison on two entity prediction datasets: Head-10 and Tail-10. We also list the same three baselines: MEAN from reference[8] and LSTM and LAN(proposed) from reference[9]. According to the results, we find all results (except Hits@10 in Head-10) of our model are better than LAN(ours). Furthermore, our combined representations of LAN+Neighbor-T achieve the best performance of MRR and Hits@1 in both datasets. Unlike MR is sensitive to the lower positions of the ranking, MRR evaluate the model more stably. Thus, we make sure that the embeddings come from Neighbor-T can enhance the entity representation.

Table 4. Evaluation results on entity prediction.

Model	Head-10					Tail-10				
	MR	MRR	Hits@10	Hits@3	Hits@1	MR	MRR	Hits@10	Hits@3	Hits@1
Mean	293	0.31	48.00	34.80	22.20	353	0.25	41.00	28.00	17.10
LSTM	353	0.25	42.90	29.60	16.20	504	0.22	37.30	24.60	14.30
LAN(propose)	263	0.39	**56.60**	**44.60**	30.20	461	0.31	**48.20**	35.70	22.70
LAN(ours)	250	0.38	55.80	43.20	29.10	434	0.31	46.50	35.20	22.30
LAN+Neighbor-T	**228**	**0.40**	55.60	44.50	**31.80**	**393**	**0.32**	47.50	**35.80**	**23.40**

Same as triple classification, the results in the first three rows come from the original paper and the results in the last two rows are our implementations.

6.3 The relevance of proportion of OOKB entities and model effectiveness

According to Table 3, we find when using Neighbor-T to enhance entity representations of LAN in different datasets, its improving capacity is different. We illustrate the relevance of proportion of OOKB entities and the improving capacity in all Tail and Both datasets in Figure 3. The figure shows that with the amount of OOKB entities increasing, the relative increasing percentage of the model is growing at the same time. Furthermore, the speed of the result growing could be beyond the linear growth mode in Both datasets. That means, when we meet large amount of OOKB entity, Neighbor-T can help LAN to achieve a stronger representation of the entity to avoid performance decreasing rapidly.

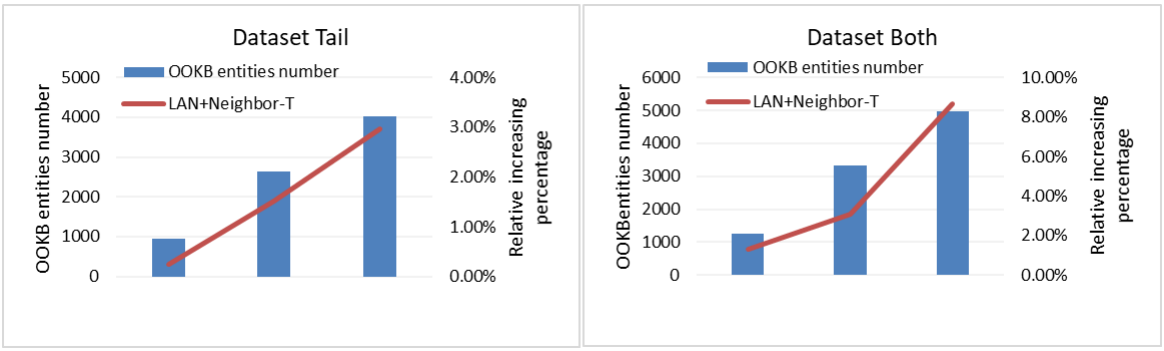

Figure 3. The relevance of proportion of OOKB entities and model effectiveness.

6.4 The influence of transformer over-fitting

As we have mentioned in the previous section, the Transformer's performance will influence the final entity embeddings. In our second training stage, we learn Transformer's parameters by predicting the true entity throughout its neighbor information. Because the Transformer has strong capacity in fitting training data, and the training amount is limited due to OOKB settings, it is easily to cause over-fitting problem. When we use it to construct the representation of OOKB entities, it may fall into some extremely fine-grained contextual semantic patterns. To figure out the impact of Transformer's performance on final entity embedding, we try different dropouts and evaluate the performance of entity prediction. Figure 4 shows that when dropout increase, the Transformer prediction accuracy will decrease, which means the model reduces its fitting capability at the same time. On the contrary, the results of MRR, Hits@10,3,1 increase. The interpretation is the objective of Transformer is to predict the entity. In the second training stage, if transformer accuracy is too high, it means the embeddings from Neighbor-T will always tend to represent the existing entities. Because the average neighbor number of FB15K is much larger than WN11, it means entities in FB15K have more information and easily to cause over-fitting problem. Thus, we assign dropout to 0.1 in triple classification task and 0.8 in the entity prediction task.

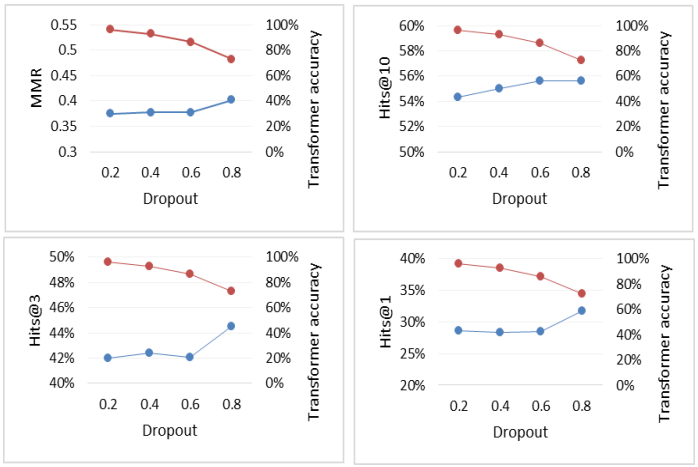

Figure 4. The influence of transformer's over-fitting problem. The red line in all four subgraphs is Transformer accuracy, while the blue line is MRR, Hits@10, Hits@3 and Hits@1 respectively.

7. CONCLUSION

In this paper, we propose a novel aggregator Neighbor-T based on LAN and evaluate it on two KBC tasks under the OOKB setting. Neighbor-T shows effectiveness in utilizing neighbors' contextual information to enhance OOKB entity representations. The extensive experiments on two tasks demonstrate that the enhanced representations from our method achieve the new state-of-the-art results on these tasks. In the future, we might consider utilizing a union model to incorporate both kinds of mentioned graph contextual information to reduce scale of the model and conduct experiments on more challenge KG tasks.

REFERENCES

[1] Miller, G. A., "WordNet: A lexical database for English," Communications of the ACM, 38, 39-41 (1995).
[2] Vrandecic, D. and Krtoetzsch, M., "Wikidata: A free collaborative knowledgebase," Communications of the ACM, 57, 78-85 (2014).
[3] Bollacker, K., "Freebase: A collaboratively created graph database for structuring human knowledge," Proc. of the 2008 ACM SIGMOD Inter. Conf. on Management of Data, 1247-1250 (2008).
[4] Bordes, A., Usunier, N., Garcia-Duran, A. and Weston, J., "Translating embeddings for modeling multi-relational data," Proc. Advances in Neural Information Processing Systems, 2787-2795 (2013).
[5] Kazemi, S. M. and Poole, D., "Simple embedding for link prediction in knowledge graphs," Proc. Advances in Neural Information Processing Systems, 4284-4295 (2018).
[6] Zhang, S., Tay, Y. and Yao, L., "Quaternion knowledge graph embeddings," Proc. Advances in Neural Information Processing Systems, 2735-2745 (2019).
[7] Vashishth, S., Sanyal, S. and Nitin, V., "Composition-based multi-relational graph convolutional networks," Proc. of Inter. Conf. on Learning Representations, (2019).
[8] Hamaguchi, T., Oiwa, H. and Shimbo, M., "Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach," Proc. of the Twenty-Sixth Inter. Joint Conf. on Artificial Intelligence, 1802-1808 (2017).
[9] Wang, P., Han, J. and Li, C., "Logic attention based neighborhood aggregation for inductive knowledge graph embedding," Proc. of the AAAI Conf. on Artificial Intelligence 33, 7152-7159 (2019).
[10] Zhao, M., Jia, W. and Huang, Y., "Attention-based aggregation graph networks for knowledge graph information transfer," Proc. of Pacific-Asia Conf. on Knowledge Discovery and Data Mining, 542-554 (2020).
[11] Devlin, J., Chang, M. W. and Lee, K., "BERT: Pre-training of deep bidirectional transformers for language understanding," Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 1, 4171-4186 (2019).

[12] Fan, M., Zhou, Q. and Zheng, T. F., "Representation learning of knowledge graphs with entity descriptions," Pattern Recognition Letters, 93, 31-37 (2016).
[13] Shi, B. and Weninger, T., "Open-world knowledge graph completion," Proc. of the AAAI Conf. on Artificial Intelligence 32 (1), (2018).
[14] Kong, F., Zhang, R. and Guo, H., "A neural bag-of-words modelling framework for link prediction in knowledge bases with sparse connectivity," The World Wide Web Conf., 2929-2935 (2019).
[15] Shah, H., Villmow, J. and Ulges, A., "An open-world extension to knowledge graph completion models," Proc. of the AAAI Conf. on Artificial Intelligence 33, 3044-3051 (2019).
[16] Teru, K. K., Denis, E. and Hamilton, W., L "Inductive relation prediction by subgraph reasoning," arXiv preprint, (2019).
[17] Yao, L., Mao, C. and Luo, Y., "KG-BERT: BERT for knowledge graph completion," arXiv preprint, (2019).
[18] Wang, Q., Huang, P. and Wang, H., "CoKE: Contextualized knowledge graph embedding," arXiv preprint, (2019).
[19] Dehghani, M., Gouws, S. and Vinyals, O., "Universal transformers," Inter. Conf. on Learning Representations, (2018).
[20] Vaswani, A., Shazeer, N. and Parmar, N., "Attention is all you need," Proc. Advances in Neural Information Processing Systems, 5998-6008 (2017).
[21] Hamilton, W., Ying, Z. and Leskovec, J., "Inductive representation learning on large graphs," Proc. Advances in Neural Information Processing Systems, 1024-1034 (2017).
[22] Wu, Z., Pan, S. and Chen, F., "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning System, 32, 4-24 (2020).
[23] Wang, Z., Zhang, J. and Feng, J., "Knowledge graph embedding by translating on hyperplanes," Proc. of the AAAI Conf. on Artificial Intelligence 14, 1112-1119 (2014).
[24] Bahdanau, D., Cho, K. and Bengio, Y., "Neural machine translation by jointly learning to align and translate," Inter. Conf. on Learning Representations, (2015).
[25] Socher, R., Chen, D. and Manning, C. D., "Reasoning with neural tensor networks for knowledge base completion," Proc. Advances in Neural Information Processing Systems, 926-934 (2013).