Knowledge Graph Completion Based on Graph Representation and Probability Model

Luwei Liu¹, Cui Zhu, Wenjun Zhu
Beijing University of Technology, No.100, Pingleyuan, Chaoyang District, Beijing, China
*blackchalk@163.com

Abstract. Knowledge Graph provides an effective scheme for unstructured knowledge on the Internet, but, to a large extent, the lack of information in the knowledge graph restricts the application of the knowledge graph. Therefore, knowledge graph completion has been an attractive research area. In recent years, many researchers apply graph convolutional neural networks to knowledge graph embedding so as to solve this problem, including R-GCN, SACN et al. The current state-of-the-art SACN uses graph convolutional neural as an encoder to make more accurate embeddings of graph nodes, and uses a convolutional network as a decoder, leading a good performance of link prediction task. In this work, we produce a novel graph representation model based on SACN---property graph convolutional network called PGCN. PGCN treats the knowledge graph as a property graph, regarding the initial embedding vector of entities and relations as the property of nodes and edges. What makes the model different is that it introduces the node clustering before convolving the nodes, so that the graph takes into account the importance of neighbors when aggregating the neighbors of nodes. We adopt the probability model Conv-TransE proposed by SACN for the modeling of relation of graph, Conv-TransE takes advantages of ConvE and use probability method which greatly improves the efficiency of the experiment and avoids the construction of the corrupted triplet. We conduct experiments on the standard datasets WN18RR and FB15k-237, and demonstrate that our new model achieves better performance.

Keywords: Knowledge graph embedding, Graph convolutional network, Link prediction.

1. Introduction
In the wake of rapid development of mobile device and web technology, the Internet generates tens of thousands of data every second. The concept of knowledge graph came into being at this time. Knowledge Graph (KG) organizes and expresses knowledge in a more visualized and richer form, which provides a more effective scheme for heterogeneous, multi-source and unstructured knowledge[1]. KG usually describes things and semantic relationships between things in the form of triples (head entities, relations, tail entities), such as (China, Capital, Beijing). Currently, there are more than 50 KGs established by various organizations around the world, and related application systems have reached more than one hundred [2], among which there are representative Freebase [3], DBpedia [4], NELL [5] and other general knowledge graphs.

However, most of these KGs are constructed semi-automatically or manually, a large number of the hidden relations between entities are not extracted or mined. For example, more than 70% of entities
in Freebase have no relationship “nationality” and “place of birth” [6] [7]. To a great extent, the lack of information constrains the development and application of KGs in various industries. Above all, it is not easy to obtain the missing information through open data on the Internet. Therefore, how to utilize the existing information in the knowledge graph to predict or infer new factual relations instead of re-extracting relations from the text has become the focus of the study.

The principle of Knowledge Graph Completion is to model the graph in an appropriate way, that is, embedding the entities and relations in the graph to the continuous vector space which is called Knowledge Graph Embedding. Then using a series of method to complete the graph on these embedding vectors. The well-known model is Trans-X, including TransE[8], TransH[9], TransD[10], TransG[11], TransM[12] et al. The main idea of these model is to regard the relation in the KGs as the translation operation of head and tail entities in the continuous vector space. The following model is based on tensor factorization, such as DisMult[13], Complex[14]. These methods are to learn representations by treating the triples as matrices or tensors and then decomposing them. In recent years, more and more researchers have introduced graph convolution network (GCN) into knowledge graph which also called graph representation learning, such as R-GCN [15], SACN [16]. GCN is an effective way to learn node embeddings, its core idea is to aggregate the information of its neighbors in the graph neighborhood [17][18]. GCN models take into account the structure information of the graph, leading a higher accuracy rate than the previous model that only focuses on the entities itself.

The lastest proposed model SACN is composed of an encoder and a decoder. The encoder called the weighted graph convolutional network (WGCN), which takes advantage of the structure and relation type of the graph to embed nodes. As the decoder, Conv-TransE is used as a decoder to reconstruct the triples in the knowledge graph to embed the relation more accurately. However, WGCN regards all neighbors equally, no neighbor differences are taken into account when aggregating neighbor’s information for a node. For large-scale graphs, a node has a large number of neighbors, while considering the aggregation of all nodes is not efficient in terms of memory and efficiency. Recently, researchers of The University of Hong Kong proposed a representation learning framework for property graphs called PGE (Property Graph Embedding) [19]. This model uses bias strategy to select neighbor nodes and assigns different bias values to neighbors, so that it can capture the impact of differences among different neighbors on node embedding.

**Fig. 1** An example of treating a KG as a property graph.

Fig.1 shows a subgraph of a Knowledge Graph. Suppose that we are going to learn the embedding vector of the entity “Yao Ming”. In real world, an entity may be associated with many other entities, which means a node (entity) might have a great number of neighbor nodes. Entity “Yao Ming” has more than those relations listed on the diagram, and these neighbor nodes might have different properties, shown in different colors in the Fig.1. In Freebase, we found that an entity could have
relations with more than 4000 entities at most, aggregating information about all these 4,000 neighbors is not an efficient way. Therefore, we consider the knowledge graph as a property graph, and utilizes property information to treat the neighbors of a node unequally, introducing the neighbor selection machine to learn more accurate node embedding efficiently.

In this paper, we proposed a new graph representation model called property graph convolutional network (PGCN) that take advantages of WGCN and PGE to embedding the node more accurately. PGCN regards the knowledge graph as a property graph, the features of the nodes and edges are both called property. Firstly, we classify the neighbors of a node based on their properties. Secondly, aggregating the neighbors of a node according to the clustering results which can capture the influence of the differences between the properties of different nodes on the embedding of the node features. For relation modeling, Conv-TransE in SACN is adopted instead of the previous mode of scoring triples. Conv-TransE takes advantage of the ConvE [20] and uses probability method, which can speed up the training speed and avoid the construction of corrupted triples. The usual way to construct corrupted triples is to randomly replace head or tail entities, which is not convincing enough (see following section).

We demonstrate that PGCN with the probability model Conv-TransE can model the KG much better and make an improvement in the task of knowledge graph completion.

In general, our contributions are summarized as follows:

- We treat KG as a property graph and present a novel graph representation model property graph convolutional network PGCN, which combines the advantages of both WGCN and PGE [19]. PGCN is similar to WGCN but with the difference that PGCN take into account the importance of neighbors when aggregating the vector representations of neighbors.
- We point out the disadvantage of defining a scoring function for each triplet fact to infer whether the triplet is valid or not. This method requires reconstructing corrupted triplet manually, which have an effect on the evaluation of the model.
- We evaluate our model on two standard datasets FB15k-237 [21] and WN18RR. The experimental results show that our method has made an improvement in MRR, Hit@1, Hit@3 and Hit@10.

2. Related Work

Knowledge Graph Completion methods proposed in recent years classified into three categories. The first is the model based on the translation, of which TransE was the founder of such models. Assuming that h and t are vector representations of head and tail entities in the triple respectively. If the triplet (h, r, t) is true, then h+rt is true. TransE has the advantage of few parameters and low computational complexity. However, TransE is unable to model one-to-many and many-to-one relation in a good way. Therefore TransH, TransD, TransG, TransM and other models are proposed later to model complex relation types more effectively. The second is the model based on the matrix decomposition, the typical work of these model is the RECAL [22] put forward by Nickel et al, they represented a triple as a third-order tensor. If a triplet is valid, the position of the tensor is 1, otherwise 0. Then, with each triple, the corresponding tensor value is decomposed into the representation of entity and relation, so that the tensor value is as close as possible to the product of two entity vectors and relation matrix in the triplet. However, RECAL has a large memory occupation and a low speed of calculation. Since that, DisMult, ComplEx are put forward later.

The third is the model based on deep learning, the previous work only focuses on the triples itself but ignores many other information in the process of knowledge graph modeling. Therefore, many researchers consider introducing graph representation learning into knowledge graph embedding. This kind of method treat KG as a graph structure, the head and tail entities are regarded as nodes in the graph structure, likewise, the relations are treated as edges in the graph structure. Graph representation learning try to embedding graph nodes to a low-dimensional vector space and keep nodes connected in the graph close to each other in the vector space. In this way the model can capture the graph structure except for triple only. A hotspot in graph representation learning is Graph Convolutional Network.
(GCN). Graph Convolutional Network aims to update the feature vector of a node by aggregating the vector of the neighbor nodes. Schlichtkrull et al used the Relational Graph Convolutional Network (R-GCN) to model the knowledge Graph. The model consists of an encoder and a decoder. Encoder adopt GCN to aggregate the neighbor information of the entity to produce the representation of the entity, and decoder adopt a decomposition model DisMult as a scoring function to model the relation. Recently Chaoshang et al proposed an End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion, they use WGCN to incorporate all of the neighbors of a node to generate the entity vector representation. A convolutional neural network model Conv-TransE was used to judge whether a triplet is valid and to model the relation in the meantime. However, WGCN did not consider the influence of different neighbors on node representation learning when aggregating neighbors, they treat all neighbors of a node are equals. Inspired by the method PGE, this work introduces a neighbor selection strategy in the process of node representation. By setting the deviation value for the neighbors of the nodes, the neighbors are treated differently, and information is selectively aggregated, the influence of the difference between different node properties on the node feature representation can also be captured. The experimental results show that better quality node representation can be obtained by sampling neighbors rather than using all neighbors, and the effect of the diverse neighbors for a node representation was captured.

Since the knowledge graph completion has reached certain extent, we consider using the previous model as a pre-training to initialize KG as a property graph, the representation of nodes and relations in the graph are regarded as property of this graph. We combine the advantages of WGCN and PGE, adding a selection strategy for neighbor nodes on the basis of WGCN. We propose a novel graph representation model property graph convolutional network (PGCN) to learn the embedding of the nodes and relation more accurately. To avoid the disadvantage of construction of the corrupted triples, we used Conv-TransE to model the relation and judge a triple is true or not. Conv-TransE takes the output of PGCN as input for 2D-convolution, and finally outputs the vector of entity with the dimension of the number of entities. Each value of the vector is a probability value. Conv-TransE maintain the transition of relationships and entities well. The most important, such probability model Conv-TransE is faster than the previous scoring model like ConvKB which need to build the wrong triple list.

3. Property Graph Convolutional Network

Up to now, there have been a lot of methods for knowledge graph embedding. We consider making full use of the model studied by predecessors as a pre-training model to initialize the entities and relations in the KG. So, we can regard the KG as a property graph. Initial vector representations of nodes and edges are treated as properties. On this basis, a graph representation model property graph convolution network PGCN is proposed. PGCN not only makes the most of the knowledge graph as the graph structure information, but also takes full advantage of the edge and node properties, so that the vector representation of entities can be better learned. We then use Conv-TransE to model relations and use probability to measure whether a triple is valid or not.

We usually let a Knowledge Graph \( G=(E, R, T) \), where \( E \) is a set of entities with \( |E|=n \), \( R \) is a set of relation between the two entities with \( |R|=m \). \( T=(h, r, t) \) is a set of triples facts in graph, where \( h \in E \) is the head entity of a triplet, \( t \in E \) is the tail entity of a triplet and \( r \in R \) is a relation between \( h \) and \( t \). Knowledge Graph Completion (link prediction) aims to infer missing triples facts based on the existing facts in KG.

Inspired by WGCN and PGE, we develop PGCN model to develop more precise vector learning for entities. Different from WGCN, we cluster the nodes based on their properties (initial vector). PGCN regards neighbors as unequal nodes and assigns different status to each neighbor node through clustering instead of clustering all neighbors without choice. Through this operation, we can integrate the influence of different neighbors on embedding in the process of learning the vector representation of entity, and the selection of different neighbors can improve the embedding learning of entity. The overall framework of PGCN is shown in the Fig.2.
At the beginning, all nodes are represented equally as black, with different colored edges representing different relations (edge properties). Suppose that the embedding of the node \( V_i \) is going to be learned. Clustering nodes based on properties was used to treat neighbors unequally. If other methods are utilized, the classification method of each node may be different, and it is difficult to guarantee the influence of each neighbor node on the current node \( V_i \). In other words, if we cluster nodes more than once, a neighbor of the node \( V_i \) is classified as similar, but it might be classified as dissimilar in another clustering, then it is difficult to measure the effect of the neighbor node on \( V_i \). Therefore, the same classification method is used for one-hop or multi-hop neighbors of node \( V_i \). The main steps are as follows.

First, we cluster all entity nodes based on the node properties. The main purpose of this step is to divide the neighbors of a node into two categories, one is similar to the current node \( V_i \) and the other is dissimilar to the current node \( V_i \).

Second, to select these different neighbors, we assign a bias value \( b_t \) to the similar neighbors and a bias value \( b_d \) to the dissimilar neighbors. The bias value indicates the probability that a node's neighbors will be selected during aggregation. The larger the value, the greater the probability of being selected. The result obtained in PGE shows that larger \( b_d \) can achieve better embedding performance, so we tend to set a larger value for \( b_d \) to collect more dissimilar neighbors. Through neighbor sampling, we obtain a set of sampled neighbors \( N_v^d \), which contain both structural information and property information. In Fig.2, the sampled neighbors are framed by a dotted line, and the neighbor which is not selected turns gray. Lastly, graph convolution operation will be conducted on the basis of this sampled neighbors \( N_v^d \).

The graph convolutional layer is similar to WGCN. In the process of aggregating neighbors, an edge property is introduced, that is, a learnable parameter \( \alpha \) is set for different edges. The difference between our work and WGCN is when we aggregate the nodes, we consider the set of sampled neighbors \( N_v \) obtained in the previous step instead of all neighbor nodes, because in the real world, the graph tends to have a large number of neighbors, choosing all nodes for aggregation is not a better choice in terms of efficiency and memory.

In particular, let \( V_i \) represent a node in the graph, \( Z_i \in R^{k_i} \) be the input vector for node \( V_i \) at the \( l \)-th layer of graph convolution network, \( k_l \) is the input dimension and \( k_{l+1} \) is the output dimension of the
l-th convolution layer. \( Z_{l}^{i+1} \in R^{k_{l+1}} \) be the output vector for node \( V_l \) at the l-th layer. Suppose that we use 2 layer of convolution, \( Z_{l}^{0} \) is the property vector (initialization vector) of node \( V_l \), our training target is to get the output vector \( Z_{l}^{2} \) of node \( V_l \) after 2-layer of convolution. We denote \( N_l \) as the sampled neighborhood set of the node \( V_l \), so the training process of each node is shown as follows:

\[
Z_{l}^{i+1} = \sigma \left( \sum_{r \in N_l} \alpha_{r} g \left( Z_{l}^{i}, Z_{r}^{i} \right) + Z_{l}^{i} W^{l} \right)
\]  

where \( Z_{l}^{i+1} \in R^{k_{l+1}} \) is the output of PGCN, \( \alpha_{r} \) is the weight of relation \( r \) at the l-th layer, edges with different properties (different relation type) have different \( \alpha_{r} \). \( g \) is typically chosen to be a linear transformation function i.e. \( g \left( Z_{l}^{i}, Z_{r}^{i} \right) = Z_{l}^{i} W^{l} \) with a weight matrix \( W^{l} \in R^{k_{l} \times k_{l+1}} \) [17], \( \sigma \) is the activation function like ReLu.

4. Probability model Conv-TransE

We use the probability model Conv-TransE which is the decoder in SACN to model relations and use probability to determine the validation of a triple. Conv-TransE maintains the translation characteristics of entities and relations based on the model ConvE. Fig.3 shows the architecture of the Conv-TransE. Since the embedding dimensions of relation and entity are both \( k \), we can stack the entity embedding and relation embedding as a matrix \( X \) to feed to the convolution layer. For a triple, we look up for the head entity embedding \( e_{h} \in R^{k} \) in the entity embedding matrix obtained from the PGCN in the previous step, and look up for the relation embedding \( e_{r} \in R^{k} \) in the relation matrix initialized by the pre-trained model. Several convolutional kernels \( \omega \) operate on the input matrix repeatedly to get the feature map \( N \). Before convolution, we need to fill in fixed number zero around the input matrix to ensure that the feature map \( F \) is of size \( k \). In detail, we let \( X = [e_{h}e_{r}] \in R^{k \times 2} \), \( X_{i,j} \) is the i-th row to the j-th row of the matrix \( X \), let \( N = [n_{1}, n_{2}, n_{3} ... n_{k}] \in R^{k} \). \( n_{i} \) is given by the following formula:

\[
n_{i} = h(\omega \cdot X_{i,j} + b)
\]  

where \( h \) is the activation function and \( b \in R \) is a bias value. Then we concatenate all the feature maps to a single vector \( P \in R^{\omega[t+1]} \), where \( t \) is the size of kernels, \( t \in \{1,2,3,...\} \). After fully-connected layers learning, we get a vector of size \( |E| \), then we take sigmoid to get the prediction vector \( P_{h} \), each term of \( P_{h} \) is a probability that represent the degree of the corresponding entity to be a correct tail entity. Finally, we obtain the prediction vector \( P_{h} \) as follows:

\[
P_{h} = \sigma(f(concat(g([e_{h},e_{r}] \ast M))))
\]  

where \( M \) is the set of kernels, \( concat \) represents a concatenation operator and \( \ast \) denotes the convolution operator. \( f \) denotes a non-linear function which transform the single vector \( P \in R^{\omega[t+1]} \) to \( P \in R^{|E|} \).

Given a valid triple \( (h, r, t) \), we get a prediction vector \( P_{h} \) with the size=|E|, all the entities get a probability value that indicate the possibility they appear as the correct tail entity of the current triple. Meanwhile, each triple \( (h, r, t) \) have a target binary vector \( I \in R^{|E|} \), the term of vector \( I \) are defined as follows:

\[
I_{i} = \begin{cases} 
1, & (h,r,t_{i}) \in \tau \\
0, & \text{otherwise} 
\end{cases}
\]  

where \( \tau \) is the valid triple set of datasets, \( t_{i} \in E \).
To avoid overfitting, we utilize label smoothing [23] to normalize the target binary vector. We also employ dropout on embeddings to reduce the problem.

![Fig.3](image)

**Fig.3** The framework of the probability model Conv-TransE. In this case, the kernel size is 3*2, the channel of kernels is 3.

5. Disadvantages of the scoring function

In previous work, most methods defined a scoring function for triples, resulting in higher scores for valid triples and lower scores for invalid triples during training. To do this, we need to construct invalid triples artificially, they usually replace the head or tail entities in the valid triplets randomly. However, there are two disadvantages:

- Because of the random replacement, the model could be fairly simple. For such a valid triple (China, the capital, Beijing), when we replace the tail entity, the randomness could make the tail entity “Beijing” be a "name", "gender" or "professional". However, this kind of triples is obviously not true. So, for a corrupted triple like (China, the capital, Shanghai), it is difficult for the model to make a right judgment, that is, to give a lower score.

- In the course of the experiment, there is a situation that the corrupted triples get the same scores as the valid triple [24]. When we calculate the Hit@K, there will be ambiguity. Assuming that we get the rank of the corrupted triple list, the top-3 are triples with the same scores as the valid triple. If we insert the valid triple into the first position of the corrupted triple list, for metric of Hit@3, it hits. But if we insert the valid triple into the fourth position, for Hit@3, it just misses it. That is to say, the insertion position of the valid triple has a great influence on the accuracy when the invalid triples get the same scores as the valid triple. The higher the insertion position, the higher the accuracy.

6. Experiments

In this section, we evaluate our model on the task of link prediction.

We follow the benchmark dataset used in the previous work: WN18RR [20] and FB15K-237 [21]. Tab.1 shows the exact statistics of the experimental datasets:

| Dataset     | #R | #E   | #Train  | #Valid | #Test  |
|-------------|----|------|---------|--------|--------|
| WN18RR      | 11 | 40,943 | 86,835  | 3,034  | 3,134  |
| FB15k-237   | 237| 14,541| 272,115 | 17,535 | 20,466 |

**Tab. 1** shows the specific data. #R, #E are the number of entities and relations. #Train, #Valid and #Test are the size of training set, validation set and test set.
WN18RR: WN18RR is selected from WN18, which is a subset of Wordnet. WN18 contains 18 relations, 40,943 entities and 151,442 triples. But the dataset involves reversible relations, which make the model surprisingly simple. Therefore, WN18RR remove all the reversible relations in WN18. Finally, it includes 11 relations, 40,943 entities and 90,330 triples.

FB15k-237: FB125K-237 is derived from FB15K, which is a subset of Freebase. Freebase is a large universal knowledge graph. FB15k selects a subset of entities in Freebase, which are also present in the Wikilinks database, and each of these entities and relations appears at least 100 times in Freebase. Similarly, FB15k-237 remove all the inverse relations.

Same as the previous work, we used MRR, Hit@1, Hit@3, and Hit@10 as the metrics for our method. For the prediction label vectors with the entity size generated by probability model Conv-TransE, we sort each entry of the vector and rank the position of the correct tail entity denoted as rank.

In the process of sorting, we filter out all valid triples following Bordes et al.

MRR: Mean square Rank. the average of the reciprocal of rank in all triples in the test set. When the value of rank is smaller, it indicates that the ranking position of the correct tail entities is higher, which proves that our method has stronger prediction ability.

Hit@K: The proportion of the correct tail entities hit in the first K of the order. The higher the value, the higher the accuracy of the algorithm. Hit@1 refers to the first-place hit ratio and determines whether the probability of the correct tail entity in each test triplet is the highest. If it is the highest, that is, the rank is 1, we record it 1, otherwise 0. Hit@3, the first 3 hit rate, judging whether the rank of the correct tail entity in each test triplet is in the top three, if so, 1 is recorded, otherwise 0. Hit@10 does the same thing.

Following ConvKb [25], we used the entity and relation vectors produced by STTransE to initialize the entity and relation vectors in PGCN and Conv-TransE. To obtain sampling neighbors $N^g$, we used DBSCAN as in jto cluster the entity nodes based on the initial vector. We used 25 to be the size of sampling neighbors $N^g$ by following GraphSAGE [26]. We adopted Adam as the optimizer and set the learning rate $\in \{0.001, 0.002, 0.003\}$, dropout rate $\in \{0.0, 0.1, 0.2, 0.3\}$. We selected the embedding size $\in \{100, 200\}$. For the kernel size t, we chose it from $\{1, 3, 5\}$, and the channel of the kernels was selected from $\{100, 200, 300\}$. We used ReLu as the g activation function.

Two-top was used in PGCN, we found the best result with the settings are: learning rate=0.003, embeddingsize=200, kernel size=3, channel of kernels=200, dropout=0.2 for WN18RR, learning rate=0.003, embedding size=200, kernel size=5, channel of kernels=100, dropout=0.2 for FB15k-237.

7. Result

Tab.2 shows the experimental results of link prediction on the two benchmark datasets WN18RR and FB15k-237. We first select four previous work DistMult, ComplEx, R-GCN and ConvE to compare with our work, the result shows that our work do the best on the metric of MRR and Hit@ {1,3,10}. In FB15k-237, compared to the best model ConvE, our approach has about 9% significant improvement in MRR and hit@3, obtains 4% improvement in Hit@1 and gains 8% improvement in Hit@10. In WN18RRR, comparing ComplEx, our model improves upon Hit@3 by margin of 7%, and upon hit@10 by margin of 8%. All these results show that graph convolutional network can enhance the vector representation learning of entity nodes, thus improving the performance of link prediction.

Especially, we reimplemented the model SACN within the same framework that they mentioned in the paper, we found that our work achieves the best performance. Specifically, In FB15k-237, our work increases the performance by 2.3% on Hit@10. In WN18RR, our work increases the performance by 3.6% on hit@3. In general, these results verify our previous analysis, selecting neighbors rather than aggregating all the neighbors helps improves the quality of node embedding.

In terms of time efficiency, we also found that the training time of each epoch decreased by 3.1%, from 6.38 minutes to 6.25 minutes for WN18RR. If the epoch was set at 1000, nearly 2 hours could be saved. For FB15k-237, the time decreased by 35%, from 1.08 minutes to 0.7 minutes, which could
save about 6 hours. The specific data is shown in Tab.3. All the above results prove that adding neighbor selection strategy in the process of graph convolution can get better embedding performance with greater time efficiency.

| Model    | MRR   | Hit@1 | Hit@3 | Hit@10 | MRR   | Hit@1 | Hit@3 | Hit@10 |
|----------|-------|-------|-------|--------|-------|-------|-------|--------|
| DistMult*| 0.24  | 0.16  | 0.26  | 0.42   | 0.43  | 0.39  | 0.44  | 0.49   |
| ComplEx* | 0.25  | 0.16  | 0.28  | 0.43   | 0.44  | 0.41  | 0.46  | 0.51   |
| R-GCN*   | 0.25  | 0.15  | 0.26  | 0.42   | -     | -     | -     | -      |
| ConvE*   | 0.32  | 0.24  | 0.35  | 0.49   | 0.46  | 0.39  | 0.43  | 0.48   |
| SACN     | 0.342 | 0.253 | 0.375 | 0.518  | 0.467 | 0.430 | 0.477 | 0.537  |
| Our work | 0.345 | 0.254 | 0.381 | 0.530  | 0.469 | 0.417 | 0.494 | 0.550  |

Tab.2 Results of Link prediction on the WN18RR and FB15k-237. Results marked (*) taken from Chao Shang. (2019).

| Model  | Batchsize | Runtime of an epoch | Batchsize | Runtime of an epoch |
|--------|-----------|---------------------|-----------|---------------------|
| SACN   | 512       | 1.08min             | 128       | 6.38min             |
| Our method | 512   | 0.7min              | 128       | 6.25min             |

Tab.3 Results of training running time on the WN18RR and FB15k-237.

8. Conclusion
In this paper, we have introduced a novel graph representation model property graph convolutional network PGCN and adopted the probability model Conv-TransE for the knowledge base completion task. PGCN regards the knowledge graph as a property graph for the first time and initializes the nodes and edges of the graph using the model proposed by previous researchers as a pre-training model. PGCN adds neighbor selection strategy to graph convolution network to improve the vector representation learning quality of entity nodes and also improve time efficiency. Then we used the probability model Conv-TransE to measure a triple is valid or not and model the relation at the same time. Conv-TransE avoids constructing the invalid triples and makes the process of validation faster.

During the experiments, we found that the accuracy of head entities is usually higher than that of tail entities. In the future, we will consider to reduce this gap so as to improve the overall accuracy. In addition, we also consider applying the knowledge graph completion method to the professional knowledge graph, such as the financial knowledge graph.

References
[1] Karie N M, Sahri N M, Haskell-Dowland P P 2020 IoT Threat Detection Advances, Challenges and Future Directions ETSecIoT (Sydney: IEEE) pp22-29
[2] Minghao H A Research of Relational Inference Algorithm Based on Knowledge Graph 2018 p 77
[3] Yuanzhuo W, Yantao J, Dawei J, Xiaolong J and Xueqi Ch 2015 Open Web Knowledge Aided Information Search and Data Mining Journal of Computer Research and Development 52 pp 456-474
[4] Kurt B, Colin E, Praveen P, Tim S and Jamie T Freebase A collaboratively created graph database for structuring human knowledge 2008 Proc. of the 2008 ACM SIGMOD Int. Conf. on Management of data (New York: Association for Computing Machinery) pp 1247-1250

[5] Sören A, Christian B, Georgi K, Jens L, Richard C and Zachary I DBpedia 2007 A nucleus for a Web of open data The Semantic Web (Berlin: Springer Berlin Heidelberg) pp 722-735

[6] Andrew C, Justin B, Bryan K, Burr S, Estevam H and Tom M 2010 Toward an architecture for never-ending language learning Proc. of the Twenty-Fourth AAAI Conf. on Artificial Intelligence (Atlanta: AAAI Press) pp1306-1313

[7] Xin D, Evgeniy G, Geremy H, Wilko H, Ni L, Kevin M, Thomas S, Shaohua S and Wei Zh Knowledge Vault 2014 A Web-Scale Approach to Probabilistic Knowledge Fusion Proc. of the 20 the ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (New York: Association for Computing Machinery) pp601–610

[8] Denis K, Stephan B and Volker T 2015 Type-Constrained Representation Learning in Knowledge Graphs Proc. of the 14th Int. Conf. on The Semantic Web (Cham: Springer International Publishing) pp640–655

[9] Antoine B, Nicolas U, Alberto G, Jason W and Oksana Y 2013 Translating embeddings for modeling multi-relational data Proc. of the 26th Int. Conf. on Neural Information Processing Systems (New York: Curran Associates Inc.) pp2787–2795

[10] Zhen W, Jianwen Zh, Jianlin F and Zhen Ch 2014 Knowledge graph embedding by translating on hyperplanes In Proc. of the 28th AAAI Con. on Artificial Intelligence pp1112-1119

[11] Guoliang J, Shizhu H and Liheng X 2015 Knowledge Graph Embedding via Dynamic Mapping Matrix In Proc. of the Association for Computational Linguistics pp 687-696

[12] Han X, Minlie H and Yu H Trans G 2015 A Generative Mixture Model for Knowledge Graph Embedding Proc. of the 54th Annual Meeting of the Association for Computational Linguistics (Berlin: Association for Computational Linguistics) pp2316-2315

[13] Nouhad A, Jean-Paul B and Jean-Paul B TransM 2004 A Structured Document Transformation Model In Proc. of the Information Systems Technology and its Applications pp53-66

[14] Bishan Y, Wen-tau Y, Xiaodong H, Jianfeng G and Li D 2015 Embedding entities and relations for learning and inference in knowledge bases Preprint

[15] Théo T, Johannes W, Sebastian R, Éric G and Guillaume B 2016 Complex embeddings for simple link prediction Proc. of the 33rd Int. Conf. on International Conference on Machine Learning (New York: JMLR.org) pp2071-2080

[16] Michael S, Thomas K, Peter B, Rianne B, Ivan T and Max W 2018 Modeling relational data with graph convolutional networks The Semantic Web (Cham: Springer) pp593-607

[17] Chao Sh, Yun T, Jing H, Jinbo B, Xiaodong H and Bowen Zh 2018 End-to-end Structure-Aware Convolutional Networks for Knowledge Base Completion Proc. of the AAAI Conf. on Artificial Intelligence pp 3060-3067

[18] Thomas K and Max W 2017 Semi-supervised classification with graph convolutional networks Preprint

[19] David D, Dougal M, Jorge A, Rafael G, Timothy H, Alán A and Ryan P 2015 Convolutional networks on graphs for learning molecular fingerprints Advances in Neural Information Processing Systems (Massachusetts: MIT Press) pp2224-2232

[20] Yifan H, Hongzhi Ch, Changji L, James Ch, Ming-Chang Y and Fan Y 2016 A Representation Learning Framework for Property Graphs 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining pp65–73

[21] Tim D, Pasquale M, Pontus S and Sebastian R 2018 Convolutional 2d knowledge graph embeddings National conference on artificial intelligence

[22] Kristina T, Danqi Ch 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 pp57–66

[23] Maximilian N, Volker T and Hans K 2011 A Three-Way Model for Collective Learning on
Multi-Relational Data  

In Proc. of the Int. Conf. on Machine Learning (Washington: Omnipress) pp 809-816

[24] Christian S, Vincent V, Sergey L, Jon S and Zbigniew W 2016 Rethinking the inception architecture for computer vision IEEE Conf. on Computer Vision and Pattern Recognition pp2818–2826

[25] Zhiqing S, Shikhar V, Partha T and Yiming Y 2020 A Re-evaluation of Knowledge Graph Completion Methods Proc. of the 58th Annual Meeting of the Association for Computational Linguistics pp5516-5522

[26] Dai Quoc N, Tu Dinh N, Dat Quoc N and Dinh P 2018 A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network Proc. of the 2018 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies pp327–333

[27] William L H, Rex Y, and Jure L 2017 Inductive representation learning on large graphs Proc. of the 31st Int. Conf. on Neural Information Processing Systems (California: Curran Associates Inc) pp1025-1035