Attention Relational Graph Convolution Networks for Relation Prediction in Knowledge Graphs

Shuo Wang¹*, Yi Zhong¹ and Chengpeng Wang¹

¹Information and Communication Engineering, Wuhan University of Technology, Wuhan, Hubei, 430070, China
* wongshuo@whut.edu.cn

Abstract. Recently, the tasks based on knowledge graph (KG), such as question answering, information retrieval, are more and more widely used. However, due to the incompleteness of the relations (links) between entities, it is very important to study the relations prediction based on KG to complete the missing relations between entities. In recent years, graph convolution networks (GCNs) have been a new method to solve the reasoning of knowledge graph. However, the existing knowledge graph relation prediction model based on GCNs fails to consider the importance between nodes. In order to obtain more abundant relations information between entities (nodes), inspired by the graph attention network (GAT), we propose an attention weighted relational graph convolutional network (denoted as AWR-GCN), which is used as an encoder of the encoder-decoder model for relationship prediction, and the decoder is a linear model. Compared with the advanced methods in the commonly used relational prediction data sets, our model has a certain performance improvement and reached advanced level.

1. Introduction
In recent years, knowledge engineering represented by knowledge graph and deep learning represented by neural network have developed rapidly. Fusion of knowledge graph and graph neural network has become an important technical idea for researchers to further improve knowledge graph learning and enhance reasoning ability of graph neural network model.

How to construct, represent, complete and apply knowledge graph better has become one of the important research directions in the field of cognition and artificial intelligence. Knowledge graph itself is a kind of graph structure data, which uses graph to build the relations between knowledge and data, and applies graph neural network technology.

Knowledge graph reasoning mainly refers to knowledge graph completion or link prediction, aiming at predicting the connected edge (relations) between two entities given, or predicting the tail/head entity given the head/tail entity and relations. In recent years, the combination of graph neural network and knowledge graph has become a new method to solve knowledge graph reasoning.

The known relational prediction methods are roughly divided into translation model[1][2][3] and convolutional neural network (CNN)[4][5]. The embedding quality of translation model is low, while CNN based model can learn more expressive embedding. However, both the translation based model and the CNN based model deal with each triples independently, so it is impossible to integrate the semantic rich potential relations near the specified entities in KG. The graph based neural network model, called R-GCN[6], is an extension of the graph convolution network (GCN) applied to relational data[7]. R-GCN learns a mapping matrix for each relation, which is used to change the influence of different relations when accumulating the weight of neighbors. The disadvantage of R-GCN is treating different
neighbor nodes equally. TuckER[8] is a fully expressed model, and sufficient bounds can be obtained in its embedding dimension.

Our thoughts are as follows. The graph attention mechanism is used to add weights to the nodes of R-GCN clustering for relations, which is used to predict the relations on KGs. An attention weighted relational graph convolutional network (AWR-GCN) is proposed, which combines the advantages of R-GCN and GAT[9]. Our model solves the problem that R-GCN does not consider the weight of nodes in different domains to the central node by assigning different weight quality (attention) to the nodes in the neighborhood.

Our architecture is an encoder-decoder model, with AWR-GCN as the encoder and linear model Tucker as the decoder. The output of AWR-GCN becomes the input of decode. Decoder scores the prediction of triples to perform link prediction task. AWR-GCN allocates different attention (weight) to nodes in the same relations by using the structure of adjacent edge and neighbor node of knowledge graph, so as to embed graph nodes more accurately.

Our contributions are summarized as follows:
1) We propose a new knowledge coding model, attention weighted relational graph convolutional network denoted as AWR-GCN, for knowledge graph relation prediction. It combines the advantages of R-GCN and GAT, adds the weight between nodes (attention coefficient), solves the disadvantage that R-GCN treats different entities in the neighborhood equally, and also improves the problem that GAT ignores edges (relations).
2) Using TuckER as the decoder. The decoder DistMult, used by R-GCN in connection prediction can only predict symmetrical relations. The decoder Tucker has a better effect on relations prediction.

2. Related Work
The relations prediction of knowledge graph is a very important research direction, which is directly applied to the improvement of knowledge base. For relational prediction, knowledge graph embedding learning is its underlying task, which obviously has a deep impact on relational prediction. For a long time, many improved models of knowledge graph embedding learning are often used in relation prediction. It can be roughly divided into the following types: 1) translation model, 2) bilinear model, 3) CNN based model, 4) graph neural network model.

Translation models, including TransE[1], TransH[2] and TransD[3], regard relations as the translation between head entities and tail entities. Bilinear models, including Rescal[10], DistMult[11], ComplEx[12] and so on, are used to calculate the credibility of potential semantics of entities and relations in vector space.

Recently, two models based on CNN, which are often mentioned recently, are used in relation prediction, namely ConvE[4] and ConvKB[5]. ConvE is a model that uses two-dimensional convolution on embedding of different embedding dimensions. It transforms head entities and relations into two-dimensional vectors, and then uses convolution layer and full connection layer to obtain interactive information. ConvKB directly splices the embedded representation of head and tail entities and relations, and then uses CNN.

R-GCN is a graph-based neural network model, which is the first to apply GCN framework to relational data modeling in link prediction. R-GCN adds support for multiple relations in graph convolution, and learns a mapping matrix for each relationship. However, due to the equal weight assigned to different nodes of the same relationship, the importance relationship between nodes is ignored.

3. Proposed Method
3.1. Background
Knowledge graph $\mathcal{G}$ is a marked and directed network, which is expressed as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{K})$.

Where $\mathcal{E}$ is the set of entities (nodes), $\mathcal{R}$ is the set of relations (edges), and $\mathcal{K}$ is the type of relations. A triple $t = (e_h, r, e_t)$, denotes the head entity $e_h$ in $\mathcal{G}$, which is connected by the relation $\mathcal{R}$ (edge) and
points to the tail entity $e_t$. In embedded triples, we encode features such as entities and relations that need to be learned. Then the triple $t$ is scored by using the scoring function $f(t)$, which makes the effective triples have higher scores than the invalid triples. For example, the score of $(Beijing, \text{capital_of}, \text{China})$ should be higher than that of $(Beijing, \text{capital_of}, \text{United Kingdom})$. The connection prediction is shown in the partial graph of KG in Figure 1, and the missing connection is inferred according to the existing triples. (For example, in Figure 1, entity Barron Trump is the son of Melania, and Melania is Donald Trump's wife. It is inferred that Barron is Donald Trump's son and brother to other children of Donald Trump.)

![Figure 1](image)

**Figure 1.** The subgraph of the knowledge graph contains the inference of the existing relationship (solid line) and hidden relationship (dashed line) between entities. Different types of entities are represented by ellipses of different colors, and different types of relationships are represented by arrows of different colors.

### 3.2. AWR-GCN encoder

AWR-GCN is an extension of R-GCN. R-GCN collects relational information from entity domain, and all information from neighbors shares the same weight. Graph attention network (GAT) can solve the shortcomings of GCNs by assigning different levels of importance to each node in its neighborhood. GAT layer is defined as $e_{ij} = a(Wx_i, Wx_j)$. Where $e_{ij}$ is the focus value of edge $(e_i, e_j)$ in $\mathcal{G}$, $W$ is a parameterized linear transformation matrix and $a$ is attention function.

Although GAT solves the problem of weight, it ignores the characteristics of relations (edge) between entities (nodes). However, for KG, the feature of relations (edge) is an indispensable part. In the knowledge graph, the relations between entities will directly affect the role of entities. For example, in Figure 1, Donald Trump plays the role of president for U.S. and father for Barron, respectively. So we use a novel method of embedding, which takes the feature attention mechanism of the included relations and adjacent nodes as the weight of R-GCN for link prediction.

The entities embedded in the attention layer are a group of features which are classified according to the type of edge and aggregate neighbor nodes, $H \in \mathbb{R}^{N_n \times F_n}$, Where $N_n$ is the number of nodes and $F_n$ is the number of features in each node. Its output is $H' \in \mathbb{R}^{N_n \times F'_n}$.

In the attention layer, relations embed a set of edge features $S \in \mathbb{R}^{N_r \times F_r}$, where $N_r$ is the number of connected edges and $F_r$ is the number of features in each connected edge. Its output is $S' \in \mathbb{R}^{N_r \times F'_r}$.

The weight matrix $W_n \in \mathbb{R}^{F_n \times F'_n}$ is used for parameterized shared linear transformation, which is applied to each node to obtain enough expression ability. For the triples $t = (e_i, r, e_j)$, the weight vector $W_n \in \mathbb{R}^{F'_n \times F_n}$ is parameterized. Define $z^{(l)}_i = W_n h_i$, vector $h_i$ denotes the embedding of $e_i$. 

$3$
Calculating the attention coefficient:

\[ e_{ij}^{(l)} = \text{LeakyReLU} \left( a_{ij}^{(l)} \left| z_i^{(l)} \right| z_j^{(l)} \right) \]  \hspace{1cm} (1)

It shows the importance of the characteristics of node \( e_i \) to node \( e_j \). (Figure 2) Next, we use softmax to normalize \( e_{ij} \):

\[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{e \in E} \exp(e_{in})} \]  \hspace{1cm} (2)

Figure 2. The attention mechanism used in our model

Finally, using the normalized concern coefficient \( \alpha_{ij} \) as the weight, a new propagation model is defined, which is used to calculate the forward update of \( e_i \):

\[ h_i^{(l+1)} = \sigma \left( \sum_{r \in R} \sum_{j \in \mathcal{N}_i} a_{ij} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right) \]  \hspace{1cm} (3)

Where \( \mathcal{N}_i \) vertex \( e_i \) is adjacent to the relation \( r \in R \) of the vertex set. The feature \( h_j^{(l)} \) of adjacent vertices is multiplied by the weight \( W_r^{(l)} \) corresponding to the type of the edge, and then multiplied by the attention coefficient \( \alpha_{ij} \). Finally, in order to retain the information of the node itself, the information transmitted from the single self-connection is added. After the activation function, it is used as the output of this layer and the input of the next layer. We not only assign different weights to each neighbor node \( e_j \) of vertex \( e_i \), but also consider the type and direction of edges. The calculation diagram of the single-node update in the AWR-GCN model is shown in Figure 2.

Figure 3. A graph used to calculate updates to a single graph node / entity (the largest green node) in the AWR-GCN model. Firstly, the information from the neighboring nodes (the smaller nodes around the central node) is collected, and each relationship type is aggregated (the incoming and outgoing edges...
are also regarded as two types of relations). Then the node information of each relationship type is used as the input of the GAT layer, and the attention coefficient is multiplied as the weight and the vector representation of the node, and then the results of each relationship type are accumulated by transforming the weight values corresponding to each relationship. Finally, the accumulated results are passed through the activation function (ReLu).

3.3. Decoder

We introduce a graph automatic encoder model, which consists of an entity encoder and a rating function (decoder). The decoder's score on a triplet indicates whether the triplet is true or not. The final goal is to infer whether the triples \((e_h, r, e_t)\) are valid by the fraction of triples \((e_h, r, e_t)\). The encoder graphs each entity \(v_t \in E\) to a real vector \(e_t \in \mathbb{R}^{F_e}\).

In our experiment, we use Tucker as the decoder (scoring function), and propose a link prediction model using Tucker decomposition for binary tensor representation of knowledge graph. The model performs well on the standard link prediction benchmark. The scoring function of \(t = (e_h, r, e_t)\) is defined as:

\[
f(e_h, r, e_t) = W \times_1 v_h \times_2 v_r \times_3 v_t
\]  

(4)

\(\times_n\) represents tensor product along sigmoid mode. \(v_h, v_t \in \mathbb{R}^{F_e}\) is the embedding vector of \(e_h\) and \(e_t\). \(v_r \in \mathbb{R}^{F_r}\) is the embedding vector of relation \(r, W \in \mathbb{R}^{F_n \times F_r \times F_e}\) is the core tensor. Then apply the Sigmoid function to each score \(f(e_h, r, e_t)\) to obtain the prediction probability \(p\).

Add a reciprocal relation to each triple in the dataset, we score the entity relation pairs \((e_h, r)\) and \((e_t, r^{-1})\) of all head and tail entities. Train a single triple \(t = (e_h, r, e_t)\) and add the reciprocal relational triple \(t^{-1} = (e_t, r^{-1}, e_h)\) to it. The loss of an entity-relations pair with all other entities is defined as:

\[
L = -\frac{1}{N_n} \sum_{i=1}^{N_n} (y^{(i)} \log(p^{(i)}) + (1 - y^{(i)}) \log(1 - p^{(i)}))
\]

(5)

Where \(N_n\) is the total number of nodes, where \(p \in \mathbb{R}^{N_n}\) is the probability vector predicted by the score function \(f(e_h, r, e_t)\), and \(y \in \mathbb{R}^{N_n}\) is the binary marker vector. For the correct triple \(y^{(i)} = 1\), conversely, if it is the wrong triple, then \(y^{(i)} = 0\).

4. Experiments and Results

4.1. Datasets

FB15k [1] and WN18[1] are usually used to evaluate link prediction algorithms. Previous work [13][14] suggested that there are a large number of inverse relations between WN18 and FB15K. To solve this problem, the corresponding subset datasets FB15k-237[15] and WN18RR[16] are created.

To evaluate our improved approach, we used the following four datasets: WN18RR, FB15k-237, WN18, FB15k.

4.2. Training Protocol

We follow the two-step training process, that is, we first train AWR-GCN to classify the aggregation of neighbor nodes according to the type of edges, and train GAT for the aggregation of nodes classified by each relation to get the attention coefficient between entities, then encode the information about graph entities and relations, and finally train the decoder model of TuckER to perform the task of relations prediction.

The optimizer we used is Adam [17] and its initial learning rate is set to 0.01. The entity and relations embedding of the last layer are both set to 200. The batch size is 128.

4.3. Evaluation Protocol

We use three evaluation indicators: mean reciprocal rank (MRR) and Hits\(_@N\), \(N \in \{1, 2, 3\}\) (that is, the proportion of effective test triple ranking in the top N prediction). Better performance is represented as higher MRR or higher Hits\(_@N\).
4.4. Results

The link prediction results for the datasets we used are shown in tables 1 and 2. AWR-GCN outperforms the previous model in using most of the indicators of the datasets. The results obtained by AWR-GCN are better than those of many deep neural networks and reinforcement learning structures, and also better than those of other linear models.

Table 1. Experimental results on FB15K-237 and WN18RR. The Hits @N value is expressed as a percentage.

|          | FB15K-237 | WN18RR |
|----------|-----------|--------|
|          | Hits @N   |        |
|          | MRR @1 @3 @10 | MRR @1 @3 @10 |
| DistMul[20] | 0.281 0.199 0.301 0.446 | 0.444 0.412 0.47 0.504 |
| ComplEx [13] | 0.278 0.194 0.297 0.45 | 0.449 0.409 0.469 0.53 |
| TransE    | 0.279 0.198 0.376 0.441 | 0.243 0.043 0.441 0.532 |
| ConvE     | 0.312 0.225 0.341 0.497 | 0.456 0.419 0.47 0.531 |
| R-GCN     | 0.248 0.151 0.264 0.417 | 0.123 0.207 0.137 0.08 |
| TuckER    | 0.358 0.266 0.394 0.544 | 0.470 0.443 0.482 0.526 |
| Ours      | 0.452 0.334 0.461 0.572 | 0.446 0.395 0.484 0.561 |

We first compare the AWR-GCN model with these six models and compare the baseline model. Compared with all the benchmarks, the performance of AWR-GCN is very good. In the FB15k-237 dataset, our AWR-GCN increased TuckER’s Hits @10 by 2.8% and by 6.7%. In the WN18RR dataset, AWR-GCN improves the Hits @10 range of TuckER by 3.5%.

In Table 2, AWR-GCN also achieves better performance. In FB15K, our AWR-GCN model improved by 5.5% Hits @10 and Hits @ 3 by 6.1% compared to R-GCN. Our AWR-GCN model increases the value of Hits @ 3 by 3.4% compared to R-GCN, in the WN18 dataset.

Table 2. Link prediction results on WN18 and FB15k. The Hits @N value is expressed as a percentage.

|          | WN18   | FB15K  |
|----------|--------|--------|
|          | MRR @1 @3 @10 | MRR @1 @3 @10 |
| DistMul[20] | 0.822 0.728 0.914 0.936 | 0.654 0.546 0.733 0.824 |
| ComplEx | 0.941 0.936 0.936 0.947 | 0.692 0.599 0.759 0.840 |
| TransE | - - - 0.892 | - - - 0.471 |
| ConvE | 0.943 0.935 0.946 0.956 | 0.657 0.558 0.723 0.831 |
| R-GCN | 0.819 0.697 0.929 0.964 | 0.696 0.601 0.760 0.842 |
| TuckER | 0.953 0.949 0.955 0.958 | 0.795 0.741 0.833 0.892 |
| Ours   | 0.961 0.951 0.963 0.961 | 0.803 0.752 0.821 0.897 |

5. Conclusion

In this paper, we propose a new graph convolution network (AWR-GCN) which takes the attention coefficient based on graph attention as weight, which is specifically aimed at the relations prediction of KG, and proves its effectiveness in the problem of link prediction. Each node in the neighborhood is assigned different importance levels, which solves the disadvantage of sharing coefficients among all nodes in GCNs. In the relational prediction task, we use AWR-GCN as the encoder and TuckER as the decoder to score the predicted triple, and achieve competitive results on the linked prediction datasets.

Reference

[1] Antoine Bordes, Nicolas Usunier, et al. 2013. Translating embeddings for modeling muti-relational data. Advances in Neural Information Processing Systems (NIPS), 2013, pages 2787–2795.
[2] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. International Conference on Learning Representations (ICLR), 2015.

[3] Theo Trouillon, Johannes Welbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In International Conference on Machine Learning (ICML), 2016, pages 2071–2080.

[4] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In Thirty-Second AAAI Conference on Artificial Intelligence (AAAI), 2018.

[5] Dennis Diefenbach, Kamal Singh, and Pierre Maret. 2018. Wdaqua-core1: a question answering service for rdf knowledge bases. In Companion of the The Web Conference 2018 on The Web Conference (WWW), 2018, pages 1087–1091. International World Wide Web Conferences Steering Committee.

[6] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In European Semantic Web Conference (ESWC), 2018, pages 593–607.

[7] Thomas N. Kipf and Max Welling. 2017. Semisupervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.

[8] Balaevi I, Allen C, Hospedales T M. TuckER: Tensor Factorization for Knowledge Graph Completion[J]. 2019.

[9] Velikovi P, Cucurull G, Casanova A, et al. Graph Attention Networks[J]. International Conference on Learning Representations (ICLR), 2018.

[10] Nickel M, Tresp V, Kriegel H P. A Three-Way Model for Collective Learning on Multi-Relational Data[C]// International Conference on International Conference on Machine Learning. Omnipress, 2011.

[11] Yang, B.; Yih, W.-t.; He, X.; Gao, J.; and Deng, L. 2014. Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575.

[12] Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, E.; and Bouchard, G. 2016. Complex embeddings for simple link prediction. In International Conference on Machine Learning, 2071–2080.

[13] Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A novel embedding model for knowledge base completion based on convolutional neural network. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), 2018, volume 2, pages 327–333.

[14] Chakraborty N, Lukovnikov D, Maheshwari G, et al. Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs[J]. 2019.

[15] Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing Text for Joint Embedding of Text and Knowledge Bases. In Empirical Methods in Natural Language Processing.

[16] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In Association for the Advancement of Artificial Intelligence.

[17] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In International Conference on Learning Representations.