Knowledge Graph Completion Based on the Joint Embedding of Entities

Luwei Liu\textsuperscript{1*}, Cui Zhu\textsuperscript{2} and Wenjun Zhu\textsuperscript{3}

\textsuperscript{1,2,3} Faculty of Information Technology, Beijing University of Technology, Beijing, China

*Corresponding author’s e-mail: llw@emails.bjut.edu.cn

Abstract. As the last step in the process of knowledge graph construction, knowledge graph completion predicts and infers new factual triples according to the existing knowledge, which is of great significance to the downstream application of the knowledge graph. The knowledge graph completion technology learns representations of entities and relations in the knowledge graph and reconstructs triples factual relations in the representation space to measure their effectiveness, so as to predict and supplement entities and relations. At present, Graph Convolutional Network (GCN) is widely used in entity modeling, which combines the graph structure information to learn the vector representation of entities. However, most algorithms only focus on static knowledge graph completion, ignoring the important role of time in the knowledge graph, and the modeling of entities is not comprehensive enough. In this paper, a knowledge graph completion based on the joint embedding of entities is proposed, which combines structural feature embedding of the graph and dynamic embedding of entities to carry out multi-level learning of entities to improve the effect of entity prediction. Structural feature embedding of the graph takes advantage of the powerful feature aggregation capability of GCN to integrate the knowledge graph as the topological structure and node features of graph data into the representation learning of entities. Dynamic embedding of entities takes advantage of the characteristics that entities will behave in different states at different times and divides the entity features into two parts for learning. One part of the features changes dynamically over time and the other part retains its static properties. In this paper, the probability model ConvTransE is used to construct the model of relations and triples, ConvTransE uses the convolutional neural network to extract the semantic information of the triples and predict the missing elements. We have conducted experiments on public datasets and achieved better results on Hit@K and MRR.

1. Introduction

Knowledge Graph (KG) organizes the unstructured and unorganized data in the Internet into standardized knowledge resources in an intuitive and clear way, so that knowledge can be further inferred and learned. However, due to the shortcomings of manual construction and relational extraction technology, the existing large-scale knowledge graph is often incomplete. To some extent, the imperfection of the knowledge graph affects the development of the knowledge graph in various fields, so the knowledge graph completion (KGC) is of great significance to the construction and application of the knowledge graph.

The early KGC technology developed from the proposal of TransE[1]. TransE models entities and relations by treating relation vectors as translational operations of head and tail entity vectors. Then models based on tensor decomposition are developed, such as DistMult[2], ComplEx[3] and so on.
Due to the powerful self-learning ability and feature extraction ability of neural network in deep learning, Conv[4], ConvTransE[5], ConvKB[6] and other models use convolutional neural network to extract deep features of entities and relationships, and learn more feature expressions than the previous shallow models. Then GCN is introduced to KGC, which takes into account the knowledge graph as the graph structure, and produces a higher accuracy than the previous models that only focus on the triple structure itself, such as R-GCN[7] and SACN[5]. SACN uses weighted graph convolutional network WGCN to model entities separately and assigns different weight parameters to each relation type in the convolution process. Because of the strong feature aggregation ability of GCN, the structural information of knowledge graph is integrated into entity representation learning, which makes entity modeling more accurate, so the effect of KGC is also improved.

However, the above methods are all aimed at static knowledge graph, ignoring the important component of the time in the knowledge graph. Most of the KGs in reality are time-sensitive, and the factual relationships in the triple are not immutable, but change with time. In order to complete the dynamic knowledge graph, TTransE[8], Know-Evolve[9], HyTE[10], TA_DistMult[11] model the timestamp in the triples from different perspectives. For the knowledge graph with constantly evolving factual relations, entities will exhibit different semantic characteristics in different periods, and thus will be associated with different entities. DE_SimplE[12] takes advantage of this characteristics of the entity to divide the feature dimension of the entity into static feature and temporal feature. A hyper-parameter is defined as the proportion of static feature and temporal feature. The temporal feature changes over time, while the static feature retains the invariable attribute. By using diachronic embedding, this method can learn the different states of entities at different times. Combined with the SimplE[13] as a scoring function, to focus more on changes in entities than previous methods. However, this method only focuses on the triple structure itself, and there is no other level of learning, so the modeling of the entity is not comprehensive enough. The way SimplE constructs negative samples also increases the uncertainty of the model (see details later).

According to the above observation, the main contributions of this paper are summarized as follows:

1) We propose a joint embedding of entities to learn entities from multiple perspectives. Combined with graph convolutional network and dynamic embedding, entities are learned from the point of view of graph and time, so as to enhance the modeling accuracy of entities and thus improve the effectiveness of knowledge graph completion.

2) We point out the disadvantage of the previous scoring function. Using the random substitution method to construct negative samples will increase the uncertainty of the model, and the same score of positive and negative samples will lead to ambiguity in metrics Therefore, we use ConvTransE to match missing entities and output probability vectors for prediction by convolutional neural network.

3) We demonstrate the effectiveness of our work on ICEWS and GDELT datasets, and the results show that we achieve better results than previous methods. Moreover, we also apply our method to the KG in the field of financing to predict the institutions conducive to the financing success of enterprises with financing needs, which proves that our method has a certain application value for improving the financing efficiency of enterprises.

2. Methods
This section details the joint embedding of entities and the advantages of the probability model ConvTransE. For entity prediction, joint embedding focuses on improving the accuracy of entity modeling, combining the advantages of GCN and diachronic embedding to learn entities from multiple perspectives. The probability model ConvTransE, uses the output probability vector to predict the missing entities, and all modules are trained by minimizing the same loss function.

2.1. Joint Embedding of entities
The process of the joint embedding of entities is shown in figure 1. Joint embedding of entities divides the entity representation learning into two modules, the vectors generated by the two modules are
averaged to get the joint embedding representation of the entities. Compared with the previous method of randomly initializing entity vectors, the model is not learned from scratch, but based on known features, which could improve the effect of knowledge graph completion to a certain extent.

When we complete the knowledge graph, we need to learn different entity characteristics for the triples at different times to adapt to the changing factual relations. Therefore, we take the advantage of diachronic embedding to divide the representation of entity into dynamic part and static part. The characteristics of the dynamic part change with the evolution of the event, while the characteristics of the static part retain the fixed attributes of the entity. As shown in figure 1, the input triplet has a specific timestamp, which is expressed in the form of year, month and day. When learning entity features, the time of the matter is integrated into dynamic learning, which is represented as solid circles of different colors in the figure. Static features are represented by yellow solid circles, representing features preserved over time. The proportion of temporal features and static features can be determined according to the experimental results. The above process is shown in equation (1):

\[
ed_{(h, r)} = \begin{cases} 
\alpha^i_h \sigma(w^i_h \tau + b^i_h), & 1 \leq i \leq \gamma k \\
\alpha^k_h, & \gamma k < i \leq k 
\end{cases}
\]

\(e_{(h, r)}\) is the embedding of the head entity \(h\) under the time \(\tau\), \(e^i_{(h, r)}\) is the \(i\)th component of \(e_{(h, r)}\). \(\alpha^i_h, w^i_h, b^i_h \in \mathbb{R}^{\gamma k}\) is the learnable parameter of the algorithm. \(k\) is the embedding dimension of the entity, \(\gamma \in [0, 1]\) is the proportion of temporal features, which is used as the hyperparameter of the model. The first \(\gamma k\) elements of \(e_{(h, r)}\) are used to capture the temporal characteristics of the entity, while the remaining \((1 - \gamma)k\) elements are used to capture the static characteristics of the entity. We can learn the characteristics of the entity and make accurate predictions at any time through the above process.

In order to learn the knowledge graph as the graph structure, we use weighted graph convolutional network WGCN to embed the entity. WGCN utilizes the graph structure and the graph nodes feature to make more accurate embeddings of entities. The schematic diagram of WGCN is shown in figure 2.
Specifically, the edges of different colors in the graph represent different relations, and each relation type will learn a different parameter in convolution. The dotted line represents the process of updating the vector by the connected nodes, as shown in equation (2):

$$h_{i+1}^l = \sigma \left( \sum_{j \in N_i} \alpha_i^l g(h_i^l, h_j^l) + h_i^l W_l \right)$$  \hspace{1cm} (2)

$h_i^l$ is the input vector of the yellow node $V_i$ in the $l$th layer, $h_{i+1}^l$ is the output vector of the node $V_i$ in the $l$th layer. $\alpha_i$ denotes the weight of each relation type. $N_i$ is the neighbours set of the node $V_i$. $g(h_i^l, h_j^l)$ represents the way that the node $V_i$ aggregate the neighbour node $V_j$, which is called the aggregation function. The details are shown in equation (3). $W_l$ is the linear transformation matrix in the $l$th layer. The way of aggregation is the accumulation of neighbour features. Equation (3) reflects the update process of the node $V_i$ vector, which means traversing the adjacent nodes and gathering the vector of the neighbour nodes to itself. At the same time, different relation weight parameters are learned for different edge types in the process of aggregation to represent the influence of neighbour features on the central node. In addition to summing the features of adjacent nodes, the central node also needs to retain its own features during the aggregation.

$$g(h_i^l, h_j^l) = h_j^l W_l$$  \hspace{1cm} (3)

Figure 2. Schematic diagram of WGCN.

2.2. Probability Model

The previous module has carried on the representation learning to the entity. Next, we need to model the relation and triple structure. In related work, scoring functions are usually designed for measuring the effectiveness of triples, such as TransE[1], DistMult[2], ConvKB[6]. These models construct the negative triples by randomly replacing the entities in the positive triples. The goal of training is to make the positive triples score as high as possible and the negative triples score as low as possible. However, random replacement of correct entities will increase the uncertainty of the model. For example, we need to construct the negative sample of the correct triples (China, Capital, Beijing) by replacing ‘Beijing’ with an entity randomly from all entities. The randomness may cause ‘Beijing’ to be replaced by ‘Lily’, ‘doctor’, ‘sports’ and other entities with little relevance, the triples after the replacement are obviously invalid, which will cause the model to become extremely simple. For the negative triple (China, Capital, Shanghai), it is difficult for the model to make correct judgments. We list the judgments of the TransE on different triples, as shown in table 1. We found that some of the negative triples scored higher than the correct triples. Therefore, random substitution of head and tail entities cannot learn negative samples well. Moreover, [14] proposed that after the activation function of ReLU, many neurons in the model such as ConvKB became zero, resulting in the same score of the negative triplet and the positive triplet, and the insertion position of the correct triplet would affect the level of the evaluation index.
In order to avoid the above problems, we use the deep network structure ConvTransE in the SACN to model the relation and triplet structure. ConvTransE convolves the input head entity vector and relation vector, and outputs a vector with the same dimension as the entity, which is used to match with all entities, and finally obtains the probability vector with the number of entities. Each entry of the probability vector measures the correctness of the triplet when the entity at the location is used as the prediction result, avoiding the construction of negative samples, and therefore the ConvTransE is called a probability model.

Table 1. The scores of different triples in TransE.

| validity | triples | score  |
|----------|---------|--------|
| positive | (Canada, Provide_humanitarian aid, Vietnam) | -11.36 |
|          | (Canada, Provide_humanitarian aid, Iraq)   | -8.59  |
|          | (Canada, Provide_humanitarian aid, Children(Canada)) | -10.88 |
| negative | (Canada, Provide_humanitarian aid, China)   | -10.53 |
|          | (Canada, Provide_humanitarian aid, Japan)   | -10.41 |
|          | (Canada, Provide_humanitarian aid, France)  | -10.08 |

Figure 3 shows the overall structure of ConvTransE. For a triple \((h, r, t)\), we obtain the vector representation \(e_h \in \mathbb{R}^k\) through the joint embedding of entities, and the relation vector \(e_r \in \mathbb{R}^k\) is obtained through random initialization. Let the input matrix \(X = [e_h, e_r] \in \mathbb{R}^{k \times 2}\), the kernel \(\omega\) will repeatedly convolve on the input matrix to obtain a series of feature maps, then we concatenate all the feature maps to obtain an independent vector \(V \in \mathbb{R}^{|\omega| \times s + 1}\), \(s\) is the size of the kernels. Then the vector \(V\) is fully connected to map the dimension to \(k\), and then multiplied by the entity matrix, the sigmoid function is used to obtain the probability vector \(P_h\) of size \(|E|\). Each item of \(P_h\) represents the probability that the entity at the corresponding position being the correct tail entity. Finally, we summarize the probability vector \(P_h\) obtained as equation (4):

\[
P_h = \sigma \left( f \left( \text{concat} \left( g \left( [e_h, e_r] \ast M \right) \right) \right) \right)
\]

Finally, all modules are trained by minimizing the binary cross-entropy loss function. As shown in equation (5). The algorithm needs to use the predicted probability vector to fit the real label vector, \(\hat{T}\) is the real label vector after smoothing, and \(p\) is the probability vector output by ConvTransE.

\[
\mathcal{L} = -\frac{1}{|E|} \sum_i \left( \hat{T}_i \cdot \log(p_i) + (1 - \hat{T}_i) \cdot \log(1 - p_i) \right)
\]
In this paper, we call our work JEE (Joint Embedding of Entities). In the following part, the ablation experiment will be designed, after dynamic embedding of the entity, it will be directly fed into ConvTransE, which is called DPM (Dynamic Probability Model) in this paper.

3. Results

For datasets, we utilize the standard datasets ICEWS14, ICEWS05-15 and GDELT followed by [13], Hit@K and MRR are used as metrics. The statistics of the datasets are shown in table 2.

| Dataset     | #Entity | #Relation | #Train | #Validation | #Test |
|-------------|---------|-----------|--------|-------------|-------|
| ICEWS14     | 7 128   | 365       | 72 826 | 8 941       | 8 963 |
| ICEWS05-15  | 10 488  | 251       | 386 962| 46 275      | 479 329|
| GDELT       | 500     | 20        | 2 735 685| 341 961    | 341 961|

Our work is implemented on Nvidia TITAN RXT. Adam is used as an optimizer to update parameters. We also utilize the Dropout and the batch normalization to improve training efficiency. When setting the experimental parameters, we set the embedding size to 200, the kernel size to 5, the number of kernels to 200, the dropout rate to 0.2, the learning rate to 1e-4 for ICEWS05-15 and GDELT, 3e-4 for ICEWS14. The epoch is set to 500 and the model was saved every 20 times.

Our experimental results on ICEWS14, ICEWS05-15, and GDELT are shown in table 3 and 4, which list the comparison of JEE and the six baselines on Hit@1, Hit@3, Hit@10, and MRR. To ensure fairness, all methods are implemented in the same environment. It can be seen from the table that JEE achieved the best results on Hit@1, Hit@3 and MRR for the three datasets. In details, compared to DE_SimpleE, JEE improved from 41.7% to 59.7% on Hit@1 regarding ICEWS14, which means that nearly 60% of the test triples could find the missing entity in Top1. For ICEWS05-15, JEE improved from 57.6% to 74.4% on Hit@3, meaning that almost 75% of the test triples could get the missing entity in Top3. In terms of MRR, JEE improved by 18.9% and 43.1% respectively on ICEWS14 and ICEWS05-15. JEE does not achieve the best performance on ICEWS14 regarding Hit@10. We consider that the number of entities and relations are limited on a small-scale knowledge graph, and the performance of the algorithm on simple datasets needs to be further strengthened. All the above experimental results show that the joint embedding of entities by combining GCN and dynamic embedding can effectively improve the knowledge completion effect. For GDELT, the improvement of JEE is more obvious. GDELT has fewer entities and relation types but has a million-level relationship, which indicates that the method proposed in this paper also has a good performance on the large-scale knowledge graph. It shows that JEE can extract rich feature information from dense knowledge relationships.

| Model       | ICEWS14 | ICEWS05-15 |
|-------------|---------|------------|
|             | Hit@K(%) | Hit@K(%)  |
|             | Hit@1  | Hit@3  | Hit@10 | MRR         | Hit@1  | Hit@3  | Hit@10 | MRR         |
| TransE      | 9.8%   | 40.5%  | 64.1%  | 0.29 | 10.1%   | 42.9%  | 65.8%  | 0.31 |
| DistMult    | 33.4%  | 51.0%  | 68.2%  | 0.45 | 36.7%   | 54.7%  | 71.9%  | 0.49 |
| HyTE        | 10.8%  | 41.6%  | 65.5%  | 0.30 | 11.6%   | 44.5%  | 68.1%  | 0.32 |
| ConvTransE  | 30.7%  | 38.8%  | 47.5%  | 0.45 | 62.7%   | 70.9%  | 78.7%  | 0.68 |
| SACN        | 46.7%  | 56.4%  | 66.2%  | 0.53 | 61.7%   | 68.8%  | 75.4%  | 0.67 |
| DE_SimpleE  | 41.7%  | 59.1%  | 73%    | 0.53 | 38.9%   | 57.6%  | 74.5%  | 0.51 |
| JEE         | 59.7%  | 64.5%  | 69.9%  | 0.63 | 69.1%   | 74.4%  | 79.8%  | 0.73 |

Table 3. Result of evaluation index on the ICEWS.
Table 4. Result of evaluation index on the GDELT.

| Model      | Hit@1(%) | Hit@3(%) | Hit@10(%) | MRR  |
|------------|----------|----------|-----------|------|
| TransE     | 0%       | 15.8%    | 31.2%     | 0.11 |
| DistMult   | 12.4%    | 22%      | 33.6%     | 0.21 |
| HyTE       | 0%       | 16.5%    | 32.6%     | 0.12 |
| ConvTransE | 41.4%    | 54.5%    | 69.2%     | 0.48 |
| SACN       | 40.3%    | 55.3%    | 69.1%     | 0.50 |
| DE_SimplE  | 14.1%    | 24.8%    | 40.3%     | 0.23 |
| JEE        | 40.2%    | **56.2%**| **70.3%** | **0.51** |

Next, we will conduct experiments on DPM and JEE to compare the influence of structural feature embedding of the graph on the experimental results. DPM directly uses ConvTransE to predict based on dynamic embedding, while JEE also combines GCN to learn the graph structure of entities on the basis of dynamic embedding. Figure 4 shows the MRR of the two models on three datasets. We found that JEE improves by 12.5%, 1.4% and 4.1% compared with DPM on three datasets after using GCN. By iteratively aggregating neighborhood information, GCN can obtain the topology structure and node features of graph, enhance the representation learning of entities, and improve the prediction ability of the knowledge graph.

In the dynamic embedding of entities, in order to further verify the necessity of temporal and static features, we design a set of parameter comparison experiments. First, the temporal feature proportion $\gamma$ is set to 0 and 1 respectively. When $\gamma$ is 0, it means that only the static part of the entity embedding is left. When $\gamma$ is 1, it means that the entity embedding are all temporal features. Secondly, we also set $\gamma$ to 0.2, 0.4, 0.6 and 0.8 respectively to verify the best $\gamma$ of different datasets. We use the above different parameters on DPM, and conduct the experiments on ICEWS14 and ICEWS05-15 regarding MRR. The result is shown in figure 5 and figure 6. From the line chart, we can clearly see that the accuracy of MRR increases with the increase of the proportion of temporal features. However, when the parameter $\gamma$ increases to 1, the accuracy decreases, which indicates the necessity of the static features. The static features can effectively reduce the over-fitting of temporal features. For ICEWS14, there are relatively few factual relations, and when $\gamma$ is 0.4, the algorithm can learn the optimal state of the entity. For ICEWS05-15, there are relatively more factual relations. When $\gamma$ value is 0.6, the algorithm can learn the best entity state. This also indicates that when there are more relationships, the triples are more dependent on time, and the temporal features $\gamma$ needs to be greater.
We also apply the proposed method to the knowledge graph in the field of financing to predict the institutions that may have financing relationship with the enterprises to be financed in the future. Experiments show that nearly 72% of enterprises can find the target financial institutions in the top 10 predictions given by the algorithm.

4. Conclusions
In this paper, a knowledge graph completion based on the joint embedding of entities is proposed. The method combines the graph feature structure with the dynamic embedding to model the entity with multi perspectives, which improves the accuracy of entity modeling and lays the foundation for the entity prediction. On the basis of entity representation learning, we use the probability model ConvTransE to model the relation and output the prediction entity, which avoids the uncertainty caused by the construction of negative samples. By designing several groups of comparative experiments, we have proved that our work can achieve better performance in knowledge graph completion and has certain application value.

References
[1] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O. (2013) Translating embeddings for modeling multi-relational data. Neural Information Processing Systems. New York. 2787–2795.
[2] Yang, B., Yih, W., He, X., Gao, J., Deng, L. (2014) Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575.

[3] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E. (2016) Complex embeddings for simple link prediction. International Conference on Machine Learning. New York. pp. 2071-2080.

[4] Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S. (2018) Convolutional 2d knowledge graph embeddings. Proceedings of the AAAI Conference on Artificial Intelligence. LA.

[5] Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B. (2019) End-to-end structure-aware convolutional networks for knowledge base completion. Proceedings of the AAAI Conference on Artificial Intelligence. Hawaii. pp. 3060-3067.

[6] Dai, Quoc, Nguyen, T.D.N., Nguyen, D.Q., Phung, D. (2018) A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. Proceedings of NAACL-HLT. New Orleans. pp. 327-333.

[7] Schlichtkrull, M., Kipf, T.N., Bloem, P., Troncy, R., Holllink, L., Tordai, A. (2017) Modeling relational data with graph convolutional networks. European semantic web conference. Portorož. pp. 593-607.

[8] Jiang, T., Liu, T., Ge, T., Sha, L., Chang, B., Li, S., Sui Z. (2016) Towards time-aware knowledge graph completion. Proceedings of COLING. the 26th International Conference on Computational Linguistics: Technical Papers. Lisbon. pp. 1715-1724.

[9] Trivedi, R., Dai, H., Wang, Y. (2017) Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. International Conference on Machine Learning. FL. pp. 3462-3471.

[10] Dasgupta, S.S., Ray, S.N, Talukdar, P. (2018) Hyte: Hyperplane-based temporally aware knowledge graph embedding. Proceedings of the 2018 conference on empirical methods in natural language processing. Brussels. pp. 2001-2011.

[11] Garcia-Durán, A., Dumancic, S., Niepert, M. (2018) Learning Sequence Encoders for Temporal Knowledge Graph Completion. EMNLP. Brussels.

[12] Goel, R., Kazemi, S.M., Brubaker, M., Poupart, P. (2020) Diachronic embedding for temporal knowledge graph completion. Proceedings of the AAAI Conference on Artificial Intelligence. New York. pp. 3988-3995.

[13] Kazemi, S.M., Poole D. (2018) SimplE embedding for link prediction in knowledge graphs. In NeurIPS. Montreal. pp. 4289–4300.

[14] Sun, Z., Vashishth, S., Sanyal S., Talukdar, P., Y Yang. (2020) A Re-evaluation of Knowledge Graph Completion Methods. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 5516-5522.