Knowledge Graph Embedding for Link Prediction and Triplet Classification

E. Shijia[1], Shengbin Jia[1], Yang Xiang[1](✉), and Zilian Ji[2]

[1] Tongji University, Shanghai 201804, People's Republic of China
e.shijia@gmail.com, {shengbinjia,shxiangyang}@tongji.edu.cn
[2] IBM China Systems and Technology Laboratory,
Shanghai 201203, People's Republic of China
jizilian@cn.ibm.com

Abstract. The link prediction (LP) and triplet classification (TC) are important tasks in the field of knowledge graph mining. However, the traditional link prediction methods of social networks cannot directly apply to knowledge graph data which contains multiple relations. In this paper, we apply the knowledge graph embedding method to solve the specific tasks with Chinese knowledge base *Zhishi.me*. The proposed method has been successfully used in the evaluation task of CCKS2016. Hopefully, it can achieve excellent performance.

Keywords: Knowledge graph · Distributed representation · Entity embedding

1 Introduction

In traditional social networks, the link prediction task is one of the important technologies to discover the relationships among users [1]. Within the link prediction of social network, the *connection* between two users is often said to be a friend relationship. However, in the knowledge graph, the knowledge network is composed of entities and relations. A connection with two entities can be denoted as a triplet (h, r, t), where h is the head entity, t is the tail entity, and the relation between them is represented as r. Different from the social networks, the *connection* in the knowledge graph is usually with a direction, *e.g.* for the triplet *(Yao Ming, born in, Shanghai)*, the relation *born in* is a way from *Yao Ming* to *Shanghai*, but we could not say *Shanghai* was born in *Yao Ming*. Therefore, the traditional link prediction methods used in social networks are not suitable for the link prediction task in the knowledge graph. In addition, because of the flexibility of Chinese language, the rule based natural language precessing (NLP) methods often require a lot of manual intervention.

In this paper, we adopt the representation learning to understand the knowledge graph provided by *zhishi.me*, and embed the entities and relations of the knowledge graph into a low dimensional vector space. The vector representation of the entities and the relations will contain the semantic relationships among them.

H. Chen et al. (Eds.): CCKS 2016, CCIS 650, pp. 228–232, 2016.
DOI: 10.1007/978-981-10-3168-7_23

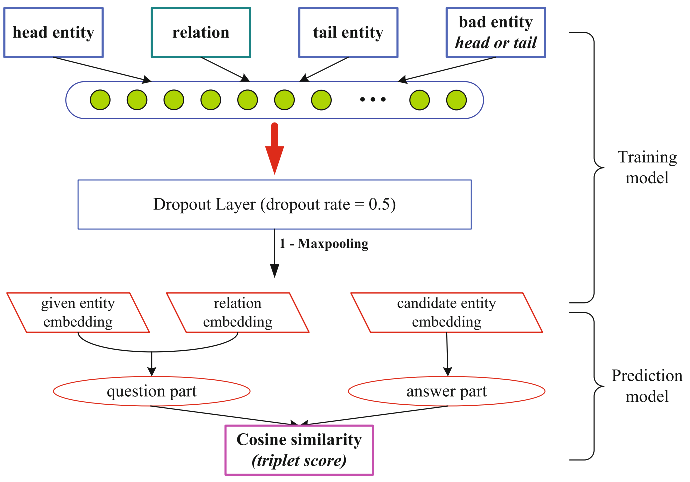

Fig. 1. The overall neural network architecture of our model

The rest of this paper is structured as follows. In Sect. 2, we describe our model architecture used in the evaluation task. In Sect. 3, we summarize the experiment setup of our model. The application of our model is presented in Sect. 4. Section 5 contains related work and finally we give some concluding remarks in Sect. 6.

2 The Embedding Model for Knowledge Graphs

In this section we describe the proposed deep neural networks to solve the LP and TC problems. Figure 1 shows the overall framework of our model. The training part aims to learn the semantic relationships among entities and relations with the negative entities (bad entities), and the goal of the prediction part is giving a *triplet score* with the vector representations of entities and relations. The following is a detailed description.

2.1 Data Preprocessing

The dataset of the evaluation task is from the Chinese knowledge base *zhishi.me*, and the basic statistics of the data are shown in Table 1. In order to meet the requirement of the evaluation task, we first number the entities and relations in turn. During the training time, different IDs represent different entities and relations. This kind of representation can be convenient for us to do the vectorize operations.

2.2 Core Architecture of Knowledge Graph Embedding

For a given triplet (h, r, t) in the training set, our model will learn the vector representations of h and t as well as the r, denoted as $\mathbf{h}$, $\mathbf{t}$ and $\mathbf{r}$. The core idea

Table 1. Data set used in the evaluation task

Dataset	#Entities	#Relations	#Triplets (Train)
zhishi.me	644699	3512	7063189

of the model is that transforming the link prediction problem into a question and answer mode, *i.e.* **h** + **r** expresses the question, and **t** is the answer, or **t** - **r** is question, and **h** expresses the answer.

Based on the above ideas, in order to learn the proper vector representations, our neural networks are trained to minimize the following loss function with the training data (illustrated by the example of tail entity prediction):

$$L = max\{0, m - cos(\mathbf{h} + \mathbf{r}, t^{+}) + cos(\mathbf{h} + \mathbf{r}, t^{-})\} \tag{1}$$

where $m > 0$ is the margin hyper-parameter, t^{+} and t^{-} denote the correct tail entity and wrong tail entity respectively. Unlike the TransE [2] or TransM [3] model that use the L_1 or L_2 norm as the dissimilarity measure, we use the cosine similarity (cos) to judge the matching degree of *question* and *answer* which can be called as **matching score**. After training with the loss function, it turns out that the loss value of the correct triplet is less than its corresponding wrong triplets. m is used to control the degree of deviation.

During the training process, at every epoch, we randomly sample a wrong entity which is from the whole entity set to each correct triplet in the training set. As a result, the four tuple (h, r, t^{+}, t^{-}) (or (h^{-}, h^{+}, r, t)) forms a training sample. As Fig. 1 shown, we add a *Dropout* layer after the *Embedding* layer to improve the generalization ability of the model and prevent overfitting [4]. Besides that, we add a 1-MaxPooling layer. The vector representation after the pooling layer is treated as the final embedding of the entity or relation which will be used in the loss function.

3 Experiment Setup

In this section, we describe the parameters and experiment environment used in this evaluation task. The parameters need to be fine tuning with different tasks.

3.1 Parameter Settings

In this evaluation task, the margin value m was 0.05, and the embedding dimension of entities and relations was 100. We also tried 200 or 1000 dimensions, and it can get better result on a small dataset (split from training set). However, on the whole dataset, it was more costly. The optimization method employed was Adam [5], and it was more computationally efficient than basic stochastic objective function (SGD). The learning rate was 0.001, and the batch size was 512 per epoch. We trained 200 epochs for the predictions of head entity and tail entity respectively.

3.2 Training Environment

The model used in this evaluation task was implemented with Keras[1]. We used a Tesla K20c GPU device to train the model. Due to time constraints, we believe our model can get better results after longer training time.

4 Applications of the Model

In the triple link prediction tasks, our model would treat all available entities as the candidates for each test sample in the test set $((h, r, \_)$ or $(\_, r, t))$. The trained model would give the matching scores to each *question* and *answer* pairs, and entities ranked at the highest top 200 could be saved as the submitted results.

As to the triplet classification task, we adopted the tail entity prediction model as the test model. For the triplet given by the test set, the model would give the matching score of the test samples. Our strategy was that if the triplet's score was greater than or equal to 0.55, it was considered to be valid, otherwise we tagged it as an invalid one.

5 Related Work

The model used in this evaluation task is related to the following two research areas.

Distributed Representation Learning. It plays an important role with the development of deep learning. The related methods can been applied to various fields, such as NLP, computer vision and image processing [6]. Especially, models based on word embedding have been achieved good performance in the field of text classification [7]. It makes it possible to train on large scale data with limited resources. Inspired by the word embedding model, such as *word2vec*[2], a lot of similar models have emerged recently. *Paragraph Vector* and *Doc2vec* [8] are extensions of *word2vec*, and they learn the vector representations of paragraphs and documents. Essentially, the core of the ideas is to make a good text representation which can express proper semantic information in a specific environment. The embedding models on knowledge graph data also try to catch the key semantic relationships hidden in the numbers of entities, and we can absorb the advantages of those models to help us learn the structure of knowledge graphs.

Knowledge Graph Completion. It aims to predict relations between entities of an existing knowledge graph. It has been several *translation* based methods, such as TransE, TransM, TransR [9] and Hole model [10]. The knowledge graph embedding models with the representative of the TransE have made remarkable achievements in the knowledge graph completion task with the specific datasets.

[1] https://keras.io.
[2] https://code.google.com/archive/p/word2vec/.

In essence, all of them try to find out a comprehensive and effective rule which *translates* head entities to tail entities. For the evaluation task in the paper, the scale of the data is far beyond the dataset used in existing experiments. Therefore, we should develop a more effective method to tackle this problem.

6 Conclusion

We describe a deep neural network method with distributed representation to solve the triplet prediction and triplet classification evaluation tasks. Our model can be trained fast with advanced GPU devices and easily extended to other similar tasks.

In addition, the entity candidates in the task is really large. If we can figure out a way to reduce the size of search space, maybe the test result will be better.

Acknowledgement. This work has been partially funded by the National Basic Reseach Program of China (2014CB340404) and the IBM SUR (2015) grant.

References

1. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inform. Sci. Technol. **58**(7), 1019–1031 (2007)
2. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, pp. 2787–2795 (2013)
3. Fan, M., Zhou, Q., Chang, E., Zheng, T.F.: Transition-based knowledge graph embedding with relational mapping properties. In: Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation, pp. 328–337 (2014)
4. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
5. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
6. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cogn. Model. **5**(3), 1 (1988)
7. Kim, Y.: Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882 (2014)
8. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. ICML **14**, 1188–1196 (2014)
9. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI, pp. 2181–2187 (2015)
10. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. arXiv preprint arXiv:1510.04935 (2015)