DTransX: A Distributed Framework for Knowledge Graph Representation Learning

Jun Ma, Guozheng Rao, Xiaowang Zhang*, and Zhiyong Feng

College of Intelligence and Computing, Tianjin University, Tianjin, China
Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
* Corresponding Author: xiaowangzhang@tju.edu.cn

Abstract. In this paper, we present a distributed framework named DTransX for knowledge graph representation learning in a hybrid parallel way. Firstly, we introduce a hybrid parallel computing model, where embedding model and data are distributed, in order to efficiently process large-scale knowledge graph. Moreover, we propose a distributed model merging method based on word frequency weight of the entity or relation to avoid semantic loss during parallel processing. Finally, we develop a decentralized architecture for parallel embedding to enhance the stability of our embedding system. The experiments show that our proposal can avoid the loss of semantics and even overcome overfitting problem of single computing as well as it exhibits significant in speeding up training.

1 Introduction

In recent years, the translation model has achieved great advances in the work of knowledge graph completion based on knowledge graph representation learning. When we build representation model to explore richer semantic associations on large-scale knowledge graph, we face tow challenges: (1) Big data: the computing power of single machine is not enough to train large-scale data in a reasonable time. (2) Large model: the size of the knowledge representation model is limited to the stand-alone memory level. In existing translation models, such as TransE [1], TransH [2], and TransR [3], all based on the assumption of stand-alone machine. At present, the parameter servers [4] [5] is the mainstream distributed machine learning framework. This architecture rely heavily on the central parameter server to maintain and update all global parameters [6].

In this poster, we propose a distributed framework for knowledge graph embedding. The large model and big data are trained by hybrid parallel computing model. And we use distributed model merging method to build a more complete and accurate embedding model. From an architectural point of view, DTransX is based on a flexible decentralized architecture. The experiments on data sets FB15K [1] and WN18 [1] showed that our system performed well on link prediction. And it observably speed up training on a larger data set, wikidata.

2 Framework of DTransX

The approach proposed in this section has been implemented in the DTransX. The framework of DTransX contains three modules, namely, *Hybrid-Parallel Configurator*, *Decentralized Trainer* and *Distributed Merging Executor* shown in the following figure.

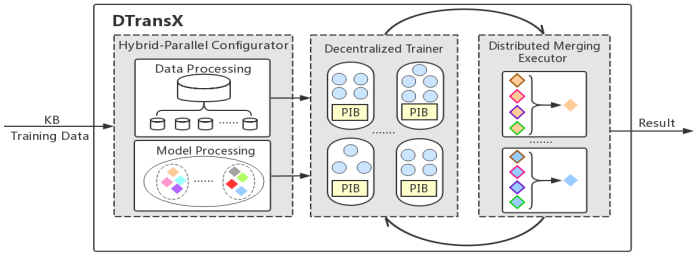

Fig. 1. The framework of DTransX

Hybrid-Parallel Configurator Hybrid-Parallel Configurator distributes data and model to computing groups and their subordinate nodes. The role of Data Processing is to shuffle the RDF triples according to the computing power of each computing group. We use Model Processing to logically parting model in each computing group, and each node construct local submodel according to the logical submodel. There are T_j computing nodes in the computing group G_j, and the model M contains $COUNT(M)$ parameters, the position of the parameter ${p_i}^{G_j}$ is expressed as $(RANK({p_i}^{G_j}), OFFSET({p_i}^{G_j}))$. The node tag $RANK({p_i}^{G_j})$ and the offset $OFFSET({p_i}^{G_j})$ are expressed as follows:

$$RANK({p_i}^{G_j}) = min(\frac{i \cdot COUNT(M)}{T_j}, T_j - 1),$$

$$OFFSET({p_i}^{G_i}) = i - \frac{COUNT(M) \cdot RANK({p_i}^{G_i})}{T_j},$$

where $i \in \{0, 1, \cdots, COUNT(M) - 1\}$.

Decentralized Trainer The architecture of our framework is decentralized. In each computing group, the nodes synchronization training, and the parameters are directly updated to the submodel, which is maintained by the corresponding node without central node scheduling. In a system with K computing groups, the information is expressed as $\{(G_j, T_j)|j \in \{0, 1, \cdots, K\}\}$ in the public information block PIB of each group.

Distributed Merging Executor After each round of training, we unify the replicas of the model, which are trained in each group, by the Distributed Merging Executor. In the first stage, each node computes address information $(RANK({p_i}^{G_j}), OFFSET({p_i}^{G_j}))$ by the information in the PIB. Then, each

node independently pull all p_i from other nodes, which is in different groups. Finally, nodes merge models by combining the word frequency weights. The merging function is defined as:

$$p_i^{new} = \sum_{j=0}^{K} {w_i}^{G_j} {p_i}^{G_j},$$

where ${w_i}^{G_j}$ is the word frequency weight of the entity or relation, which corresponding to ${p_i}^{G_j}$ in the data set on G_j group.

3 Experiments and Evaluations

Experiments are implemented in a cluster running linux system, and we build system by C++ and MPI.

We evaluated the performance of the link prediction on FB15K and WN18, and used Meak Rank and Hit@10 as evaluation methods under the settings of "Raw" and "Filter". All experiments were performed under the same hyperparameters, with the dimension d=100, the margin m=1, and the learning rate η= 0.001. The baseline includes TransE, TransH, and TransR. These classical translation models can be trained in the DTransX framework, denoted by DTransE, DTransH, and DTransR in the Table 1.

Table 1. Link prediction result.

DataSets	*FB*15*K*				*WN*18			
metirc	*MeanRank*		*Hit*@10(%)		*MeanRank*		*Hit*@10(%)	
	Raw	*Filter*	*Raw*	*Filter*	*Raw*	*Filter*	*Raw*	*Filter*
TransE	237.884	139.180	51.5	71.2	450.549	437.035	80.5	85.3
DTransE	**226.787**	**120.059**	50.6	71.0	**361.165**	**349.120**	**81.7**	**87.4**
TransH	235.632	126.594	53.3	75.8	458.215	445.012	80.8	89.5
DTransH	**225.567**	**121.970**	50.9	**75.9**	**366.932**	**365.283**	80.2	88.9
TransR	198.164	95.651	62.2	82.1	432.381	375.209	85.9	90.1
DTransR	**180.498**	**90.237**	62.2	81.0	**402.746**	**350.285**	85.1	**94.2**

In the process of training, the unbalanced word frequency of the entity or relation leads to different training degrees of the corresponding model vector, so we use the word frequency as the weight to coordinate the training effect of the model vector. In the experiment, link prediction performance in DTransX is as good as the corresponding translating model on the standard data set, and some even outperform it. The explanation is that, in our framework, each computing group uses different data sets to train different model, so multiple models merging can reduce the risk of single model falling into local minima, thus improving the generalization of the whole model.

We test training efficiency on the wikidata data set (about 68902801 lines) with different numbers of groups and nodes. The results are shown in Table2.

And we can observe that training time continues to decrease as the increasing number of groups or nodes.

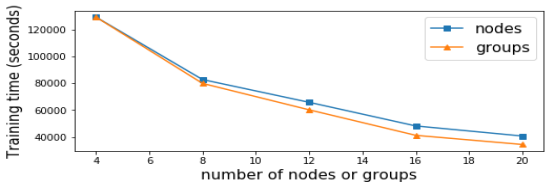

Fig. 2. Training time along with number of nodes and groups

4 Conclusions

In this poster, we present a hyprid parallel representation learning in knowledge graph to improve the performance of training without loss of semantics during distributed training. As an advantage, our approach is independent of data and embedding models through our experiments compare the three classical embedding models (TransE, TransH, and TransR). Due to its strong expansibility, we believe it is helpful in representation learning in large-scale data. In the future work, we further expand DTransX to support more other types of knowledge representation learning models, such as ConvE, R-GCN, etc.

5 Acknowledgments

This work is supported by the National Key Research and Development Program of China (2017YFC0908401) and the National Natural Science Foundation of China (61672377,61972455). Xiaowang Zhang is supported by the Peiyang Young Scholars in Tianjin University (2019XRX-0032).

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: *Proc. of NIPS*, pp. 2787–2795 (2013).
2. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: *Proc. of AAAI*, pp. 1112–1119 (2014).
3. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: *Proc. of AAAI*, pp. 2181–2187 (2015).
4. Li, M., Andersen, D.G., Park, J.W., Smola, A.J., et al.: Scaling distributed machine learning with the parameter server. In: *Proc. of OSDI*, pp. 583–598 (2014).
5. Huang, Y., Jin, T., Wu, Y., Cai, Z., et al.: Flexps: Flexible parallelism control in parameter server architecture. In: *Proc. of PVLDB*, pp. 566–579 (2018).
6. Lian, X., Zhang, C., Zhang, H., Hsieh, C., Zhang, W., Liu, J.: Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent. In: *Proc. of NIPS*, pp. 5330–5340 (2017).