Continual Learning Knowledge Graph Embeddings

Final Report

Yang Yiliu

1155157082

Abstract

Knowledge Graph (KG) Embedding is a method to encode the entities and relations of KGs into a vector or vector-like space to support question answering, KG completion, and recommender systems. There have been many papers focusing on static KG embedding learning. However, in the real world, the KGs always change over time. That means most existing models must retrain on whole edges if there is any update on the KG, which cost a lot of time. In order to reduce time consumption and improve the model accuracy, this report presents two training improvement methods, which significantly reduce the catastrophic forgetting during incremental updating and improve the model accuracy.

1. Introduction

A knowledge graph (KG) is a graph structure that contains knowledge in the form of head-relation-tail triples, where head and tail are entities and the relation is an edge between them. [1] Knowledge graphs are being widely used in various domains such as recommender systems [2], question-answering bots [3], and drug discovery [4].

Reasoning over such KGs is a common task of AI systems. For example, if we ask Turing Award --- Win --- ?, which means who won the Turing Award, the goal for a reasoning system based on the KGs is expected to predict Geoff Hinton, Don Knuth, etc. Nevertheless, there could be multiple unobserved edges in the original KGs, which means there could be no link connection between Geoff Hinton and Turing Award. At the same time, the reasoning system should be able to reason out there is a link connection. The common method to finish such a task is to embed every entity and relation into a vector or vector-like space and convert the link predictions into algebraic or neural operations.

However, most existing models are designed for static knowledge graphs. In the real world, KGs are often dynamic and updated over time by adding and removing edges. To fix this issue, most real-world applications of KGs have to retrain the whole model to fit new edges at a high cost. Embedding dynamic Knowledge Graphs with low time cost is an essential and practical problem to be solved.

There are only a few models that support the continual learning of KGs, such as puTransE [5], DKGE [6], and RotatH [7]. However, these models change not only the training method but also the embedding method. Changing the embedding method means these models cannot utilize the existing static KG models. In order to reduce the time consumption of retraining and improve the link prediction performance, this report presents two training improvement methods, which significantly reduce the catastrophic forgetting during incremental updating and improve the model accuracy.

2. Description

In this section, we define the problem of continual learning KGs embeddings as two sub-problems, i.e., learning from scratch and online learning. Let a KG $G^T = \{(h,r,t)\} \subseteq E \times R \times E$, where $\{(h,r,t)\}$ represents a set of triples, h means a head entity, r is a relation, t is a tail entity, E and R are the sets of all entities and relations, respectively. And T is the current time step. Then we define learning from scratch as follows:

Learning from Scratch: Given a KG G^T, learning from scratch uses a KG embedding method to learn the embeddings of all entities and relations from all edges in G^T.

At time step T+1, G^T becomes G^{T+1} with an update ΔG^{T+1} involving addition and deletion of edges. Here we define online learning as follows:

Online Learning: Given The KGs G^T and G^{T+1} with ΔG^{T+1} as well as the embeddings of entities and relations at time step T. An online learning method should learn new embeddings at time step T+1 efficiently without retraining the whole edges in G^{T+1}.

The basic idea of continual learning on KGs is to adapt learning from scratch on the first dataset. Then for each updating edge dataset, the model adapts online learning on these datasets with the time order. There have been many papers focusing on from-scratch

learning, such as TransE [8], DistMult [9], ComplEx [10], and RotatE [11], which are also known as embeddings methods. However, there is no research that only focuses on online learning of the KGs. The existing online learning models of the KGs did not use the original embedding methods but modified the embedding methods to cooperate with their online learning method. For example, puTransE [5] modified TransE to cooperate with its online learning method. DKGE [6] and RotatH [7] modified RotatE as their embedding method. Nevertheless, our model does not specify any embedding methods, so our model can easily use any embedding methods while only changing lines of code. And when there is a new embedding method, this method can be easily modified into the online version.

3. Embedding Methods

Since there have been many methods to embed KGs and this report does not focus on the embedding methods, we just briefly introduce what KG embedding methods are and take TransE as an example.

If we have an edge (h, r, t), where h and t are entities, r is a relation, and the embed form of h, r, t are **h**, **r**, **t** respectively, the goal of an embedding method is to fit $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$. In other words, the purpose of the embedding methods is to minimize the distance between $\mathbf{h}+\mathbf{r}$ and **t**, i.e., Dis($\mathbf{h}+\mathbf{r}$, **t**), which is also called the distance of this edge. For example, in TransE, the embedding of entities and relations are vectors with a particular dimension. Then the addition operator in this embedding space comes to vector addition. The distance between $\mathbf{h}+\mathbf{r}$ and **t** is defined as $||h + r - t||_2$.

However, this definition will cause an issue. If we only try to minimize the distance of edges, fixing all entities and relations into the original points will be one of the best situations, which is not what we want. To deal with this issue, we introduce negative sampling. The idea of negative sampling is to randomly sample edges that did not occur in the edge set, and maximize the distances of these negative edges. Finally, the overall goal of embedding methods is to minimize the distances of positive edges and maximize the distances of negative edges.

4. Online Learning

4.1 Elastic Weight Consolidation

Recall that online learning is an efficient method of updating a model using the updated data without retraining from the beginning. That means we can utilize the trained model on the time step T, and use some methods to update this model in order to fit in the data on the time step T+1. However, Simply training the model with the updated data will cause catastrophic forgetting. Here we introduce Elastic Weight Consolidation (EWC) [12] to overcome this phenomenon.

EWC is an improvement on traditional machine learning. Assume that there are two tasks, A and B, and a model M which has been trained with A. The basic idea of EWC is to train M with B without forgetting A. The implementation method of EWC is to introduce an additional loss item which is defined as $L_{model} = \lambda \sum_i F_i(\theta_i - \theta_{i,A})^2$, where F_i is the ith element of diagonal in Fisher Information matrix, which represents the important degree of a certain parameter and λ is a hyper-parameter. F_i can be approximately calculated as $(\frac{\partial \mathcal{L}(\theta_i)}{\partial \theta_i})^2$

For a KG embedding method, all of the parameters could be grouped into 3 classes, i.e., entity embeddings, relation embeddings, and model parameters. According to what we have observed, the relation usually does not shift over time, which means that the relation embeddings trained in the new dataset are usually the same as those in the old dataset. Based on this observation, we designed to freeze the relation embeddings when finishing training the relation for the first time. To implement EWC on KG embedding methods, we let all parameters except relation embeddings, i.e., entity embeddings and model parameters, fit in the EWC model. Hence, the EWC loss of an embedding model is $L_{model} = \sum_{\theta \in \theta_E \cup \theta_M} \lambda F_\theta (\theta - \theta^{T-1})^2$, where θ^{T-1} is the parameter on the time step T-1.

4.2 Edges Replay

Except for EWC, replaying some old edges when training will also reduce the phenomenon of catastrophic forgetting. However, replaying more edges means it will take more time. What we want to do is to improve the model accuracy with fewer edges replaying. Here we propose a strategy that makes use of the old model and new edges to calculate the influence degrees of entities, and sample replay edges based on the influenced degree.

Here we define the influence degree of a new edge as the distance of this edge. Then the influence degree of an entity is the summation of all influence degrees of edges involving this entity, i.e., $I_{entity} = \sum_{edge \in N(entity)} I_{edge}$ and $I_{edge} = Dis(edge)$. Finally, the replaying edges are sampled based on the summation of influence degrees of two endpoints of the edge, i.e., $W_{edge} = I_{head\ entity} + I_{tail\ entity}$.

The idea behind this definition is easy to understand. If there is one edge with a large distance, when it is trained, it will influence much to the entities located at the endpoints of this edge. Therefore we have to replay more existing edges to make sure these entities are stable and not too far away from the embeddings in the previous time step.

Overall, the loss function to be optimized consists of three parts, which are new edges loss, replay edges loss, and EWC loss, i.e., $\mathcal{L}_{all} = \mathcal{L}_{new} + \mathcal{L}_{replay} + \mathcal{L}_{model}$.

5. Experimental Result

In this section, we present experiments to show the effectiveness and accuracy of our model in link prediction tasks compared with the other state-of-the-art online learning models. We conducted experiments on two datasets, YAGO-3SP [13] and IMDB-30SP. These two datasets are real-world datasets with 2 and 29 update edges respectively. With the original dataset, there are 3 and 30 snapshots for each dataset. The statistical data of these two datasets are shown in the below table.

Datasets	#Entities	#Edges	#Relations	#Add	#Del	#Train	#Valid	#Test
YAGO-3SP	27,009	130,757	37	950	150	124,757	3,000	3,000
IMDB-30SP	243,148	627,096	14	9,379	2,395	621,096	3,000	3,000

#Entities, #Edges, #Relations are the number of entities, edges, and relations in the whole dataset respectively. #Add, #Del, #Train, #Valid, #Test, are the number of adding edges, deleting edges, training edges, validating edges, and testing edges in each snapshot respectively.

Since our model can fit in any embedding methods, we use TransE and DistMult as our embedding methods. For baselines, we compared our model with online models puTransE and DKGE, static models TransE and DistMult (learning from scratch).

For the evaluation metrics, we take link prediction as the evaluation method. Link prediction [14] in a KG is defined as the task of predicting an entity that has a specific relation with another given entity. This task consists of two parts. One is given the head entity and the relation, then predicts the tail entity (denoted as (h,r,?)). and another is given the tail entity and the relation, then predicts the head entity (denoted as (?,r,t)). Rather than requiring the best result, this task usually ranks a set of candidate entities from the KG, which is the rank of the actual answer in the predicted answer. There are two evaluation metrics to evaluate the model on the whole test set. The Mean Reciprocal Rank (MRR) is the average multiplicative inverse of the ranks for all head entities and tail entities in test edges. Another is the Hits@10, the proportion of ranks not larger than 10 for all head entities and tail entities in the test edges. The hyper-parameters are set as the same for all baselines and models. And the replay ratio is set to be 10%, which means replay 10% of the old edges. We also recorded the time consumption for each model to train. Since the limit of the size of a table, we only show the first three snapshots' data for IMDB-30SP. The detailed data can be found in the below table. Please be noted that DistMult (Online) and TransE (Online) are our models.

		YAGO-3SP			IMDB-30SP		
		MRR	Hits@10	Time (s)	MRR	Hits@10	Time (s)
Snapshot 1	puTransE (Online)	0.180	0.262	N/A	0.122	0.188	N/A
	DKGE (Online)	0.460	0.545	953	0.381	0.569	6,950
	TransE (Static)	0.348	0.508	132	0.330	0.499	1,531

	DistMult (Static)	0.518	0.567	219	0.395	0.581	3,957
Snapshot 2	puTransE (Online)	0.186	0.259	N/A	0.119	0.182	N/A
	DKGE (Online)	0.440	0.539	191	0.380	0.567	350
	TransE (Static)	0.300	0.460	144	0.323	0.492	1,721
	DistMult (Static)	0.517	0.570	230	0.395	0.582	4,122
	TransE (Online)	0.297	0.458	8	0.322	0.489	63
	DistMult (Online)	0.511	0.566	20	0.391	0.579	212
Snapshot 3	puTransE (Online)	0.173	0.247	N/A	0.123	0.187	N/A
	DKGE (Online)	0.442	0.542	163	0.377	0.561	423
	TransE (Static)	0.304	0.460	152	0.323	0.492	1,867
	DistMult (Static)	0.517	0.571	233	0.397	0.586	4,201
	TransE (Online)	0.297	0.457	5	0.319	0.484	57
	DistMult (Online)	0.507	0.566	14	0.389	0.581	196

As the table shows, compared with the static baselines, our models lose a little bit of accuracy but significantly reduce the training time. Compared with online baselines, our models gain more accuracy and take less time. The training time of puTransE is not given because the accuracy of puTransE is extremely low and comparing with it makes no sense. Finally, the most important point of our model is that our model does not depend on any specific embedding method. The static embedding methods can be easily converted into the online version. If there is a more accurate model, the online version of this model will also be more accurate.

6. Conclusion

In this report, we presented two training improvement methods for continual learning in KG embeddings, independent of the embedding methods and do not specify any embedding method. Compared with state-of-the-art static and online KG embedding learning models, our model has a better efficiency in online learning with an acceptable accuracy loss. As for future work, we will study how to do the online training without freezing the relation embeddings since our model currently cannot detect the shift of the relation embeddings. Besides, we plan to implement our model for Question Answering (QA) tasks.

Reference:

[1] Ren, Hongyu, Hanjun Dai, Bo Dai, Xinyun Chen, Denny Zhou, Jure Leskovec, and Dale Schuurmans. "SMORE: Knowledge Graph Completion and Multi-hop Reasoning in Massive Knowledge Graphs." *arXiv preprint arXiv:2110.14890* (2021).

[2] Wang, H., Zhang, F., Xie, X., and Guo, M. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the International World Wide Web Confer- ence (WWW)*, 2018.

[3] Ren, H. Snap-stanford KGReasoning: Multi-hop Reasoning on KGs. https://github.com/snap- stanford/KGReasoning, 2021.

[4] Zitnik, M., Agrawal, M., and Leskovec, J. Modeling polypharmacy side effects with graph convolutional net- works. *Bioinformatics*, 2018.

[5] Y. Tay, A. Luu, and S. C. Hui, “Non-parametric estimation of multiple embeddings for link prediction on dynamic knowledge graphs,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, no. 1, 2017.

[6] T. Wu, A. Khan, M. Yong, G. Qi, and M. Wang, “Efficiently embedding dynamic knowledge graphs,” Knowledge-Based Systems, vol. 250, p. 109124, 2022.

[7] Y. Wei, W. Chen, Z. Li, and L. Zhao, “Incremental update of knowledge graph embedding by rotating on hyperplanes,” 2021 IEEE International Conference on Web Services (ICWS), 2021.

[8] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-Relational Data, in: NIPS, 2013, pp. 2787–2795.

[9] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575, 2014.

[10] The´o Trouillon, Johannes Welbl, Sebastian Riedel, E´ric Gaussier, and Guillaume Bouchard. Com- plex embeddings for simple link prediction. In International Conference on Machine Learning, pp. 2071–2080, 2016.

[11] Sun, Zhiqing, et al. "Rotate: Knowledge graph embedding by relational rotation in complex space." arXiv preprint arXiv:1902.10197 (2019).

[12] Kirkpatrick, James, et al. "Overcoming catastrophic forgetting in neural networks." Proceedings of the national academy of sciences 114.13 (2017): 3521-3526.

[13] F. Mahdisoltani, J. Biega, F. M. Suchanek, YAGO3: A Knowledge Base from Multilingual Wikipedias, in: CIDR, 2015.

[14] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge Graph Embedding: A Survey of Approaches and Applications, IEEE Transactions on Knowl- edge and Data Engineering 29 (12) (2017) 2724–2743.