Continual Learning Knowledge Graph Embeddings

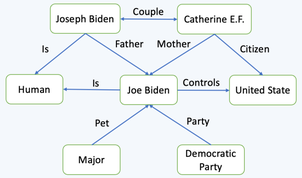
Yang Yiliu (SID: 1155157082)

Supervisor: Prof. James Cheng

Introduction

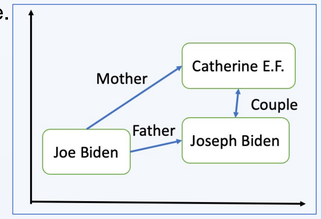
A Knowledge Graph (KG) is a graph structure that contains knowledge in the form of head-relation-tail triples, where the head and tail are entities being connected with the relation.

Edge list:
Joe Biden – is – Human,
Joseph – Father – Biden,
Biden – Controls – U.S.,
…



Knowledge Graph Embeddings

However, there could be unobserved edges in KGs (Catherine E.F. – Is – Human).
In order to predict these missing edges, the most common method is to embed entities and relations into a low-dimension vector space.

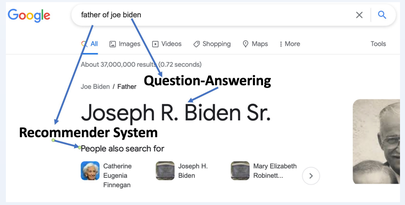


If there is an edge (h,r,t), The goal of an embedding model is to fit **h+r ≈ t**.

To evaluate an edge in the embedding model, we need a score function. Different embedding methods have different score functions. Then the goal is converted to maximize the score.

Embedding Method	Score Function
TransE [1]	−‖h+r−t‖
DistMult [2]	<h,r,t>

Applications:
1. Question-Answering bot



2. Recommender Systems

…

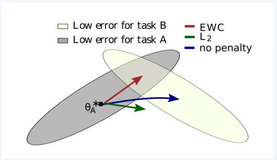
Continual Learning

Most existing models focus on embedding static KGs. If there is an update, these models must retrain with the whole edge set, which takes a lot of time. However, real-world KGs are always updated over time. In order to reduce time consumption, there is an urgent need for a model that supports continual learning.

Only a few models are supporting continual learning of KGs, such as puTransE [3], and DKGE [4]. But they both specified the embedding method. puTransE took TransE as its embedding method while DKGE modified RotatE [5] to fit its continual learning algorithm. Currently, there is no common method to deal with continual learning on KGs that applies to all embedding methods. Here we present two training improvements that apply to all embedding methods. Compared with state-of-the-art models, our model has better accuracy with less training time.

Elastic Weight Consolidation

Elastic Weight Consolidation (EWC) [6] is an improvement on traditional machine learning. And we discovered that the modified EWC can be used in continual learning for knowledge graph embeddings.



Assume that there are two tasks A and B, and a model M which has been trained on A. Simply training M on B will cause a phenomenon called Catastrophic Forgetting, which means the model will completely forget A. The goal of EWC is to train M on B without forgetting A.

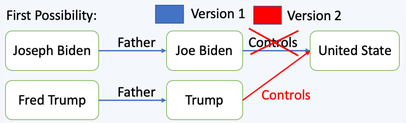
The idea of EWC is to assign importance degrees F_i to each parameter and penalize the model if the parameter changes a lot. The derivation of the importance degree can be seen in [6]. F_i is defined as $F_i = (\frac{\partial \mathcal{L}(\theta_i)}{\partial \theta_i})^2$, where $\mathcal{L}$ is the loss function of this model. Then the new loss function is defined as $\mathcal{L} = \mathcal{L}_{Original} + \lambda \sum_i F_i(\theta_i - \theta_i^*)^2$, where λ is a hyper-parameter and θ_i^* is the old parameter, i.e., parameter trained on the older edge set.

Edges Replay

Except for EWC, replaying some old edges when training can also reduce catastrophic forgetting. At the same time, replaying more edges means taking more time. There is a trade-off between training time and model accuracy. 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 weights calculated by the influenced degrees.

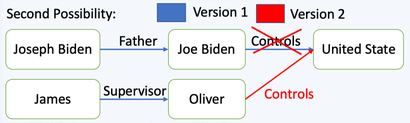
We define the influence degree of a new edge as the reciprocal of the edge score. Recall that the less score is, the better an edge fits in the vector space. $I_{edge} = \frac{1}{Score(edge)}$. Then the influence degree of an entity is defined as $I_{entity} = \sum_{edge \in N(entity)} I_{edge}$. Finally, the replaying weights for edges is defined as $W_{edge} = I_{head\ entity} + I_{tail\ entity}$.

The idea of these definitions are easy to understand. Take these two KGs as examples.



1) Biden Controls U.S. in the first KG. Then Trump win the election and becomes the new president in the second KG. The entities are less needed to be updated.

2) But what if I win the election? The score of (Oliver – Controls – U.S.) is significantly low. Then all entities that connected to me are needed to be updated since it changes a lot.



Finally, the replaying edges are sampled based on the replaying weights for edges. Overall, the loss function of our model consists of two parts, i.e., the EWC loss, and the score loss of new and replaying edges.

Experimental Results

YAGO-3SP and IMDB-30SP are two real-world datasets that consist of Wikipedia data and movie data. They both are updated over time and have 3 and 30 snapshots respectively.

We implemented our two improvements on DistMult [2] and tested this model (DistMult-OL) on the above two real-world datasets. The evaluation data is shown in the below table.

The evaluation data shows that compared with the state-of-the-art online models, our model takes less training time and have greater accuracy.

Question: (? – Party - Trump):

Rank	Answer	Score
1	(Green–Party–Trump) ✗	0.6
2	(Republican–Party–Trump) ✓	0.5
3	(Democratic–Party–Trump) ✗	0.2

So, the rank of this question is 2.

$$\text{MRR} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Rank_i}$$

$$Hits@10 = \frac{1}{N}\sum_{i=1}^{N} Id(Rank_i \le 10)$$

		YAGO-3SP			IMDB-30SP		
		MRR	Hits@10	Time (s)	MRR	Hits@10	Time (s)
Snapshot 1	puTransE	0.180	0.262	N/A	0.122	0.188	N/A
	DKGE	0.460	0.545	953	0.381	0.569	6,950
	DistMult	0.518	0.567	219	0.395	0.581	3,957
Snapshot 2	puTransE-OL	0.186	0.259	N/A	0.119	0.182	N/A
	DKGE-OL	0.440	0.539	191	0.380	0.567	350
	DistMult	0.517	0.570	230	0.395	0.582	4,122
	DistMult-OL	**0.511**	**0.566**	**20**	**0.391**	**0.579**	**212**
Snapshot 3	puTransE-OL	0.173	0.247	N/A	0.123	0.187	N/A
	DKGE-OL	0.442	0.542	163	0.377	0.561	423
	DistMult	0.517	0.571	233	0.397	0.586	4,201
	DistMult-OL	**0.507**	**0.566**	**14**	**0.389**	**0.581**	**196**

Conclusion

In this poster, we presented two training improvements for continual learning in KG embeddings, which do not specify any embedding methods. Compared with state-of-the-art static and online KG models, our model has a better efficiency in online learning with an acceptable accuracy loss.

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-Relational Data, in: NIPS, 2013, pp. 2787–2795.
[2] 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.
[3] 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.
[4] T. Wu, A. Khan, M. Yong, G. Qi, and M. Wang, "Efficiently embedding dynamic knowledge graphs," Knowledge-Based Systems, vol. 250, p. 109124, 2022.
[5] Sun, Zhiqing, et al. "Rotate: Knowledge graph embedding by relational rotation in complex space." arXiv preprint arXiv:1902.10197 (2019).
[6] Kirkpatrick, James, et al. "Overcoming catastrophic forgetting in neural networks." Proceedings of the national academy of sciences 114.13 (2017): 3521-3526.