CS249 2022Spring Group5 Report: Adversarial Attacks on Knowledge Graph Embeddings

Yikai Zhu
zhuyikai@g.ucla.edu
University of California, Los Angeles
Los Angeles, California, USA

ABSTRACT

Knowledge graph embedding(KGE) is a technique for learning continuous embeddings for entities and relations in the knowledge graph. Despite the widespread use of KGE models, little is known about the security vulnerabilities that might disrupt their intended behaviour. In this project we study the problem of generating data poisoning attacks against KGE models for the task of link prediction in knowledge graphs. We implement state-of-art attack methods and run experiments on FB15k-237 and WN18RR. We further analyse why "Direct attack" method works and propose an improved method based on our analyse. In addition, we propose a method to attack the global structure of knowledge graph. Our experiments show that the proposed strategy outperform the origin method. However, all the methods proposed don't drop the performance too much. In the end, we summary the pros and cons of the existing methods and come up with some ideas about future work.

KEYWORDS

knowledge graph embedding, adversarial attack

ACM Reference Format:
Yikai Zhu. 2018. CS249 2022Spring Group5 Report: Adversarial Attacks on Knowledge Graph Embeddings. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Knowledge graph, as a well-structured effective representation of knowledge, plays a pivotal role in many real-world applications such as web search [7], question answering[8, 19], and personalized recommendation [20]. It is constructed by extracting information as the form of triple from unstructured text using information extraction systems. Each triple (h_i, r_i, t_i) represents a relation r_i between a head entity h_i and a tail entity t_i.

Although such triples can effectively record abundant knowledge, their underlying symbolic nature makes them difficult to be directly fed to many machine learning models. Hence, knowledge graph embedding, which projects the symbolic entities and relations into continuous vector space, has quickly gained significant attention[4, 9, 11, 14, 18].

Despite the increasing success and popularity of Knowledge graph embedding models, their robustness has not been fully analyzed. In fact, many knowledge graphs are built upon unreliable or even public data sources. For instance, the well known Freebase[3] harvests its data from various sources including individual, user-submitted wiki contributions. The openness of such data unfortunately would make KGE models vulnerable to malicious attacks. When being attacked, biased knowledge graph embeddings would be generated, leading to serious impairment and financial loss of many downstream applications. Therefore, there is strong need for the analysis of the vulnerability of knowledge graph embeddings.

Designing data poisoning attacks against KGE models has three main challenges. First, the naive approach of re-training a new KGE model for each candidate perturbation is computationally prohibitive. Second, it is computationally intractable to enumerate through all candidate adversarial additions. Furthermore, due to unique characteristics characteristics of knowledge graph and it embedding models, existing adversarial attack methods on graph data cannot be directly applied to attack KGE models.

In this project, we systemically investigate the sensitivity of KGE model. First, we analyze the mathematical explanation of one state-of-art method, which the original paper does not give. Based on the mathematical explanation, we optimize the existing method. Second, we implement experiments on four KGE models - DistMult, ComplEx, TransE and RotatE in two benchmark datasets - WN18RR and FB15k-237. We analyze efficiency and performance for each method. Our experiment shows our proposed method outperform the origin method.

2 RELATED WORK

2.1 Knowledge Graph Embeddings

[5] and [10] provide a comprehensive survey of KGE models. Existing methods can be roughly divided into two categories: translational distance models and semantic matching models. Translational distance models measure the plausibility of a fact as the distance between two entities after a translation carried out by the relation. TransE [4], TransH [16] and TransR [9] are the representative approaches in this category. Semantic matching models measure plausibility of facts by matching latent semantics of entities and relations embodied in their vector space representations. The typical models include RESCAL [11], DistMult [18] and ComplEx [15]. The attack strategy proposed in this paper can be used to attack most of the existing KGE models while we will use the popular

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

models DistMult[18], ComplEx[15], TransE[4] and RotatE[14] in our experiment.

2.2 Data Poisoning Attack

Data poisoning attacks, such as those in [2, 13], are a family of adversarial attacks on machine learning methods. In these works, the attacker can access the training data of the learning algorithm, and has the power to manipulate a fraction of the training data in order to make the trained model meet certain desired objectives.

[1] is now the state-of-art adversarial attack method and can be applied to any KGE models for adversarial deletion and addition. The main idea of this paper is to find the most influential triple (h_i, r_i, t_i) in the training set, then find an entity t_i' which has the least cos similarity with t_i. To find the most influential triple, they propose to measure the similarity with the feature of target triple $g(\hat{h}, \hat{r}, \hat{t})$ and the feature of the candidate triple $g(h_i, r_i, t_i)$, where g is a function outputs the feature of triple. They use the state-of-art KGE scoring functions without reduction over the embedding dimension. For example, For the DistMult model, the triple feature vector is $\boldsymbol{g} := \boldsymbol{h} \circ \boldsymbol{r} \circ \boldsymbol{t}$. In addition, they use $\boldsymbol{g}(h, r, t) := \nabla_{\theta} \mathcal{L}((h, r, t), \widehat{\theta})$. For similarity, they try dot product, cos similarity and L2 distance. However, it has some major drawbacks. The second g function need to calculate gradient for every candidate triple. For each target triple, there are about 100 ~ 1000 candidate triples, which is really computationally prohibitive. It is so inefficient that we cannot use it to attack all the test triples.

[21] proposed an efficient method named Direct Attack. Their method only calculates gradient for every target triple and is more efficient than [1]. Given an target triple $\hat{h}, \hat{r}, \hat{t}$ and the score function $f(\hat{h}, \hat{r}, \hat{t})$. [21] try to find a triple $(\hat{h}, r', t')$ which can help to minimize the score of target triple $f(\hat{h}, \hat{r}, \hat{t})$. The optimal embedding shifting vector of $\hat{h}$ is $\epsilon_h^* = -\frac{\partial f\left(\hat{h}, \hat{r}, \hat{t}\right)}{\partial \hat{h}}$. They sample some negative triples, calculate the score $f(\hat{h} + \epsilon_h^*, r', t') - f(\hat{h}, r', t')$ for each triple and choose the triple with the maximum score. However, in their paper they only explain the basic intuition behind the attack function and do not give a mathematics explanation.

In addition, CRIAGE [12] is another adversarial additions against KGE models. But CRIAGE is only applicable to multiplicative models. The official implementation of CRIAGE has lots of bugs. Therefore, we won't implement it in our experiments.

3 ATTACK METHOD

3.1 Problem Definition

Let us consider a knowledge $\mathcal{KG}$ with a training set denoted as $\{(h_i, r_i, t_i)\}_{n=1}^{N}$. Consider a targeted set denoted as $\left\{\left(\hat{h}, \hat{r}, \hat{t}\right)\right\}_{m=1}^{M}$. We use $\boldsymbol{h_i}, \boldsymbol{r_i}, \boldsymbol{t_i}$ to denote the embedding of the head entity h_i, relation entity r_i and tail entity t_i. Our task is to minimize the plausibility of all the triples in $\{(h_i, r_i, t_i)\}_{m=1}^{M}$, by making adding perturbation facts on the training set. Furthermore, we assume the attacker is only capable of making M perturbations.

We focus on the white-box attack setting where the attacker has full knowledge of the victim model architecture and access to the learned embeddings. However, they cannot perturb the architecture or the embeddings directly; but only through perturbations in the training data.

3.2 Direct Attack and Optimization

First we will give the mathematics explanation of the method proposed by [21] and try to optimize it. For candidate triple $(\hat{h}, r', t')$, if we optimize the score of this triple $f(\hat{h}, r', t')$, we want that the embedding of target head $\hat{h}$ will move towards ϵ_h^*. Specifically, we want to maximize $\frac{\partial(\hat{h}, r', t')}{\partial \hat{h}}$. The score function is an approximation of maximizing the dot product of expected direction and its gradient if we consider Taylor series:

$$f(\hat{h} + \epsilon_h^*, r', t') - f(\hat{h}, r', t') \approx \epsilon_h^{*T} \frac{\partial(\hat{h}, r', t')}{\partial \hat{h}}$$

With this mathematics explanation, we can further optimize it by changing the score function to $f(\hat{h} + \epsilon_h^*, r', t') - f(\hat{h} - \epsilon_h^*, r', t')$ since central difference is a better approximation than forward difference. We will name this optimization as "central difference".

In addition, [21] only try to attack the entity instead of the relation since the number of facts that a relation involves is much larger than the number of facts that an entity involves and therefore it is easier to attack the entity. However, since every relation involves a great number of facts, much more triples might be influenced if we focus on attacking the relation. We use the score function $f(h', \hat{r} + \epsilon_r^*, t') - f(h', \hat{r} - \epsilon_r^*, t')$ to choose the triple which can best attack the relation and name this method as "direct relation attack".

3.3 Least Confidence Attack

We notice the basic idea of [1] is to find the most influential triple in the training set and then find the contradict triple. Why not just inject some contradict to the training set directly? So we consider to add the contradict triple of itself using the same method as [1]. However, the performance of this attack is almost the same as random addition. Therefore, we think that using cos similarity to find the contradict fact is not helpful. We should find another way to find the contradict.

A simple way to find $e = \arg\min f(h, r, t')$ as the contradict entity. We then add (h, r, e) as the contradict triple and try to confuse the model by this way. However, this method does not perform as well as expected since the number of contradict is relatively small and these contradict is random and unorganized. Another way is to inject contradict is to add those triples with least score to the training dataset. We first sample some candidate negative triples, and then pick K triples which the original model will give lowest scores. We name this method as "least confidence".

4 EXPERIMENT

Since FB15k and WN18 suffer from data leakage problem[6], We evaluate the effectiveness of the proposed attack strategies in degrading the KGE model's predictions on all test triples on FB15k-237 and WN18RR. We evaluate our attacks on four state-of-art KGE models-DistMult, ComplEx, RotatE and TransE. The number of perturbations is set to number of test triples for all attack methods. We follow the state-of-art protocol to evaluate poisoning attacks

Table 1: Reduction in MRR and Hits@1 in FB15k-237. Lower values indicate better results; best results for each model are in bold. First block of rows are the baseline attacks with random additions; second block is state-of-art attacks; remaining are the proposed attacks. For each block, re report the best performance as well as the percentage relative to the origin version; computed as (poisoned − original)/original

		DistMult		ComplEx		TransE		RotatE	
		MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1
origin		100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Baseline Attacks	Random_n	97.48	96.25	97.85	97.05	62.71	27.71	94.43	94.23
	Random_g	97.69	97.07	98.75	98.57	69.34	37.16	95.19	95.31
	Direct	95.32	93.76	97.23	96.17	52.90	10.41	96.77	97.34
	Dot	96.39	94.26	98.15	96.87	95.65	93.50	98.59	98.55
	L2	97.41	95.63	98.09	97.39	95.87	92.47	99.28	99.15
	Cos	97.28	94.87	98.68	98.47	95.88	92.88	98.80	97.42
Proposed Attacks	Central_Diff	93.88	91.07	97.22	96.51	54.81	13.28	95.13	94.48
	Direct_Rel	93.61	**90.92**	**94.67**	**91.85**	**49.93**	**5.96**	**89.53**	**79.62**
	Least_Conf	**92.69**	91.31	96.01	96.01	56.87	17.34	97.31	96.94

Table 2: Reduction in MRR and Hits@1 in wn18rr. Lower values indicate better results; best results for each model are in bold. First block of rows are the baseline attacks with random additions; second block is state-of-art attacks; remaining are the proposed attacks. For each block, re report the best performance as well as the percentage relative to the origin version; computed as (poisoned − original)/original

		DistMult		ComplEx		TransE		RotatE	
		MRR	H@1	MRR	H@1	MRR	H@1	MRR	H@1
origin		100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Baseline Attacks	Random_n	98.04	99.50	98.10	98.26	88.18	27.18	96.23	95.06
	Random_g	99.17	99.67	99.21	99.13	94.12	43.69	100.11	99.89
	Direct	98.71	99.42	99.12	99.17	87.53	**0.97**	97.19	96.24
	Dot	94.82	90.25	96.16	93.69	91.39	36.90	95.90	92.98
	L2	88.22	77.36	88.19	76.91	90.36	35.92	95.11	91.72
	Cos	**87.69**	**75.66**	**88.06**	**75.18**	90.90	38.84	**94.44**	**90.93**
Proposed Attacks	Central_Diff	98.43	99.09	98.81	98.93	**87.46**	9.71	98.23	97.65
	Direct_Rel	99.02	98.88	100.00	98.86	92.52	31.07	99.84	99.20
	Least_Conf	98.00	96.82	95.97	95.97	91.60	14.56	99.22	99.09

[17]-we train a victim KGE model on the original dataset; generate adversarial additions using one of the attack method; perturb the original dataset; and train a new KGE model on the perturbed dataset. The hyperparameters for victim and poisoned KGE models are same.

Baselines: We evaluate our attacks against baseline methods based on random addition and the state-of-art poisoning attacks. *Random_n* adds a random triple from the neighbourhood of the target triple. *Random_g* adds a random triple globally. *Direct* is the adversarial addition attack proposed in [21]. CRIAGE will run into numpy.linalg.LinAlgError:Singular matrx error. Therefore we don't implement this method in our experiment. *Dot*, *L2*, *Cos* is the method proposed in [1]. We only implement the instance similarity method since we find that gradient similarity is so inefficient that run time exceeds 24 hours. In the original paper, they only attack 100 triples and that's why they can run their method.

The source code implementation of our project is available at github [1]. For the details of *Dot*, *L2*, *Cos*, we follow the implementation from github[2] and do some modification to adjust their method into our KGE model implementations.

4.1 Comparison with Baselines

Overall Attack Performance: For FB15k-237 dataset, we observe that the proposed strategies for adversarial additions successfully outperform the baseline attacks from Table 1. *Direct_Rel* performs best in all models, especially on TransE. For WN18RR dataset, results are completely different. Our methods work well only on TransE. That is because the score function of TransE is $h + r - t$ and the gradient of relation and entity is not related to others and thus are more vulnerable for gradient-based methods. *Cos* proposed by [1] is better than other models. We think this is because entities in wn18rr are more sparse FB15k-237 and therefore model can still

[1] https://github.com/zyksir/AdversarialAttackOnKGE
[2] https://github.com/PeruBhardwaj/AttributionAttack/

learn the graph structure with a small disturbance. [1] finds the most influential triple for each target triple and therefore disturbs the global graph structure.

The different between wn18rr and FB15k-237 can be explained by the difference in the graph structure of these two datasets. wn18rr is much more sparse than FB15k-237. If we corrupt the most influential triple, we corrupt the reasoning process and then the model might fail to infer the target triple. In FB15k-237, it is dense and it's hard to corrupt the model by add contradicts to one triple in the training set. The model can still infer the target triple by other triples. While the shifting methods have better performance since with a little shifting, the entity might have different meaning.

We find *Central_Diff* works similar to *Direct*. This is because during the process of generating the noise we randomly choose some candidate triples, which introduces uncertainty. In addition, if we add another triple in the training set, the final embedding will also change and the best direction calculated before will change as well. However, the key idea behind the *Direct* does become more clear after we give the mathematical explanation. In addition, the success of *Direct_Rel* shows that the assumption proposed by [21] is wrong. They claim the innate characteristics relation is more stable than entity. While our experiment shows that if we focus on attacking the relation, the performance will drop more. One relation involves more facts than one entity, which means more facts will get involved and even the global structure of the graph might get disturbed if one relation get disturbed.

The attack methods on DisMult, ComplEx and RotatE actually work similar to the random methods. We think this is because we focus on attacking the single triple and do not disturb the global graph structure of Knowledge Graph. This should be the guidance of future work.

In addition, we try to enlarge the search space of those methods which need to randomly sample some candidate triples. The original search space of *Direct* is $n_{relation} \times n_{entity}/20$ for each triple. After we enlarge the search space by 4 times, the performance does not change too much.

4.2 Efficiency Analysis

Table 3: Time Cost(seconds) For each method on different datasets. We report the average time cost of 4 models

Attack Method	FB15k-237	wn18rr
Random_n	0.19	0.04
Random_g	0.19	0.03
Direct	631.11	85.32
Dot	103.97	16.95
L2	104.52	16.75
Cos	103.35	16.76
Central_Diff	627.94	85.47
Direct_Rel	635.84	88.11
Least_Conf	1003.06	296.83

We can see from the table that the efficiency of *Direct* is the worst. For other methods proposed by [17] which need to calculate the gradient of every candidate triple. We implement them only to find it might run more than 24 hours. Even *Direct* is slow, we find using 'random.choices' to choice candidates is time consuming. In reality, this can easily be optimized by choosing the candidate triples in ahead. In summary, for gradient-based method, we should restrict the number of calculate the gradient and try to use approximation.

5 CONCLUSION

We present a complete study on the vulnerability of existing KGE methods and propose a collection of data poisoning attack strategies for different attack scenarios. Experiment results on two benchmark dataset shows the vulnerability of relation in FB15k-237 and the importance of attacking the global structure. There are many other adversarial attack methods designed to attack the structure of graph and those methods might better disturb the KGE models on wn18rr dataset. We will leave this as a feature work.

REFERENCES

[1] Peru Bhardwaj, John Kelleher, Luca Costabello, and Declan O'Sullivan. 2021. Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods. *arXiv preprint arXiv:2111.03120* (2021).

[2] Battista Biggio, Blaine Nelson, and Pavel Laskov. 2012. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, 1247–1250.

[4] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*. 2787–2795.

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.

[6] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[7] Jens Graupmann, Ralf Schenkel, and Gerhard Weikum. 2005. The SphereSearch engine for unified ranked retrieval of heterogeneous XML and web documents. In *Proceedings of the 31st international conference on very large data bases*. VLDB Endowment, 529–540.

[8] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 221–231.

[9] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

[10] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2015. A review of relational machine learning for knowledge graphs. *Proc. IEEE* 104, 1 (2015), 11–33.

[11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data.. In *ICML*, Vol. 11. 809–816.

[12] Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 2019. Investigating Robustness and Interpretability of Link Prediction via Adversarial Modifications. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 3336–3347. https://doi.org/10.18653/v1/N19-1337

[13] Jacob Steinhardt, Pang Wei W Koh, and Percy S Liang. 2017. Certified defenses for data poisoning attacks. *Advances in neural information processing systems* 30 (2017).

[14] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197* (2019).

[15] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International conference on machine learning*. PMLR, 2071–2080.

[16] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

[17] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing* 17, 2 (2020), 151–178.

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

[19] Wentau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base. In *IJCNLP*. Beijing, China, 1321–1331.

[20] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 353–362.

[21] Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. 2019. Data poisoning attack against knowledge graph embedding. *arXiv preprint arXiv:1904.12052* (2019).