Adversarial Explanations for Knowledge Graph Embeddings - Supplementary Material

1 Code

The python code regarding the experiments conducted in the paper is based on the modification of different repositories (e.g. [Bhardwaj *et al.*, 2021; Dettmers *et al.*, 2018] and the respective references are provided in *Readme.md*. The implementation of our approach uses the java code of AnyBURL [Meilicke *et al.*, 2019] with some adaptions and an additional class that computes the attacks. Within the *code* folder we provide everything needed to reproduce the results.

Code for the individual files in regard to Python can be found in *KGEAttack/ConvE* where *main.py* provides the base code for training/evaluation (We kept the naming convention from the other repositories). Connecting our attacks to the KGE training steps takes place in *explain_process.py* and *explain_prepare.py*. The attack is implemented in the Java class *x.y.z.explain.attack.Explain.java* which can be found in the zipped java source code. Please note that for code execution the compiled version is already placed at the required location.

1.1 Setup

The code is tested and run under the Ubuntu operating system. As a setup we suggest using a fresh conda environment. Please ensure that you have access to a GPU and respective CUDA software. You can find the requirements in *Readme.md*. Alternatively, simply install the environment in *conda_env.yaml* where we dumped the environment that we used during the experiments.

1.2 Running the experiments

We provide bash scripts to reproduce all experimental results in the folder *KGEAttack*. For instance, for reproducing the results for FB15k-237 and ComplEx run:

```
bash preprocess.sh
bash FB15k-237_complex_explain_run.sh
```

This will create different folders where in *ConvE/logs* you will find the log files with the evaluation results. For instance, the results for our approach in the deletion setting can be found in *ConvE/logs/expl_del_complex_FB15k-237_[...].log*. The target evaluation based on the epoch where the model achieved the highest valid MRR is appended to the end of the log files. You will find results marked with $[test]$ which is the old protocol. You will also find the results marked with $[test_delete_filter_orig]$ and $[test_add_prot]$. The former refers to the deletion protocol and the latter to the addition protocol as explained in the main text.

You can also find more meta information in regard to our method in the created folder *data/expl_prep_complex_FB15k-237_[...]*. For instance, you can find the deletions and human readable rules according to the theories in *delete-verbose.txt*.

Please note that Pytorch reproducibility can only be ensured for the same GPU and the same machine when using random seeds. We therefore ran all the bash scripts 5 times and report average results as explained in the main text.

	FB15k-237		WN18RR	
	H@1	MRR	H@1	MRR
ComplEx	0.257	0.348	0.466	0.505
DistMult	0.255	0.343	0.441	0.481
ConvE	0.228	0.318	0.407	0.440

Table 1: Standard results on the validation set.

2 Training Details

We use ComplEx [Trouillon *et al.*, 2016], DistMult [Yang *et al.*, 2015] and ConvE [Dettmers *et al.*, 2018]. We inherit the hyperparameter specifications and training details from [Bhardwaj *et al.*, 2021] who tuned the hyperparameters first to achieve best results on the original validation set. In particular, we use $1vsAll$ training with a Cross Entropy loss. For ComplEx and DistMult, N3 regularization is used. For the FB15k-237 dataset, the Adam optimizer is used with a learning rate of 0.005. For WN18RR, Adam is used with a learning rate of 0.01 for ConvE and DistMult and 0.005 for ComplEx. Furthermore, an embedding dimensionality of 200 is used for ConvE and DistMult and 100 for ComplEx.

As mentioned in the main text *all* experimental results presented are averages over 5 runs. When training the KGE models, in the original run and after the attack, we validate on the normal validation set every 20'th epoch while training for a maximum number of 100 epochs. The final target evaluation results are based on the epoch where the models had the highest valid MRR. We show exemplary these best validation results for one run in Table 1.

In regard to the results of GR [Lawrence *et al.*, 2021] and

direct-del/add [Zhang *et al.*, 2019] we follow code provided from [Bhardwaj *et al.*, 2021]. For direct-add the authors used two downsampling factors (5 and 20) we found that 5 produced better results (stronger degradations) and therefore used this factor in favor of the method. The respective implementations can be found in *gr_del.py*, *ijcai_del.py*, and *ijcai_add.py* (direct).

3 Running our approach standalone

You can run our approach outside of the experimental protocol. This makes it easy to include it in applications or alternative experimental settings. It requires as first argument the path to the folder where the target.txt file is located (the file that lists the triples that should be attacked) and as second argument the path to the folder of the dataset, where train.txt, valid.txt, and test.txt are located (the latter two can also be empty files, however, there need to be such files). This folder that contains target.txt will alos be used as output folder, where the resulting addition.txt and delete.txt files will be stored. Here is an example call (should be in one line):

```
java -cp AnyBURL-ATTACK.jar
x.y.z.attack.Explain
exp/attack/fb237 data/FB15-237
```

In case you encounter a problem related to using a different java version, you can build the jar file on your own. The sources are also available.

References

[Bhardwaj *et al.*, 2021] Peru Bhardwaj, John Kelleher, Luca Costabello, and Declan O'Sullivan. Adversarial attacks on knowledge graph embeddings via instance attribution methods. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8225–8239. Association for Computational Linguistics, 2021.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 1811–1818, 2018.

[Lawrence *et al.*, 2021] Carolin Lawrence, Timo Sztyler, and Mathias Niepert. Explaining neural matrix factorization with gradient rollback. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021*, pages 4987–4995. AAAI Press, 2021.

[Meilicke *et al.*, 2019] Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI/AAAI Press, 2019.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2071–2080. JMLR.org, 2016.

[Yang *et al.*, 2015] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations*, 2015.

[Zhang *et al.*, 2019] Hengtong Zhang, Tianhang Zheng, Jing Gao, Chenglin Miao, Lu Su, Yaliang Li, and Kui Ren. Data poisoning attack against knowledge graph embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4853–4859. International Joint Conferences on Artificial Intelligence Organization, 7 2019.