Explaining Link Predictions in Knowledge Graph Embedding Models with Influential Examples

Adrianna Janik
Accenture Labs, Dublin
adrianna.janik@accenture.com

Luca Costabello
Accenture Labs, Dublin
luca.costabello@accenture.com

1 Introduction

Link prediction is a common task in knowledge graphs, and often tackled with knowledge graph embedding (KGE) models, such as ComplEx, DistMult, TransE [1–3], and others. However, these models lack direct interpretability [4], which is crucial for applications in critical domains like drug discovery and medicine [5]. Existing explainability methods for KGE models [6–9] are limited, and their evaluation approaches and datasets vary. Moreover, the human readability of explanations is often overlooked. Model explainability has been approached differently [10, 11], e.g., via gradient-based methods, perturbations method, or via training simplified model from the original black-box model to provide a model that is transparent and therefore could be interpreted more easily. Other approaches focus on providing the most similar training examples for the predicted one – these we call example-based explanations [12–14] and the ones that are the most important we call influential examples. This form of explanations is coherent with human understanding of cases and therefore it makes this form of explanations more understandable for the user. In this work, we propose a new method that generates explanations for link predictions in KGE models using influential examples.

However, achieving explainability in KGE models presents several challenges. First, there is a lack of evaluation protocols, metrics, and benchmark datasets specifically designed for assessing the explainability of these models. Another challenge for explainability arises from lack of ground-truth explanations apart from synthetic datasets and limited user studies on explanations in the existing literature. Last but not least the model's predictions are ranked based and are not calibrated to represent probabilities directly, which makes it difficult to interpret the results quickly [15].

At the moment, understanding the factors contributing to predictions made by a Knowledge Graph Embedding (KGE) model is a challenging task, the class of models is not designed with transparency in mind [16]. One approach to address this challenge is to explore specialised interpretability methods that can be applied post-hoc (after-training). For example, we can investigate the adaptation of existing interpretability techniques designed for other machine learning models [16].

To address the lack of interpretability in KGE models, we propose post-hoc interpretability method. Our goal is to provide explanations that link predictions back to the original graph, highlighting the links and nodes that contribute the most to a given prediction. These explanations should be understandable to users and provided quickly and efficiently.

Related Work The most basic way to identify influential triples would be to perform a simple search over all possible triples that could be removed from the dataset and perform retraining after each such modification of the dataset. This approach is very inefficient as it requires many retrainings of the model. For example, if explanation size, we are interested in, is equal to $|e| = 1$ we need n retrainings of the model for each triple, when n is the number of triples in the training dataset. The number of retrainings is increasing if we allow the explanation to be greater than 1, $|e| > 1$.

Basic principle of the majority of explanation methods presented below is as follows: they try to identify such existing links in the graph that their removal will strongly decrease the probability of the predicted link (this holds for ExplaiNE [8] and GNNExplainer [17] but not for GraphLIME [18]).

A. Janik et al., Explaining Link Predictions in Knowledge Graph Embedding Models with Influential Examples (Extended Abstract). Presented at the Second Learning on Graphs Conference (LoG 2023), Virtual Event, November 27–30, 2023.

Apart from work on explainability of Knowledge Graph Embedding models we would like to bring attention to an overlapping subject of robustness and adversarial attack approaches for Knowledge Graph Embedding models, where influential triples are sought after for a differnt reason. In recent work, Bhardwaj et al [7] explored methods of poisoning KGEs with relation inference patterns, which aims at targeting influential triples and design attacks based on it. Another work by Betz et al [9] introduced adversarial explanations where they identify regularities in the knowledge graph and plan attacks based on them. In [19], authors investigated robustness of knowledge graph embedding models with regards to removal or addition of an influential triple to the training set.

GradientRollback [6] works by storing gradient updates in a separate influence matrix per every training example t (during training) and also per every unique entity and relation in a triple. It then refers to this gradient update matrix during the explanation phase. The influence updates regarding the training triple (t) are subtracted from the parameters matrix to obtain a new parameter matrix that simulates the situations of retraining the model without t. This approach requires enabling a special training mode, and requires much more memory than the initial dataset size to store training artefacts. This method although time and memory consuming traces parameters in the training leading to probably more accurate results on the expense of the high memory cost.

With the following shortcomings in mind we posed the following **research question** *How to provide pertinent explanations for KGE models trained on large knowledge graphs with reasonable time/memory constraints?*

2 Method

In this section we introduce the intuition behind the proposed approach, notation, and our heuristics to obtain influential examples and evaluation protocol.

Intuition We propose ExamplE, a post-hoc, local explanation approach that explains Knowledge Graph Embedding predictions. Our approach is based on the assumption that to explain why a certain link between two entities is predicted as plausible, we have to look at the latent space representation of that triple (individually at its subject, object, and predicate embeddings) and try to "reverse-engineer" the training samples that the pattern was extracted from. For example we are interested in the past cases that contributed to this prediction the most, e.g., if our knowledge graph contains patients and we want to make prediction on a single patient we would like our method to retrieve past patients that are the most similar to the one we are predicted.

Preliminaries Let us introduce key concepts and the notation used throughout the article. Let $\mathcal{G}$ be a knowledge graph, denoted as $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$, where $\mathcal{E}$ is a set of entities, $\mathcal{R}$ is a set of predicates, and finally $\mathcal{T}$ is a set of statements - triples defining specific links between entities $\mathcal{E}$ with types of relations $\mathcal{R}$, e.g.: triple $t_{(s,p,o)} \in \mathcal{T}$ represents a directed edge in the knowledge graph $\mathcal{G}$, where s is the head entity (subject), p is the relation (predicate), and o is the tail entity (object). Let e be an entity in $\mathcal{G}$. The 1-hop neighborhood of entity e, denoted as $N(e, 1)$, is defined as: $N(e, 1) = \{(s, p, o) : (s, p, o) \in \mathcal{G}, e \in \{s, o\}\}$, it contains such triples in graph $\mathcal{G}$ that either their subject or their object is the same as the entity e for which the neighbourhood is being derived. Consequently we will define an n-hop neighbourhood of an entity, denoted as $N(e, n)$ as: $N(e, n) = \{(s, p, o) : (s, p, o) \in \mathcal{G}; s, o \in S' \cup O'\} \cup \{N(e, n-1)\}$, where $S' = \{s : (s, p, o) \in N(e, n-1)\}$ and $O' = \{o : (s, p, o) \in N(e, n-1)\}$. It contains triples from the $n-1$ neighbourhood and triples that are connected to them. Building on top of this formalisation we will define a 1-hop neighbourhood of a triple $t_{(s,p,o)}$ as: $N(t_{(s,p,o)}, 1) = \{N(s, 1) \cup N(o, 1) \setminus t\}$ and consequently we will define n-hop neighbourhood of a triple $t_{(s,p,o)}$ as: $N(t_{(s,p,o)}, n) = \{N(t, n-1) \cup \{(s, p, o) : (s, p, o) \in \mathcal{G}; s, o \in S' \cup O'\}\}$ where $S' = \{s : (s, p, o) \in N(t_{s,p,o}, n-1)\}$ and $O' = \{o : (s, p, o) \in N(t_{s,p,o}, n-1)\}$.

2.1 ExamplE algorithm:

ExamplE is an example-based heuristics that consists of four steps: sampling, filtering for examples, aggregating for prototype and assembling the Explanation Graph. **Prerequisites:** Calibrated Knowledge Graph Embedding model, returning probability estimates as predictions, following [15], in this way we are ensuring that the predictions are bounded and are as close to the real probabilities as the current SOTA allows. **Input:** Trained KGE model M, target triple t, training graph $\mathcal{G}$. **Output:** Explanation Graph EG, sorted influential triples E.

1. **Latent space sampling:** Given, s_t is a subject of the target triple, and f is a method of KGE model to obtain embedding of an element; and s_i is another entity in $\mathcal{G}$ different than s_t, and a distance measure: $dist(\mathbf{s_t}, \mathbf{s'})$ we can define an ordered set of: $S_{s_t} = \{s_i : dist(f(s_t), f(s_{i-1})) \leq dist(f(s_t), f(s_i)) \forall s_i \in \mathcal{E}\}$ and consquently set S_m as an ordered set $S^m = \{s_1, s_2, ..., s_m\}$, $S^m \subset S_{s_t}$, where elements of the set are entities with the same ordering as in set S_{t_s}, as described above. We will also define an ordered set with distances between subject entity and other entities as: $D_{S_{s_t}} = \{(s_i, d_i) : d_i = dist(f(s_t), f(s_{i-1})) \leq dist(f(s_t), f(s_i)) \forall s_i \in \mathcal{E}\}$ We will now repeat the same operation for the object o_t of the target triple to obtain set $O^m = \{o_1, o_2, ..., o_m\}$, $O^m \in O_{o_t}$, analogically to the target triple subject we will define ordering for the object entities as follows: $O_{o_t} = \{o_i : dist(f(o_t), f(o_{i-1})) \leq dist(f(o_t), f(o_i)) \forall o_i \in \mathcal{E}\}$ Similarly we will also save distances to the object entity for the other entities in set $D_{O_{o_t}}$ as below and $D_{P_{p_t}}$ for predicates (omitted for brevity): $D_{O_{o_t}} = \{(o_i, d_i) : d_i = dist(f(o_t), f(o_{i-1})) \leq dist(f(o_t), f(o_i)) \forall o_i \in \mathcal{E}\}$
2. **Filtering for example triples:** The second step is to obtain the Cartesian product of sets S^m and O^m to create a set of candidate triples with the target triple predicate p, as denoted below: $eG_t = \{S^m \times O^m : (s_i, p, o_i) \in \mathcal{G}\}$
3. **Aggregating for prototype:** Obtain N-hop neighborhoods of example triples (eG_t) from the step above, and aggregate into a prototype graph pG_t following strict or permissive strategy. By strict - take the intersection of sets of triples between n-hop neighbourhoods of examples and target triple, as: $pG_t = \cap_{i=1}^{len(eGT_t)} N(t_{(s,p,o)_i}, n) \cap N(t_{(s,p,o)}, n)$ and by permissive - take the union of sets of triples between n-hop neighbourhoods of examples and intersect it with the target triple n-hood neighbourhood, as: $pG_t = \cup_{i=1}^{len(\mathcal{T})} N(t_{(s,p,o)_i}, n) \cap N(t_{(s,p,o)}, n)$
4. **Assembling the Explanation Graph:** the last step combines the results of the previous steps into an Explanation Graph: $EG = pG_t \cup eG_t \cup \{t\}$ Influential triples are the ones that are identified in step 2 in eG_t, and their scores are derived as follows: $E = \{((s,p,o), \frac{D_{S_{s_t}}[s] + D_{O_{o_t}}[o] + D_{P_{p_t}}[p]}{3}) : \forall (s,p,o) \in eG_t\}$. The scores are used to sort the triples, and the most influential ones have the lowest score (are most similar to the target triple).

2.2 Evaluation Protocol

To evaluate the method we generated explanations for TransE model on two different datasets with our proposed approach and also using a baseline approach and then modified the training dataset based on the explanations and retrained the models. To evaluate our explanation approach we consider only generation of influential examples with associated scores specifically eG_t. The choice of a model is arbitrary and the method works independently of this choice.

Baseline: As a baseline we utilized random explanation approach where we limit triples only to the ones with same predicate without any constrains on how and if such triple is connected to the target triple. We can denote the baseline in a mathematical formulation as follows. First, we define a set E of all triples that have the same predicate as target triple t: $E = \{(e', p', e'') : (e', p', e'') \in \mathcal{G}, p' = p\}$ Then, we draw a random sample of n triples from set E as a baseline explanation, equal in a size to the number of triples obtained from our heuristics. The sample is drawn without replacement assuming uniform distribution.

Metrics In this work we used a metric called Probability Difference measured as percentage. It is similar to the metric used in [6] with the difference that PD used in this work takes into account both explanations that increase the score after retraining and the ones that decrease the triple scores after retraining. It is used, to measure the difference between prediction scores obtained by originally trained KGE model M on a target triple t and a prediction obtained from the model trained on the dataset without explanation of such training triple, it is defined as follows: $PD = \frac{(M(t) - M'(t)) * 100}{M(t)}$, where $M(t)$ denotes KGE model prediction on target triple t. The choice of this metric is dictated by easier interpretation of this metric.

3 Results

To analyze the performance of our approach we proposed two experiments. **1) Remove-and-Retrain (ROAR)** [20]: We removed explanation from the dataset and retrained the model (ROAR protocol)

on the modified dataset without explanation for two cases removing only the most influential example triple and removing full set of examples returned. **2) Reversed-Remove-and-Retrain (rev-ROAR)**: We removed all triples with same predicate as Target Triple (set E) and instead added only the explanation. In this scenario we wanted to test whether model can recover from a loss of majority of its examples. We also explored two cases leaving only the most influential example triple and leaving the full set of examples returned by the method.

Note: Both models: original and the one retreined in each experiment were trained exactly the same, with SOTA hyperparameters for the TransE model on the dataset with early stopping. This has the following consequence of evaluation time being very long, for the four scenarios presented we had to train 5 models per dataset per target triple. For FB15k-237 trained with TransE model we got 0.20 Hits@1 (H@1), and 0.30 Mean Reciprocal Rank (MRR). For WN18RR trained with TransE we obtained 0.05 H@1 and 0.22 MRR, the results are reported in Appendix for this dataset. We have also trained other models and generated explanations with our method for them, their performance is reported in the Appendix along with example explanations obtained for them.

epoch	**average**	rev-ROAR [%]		ROAR [%]	
		1	all	1	all
10 ours	-0.507	69.694	63.971	**0.106**	5.717
rand.	-0.478	73.757	62.186	0.013	**14.647**
20 ours	-0.066	25.457	23.819	-0.0	-0.009
rand.	0.007	27.968	21.456	0.001	-0.011
30 ours	-0.181	10.911	8.27	-0.003	0.013
rand.	-0.348	11.117	8.388	-0.001	-0.073
40 ours	0.009	5.312	4.735	0.0	0.034
rand.	0.096	5.435	4.154	0.002	0.048
50 ours	-0.049	2.707	2.227	0.03	0.036
rand.	0.021	2.863	2.196	0.031	0.022
60 ours	-0.112	1.635	1.456	-0.002	-0.016
rand.	-0.057	1.718	1.332	-0.001	-0.007
70 ours	-0.135	0.894	0.786	-0.0	-0.011
rand.	-0.072	0.94	0.721	-0.0	-0.015
80 ours	0.053	0.574	0.496	-0.001	-0.001
rand.	0.002	0.599	0.456	-0.001	-0.038
90 ours	-0.153	0.384	0.325	-0.0	0.028
rand.	-0.421	0.402	0.303	-0.0	-0.0
100 ours	-0.153	0.23	**0.203**	-0.007	-0.01
rand.	-0.03	**0.244**	0.176	-0.007	-0.005

Table 1: TransE on Fb15k-237. Probability difference between original model and models retrained using ROAR and rev-ROAR w.r.t. most influential triple (1) and all explaining triples (all). When retraining the model with a single triple of given predicate (rev-ROAR-1) the model recovers from it's initial 70% probability drop at epoch 10th to 0.2 difference at epoch 100th. The same pattern but faster is present for the training with full explanation. In the ROAR experiment removing a single triple has only influence epoch 10th of training with 0.1% probability drop, this is increased when all triples are removed to 6%.

4 Discussion

In this work we introduce a novel heuristics to generate explanations for knowledge graph embedding models. It works by generating influential examples from the constrained latent space search. For future work we plan to compare how this approach work on the GNNs. One disadvantage, that we are aware of, is that our approach is a heuristics, we are doing a follow-up research on how to provide estimation guarantees for this approach. The research question posed at the beginning of the article states two component of the approach 1) its quality and 2) complexity. To evaluate the quality of the method in our future work we will compare the results with other methods in the literature. The second part of the question was a motivation for developing our approach, mainly in relation to [6]. In appendix we include the time it takes to obtain explanations. We were able to reduce the time from minutes [6] to seconds in this work.

Overall, the evaluation results support the effectiveness of our explanation approach based on influential examples. By removing these examples and retraining the model, we observed a decrease in the plausibility of the target triple, indicating the importance of the identified examples in the original prediction. This demonstrates the potential of our method to provide meaningful and informative explanations for link predictions in knowledge graph embedding models.

References

[1] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 2071–2080, New York, NY, USA, June 2016. JMLR.org. 1

[2] 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, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. 1

[4] Federico Bianchi, Gaetano Rossiello, Luca Costabello, Matteo Palmonari, and Pasquale Minervini. Knowledge graph embeddings and explainable AI. 2020. doi: 10.3233/SSW200011. URL http://arxiv.org/abs/2004.14843. 1

[5] Luca Costabello, Sumit Pai, Nicholas McCarthy, and Adrianna Janik. Knowledge graph embeddings tutorial: From theory to practice, September 2020. URL https://doi.org/10.5281/zenodo.4268208. https://kge-tutorial-ecai2020.github.io/. 1

[6] Carolin Lawrence, Timo Sztyler, and Mathias Niepert. Explaining neural matrix factorization with gradient rollback. 2020. URL http://arxiv.org/abs/2010.05516. 1, 2, 3, 4, 7

[7] Peru Bhardwaj, John Kelleher, Luca Costabello, and Declan O'Sullivan. Poisoning Knowledge Graph Embeddings via Relation Inference Patterns. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1875–1888, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.147. URL https://aclanthology.org/2021.acl-long.147. 2

[8] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. ExplaiNE: An approach for explaining network embedding-based link predictions. 2019. URL http://arxiv.org/abs/1904.12694. 1

[9] Patrick Betz, Christian Meilicke, and Heiner Stuckenschmidt. Adversarial Explanations for Knowledge Graph Embeddings. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 2820–2826, Vienna, Austria, July 2022. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-1-956792-00-3. doi: 10.24963/ijcai.2022/391. URL https://www.ijcai.org/proceedings/2022/391. 1, 2

[10] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in Graph Neural Networks: A Taxonomic Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(05): 5782–5799, May 2023. ISSN 0162-8828. doi: 10.1109/TPAMI.2022.3204236. URL https://www.computer.org/csdl/journal/tp/2023/05/09875989/1GqajxgkWcM. Publisher: IEEE Computer Society. 1

[11] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. 58:82–115. ISSN 1566-2535. doi: 10.1016/j.inffus.2019.12.012. URL http://www.sciencedirect.com/science/article/pii/S1566253519308103. 1

[12] Mark T. Keane and Eoin M. Kenny. How case-based reasoning explains neural networks: A theoretical analysis of XAI using post-hoc explanation-by-example from a survey of ANN-CBR twin-systems. In Kerstin Bach and Cindy Marling, editors, *Case-Based Reasoning Research and Development*, Lecture Notes in Computer Science, pages 155–171. Springer International Publishing. ISBN 978-3-030-29249-2. doi: 10.1007/978-3-030-29249-2_11. 1

[13] R. Caruana, H. Kangarloo, J. D. Dionisio, U. Sinha, and D. Johnson. Case-based explanation of non-case-based learning methods. *Proceedings. AMIA Symposium*, pages 212–215, 1999. ISSN 1531-605X.

[14] Tim Miller. Explanation in Artificial Intelligence: Insights from the Social Sciences, August 2018. URL http://arxiv.org/abs/1706.07269. arXiv:1706.07269 [cs]. 1

[15] Pedro Tabacof and Luca Costabello. Probability calibration for knowledge graph embedding models. 2020. URL http://arxiv.org/abs/1912.10000. 1, 2

[16] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models. 2018. URL http://arxiv.org/abs/1802.01933. 1

[17] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. 32:9240–9251, 2019. ISSN 1049-5258. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7138248/. 1

[18] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. GraphLIME: Local interpretable model explanations for graph neural networks. 2020. URL http://arxiv.org/abs/2001.06216. 1

[19] Pouya Pezeshkpour, Yifan Tian, and Sameer Singh. 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)*, pages 3336–3347. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1337. URL https://aclanthology.org/N19-1337. 2

[20] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d{\textbackslash}textquotesingle Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 3

A Appendix

Method	time/triple
ExamplE	5.09 s
GradientRollback	6±7 min

Table 2: Time to obtain explanations for triples for model trained on different FB15k-237 dataset according to ExamplE and GradientRollback [6]

epoch	**average**	rev-ROAR [%]		ROAR [%]	
		1	all	1	all
10 ours	0.0	13.515	15.018	**1.118**	**1.696**
rand.	-0.004	-	-	0.089	-0.306
20 ours	-0.004	26.975	29.598	0.008	3.5
rand.	-0.027	-	25.585	-0.021	-0.058
30 ours	0.018	34.791	37.518	0.031	1.198
rand.	0.005	34.384	-	-0.046	0.035
40 ours	-0.004	38.778	42.265	0.048	0.041
rand.	0.02	-	-	0.021	0.036
50 ours	0.006	40.661	44.588	0.038	0.095
rand.	0.025	42.159	41.134	-0.01	0.004
60 ours	0.015	42.071	45.949	0.054	0.018
rand.	0.042	43.586	42.388	0.011	0.028
70 ours	-0.032	43.149	46.509	0.071	0.021
rand.	-0.309	44.506	-	0.005	-0.002
80 ours	-0.017	42.835	46.048	0.025	0.017
rand.	-0.252	44.057	-	0.001	-0.02
90 ours	0.047	42.512	45.655	0.01	0.012
rand.	0.058	43.58	-	0.011	0.013
100 ours	-0.008	41.778	**44.503**	0.023	0.002
rand.	0.01	**42.673**	41.57	0.021	0.014

Table 3: TransE on WN18RR - Probability difference between original model and models retrained using two different scenarios ROAR and rev-ROAR considering most influential triple (1) and all triples from the obtained explanation (all). We can see that when retraining the model with only a single triple of given predicate (rev-ROAR-1) the model cannot recover from initial 14% probability drop at epoch 10th instead it worsen to reach it's peak at around epoch 70th (amounting to 43%) to settle on nearly 42% difference at epoch 100th, we can observe the same pattern but faster for the training with full explanation. On the other hand when we look at the ROAR experiment we can see that removing a single triple has only influence epoch 10th of training with 1.1% probability drop, this is increased when all triples are removed to 1.7%, the difference above epoch 10th is smaller than 1%.

Table 4 lists training parameters for models used in the experiments.

model/dataset	WN18RR	Fb15k-237
TransE	k=350, eta=30	k=400, eta=30

Table 4: Parameters used for model training, trained with early stopping for 4000 epochs using Adam optimizer with lr=0.0001, multiclass-nll loss, seed=0, regularizer L2 with lambda=0.0001

B Computational Complexity Analysis

We have analyzed computational complexity of the ExamplE in the scenario of batch explanations. ExamplE requires access to the training dataset so space complexity starts from $O(t)$. Given: e - number of entities (e.g.: 14,541 in Fb15k-237). k - embedding vector dimension (e.g. k=400, TransE

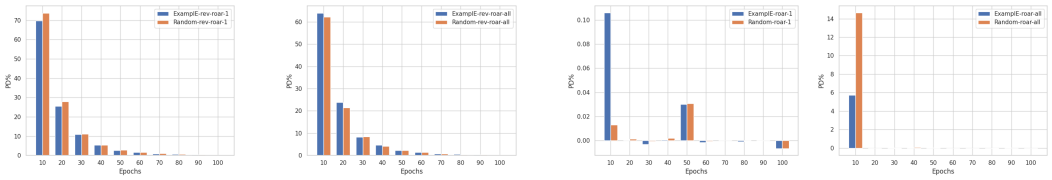

Figure 1: Probability differences between ExamplE and random baseline, TransE trained on Fb15k-237.

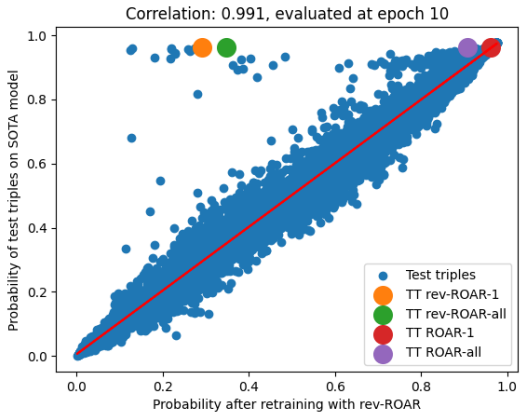

Figure 2: TransE on Fb15k-237 - Probabilities correlation before and after retraining model with reversed ROAR only explanations is left in the training dataset among triples with same predicates as target triple. Above epoch 20 the Pearson correlation coefficient is 1 and predictions are perfectly correlated.

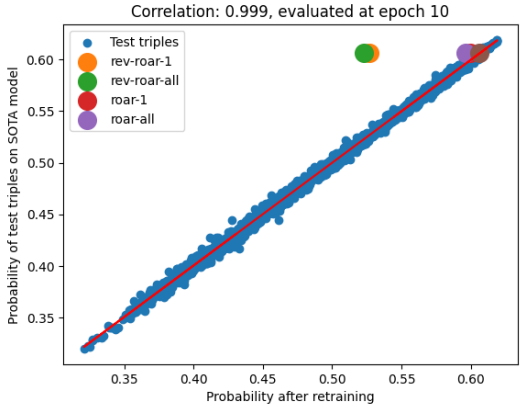

Figure 3: TransE on WN18RR - Probabilities correlation before and after retraining model with ROAR.

Approach	TransE		ComplEx		DistMult		ConvE	
	H@1	MRR	H@1	MRR	H@1	MRR	H@1	MRR
Fb15k-237	0.20	0.30	0.21	0.31	0.21	0.30	0.21	0.30
WN18RR	0.05	0.22	0.47	0.50	0.43	0.47	0.44	0.47

Table 5: MRR and Hits@1 on the Fb15k-237 and WN18RR benchmark datasets.

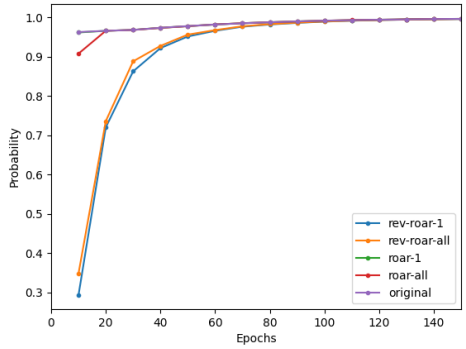

Figure 4: TransE on Fb15k-237. Target triple probability across different epochs. We can see that the probability difference between models before and after explanation removal is the lowest at the beginning of the training. It changes in such a way that model can recover it's predictive ability to predict on a given triple. It means that to make an evaluation of an explainability method, one has to consider the time aspect of the prediction.

on Fb15k-237). m - number of nearest neighbours considered (parameter of ExamplE default m=25). t - number of triples in the train set. x - number of examples, as explanation $x << t$. we can split computational complexity into steps: 1) Sampling - this step is entirely dependent on the nearest neighbour algorithm implementation, in the experiments we used implementation provided in sklearn, which by default tries to adjust parameters for best efficiency. In the worst case scenario it uses a brute force approach which complexity of the prediction time is $O(e \times k \times m)$ with negligible complexity of initialization of the algorithm and negligible space complexity too. In the best case scenario kNN algorithm tries to adjust the inner data structure for optimized inference time with the cost of initialization and space e.g. in the case of KD-Tree it is $O(k \times e \times log(e))$ of extra initialization time and $O(k \times e)$ space with a benefit of inference time being $O(m \times log(e))$. ExamplE needs

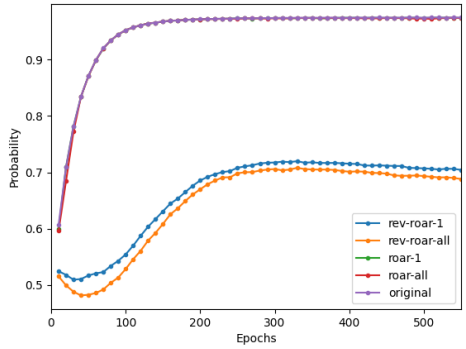

Figure 5: TransE on WN18RR. Target triple probability across different epochs.

	initialization	
step	time	sapce
$sampling_{KD-Tree}$	$O(k \times e \times log(e))$	$O(k \times e)$
$sampling_{brute-force}$	$O(1)$	$O(1)$

Table 6: Computational time and space complexity of initialization phase.

	prediction	
step	time	sapce
$sampling_{KD-Tree}$	$O(m \times log(e))$	$O(1)$
$sampling_{brute-force}$	$O(e \times k \times m)$	$O(1)$
product	$O(m^2)$	$O(1)$
mapping	$O(t)$	$O(t)$
filtering	$O(min(t, m^2))$	$O(1)$
post-processing	$O(x)$	$O(1)$

Table 7: Computational time and space complexity of prediction phase.

to find m nearest neighbours for both subject and object entity (in the default case) in this step. 2) Filtering for example triples - in this step we need to take a cartesian product of obtained sets of neighbours in step 1: which leaves us with $O(m^2)$ (default case, in full case it is $O(m^3)$ if we are considering predicates embedding as well) and forces us to filter examples according to the dataset. First we are mapping it into a tuples ($O(t)$, where t is a number of train triples), than we utilize sets intersections implementation in Python with complexity of $O(min(t, m^2))$. In the post-processing step we compute the score per each example obtained. The computational complexity in batch explain is always dependent on the number of target triples to obtain explanations for.

S	P	O	Score
Billy Idol	languages spoken, written, or signed	English	TT
Johnny Marr	languages spoken, written, or signed	English	0.00075
Chester Bennington	languages spoken, written, or signed	English	0.00076
Morrissey	languages spoken, written, or signed	English	0.00077
Loreena McKennitt	languages spoken, written, or signed	English	0.00077
Gordon Lightfoot	languages spoken, written, or signed	English	0.00080
Alan Stivell	languages spoken, written, or signed	English	0.00080
Robert Plant	languages spoken, written, or signed	English	0.00080
Oleg Skripka	languages spoken, written, or signed	French	0.00091
Alan Stivell	languages spoken, written, or signed	French	0.00091
Oleg Skripka	languages spoken, written, or signed	Russian	0.00097
Oleg Skripka	languages spoken, written, or signed	Ukrainian	0.00117

Table 8: Example explanation for a test triple in CODEX-M dataset - first row represents Target Triple (TT). The lower the score, the closer is example to the Target Triple.

S	P	O	Score
Artie Lange	/influence/influence_node/influenced_by	Jackie Gleason	TT
George Carlin	/influence/influence_node/influenced_by	Danny Kaye	0.17232
Conan O'Brien (aka Big Red)	/influence/influence_node/influenced_by	Danny Kaye	0.19481
Conan O'Brien (aka Big Red)	/influence/influence_node/influenced_by	Steve Allen	0.24502
Bill Maher	/influence/influence_node/influenced_by	Steve Allen	0.25498

Table 9: Example explanation for a test triple in Fb15k-237 dataset - first row represents Target Triple (TT). The lower the score, the closer is example to the Target Triple.

S	P	O	Score
02314321	_hypernym	08102555	TT
02314001	_hypernym	08102555	0.02788
02313495	_hypernym	08102555	0.04839
01928360	_hypernym	08102555	0.05155
02314717	_hypernym	08102555	0.06077
01928737	_hypernym	08102555	0.06294
02321759	_hypernym	02316038	0.09046

Table 10: Example explanation for a test triple in WN18RR dataset - first row represents Target Triple (TT). The lower the score, the closer is example to the Target Triple.