A Simplified Benchmark for Non-ambiguous Explanations of Knowledge Graph Link Prediction using Relational Graph Convolutional Networks*

Nicholas Halliwell[1], Fabien Gandon[1], and Freddy Lecue[1,2]

[1] Inria, Université Côte d'Azur, CNRS, I3S, France
[2] CortAIx, Thales, Montreal, Canada
{nicholas.halliwell,fabien.gandon,freddy.lecue}@inria.fr

Abstract. Relational Graph Convolutional Networks (RGCNs) identify relationships within a Knowledge Graph to learn real-valued embeddings for each node and edge. Recently, researchers have proposed explanation methods to interpret the predictions of these black-box models. However, comparisons across explanation methods is difficult without a common dataset and standard evaluation metrics to evaluate the explanations. In this paper, we propose a method, including two datasets (Royalty-20k and Royalty-30k), to benchmark explanation methods on the task of explainable link prediction using Graph Neural Networks. We report the results of state-of-the-art explanation methods for RGCNs. [3]

Keywords: link prediction · Explainable AI · knowledge graphs.

1 Need for Explaining Links Predicted by RGCNs

Knowledge Graphs often represent facts as triples in the form *(subject, predicate, object)*, where a *subject* and *object* represent an entity, linked by some *predicate*. Link prediction is used to discover new facts from existing ones. Graph embedding algorithms can be used for link prediction, learning a function to map each subject, object, and predicate to a low dimensional space. A Relational Graph Convolutional Network (RGCN) [4] leverages Graph Convolutional Networks [3] with a scoring function such as DistMult [5] as an output layer, returning a probability of the input triple being a fact.

Recently, methods have been developed to explain the predictions of Graph Neural Networks. GNNExplainer [6] explains the predictions of any Graph Neural Network, learning a mask over the adjacency matrix to identify the most informative subgraph. ExplaiNE [2] quantifies how the predicted probability of a link changes when weakening or removing a link with a neighboring node.

[3] These datasets are available: https://gitlab.com/halliwelln/royalty-datasets

The authors of ExplaiNE acknowledge the difficulty in measuring the quality of explanation generated, and a lack of available datasets with ground truth explanations [2]. This approach is evaluated with four datasets, none of which include ground truth explanations. Additionally, computing ground truth explanations for these datasets is non-trivial.

On the task of link prediction on Knowledge Graphs, GNNExplainer remains untested. Neither GNNExplainer or ExplaiNE use the same dataset in their experiments. On the task of explainable link prediction on Knowledge Graphs, no standard datasets or scoring metrics exist to evaluate explanation methods.

In this work, we propose a method, including two datasets (Royalty-20k and Royalty-30k) to evaluate explanations methods for link prediction using RGCNs. We propose the use of a scoring metric on these datasets, and perform a benchmark comparing the explanations of ExplaiNE and GNNExplainer.

2 Building the Royalty Datasets with Explanations

We define ground truth explanations as a single justification for an entailment. We use an open-source semantic reasoner with rule-tracing capabilities [1] to generate ground truth explanations for each rule, allowing for each set of explanations to be precisely controlled and selected. We rely on a set of rules equivalent to strict Horns clauses i.e. disjunctions of literals with exactly one positive literal l_c, all the other l_i being negated: $\neg l_1 \vee ... \vee \neg l_n \vee l_c$. The implication form of the clause can be seen as an inference rule assuming that, if all l_i hold (the antecedent of the rule), then the consequent l_c also holds, denoted $l_c \leftarrow l_1 \wedge ... \wedge l_n$. In our case, each literal is a binary predicate capturing a triple pattern of the Knowledge Graph with variables universally quantified for the whole clause.

For a given Knowledge Graph and a given set of rules, the semantic reasoner performs a forward chaining materialization of all inferences that can be made. Each time the engine finds a mapping of triples $T_1, \ldots, T_n$ making the antecedent of a rule true, it materializes the consequent triple T_c, and records the explanations in the form $T_c \leftarrow (T_1, \ldots, T_n)$, where T_c is a generated triple, and triples $T_1, \ldots, T_n$ are its explanation.

For example, under the *hasGrandparent* rule, if two triples (*Princess Marie Anne*, *hasParent*, *Louis XIV*), and (*Louis XIV*, *hasParent*, *Anne of Austria*) are identified in a Knowledge Graph, the semantic reasoner generates a new triple (*Princess Marie Anne* , *hasGrandparent*, *Anne of Austria*). We compute explanation triples in a similar fashion for each logical rule defined in each dataset. In this example, the explanation for why (*Princess Marie Anne*, *hasGrandparent*, *Anne of Austria*) is a valid fact is because (*Princess Marie Anne*, *hasParent*, *Louis XIV*), and (*Louis XIV*, *hasParent*, *Anne of Austria*).

This approach to generating ground truth explanations can be applied to many Knowledge Graphs and many sets of rules from different domains. It also corresponds to the RGCN explanation methods, which provide explanations in the form of a set of evidence triples.

Dataset	Predicate	# Triples	# Rule Generated Triples	# Unique Entities	Explanation Cardinality	Predicate Property
Royalty-20k	hasSpouse	7,526	3,763	6,114	1	Symmetric
	hasSuccessor	6,277	2,003	6,928	1	Inverse
	hasPredecessor	6,277	2,159	6,928	1	Inverse
	Full data	20,080	7,924	8,861	-	-
Royalty-30k	hasSpouse	7,526	3,763	6,114	1	Symmetric
	hasGrandparent	7,736	7,736	4,330	2	Chain
	hasParent	15,472	0	-	-	-
	Full data	30,734	11,499	11,483	-	-

Table 1: Royalty datasets: Breakdown of each predicate in the dataset. # of Triples denotes the total number of triples with that predicate. Explanation Cardinality denotes the number of triples in the ground truth explanation set.

There may be several ways to define these logical rules. For example, *hasSuccessor* can be defined using its inverse relation *hasPredecessor*. In some cases, *hasPredecessor* could be correlated to the *hasParent* relation and therefore considered an explanation. Both explanations could be correct, thus the optimal explanation is ambiguous. In this work, we build two datasets with non-ambiguous explanations, where each logical rule is designed to give one and only one ground truth explanation for each triple in the training and test sets. This prevents an explanation method from having to arbitrarily select between alternative explanations. Table 1 gives dataset statistics for each predicate. Each rule is detailed below.

Spouse Entity X is the spouse of Y if Y is the spouse of X, more formally, $hasSpouse(X, Y) \leftarrow hasSpouse(Y, X)$. This is a symmetric relationship. A predicate p is said to be symmetric for some subject s and object o if and only if $(s, p, o) \leftarrow (o, p, s)$. There are $7,526$ triples with the *hasSpouse* predicate in each dataset, $3,763$ of which are generated by rules. Note this rule is the same for both datasets.

Successor and Predecessor A successor in the context of royalty is one who immediately follows the current holder of the throne. X is the successor of Y if Y is the predecessor of X. Equivalently, $hasSuccessor(X, Y) \leftarrow hasPredecessor(Y, X)$. Likewise, a predecessor is defined as one who held the throne immediately before the current holder. X is the predecessor of Y if Y is the successor of X. Equivalently, $hasPredecessor(X, Y) \leftarrow hasSuccessor(Y, X)$. Indeed *hasSuccessor* and *hasPredecessor* follow an inverse relationship. A predicate p_1 is the inverse of p_2 if and only if $(s, p_1, o) \leftarrow (o, p_2, s)$. Therefore triples with the *hasSuccessor* predicate are used to explain the triples with the *hasPredecessor* predicate and vice-versa. There are $6,277$ triples with *hasSuccessor* pred-

icate, 2,003 of which are generated by rules. Similarly, there are 6,277 triples with *hasPredecessor* predicate, 2,159 of which are generated by rules.

Grandparent We define *hasGrandparent* using a chain property pattern. A predicate p is a chain of predicates p_i if and only if $(s, p, o) \leftarrow (s, p_1, s_2) \wedge ... \wedge (s_n, p_n, o)$. Y is the grandparent of X if Y is the parent of X's parent P. Equivalently, $hasGrandparent(X, Y) \leftarrow hasParent(X, P) \wedge hasParent(P, Y)$. There are 7,736 triples with *hasGrandparent* predicate, all of which are generated by rules.

3 Evaluating Explanations of RGCNs

In this benchmark, ExplaiNE and GNNExplainer were asked to identify existing triples in the graph as candidate explanations. We use the Jaccard similarity to score the similarity between predicted and ground truth explanations. This metric is appropriate, as it compares the triples in common between the predicted and ground truth without considering the order. Furthermore, this metric can be applied to data from different domains without relying on further assumptions. Benchmark results can be found in Table 2.

As an example of one of ExplaiNE's errors, for some triple (*Albert III, Count of Everstein*, *hasSpouse*, *Richeza of Poland*) and its ground truth explanation (*Richeza of Poland*, *hasSpouse*, *Albert III, Count of Everstein*), ExplaiNE predicted a first degree neighbor (*Richeza of Poland*, *hasSpouse*, *Alfonso VIII of Leon and Castile*) to be its explanation. Note the incorrectly predicted triple uses the *hasSpouse* predicate but in a wrong way.

Across both datasets, we observe GNNExplainer had the highest Jaccard and F_1-Score performance on the *hasSpouse* predicate. GNNExplainer's best performing predicates were *hasSpouse*, *hasSuccessor* and *hasPredecessor*. On the Royalty-30k dataset, we observe performance drops across all metrics on the *hasGrandparent* predicate. We found GNNExplainer to have a recall of 1 for each predicate, meaning the triples in the ground truth explanation are always identified, however, many false positives are included in the predicted explanation set. Indeed a recall of 1 can be trivially achieved by including the entire input graph in the predicted explanation set. This was not the case for the predicted explanations of GNNExplainer, however, the predicted explanation sets were often too large (up to 20 triples on average), and had many false positives.

Table 2 also reports the results of ExplaiNE on the Royalty datasets. Similar to GNNExplainer, we observe the best Jaccard and F_1-Score performance on the *hasSpouse* predicate. On the Royalty-30k dataset, we see lower relative performance on the predicates where larger explanations need to be predicted, such as the *hasGrandparent* rule.

Overall, we find ExplaiNE outperformed GNNExplainer in terms of Jaccard score for all rules across both datasets. While GNNExplainer outperforms ExplaiNE in terms of F_1 score on *hasSpouse*, and *hasGrandparent*, along with the Royalty-30k full dataset, this is likely due to GNNExplainer's high recall, indicating that the F_1 metric is not appropriate for this task.

Dataset	Models	Metrics	Predicates				
			Spouse	Successor	Predecessor	Grandparent	Full set
Royalty-20k	RGCN	Accuracy	0.802	0.74	0.75	-	0.746
	GNN Explainer	Precision	0.656	0.182	0.182	-	0.277
		Recall	1.0	1.0	1.0	-	1.0
		F_1	**0.792**	0.307	0.308	-	0.433
		Jaccard	0.328	0.178	0.178	-	0.184
	ExplaiNE	Precision	0.754	0.319	0.368	-	0.397
		Recall	0.571	0.317	0.365	-	0.577
		F_1	0.65	**0.318**	**0.366**	-	**0.47**
		Jaccard	**0.388**	**0.314**	**0.363**	-	**0.274**
Royalty-30k	RGCN	Accuracy	0.802	-	-	0.75	0.703
	GNN Explainer	Precision	0.656	-	-	0.067	0.261
		Recall	1.0	-	-	1.0	1.0
		F_1	**0.792**	-	-	**0.125**	**0.414**
		Jaccard	0.328	-	-	0.133	0.174
	ExplaiNE	Precision	0.754	-	-	0.101	0.363
		Recall	0.571	-	-	0.135	0.412
		F_1	0.65	-	-	0.115	0.386
		Jaccard	**0.388**	-	-	**0.135**	**0.216**

Table 2: Benchmark results on Royalty-20k and Royalty-30k: Link prediction results for RGCN, and explanation evaluation for GNNExplainer and ExplaiNE. Best F_1 and Jaccard scores per predicate denoted in bold.

References

1. Corby, O., Gaignard, A., Faron Zucker, C., Montagnat, J.: KGRAM Versatile Inference and Query Engine for the Web of Linked Data. In: IEEE/WIC/ACM International Conference on Web Intelligence. pp. 1–8. Macao, China (Dec 2012)
2. Kang, B., Lijffijt, J., Bie, T.D.: Explaine: An approach for explaining network embedding-based link predictions. CoRR **abs/1904.12694** (2019)
3. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations, ICLR (2017)
4. Schlichtkrull, M.S., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., Navigli, R., Vidal, M., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., Alam, M. (eds.) European Semantic Web Conference, ESWC (2018)
5. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR (2015)
6. Ying, Z., Bourgeois, D., You, J., Zitnik, M., Leskovec, J.: Gnnexplainer: Generating explanations for graph neural networks. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) Advances in Neural Information Processing Systems (2019)