Linked Data Ground Truth for Quantitative and Qualitative Evaluation of Explanations for Relational Graph Convolutional Network Link Prediction on Knowledge Graphs

Nicholas Halliwell, Fabien Gandon, Freddy Lecue

Linked Data Ground Truth for Quantitative and Qualitative Evaluation of Explanations for Relational Graph Convolutional Network Link Prediction on Knowledge Graphs

Nicholas Halliwell
Inria, UCA, CNRS, I3S
Sophia Antipolis, France
nicholas.halliwell@inria.fr

Fabien Gandon
Inria, UCA, CNRS, I3S
Sophia Antipolis, France
fabien.gandon@inria.fr

Freddy Lecue
CortAIx, Thales
Montreal, Canada
freddy.lecue@thalesgroup.com

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 for link prediction remains difficult, as there is neither a method nor dataset to compare explanations against. Furthermore, there exists no standard evaluation metric to identify when one explanation method is preferable to the other. In this paper, we leverage linked data to 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. In particular, we rely on the Semantic Web to construct explanations, ensuring that each predictable triple has an associated set of triples providing a ground truth explanation. Additionally, we propose the use of a scoring metric for empirically evaluating explanation methods, allowing for a quantitative comparison. We benchmark these datasets on state-of-the-art link prediction explanation methods using the defined scoring metric, and quantify the different types of errors made with respect to both data and semantics.

CCS CONCEPTS

• **Mathematics of computing** → **Computing most probable explanation**; • **Computing methodologies** → **Knowledge representation and reasoning**; *Neural networks.*

KEYWORDS

link prediction, Explainable AI, knowledge graphs, graph neural networks.

ACM Reference Format:
Nicholas Halliwell, Fabien Gandon, and Freddy Lecue. 2021. Linked Data Ground Truth for Quantitative and Qualitative Evaluation of Explanations for Relational Graph Convolutional Network Link Prediction on Knowledge Graphs. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT '21), December 14–17, 2021, ESSENDON, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3486622.3493921

WI-IAT '21, December 14–17, 2021, ESSENDON, VIC, Australia

ACM ISBN 978-1-4503-9115-3/21/12…$15.00
https://doi.org/10.1145/3486622.3493921

1 INTRODUCTION

Knowledge Graphs [3] are used on tasks such as search engine enhancement, question answering, and product recommendation. 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*.

Knowledge Graphs can be incomplete or evolving, hence they often do not explicitly contain every available fact. Link prediction on Knowledge Graphs [13] is used to identify unknown facts from existing ones. A typical way to perform link prediction on Knowledge Graphs involves the use of graph embeddings algorithms. Such algorithms learn a function mapping each subject, object, and predicate to a low dimensional space. A scoring function is defined to quantify if a link (relation) should exist between two nodes (entities). DistMult [14] learns a diagonal matrix for each relation. A Relational Graph Convolutional Network (RGCN) [12] leverages Graph Convolutional Networks [8] with the scoring function from DistMult as an output layer, returning a probability of the input triple being a fact.

Often the prediction alone is not enough to support decision-making. Users must understand why the model arrived at a particular decision [2]. Therefore, there has been a push to explain the predictions of these algorithms, creating the task of explainable link prediction. The term explanation is commonly defined as a statement that makes something clear. Throughout this paper, we use the term explanation to refer to a set of observations (triples) that provides an understanding of the black-box model's predictions. In other words, we define ground truth explanations as a set of triples that cannot be ignored when justifying the suggestion of adding a targeted link to the graph.

This paper focuses on explainable link prediction specifically on Knowledge Graphs, in particular: ExplaiNE [7] quantifies how the predicted probability of a link changes when weakening or removing a link with a neighboring node, while GNNExplainer [15] explains the predictions of any Graph Neural Network, learning a mask over the adjacency matrix to identify the most informative subgraph. ExplaiNE and GNNExplainer share the idea of providing a selection of existing triples to the user as an explanation.

These state-of-the-art explanation methods have no common datasets used as benchmarks, and have no standard evaluation metrics to measure explanation quality. This prevents quantitative evaluation and comparisons across explanation methods. In this paper, we propose a method, along with two datasets, Royalty-20k, and Royalty-30k, to quantitatively evaluate explanation methods on the task of link prediction using Graph Neural Networks. These

datasets includes ground truth explanations, allowing for comparisons with predicted explanations. Additionally, we propose the use of an evaluation metric, leveraging a similarity between the predicted and ground truth explanation to measure the quality of explanation. Lastly, we benchmark state-of-the-art explanation methods using the proposed dataset and evaluation metric, and quantify the different types of errors made in terms of both data and semantics.

This paper is organized as follows: Section 2 provides an overview of related work on the task of Knowledge Graph embeddings, and explainable link prediction, along with their shortcomings. Section 3 describes a generic approach to generate datasets with ground truth explanations, and a metric for empirical evaluation. Section 4 applies this approach to construct two datasets, outlining the rules that define each dataset. Section 5 details the benchmark performed on the Royalty datasets, and reports the results. Lastly, Section 6 concludes this work, outlining opportunities for future work. All the resources used and produced in this work are available online including the download link for the reasoner, code for this paper and datasets.[1]

2 RELATED WORK: LOOKING FOR A GROUND TRUTH

Knowledge Graph embeddings. Knowledge Graph embedding algorithms learn continuous vectors for each subject, predicate and object. The loss functions are often designed to capture specific algebraic properties of predicates (symmetric, reflexive, transitive, etc). A scoring function is defined to assign a value to each triple based on if the subject, predicate, and object form a valid fact. Typically, the scoring function is included in the loss function. We refer the reader to a recent survey [6] for further details.

A Relational Graph Convolutional Networks (RGCN) [12] can be used to learn embeddings and perform link prediction on Knowledge Graphs. The RGCN performs embedding updates for a given entity by multiplying the neighboring entities with a weight matrix for each relation in the dataset, and summing across each neighbor and relation. A weight matrix for self connections is also learned, and added to the neighbor embedding summation.

Indeed there are other approaches for link prediction i.e. rule based, this work however focuses on link prediction on Knowledge Graphs using Graph Neural Networks.

Explainable link prediction. Few algorithms exist to understand the predictions of Knowledge Graph embedding algorithms. For a given embedding model and some scoring function g, ExplaiNE [7] computes the gradient of the scoring function with respect to the adjacency matrix. Indeed this measures the change in score due to a small perturbation in the adjacency matrix, that is, how much will the score change if a link is added or removed between two given nodes. Formally, given two nodes i, j serving as prediction candidates, and two nodes k, l serving as a candidate explanation, the score assigned to node pair k, l is given by:

$$\frac{\partial g_{ij}}{\partial a_{kl}}(\mathbf{A}) = \nabla_{\mathbf{X}} g_{ij}(\mathbf{X}^*)^T \cdot \frac{\partial \mathbf{X}^*}{\partial a_{kl}}(\mathbf{A}), \tag{1}$$

[1] https://github.com/halliwelln/Explain-KG

where $\mathbf{X}^*$ is the optimal embedding matrix, and a_{kl} is an element of the adjacency matrix $\mathbf{A}$.

To explain the predictions of any Graph Neural Network, GNNExplainer [15] learns a mask over the input adjacency matrix to identify the most relevant subgraph. This is achieved by minimizing the cross entropy between the predicted label using the input adjacency matrix, and the predicted label using the masked adjacency matrix. The objective function minimized by GNNExplainer is:

$$\min_{\mathbf{M}} - \sum_{c=1}^{C} \mathbb{1}[y = c]\, log P_{\Phi}(Y = y | \mathbf{A}_c \odot \sigma(\mathbf{M}), \mathbf{X}_c), \tag{2}$$

where $\mathbf{M}$ is the mask learned and $\odot$ denotes element-wise multiplication.

Explanation quality. The weak point of the empirical evaluation of these explanation methods is often explanation quality. The authors of ExplaiNE acknowledge the difficulty in measuring the quality of explanation generated and a lack of available datasets with ground truth explanations [7]. Moreover, they rely on the assumption that the explanation can be found using one of the 1^{st} degree neighbors. On the task of movie recommendation, ExplaiNE measures the quality of explanations using the average Jaccard similarity between the genres for a given recommended movie, and the set of genres from the top 5 ranked explanations computed. A p-value is computed to estimate the significance of the average. It is unclear how this evaluation method generalizes to tasks outside of movie recommendation. Ideally, a performance metric would not have to rely on such assumptions and would generalize to other tasks.

Ground truth. In general, ground truth does not exist for explanations. For the task of node classification, GNNExplainer uses simulated data with ground truth explanations in the form of connected subgraphs. The explanation accuracy of each node's predicted label is then computed. However, no insight is provided on how to simulate ground truth data for the task of link prediction. Furthermore, GNNExplainer has not been benchmarked by its authors on the task of explainable link prediction on Knowledge Graphs.

Datasets. Datasets with explanations are not available for the previously mentioned approaches to measure the quality of explanations. The authors of ExplaiNE benchmark their approach with 4 datasets: Karate, DBLP, MovieLens, and Game of Thrones networks. These datasets do not include ground truth explanations. Additionally, it is non-trivial to define ground truth explanations on these networks. Without a dataset containing ground truth explanations, it is difficult to recognize if explanation methods such as ExplaiNE and GNNExplainer, are generating high quality explanations. Furthermore, these algorithms use different approaches to evaluating explanations. There is no standard quantitative metric to measure the quality of explanations generated, making comparisons of the methods difficult.

Contributions. Our contributions include a method to quantitatively evaluate explanation methods on the task of link prediction on Knowledge Graphs. Additionally, we propose two datasets, Royalty-20k, and Royalty-30k, that includes ground truth explanations for each observation. Furthermore, we propose the use of

a scoring metric leveraging the similarity between predicted and ground truth explanations, allowing for quantitative comparisons across explanation methods. Lastly, we benchmark state-of-the-art explanation methods, using the proposed dataset and metrics, and quantify the different types of errors made in terms of both data and semantics.

3 GENERATING GROUND TRUTH EXPLANATIONS FOR EVALUATION

3.1 Inference Traces as Explanations

We introduce a generic approach to generate datasets with ground truth explanations. We propose to view the ground truth generation as equivalent to computing a single justification for an entailment. We selected the single-all-axis glass-box category of algorithms [4] that computes a single justification for a triple we will then try to predict instead of inferring. A small and exact set of explanations are needed, that of which must be precisely controlled and selected. Therefore, we select an open-source semantic reasoner with rule-tracing capabilities [1] to generate ground truth explanations for chosen rules, without needing manual annotations. In essence, this tracing pinpoints the input triples that caused the generation of a triple we will then try to predict and explain.

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 instance, $hasGrandparent(X, Z) \leftarrow hasParent(X, Y) \wedge hasParent(Y, Z)$.

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. Note that using reasoning to generate explanations is independent of the algorithm used on the link prediction task.

Indeed this forms an intuitive explanation for graph data, a recent study shows users prefer example based explanations [5]. This generic approach to generating ground truth explanations can be applied to many Knowledge Graphs and many sets of rules. In this work, we focus on non-ambiguous explanations i.e. logical rules that are carefully constructed to give only one ground truth explanation. These rules and datasets were designed to construct explanations containing triples that cannot be ignored when justifying the suggestion of adding a targeted link to the graph. To our knowledge, this approach to generate ground truth explanations has not been previously applied to the task of explainable link prediction on Knowledge Graphs using Graph Neural Networks.

3.2 Explanation Evaluation Metric

To our knowledge, there is no standard evaluation metric to measure the quality of explanations generated by link prediction explanation methods. A standard evaluation metric is needed to identify when one explanation method is preferable to the other. This metric must compare the predicted explanation set to a ground truth explanation set, and assign a similarity score to these two sets.

One way to measure the similarity between a predicted and ground truth explanation set would be to use the Jaccard similarity between a ground truth explanation set E and a predicted set of explanations $\hat{E}$ is:

$$J(E, \hat{E}) = \frac{|E \cap \hat{E}|}{|E \cup \hat{E}|} = \frac{|E \cap \hat{E}|}{|E| + |\hat{E}| - |E \cap \hat{E}|}. \quad (3)$$

In this context, a Jaccard similarity of 1 means the predicted set of the explanation method $\hat{E}$ exactly matches the ground truth set E. Similarly, when E and $\hat{E}$ have no elements in common, the Jaccard similarity is 0. We feel this metric is appropriate, as both ExplaiNE and GNNExplainer are asked to only identify existing triples in the graph to serve as an explanation, therefore only set similarity need be considered.

As an example, let E ={(*Abel, King of Denmark, hasParent, Berengaria of Portugal*), (*Berengaria of Portugal, hasParent, Sancho I of Portugal*)} and $\hat{E}$ ={(*Abel, King of Denmark, hasParent, Berengaria of Portugal*), (*Valdemar II of Denmark, hasParent, Sophia of Minsk*)}. Hence $J(E, \hat{E}) = 0.333$, as they share only one triple in common.

This metric has several nice properties; the Jaccard similarity penalizes a set of candidate explanations when the cardinality differs from the ground truth explanation set. Additionally, the order of the explanations is not considered. Metrics like ROUGE-N [10] or BLEU [11] used in Natural Language Processing (NLP) to compare translations against multiple references adds complexity with no immediate benefit in our case.

A second way to measure explanation quality is to consider the precision, recall and F_1-Score of each explanation method, where $precision = \frac{tp}{tp+fp}$, $recall = \frac{tp}{tp+fn}$, and $F_1 = 2 \cdot \frac{precision \cdot recall}{precision+recall}$. In this context, a false positive (fp) corresponds to a triple predicted to be in the explanation set but shouldn't be. Similarly, a false negative (fn) corresponds to a triple that is predicted to not be in the explanation set but should be. Lastly a true positive (tp) corresponds to a triple that is correctly predicted to belong in the explanation set.

The precision answers the following question; given that a triple is predicted to be in the explanation set, what are the chances that it actually belongs in the explanation set? Furthermore, the recall can be interpreted as how many triples the model was able to correctly identify as belonging in the explanation set. The traditional F_1-Score computes the harmonic mean between the precision and recall, incorporating both of these metrics when evaluating the effectiveness of explanation retrieval. To compare two sets, the Jaccard similarity forms a more intuitive scoring metric, thus we use this metric in addition to precision, recall and F_1-Score in comparing explanation methods.

4 EXTRACTING AND GENERATING THE ROYALTY DATASETS

Applying the method of Section 3.1, we build two datasets (Royalty-20k and Royalty-30k), a collection of 20,080 and 30,734 triples

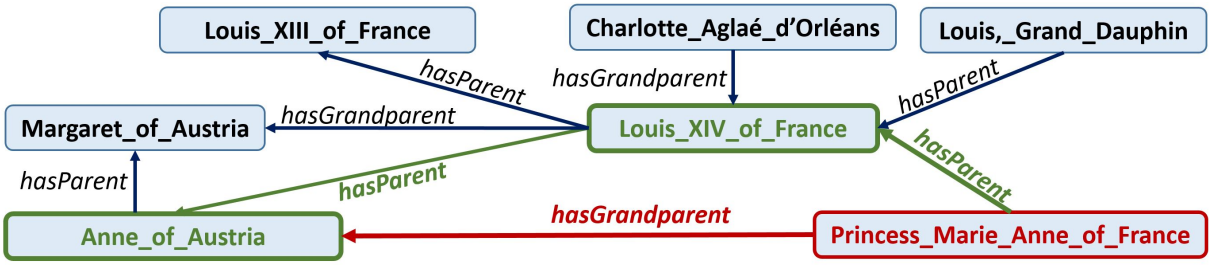

Figure 1: A triple (*Princess Marie Anne of France, hasGrandparent, Anne of Austria*) plotted in red with its explanation set in green {(*Princess Marie Anne of France, hasParent, Louis XIV of France*), (*Louis XIV of France, hasParent, Anne of Austria*)}, and neighboring triples.

respectively, containing royal family members from DBpedia [9]. The triples and explanations are derived from a set of rules introduced later in this section. We use family members to construct datasets with ground truth explanations, as the logical rules can be easily understood, and no prior domain knowledge is needed.

An example from the Royalty-30k can be seen in Figure 1. Take two entities *Princess Marie Anne of France*, and *Anne of Austria*, that we wish to predict the link *hasGrandparent* between. *Anne of Austria* is the grandparent of *Princess Marie Anne of France*, because *Louis XIV of France* is the parent of *Princess Marie Anne of France*, and *Anne of Austria* is the parent of *Louis XIV of France*.

Each example in the Royalty datasets consists of a triple e.g. (*Princess Marie Anne of France*, *hasGrandparent*, *Anne of Austria*) and a set of triples defining its ground truth explanation e.g. {(*Princess Marie Anne of France*, *hasParent*, *Louis XIV of France*), (*Louis XIV of France*, *hasParent*, *Anne of Austria*)}.

4.1 Royalty Datasets Rule Generation

In this work we focus on 4 logical rules based on family relationships: *hasSpouse*, *hasSuccessor*, *hasPredecessor*, and *hasGrandparent*. The predicate of each triple used on the link prediction task is in the consequent of one of these rules. The associated explanation set consists of the triples that triggered the rule.

Indeed there may be several ways to define these logical rules. For example, *hasSuccessor* can be defined using its inverse relation *hasPredecessor*. However, *hasPredecessor* in some cases, could be correlated to the *hasParent* relation and therefore considered an explanation. Both explanations could be correct in many cases, thus the optimal explanation would be ambiguous. Therefore in this work, we define all rules in both datasets such that there is one and only one possible explanation set for each predicate. This prevents an explanation method from having to arbitrarily select between alternative explanations, and ensures a better evaluation and understanding of the explanation techniques.

We define the Royalty-20k dataset using rules for *hasSpouse*, *hasSuccessor*, and *hasPredecessor* predicates. The Royalty-30k dataset is defined using rules for *hasSpouse* and *hasGrandparent* predicates. We create two datasets, separating *hasSuccessor* and *hasPredecessor* from *hasGrandparent*, to avoid having multiple ways to explain a predicate. Each rule is detailed below.

Spouse. Some entity X is the spouse of Y if Y is the spouse of X. i.e. $hasSpouse(X, Y) \leftarrow hasSpouse(Y, X)$. This is a symmetric relationship. 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, 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* predicate, 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* to use a chain property pattern, detailed in Section 4.2. 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. Note *hasParent* is provided by the DBpedia data and not defined by any external logical rule.

4.2 Dataset Specifics

Many of these rules have similar structures because of the algebraic properties of the predicate of the triples they generate. A predicate p is said to be symmetric for some subject s and object o if and only if $(s, p, o) \leftarrow (o, p, s)$. A predicate p_1 is the inverse of p_2 if and only if $(s, p_1, o) \leftarrow (o, p_2, s)$. Lastly, a predicate p is a chain of predicates p_i if and only if $(s, p, o) \leftarrow (s, p_1, s_2) \wedge ... \wedge (s_n, p_n, o)$.

Table 1 gives the details of each predicate. The "# of Triples" column denotes the total number of triples in the dataset with that predicate. The "# of Rule-Generated Triples" column denotes the number of triples that were generated from a triggered rule. These are triples not listed on DBpedia, and thus generated by the semantic reasoner. The "# Unique Entities" column denotes the number of unique nodes in the graph for a given predicate, including the nodes in the associated explanation triples. Furthermore, the explanation

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.

cardinality (Expl. Cardinality) column gives the number of triples in the ground truth explanation sets for each triple inferred by a given rule. This is determined by the definition of the logical rule. Lastly, the Predicate property column describes the algebraic properties of the relations generated by the rules.

5 BENCHMARK

5.1 Experiment Details

One way to perform link prediction on Knowledge Graphs is to learn an embedding for each entity and relation. For this experiment, we use a Relational Graph Convolutional Network (RGCN) [12] to learn embeddings. We chose to use this algorithm, as it can be used with multiple explanation methods without the need for any further adaptations. GNNExplainer is only defined for Graph Neural Networks, hence a GNN must be used on the link prediction task. ExplaiNE requires a model that takes an adjacency matrix as input. The RGCN meets both of these requirements. Additionally, the scoring function has a meaningful interpretation, returning the probability that the input triple is a fact. For a fair comparison of explanation methods, both approaches must use the same embeddings. We fix the number of dimensions to 25, and use a learning rate of 0.001 for all rules. The number of epochs used to train the RGCN varied per rule, we use between 50 and 2000, as this gave the best performance on the task of link prediction.

We use ExplaiNE [7] and GNNExplainer [15] to explain the predictions of the RGCN. Model performance is reported on the full dataset, and for each predicate subset. We report the accuracy of the RGCN as a performance metric on the task of link prediction.

ExplaiNE relies on the assumption that an optimal explanation can be found using one of the adjacent neighbors. We drop this assumption on our experiment, and allow ExplaiNE to pick any observed triple in the graph as a possible explanation candidate. Note that using the gradient of the scoring function with respect to the adjacency matrix, ExplaiNE requires no hyper-parameter tuning.

We train the GNNExplainer using a learning rate of 0.001 for each rule, which was the best performing learning rate from the set $\{0.00001, 0.0001, 0.001\}$. We use between 10 and 30 iterations for each observation. We use 3-fold cross validation for both models, and report results of the best performing fold.

5.2 Link Prediction-Results

The first section of Table 2 reports results on the Royalty-20k dataset. The topmost row reports the performance of the RGCN on the task of link prediction. We observe the highest accuracy on the *hasSuccessor* predicate, and performance dropping across each of the other predicates. Overall, we see similar results for *hasSuccessor*, and *hasPredecessor*.

The second section of Table 2 reports results on the Royalty-30k dataset. From the topmost row we can see the performance of the RGCN on the task of link prediction. We observe the highest accuracy on the *hasGrandparent* predicate, which follows a chain property. We observe the lowest performance on the *hasSpouse* predicate.

5.3 Quantitative Evaluation of Link Prediction Explanation

GNNExplainer. From Table 2, we can see the performance of GNNExplainer on the task of explainable link prediction. On both datasets, we observe its best Jaccard and F_1-Score performance on the *hasSpouse* predicate. Note that these predicates, *hasSpouse*, *hasSuccessor* and *hasPredecessor* all have an explanation cardinality of 1, meaning the ground truth explanation set contains 1 triple. On the Royalty-30k dataset, we observe performance drops on the *hasGrandparent* predicate. Recall this predicate follows a chain property and has an explanation set with 2 triples, forming a path.

Note GNNExplainer has a recall of 1 for all rules. Indeed this means GNNExplainer was able to correctly identify triples that belong to the explanation. However, the predicted explanation sets were often too large (up to 20 triples on average, for some rules), and had many false positives: a recall of 1 can be trivially achieved by including the entire input graph in the predicted explanation. This was not the case for the predicted explanations of GNNExplainer, however, the cardinality of the predicted explanation set of GNNExplainer was often larger than the ground truth cardinality.

ExplaiNE. Lastly, Table 2 reports the performance of ExplaiNE on the task of explainable link prediction. On both datasets, again

Dataset	Models	Metrics	Predicates				
			Spouse	Successor	Predecessor	Grandparent	Full set
Royalty-20k	RGCN	Accuracy	0.682	0.696	0.692	-	0.623
	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	0.682	-	-	0.713	0.621
	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.

we observe the best Jaccard and F_1-Score performance on the *hasSpouse* predicate. In general, we observe that rules with a similar explanation structure had similar performance. On the Royalty-30k dataset, we see lower relative performance on the predicates where larger explanations need to be predicted, e.g. *hasGrandparent*.

GNNExplainer vs. ExplaiNE. Overall, we find ExplaiNE outperformed GNNExplainer in terms of Jaccard score for all rules across both datasets. Additionally, we find GNNExplainer outperforms ExplaiNE in terms of F_1 score on the *hasSpouse*, and *hasGrandparent* predicate subsets, along with the Royalty-30k full dataset. This is likely due to the high recall of GNNExplainer's predicted explanations. This is evidence the F_1 score is not a good metric in that specific configuration.

5.4 Qualitative Evaluation of Link Prediction Explanation

Table 3 gives a breakdown of each explanation method's most frequent error by subset. Each row of this table can be read as follows: Under the *hasSpouse* subset for example, the most common predicate across ExplaiNE's incorrectly predicted explanations was *hasSpouse*, and this predicate was observed in 100% of errors. This error occurs when ExplaiNE predicts the wrong subject or object in the explanation. For GNNExplainer, *hasSpouse* was also the most common predicate amongst incorrectly predicted explanations, also accounting for 100% of errors. Indeed this is possible on the *hasSpouse* subset, as under this subset, there is only one possible predicate to predict (*hasSpouse*).

As an example of one of ExplaiNE's errors, for some triple (*Albert III, Count of Everstein*, *hasSpouse*,*Richeza of Poland*) and its 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.

Table 4 reports the most frequently missing predicate from ExplaiNE's errors. Each row denotes the predicate subset, the ground truth predicates defining the rule, and the percentage of triples not containing the ground truth predicate(s). For example, under the *hasSuccessor* subset of the Royalty-20k dataset, 78% of ExplaiNE's errors did not contain *hasPredecessor*. Note we do not report the most frequently missing predicate for GNNExplainer, as the recall for each subset was 1, the ground truth triple(s) were always included in the predicted explanation. Therefore, the percent missing is 0 for each subset.

The first row of Figure 2 shows histograms of predicate counts of ExplaiNE's incorrectly predicted explanations. For example, under the *hasSuccessor* subset of the Royalty-20k dataset, *hasSuccessor* was the most frequently predicted predicate amongst ExplaiNE's incorrect explanations. From this we can conclude on this subset, ExplaiNE's most frequent error occurred by predicting the wrong predicate. We also observe this phenomenon under the *hasPredecessor* subset of the Royalty-20k dataset, and the *hasGrandparent* subset of the Royalty-30k dataset. ExplaiNE incorrectly predicts explanations to have the same predicate as the input triple (the triple we want an explanation for). Furthermore, on the Royalty-20k and Royalty-30k full data, ExplaiNE's errors most frequently contained

Most Frequently Predicated Predicate					
		ExplaiNE		GNNExplainer	
Dataset	Predicate	Most Frequent Predicate	% of Error	Most Frequent Predicate	% of Error
$Royalty-20k$	*hasSpouse*	*hasSpouse*	100%	*hasSpouse*	100%
	hasSuccessor	*hasSuccessor*	78%	*hasPredecessor*	50%
	hasPredecessor	*hasPredecessor*	73%	*hasSuccessor*	50%
$Royalty-30k$	*hasSpouse*	*hasSpouse*	100%	*hasSpouse*	100%
	hasGrandparent	*hasParent*	54%	*hasParent*	50%

Table 3: Most frequent predicate across incorrectly predicted explanations, along with the percentage of error by subset.

ExplaiNE: Most Frequently Missing Predicate			
Dataset	Predicate	Ground Truth	% Missing
$Royalty-20k$	*hasSpouse*	*hasSpouse*	0%
	hasSuccessor	*hasPredecessor*	78%
	hasPredecessor	*hasSuccessor*	73%
$Royalty-30k$	*hasSpouse*	*hasSpouse*	0%
	hasGrandparent	*hasParent* *hasParent*	27% 27%

Table 4: ExplaiNE's most frequently missing predicate. Each row denotes the predicate subset, the ground truth predicates defining the rule, and the percentage of triples not containing the ground truth predicate(s)

the *hasSpouse* predicate. We can conclude that ExplaiNE's use of the gradient of the score with respect to the adjacency matrix assigns a large gradient to triples with the same predicate as the input, and to first degree neighbors of the input subject and object.

The second row of Figure 2 shows histograms of predicates counts of GNNExplainer's incorrectly predicted explanations. Under the *hasSuccessor*, *hasPredecessor* subsets, GNNExplainer's errors were uniform. Incorrectly predicting an explanation to contain either *hasSuccessor* or *hasPredecessor* predicates was equally likely. We can conclude from this that a triple with an incorrect predicate was equally likely to be predicted by GNNExplainer as a triple with the correct predicate and incorrect subject and/or object. Similar to ExplaiNE, the most frequent error on *hasGrandparent* occurred by incorrectly predicting the same predicate as the input triple. On the Royalty-30k full dataset, the majority of GNNExplainer's errors were triples using the *hasGrandparent* predicate.

5.5 Discussion

From GNNExplainer's high recall, we conclude the explanations of this method identifies all triples belonging in the explanation set. However, many irrelevant triples are also included in the predicted explanation set that shouldn't be, and often the size of the true explanation set is overestimated.

More generally, our method allows us to see that GNNExplainer and ExplaiNE do not always make the same types of mistakes: one may often choose an irrelevant relation type while the other may often pick the right type of relation but with the wrong arguments. This is useful to evaluate the impact of the choices made in Equations 1 and 2, and to propose and evaluate new methods addressing the shortcomings.

From this experiment, we can see the importance of the Royalty-20k and Royalty-30k datasets, along with the method we use to generate it. This experiment shows that state-of-the-art explanation methods do not always give accurate explanations. There are many approaches to generating explanations, however, they must be evaluated with a ground truth dataset and quantitative metric. Our method, dataset, and metric allow researchers to develop new explanation methods and quantitatively evaluate their explanations in a way they were previously unable to.

6 CONCLUSION

On the task of explainable link prediction, there is no standard dataset available to quantitatively compare explanations, as no standard method exists to generate datasets with explanations. Additionally, there is no standard evaluation metric to determine when one explanation method is preferable to the other. In this work, we propose a method, including two datasets (Royalty-20k, and Royalty-30k), to compare predicted and ground truth explanations. Furthermore, we propose the use of an evaluation metric, leveraging the Jaccard similarity between the predicted and ground truth explanation for quantitative comparisons across explanation methods. Lastly, we benchmark two state-of-the-art explanation methods, ExplaiNE and GNNExplainer, and perform a quantitative analysis on their predicted explanations using the Royalty datasets and the aformentioned evaluation metric. As a result, we are able

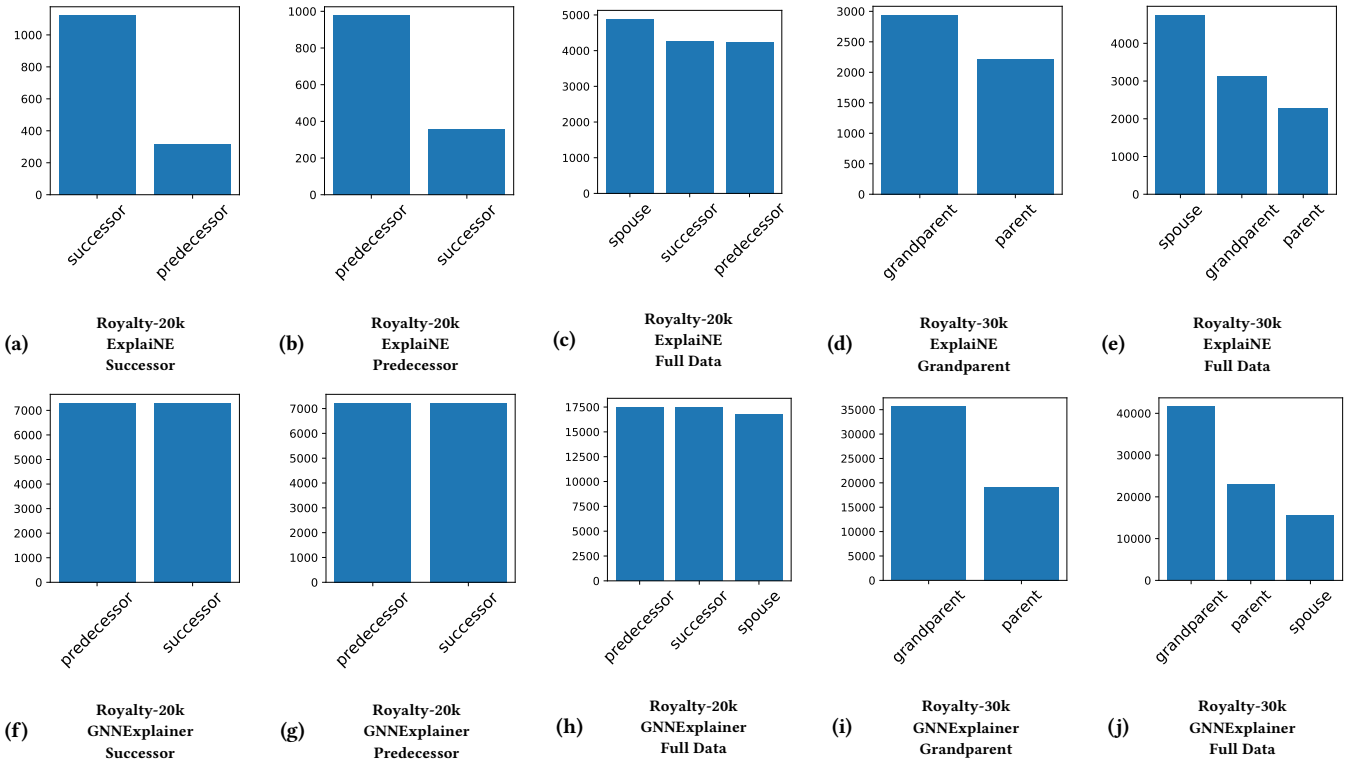

Figure 2: Predicate Frequency Count on Incorrectly Predicted Explanations. Note *hasSpouse* was omitted as only one predicate could be predicted (*hasSpouse*).

to identify and quantify the different types of errors they make in terms of both data and semantics.

This paper provides opportunities for future extensions. This could involve using the ground truth explanations to distinguish between model error and an explanation method error, i.e., determining if the RGCN is causing the explanation method (ExplaiNE/GNNExplainer) to produce incorrect explanations, or if the error is coming from the explanation method itself.

REFERENCES

[1] Olivier Corby, Alban Gaignard, Catherine Faron Zucker, and Johan Montagnat. 2012. KGRAM Versatile Inference and Query Engine for the Web of Linked Data. In *IEEE/WIC/ACM Int. Conference on Web Intelligence*.

[2] Randy Goebel, Ajay Chander, Katharina Holzinger, Freddy Lécué, Zeynep Akata, Simone Stumpf, Peter Kieseberg, and Andreas Holzinger. 2018. Explainable AI: The New 42?. In *Machine Learning and Knowledge Extraction - Second IFIP CD-MAKE (LNCS, Vol. 11015)*, Andreas Holzinger, Peter Kieseberg, A Min Tjoa, and Edgar R. Weippl (Eds.). Springer.

[3] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, et al. 2020. Knowledge graphs. *preprint arXiv:2003.02320* (2020).

[4] Matthew Horridge. 2011. *Justification based explanation in ontologies.* Ph.D. Dissertation. University of Manchester, UK.

[5] Jeya Vikranth Jeyakumar, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani B. Srivastava. 2020. How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods. In *Advances in Neural Information Processing Systems*.

[6] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2020. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. *CoRR* abs/2002.00388 (2020). arXiv:2002.00388

[7] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. 2019. ExplaiNE: An Approach for Explaining Network Embedding-based Link Predictions. *CoRR* abs/1904.12694 (2019). arXiv:1904.12694

[8] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations, ICLR*.

[9] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web* (2015).

[10] Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL*, Marti A. Hearst and Mari Ostendorf (Eds.). The Association for Computational Linguistics.

[11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Meeting of the Association for Computational Linguistics*. ACL.

[12] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *European Semantic Web Conference, ESWC*, Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam (Eds.).

[13] Benjamin Taskar, Ming Fai Wong, Pieter Abbeel, and Daphne Koller. 2003. Link Prediction in Relational Data. In *Advances in Neural Information Processing Systems*.

[14] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *International Conference on Learning Representations, ICLR*.

[15] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems*.