Evaluating and improving explanation quality of graph neural network link prediction on knowledge graphs

Nicholas Halliwell

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

THÈSE DE DOCTORAT

Évaluation et Amélioration de la Qualité d'Explication de la Prédiction des Liens par Réseau Neuronal sur les Graphes de Connaissances

Nicholas HALLIWELL

Inria, CNRS, I3S

Présentée en vue de l'obtention du grade de docteur en Informatique **d'**Université Côte d'Azur
Dirigée par : Fabien Gandon
Soutenue le : 23.11.2022

Devant le jury, composé de :
Adrien Coulet, CR, HDR, Inria
Fabien Gandon, DR, Inria
Freddy Lecue, Dr., Inria, Thales CortAIx
Fatiha Saïs, PR, Université Paris Saclay
Andrea Tettamanzi, PR, Université Côte d'Azur

Évaluation et Amélioration de la Qualité d'Explication de la Prédiction des Liens par Réseau Neuronal sur les Graphes de Connaissances

COMPOSITION DU JURY

Rapporteurs

Adrien Coulet, CR, HDR, Inria

Fatiha Saïs, PR, Université Paris Saclay

Examinateurs

Andrea Tettamanzi, PR, Université Côte d'Azur

Directeur de thèse

Fabien Gandon, DR, Inria, Université Côte d'Azur, CNRS, I3S

Invités

Freddy Lecue, Dr., Inria, Thales CortAIx

RÉSUMÉ

Les graphes de connaissances contiennent une collection de faits sur monde réel, où les nœuds représentent des entités liées par certaines relations. Ils sont utilisés dans de nombreuses applications, notamment la recherche d'informations, l'intégration de données ou les agents conversationneles. Cependant, les graphes de connaissances sont souvent incomplets. La prédiction de liens sur les graphes de connaissances est alors utilisée pour inférer de nouveaux faits à partir de ceux existants. Plusieurs modèles existent pour effectuer la prédiction de liens sur les graphes de connaissances. Une méthode actuellement étudiée est l'utilisation de réseaux convolutifs de graphes relationnels (RGCNs). Mais ces modèles sont essentiellement traités comme des boîtes noires, où aucune information n'est donnée à l'utilisateur sur la raison pour laquelle le modèle a fait une prédiction particulière.

Souvent, la prédiction seule n'est pas suffisante pour les utilisateurs du modèle qui doivent prendre des décisions. Sans explication sur la raison pour laquelle le modèle a fait une prédiction, ces modèles ne peuvent pas être adoptés dans des contextes réels tels que la banque, le marketing et le diagnostic médical. Récemment, des chercheurs ont proposé plusieurs algorithmes, ou méthodes, pour expliquer pourquoi un modèle de prédiction de liens RGCN a pris une décision particulière. Ces méthodes d'explication sont appliquées à posteriori à un modèle de prédiction. Mais, les différentes méthodes d'explication ne renvoient pas toujours la même explication pour la même entrée. De plus, plusieurs méthodes d'explication existent pour expliquer les résultats des RGCN pour la prédiction de liens sur les graphes de connaissances. Le principal inconvénient de ces méthodes d'explication est l'évaluation empirique de la qualité de l'explication. Il n'existe aucun ensemble de données standard pour comparer quantitativement les explications entre les méthodes d'explication. De plus, il n'existe aucune mesure d'évaluation standard pour quantifier la qualité de

l'explication donnée à l'utilisateur. Cela rend les comparaisons entre les méthodes d'explication difficiles.

Dans cette thèse, nous proposons une méthode permettant de générer plusieurs ensembles de données avec une vérité terrain pour les explications afin d'évaluer quantitativement la qualité des explications produites par les méthodes d'explication post hoc pour les RGCNs. De plus, nous proposons plusieurs mesures de notation pour quantifier la qualité de l'explication donnée à l'utilisateur. Les résultats du benchmark de plusieurs méthodes d'explication de pointe montrent que ces méthodes ne produisent souvent pas d'explications de haute qualité. Nous proposons un RGCN qui incorpore les explications de la vérité terrain dans l'intégration des graphes. Nous constatons cependant que la qualité des explications reste faible, en particulier pour les explications longues. Enfin, nous proposons un modèle Séquence-à-Séquence qui apprend à générer des explications pour un RGCN. Nous constatons des améliorations significatives de la qualité des explications par rapport aux méthodes d'explication de l'état de l'art, en particulier pour les explications plus longues.

Mots-clés

graphe de connaissances, apprentissage automatique, évaluation des explications, prédiction de liens, Réseaux neuronaux de graphes, modèles de séquence à séquence

ABSTRACT

Knowledge Graphs contain a collection of real world facts, where nodes represent entities linked by some relationship. They are used in many applications, including information retrieval, data integration, and chat-bots. Knowledge Graphs are often incomplete. Link prediction on Knowledge Graphs is used to infer new facts from existing ones. Several models exist to perform link prediction on Knowledge Graphs, one popular method is the use of Relational Graph Convolutional Networks (RGCNs). These models are treated as a black box, where no information is given to the user as to why the model has made a particular decision.

Often the prediction alone is not enough for users of the model who need to make decisions. Without an explanation as to why the model has made a prediction, these models cannot be adopted into real world settings such as banking, marketing, and medical diagnosis. Recently researchers have proposed several algorithms, or explanation methods, to explain why a black box link prediction model has made a particular decision. These explanation methods are applied to a model post hoc. The different explanation methods do not always return the same explanation for the same input. Moreover, multiple explanation methods exist to explain the results of RGCNs for link prediction on Knowledge Graphs. The main drawback of these explanation methods is the empirical evaluation of explanation quality. No standard dataset exists to quantitatively compare explanations across explanation methods. Additionally, no standard evaluation metrics exist to quantify the quality of explanation given to the user. This makes comparisons across explanation methods difficult.

In this thesis, we propose a method to generate several datasets with ground truth explanations to quantitatively evaluate the quality of explanation produced by post hoc explanation methods for RGCNs. Additionally, we propose several scoring metrics to quantify the quality of explanation given to the user. Benchmark results of several

state-of-the-art explanation methods show that these methods often do not produce high quality explanations. We propose an RGCN that incorporates ground truth explanations into the graph embeddings. We find however that explanation quality was still low, especially on longer explanations. Lastly, we propose a Sequence-to-Sequence model that learns to generate RGCN explanations. We find significant improvements of explanation quality over state-of-the-art explanation methods, particularly on longer explanations.

Keywords

knowledge graph, machine learning, explanation evaluation, link prediction, graph neural networks, sequence to sequence models

ACKNOWLEDGMENTS

First I would like to thank my supervisors, Fabien Gandon and Freddy Lecue, for their advice, patience, and support throughout the duration of this thesis. Thank you Fabien, for your guidance, and helping me make my research the very best it can be. Your dedication throughout this thesis has shaped me into the researcher I am today. Thank you Freddy, for your mentorship, and providing me the opportunity to intern on your team at CortAIx in Montreal. This experience had a big impact on my research career, and will be invaluable going forward.

I would like to express gratitude to each member of the jury, Andrea Tettamanzi, Adrien Coulet, and Fatiha Saïs for taking the time to review this thesis. I would like to thank Serena Villata, for bringing me all the way to France, and for mentoring me at the start of my research career.

I am thankful for every member of the WIMMICS team I have met since I joined in July 2018. I would also like to thank the team developing CORESE, and the team supporting the NEF cluster. This thesis would not have been possible without them.

I would also like to thank the every member of the CortAIx team at Thales-Montreal that I was lucky enough to work alongside. I thank Damien Dalla Rosa for answering my numerous Tensorflow questions, and for our many discussions involving hamburgers.

I thank Gaël Varoquaux for your mentorship. Your tutelage taught me invaluable lessons about how to be a researcher.

Lastly, I thank my parents for their constant support, and my brother, Derek, who will now have to address me as “Dr. Halliwell.”

Contents

Page

List of Tables

List of Figures

Chapter 1

INTRODUCTION

1.1 On the need to explain automated decisions

Deep learning models are used to serve automated decisions in applications such as fraud detection Dhankhad et al. [2018], Randhawa et al. [2018], credit scoring Dumitrescu et al. [2022], Lessmann et al. [2015], Henley and Hand [1997], and medical diagnosis Patrício et al. [2022], Choudhury and Gupta [2019], Varoquaux and Cheplygina [2022]. Users receiving an automated decision such as hospital patients, credit applicants, and practitioners debugging the model want to know why the model preferred one outcome over another. No user is willing to receive a medical diagnosis from an automated system unless the outcome is explained and justified.

In 2016, Defense Advanced Research Projects Agency (DARPA) launched an explainable AI initiative calling on researchers to create machine learning techniques that produce more explainable models, where humans could understand and trust their rationale. In 2018, the European Union enacted the General Data Protection Regulation (GDPR) to give individuals protective rights to their personal data, including a "right to explanation" of automated decisions.

Deep learning models can be applied to many tasks, including image classification, machine translation, and link prediction. These models are too often treated as black boxes, where no insight is given as to how they make decisions. Recently, researchers have proposed algorithms, or explanation methods, that explain to the user why a model makes a given decision. The weak point of these explanation methods is the empirical evaluation of explanations returned to the user.

1.2 Evaluating explanations

In many domains, there is no benchmark to quantitatively evaluate the quality of explanation produced by the model. Current state-of-the-art explanation methods do not use any common datasets or scoring metrics to quantitatively evaluate explanation quality. Consequently, it is difficult to determine if an explanation method is producing accurate explanations, and when to prefer one method over another. One solution is to create a standard dataset with ground truth explanations, which would allow researchers to develop standard scoring metrics that could be used to evaluate the explanations from state-of-the-art explanation methods.

Ground truth explanations can be difficult to define for some domains. Even if they could be defined, there can be more than one way to explain the prediction of a black box model. Some natural questions stem from this idea of defining ground truth explanations:

- How do current state-of-the-art explanation methods perform when there is only one ground truth explanation to choose from? That is, does one explanation method produce more accurate explanations when each observation has one and only one ground truth explanation?

- How do current state-of-the-art explanation methods perform when there are multiple ground truth explanations to choose from? In other words, does one explanation method produce more accurate explanations when each observation has more than one ground truth explanation to choose from?

- Can ground truth explanations be used during training to improve the quality of post hoc explanations?

- Lastly, can an explanation method be developed that outperforms current state-

of-the-art explanation methods?

1.3 Contributions of this thesis

This thesis is outlined as follows; Chapter 2 provides an overview of all related work used in this thesis. Chapter 3 justifies the need for ground truth explanations, and we discuss the difficulties that arise when measuring explanation quality. In Chapter 4, we construct two datasets where each observation has one and only one ground truth explanation. We then benchmark state-of-the-art explanation methods on these datasets. In Chapter 5, we construct a dataset where each observation has multiple ground truth explanations. We then benchmark state-of-the-art explanation methods on this dataset. Chapter 6 proposes to train a model using information from ground truth explanations. Empirical results show improved explanation prediction performance for several explanation methods. In Chapter 7, we propose a sequence model that generates explanations, and propose sanity checks to verify what the model has learned. Empirical results show improved performance over state-of-the-art, specifically on explanations where previous approaches struggled. Lastly, Chapter 8 concludes the thesis and provides opportunities for future work.

1.4 Published results

The results of this thesis were published in several international venues:

- N. Halliwell, F. Gandon, and F. Lecue. A Simplified Benchmark for Non-ambiguous Explanations of Knowledge Graph Link Prediction using Relational Graph Convolutional Networks. International Semantic Web Conference, Oct. 2021a. URL `https://hal.archives-ouvertes.fr/hal-03339562`. Poster

- N. Halliwell, F. Gandon, and F. Lecue. Linked Data Ground Truth for Quantitative and Qualitative Evaluation of Explanations for Relational Graph Con-

volutional Network Link Prediction on Knowledge Graphs. In *International Conference on Web Intelligence and Intelligent Agent Technology*, Melbourne, Australia, Dec. 2021c. doi: 10.1145/3486622.3493921. URL `https://hal.archives-ouvertes.fr/hal-03430113`

- N. Halliwell, F. Gandon, and F. Lecue. User Scored Evaluation of Non-Unique Explanations for Relational Graph Convolutional Network Link Prediction on Knowledge Graphs. In *International Conference on Knowledge Capture*, Virtual Event, United States, Dec. 2021b. doi: 10.1145/3460210.3493557. URL `https://hal.archives-ouvertes.fr/hal-03402766`

- N. Halliwell, F. Gandon, and F. Lecue. A Simplified Benchmark for Ambiguous Explanations of Knowledge Graph Link Prediction using Relational Graph Convolutional Networks. 36th AAAI Conference on Artificial Intelligence, Feb. 2022a. URL `https://hal.archives-ouvertes.fr/hal-03434544`. Poster

- N. Halliwell. Evaluating Explanations of Relational Graph Convolutional Network Link Predictions on Knowledge Graphs. In *AAAI 2022 - 36th AAAI Conference on Artificial Intelligence*, Vancouver, Canada, Feb. 2022. URL `https://hal.archives-ouvertes.fr/hal-03454121`

- N. Halliwell, F. Gandon, F. Lecue, and S. Villata. The Need for Empirical Evaluation of Explanation Quality. In *AAAI 2022 - Workshop on Explainable Agency in Artificial Intelligence*, Vancouver, Canada, Feb. 2022c. URL `https://hal.archives-ouvertes.fr/hal-03591012`

- N. Halliwell, F. Gandon, and F. Lecue. Impact of Injecting Ground Truth Explanations on Relational Graph Convolutional Networks and their Explanation Methods for Link Prediction on Knowledge Graphs. In *WI-IAT 2022 - The*

21st IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Niagara Falls / Hybrid, Canada, Nov. 2022b. URL https://hal.archives-ouvertes.fr/hal-03771424

Chapter 2

RELATED WORK

This chapter outlines related work and concepts used throughout the thesis, including the type of data used, Knowledge Graphs, the machine learning task used in the thesis, link prediction, the algorithms used to perform the link prediction task, Deep Learning, and the algorithms used to explain the predictions of Deep Learning models, and black box explanation methods.

2.1 Knowledge Graphs

Knowledge Graphs (KGs) Hogan et al. [2020], Ji et al. [2020] are used on tasks such as search engine enhancement, question answering, and product recommendation. Knowledge Graphs represent facts as triples in the form *(subject, predicate, object)*, where a *subject* and *object* represent a real-world entity, linked by some *predicate*. In other words, KGs represent nodes (entities) in a graph, where the links connecting the nodes are not necessarily the same. Resource Description Framework (RDF) Cyganiak et al. [2014] defines a graph based Linked Data model to represent KGs on the Web. The Semantic Web uses several standards, including RDF, to link data found on the Web. Linked Data Berners-Lee [2006], is a specific type of KG following a set of principles for publishing the data on the Web, ensuring the KG is machine and human readable, freely accessible, and non-proprietary.

2.2 Link Prediction

Knowledge Graphs often do not explicitly contain every available fact. Link prediction on Knowledge Graphs is used to identify unknown facts from existing ones.

Recently, researchers have proposed the use of graph embeddings Yang et al. [2015], Bordes et al. [2013], Wang et al. [2014], Trouillon et al. [2016], Nickel et al. [2016] for link prediction on Knowledge Graphs. 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).

Rule based link prediction approaches Barati et al. [2017], Galárraga et al. [2013, 2015], Muggleton [1995], Ott et al. [2021] can also be used on Knowledge Graphs, where logical rules are extracted by the model. This thesis focuses on Graph Neural Network based link prediction models, detailed below. For more details on link prediction methods, we refer the reader to a recent survey Wang et al. [2017].

Another common approach to link prediction on Knowledge Graphs involves the use of Graph Neural Networks (GNNs) such as Graph Convolutional Networks Kipf and Welling [2017] (GCNs) or Relational Graph Convolutional Networks Schlichtkrull et al. [2018] (RGCNs) that learn a function mapping each subject, object, and predicate to a low-dimensional space. This thesis focuses on the use of RGCNs for link prediction on Knowledge Graphs.

2.3 Deep Learning

2.3.1 Feedforward Neural Network

In general, deep learning models (neural networks) perform supervised learning, where some function f is learned to map a set of inputs $\mathbf{X} \in \mathbb{R}^{n \times m}$ to a set of labels $\mathbf{y} \in \mathbb{R}^n$. Typically f is parameterized by some set θ, and an optimal set of parameters θ^* are found by minimizing a loss function $\mathcal{L}$, hence $\theta^* = \underset{\theta}{\operatorname{argmin}}\ \mathcal{L}(f(\mathbf{X}, \theta), \mathbf{y})$. For neural networks, a given parameter $\theta_i \in \theta$ corresponds to one node (neuron) in the network.

The most common form of neural network consists of a sequence of consecutive layers, where the output of the previous layer is passed as input into the next layer. Neural network layers can fall into one of three categories; input, hidden, and output layers. Input layers take training data as input, and pass this data through to the next layer. Hidden layers take the previous layer as input, and perform a nonlinear transformation. Lastly, output layers, typically consist of softmax layer, which takes the nonlinear representation from the hidden layer as input, and converts this representation into a probability distribution across each possible label. Neural networks with many hidden layers are termed deep neural networks.

A common type of neural network layer is a fully connected layer, where every neuron in the current layer connects to a neuron the next layer. Formally, a fully connected layer i is defined by $\mathbf{a}_i = \phi(\mathbf{W}_i\mathbf{x}_{i-1} + \mathbf{b}_i)$, where $\mathbf{a}_i \in \mathbb{R}^a$ is the output vector with dimensionality a, $\mathbf{W}_i \in \mathbb{R}^{a \times b}$ is the parameter matrix for layer i, $\mathbf{x}_{i-1} \in \mathbb{R}^b$ is the output from the previous layer with dimensionality b, $\mathbf{b}_i$ is the bias, and ϕ is a non-linear activation function such as a ReLu, sigmoid, hyperbolic tangent, etc. These models are typically optimized using Stochastic Gradient Descent, or one of its variants Duchi et al. [2011], Zeiler [2012], Kingma and Ba [2015]. Neural network with only fully connected layers as hidden layers are termed feedforward neural networks.

2.3.2 Prototype Network

Prototype networks Li et al. [2018] are a specific type of neural network architecture that learn "prototype" vectors, representing typical observations from the training data. These prototype vectors serve as explanations for why the model made a decision. The Prototype network architecture can be visualized in Figure 2.1. It consists of an autoencoder (the encoder defined as $f : \mathbb{R}^p \to \mathbb{R}^q$ and the decoder, defined as $g : \mathbb{R}^q \to \mathbb{R}^p$), a prototype layer $p : \mathbb{R}^q \to \mathbb{R}^m$, and a dense (fully-connected)

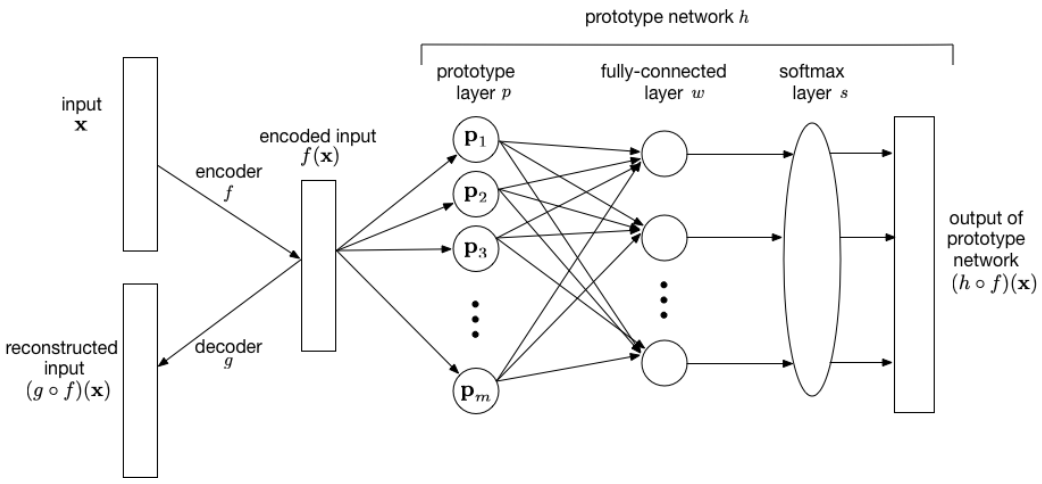


Figure 2.1: Prototype Network Architecture Li et al. [2018].

layer $w : \mathbb{R}^m \to \mathbb{R}^K$ that feeds into a softmax layer. The prototype layer takes as input encoded training points, denoted $f(\mathbf{x}_i)$, and computes the L^2 distance between $f(\mathbf{x}_i)$ and m prototype vectors, denoted $\mathbf{p}_1, \ldots, \mathbf{p}_m \in \mathbb{R}^q$. The overall network is given by $h : \mathbb{R}^q \to \mathbb{R}^K$. In this prototype network architecture, observations are classified based on their distance to a prototypical observation, and the loss function ensures that each prototype vector is similar to an encoded training point. We denote the data set $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $y_i \in \{1, \ldots K\}$, and K being the number of classes.

The Prototype loss function Li et al. [2018] is broken down into the following four parts below. First, $E(h \circ f,\ D)$ (Equation 2.1) penalizes misclassified observations, and $R(g \circ f,\ D)$ (Equation 2.2) is the reconstruction error of the autoencoder.

$$E(h \circ f,\ D) = \frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} -\mathbb{1}[y_i = k] log((h \circ f)_k(\mathbf{x}_i)), \tag{2.1}$$

$$R(g \circ f,\ D) = \frac{1}{n} \sum_{i=1}^{n} ||(g \circ f)(\mathbf{x}_i) - \mathbf{x}_i||_2, \tag{2.2}$$

Two regularization terms are used, i.e., R_1 (Equation 2.3), which forces each prototype vector to be as close as possible to one encoded training point, and R_2 (Equation 2.4), which forces every encoded training point to be as close as possible to one prototype vector.

$$R_1(\mathbf{p}_1, \ldots, \mathbf{p}_m,\ D) = \frac{1}{m} \sum_{j=1}^{m} \min_{i \in [1,n]} \|\mathbf{p}_j - f(\mathbf{x}_i)\|_2, \tag{2.3}$$

$$R_2(\mathbf{p}_1, \ldots, \mathbf{p}_m,\ D) = \frac{1}{n} \sum_{i=1}^{n} \min_{j \in [1,m]} \|f(\mathbf{x}_i) - \mathbf{p}_j\|_2. \tag{2.4}$$

The complete loss function is given by

$$\begin{aligned} L((f, g, h), D) = E(h \circ f,\ D) + \lambda_0 R(g \circ f,\ D) + \\ \lambda_1 R_1(\mathbf{p}_1, \ldots, \mathbf{p}_m,\ D) + \lambda_2 R_2(\mathbf{p}_1, \ldots, \mathbf{p}_m,\ D), \end{aligned} \tag{2.5}$$

where λ_0, λ_1, λ_2 are hyperparameters.

2.3.3 Long short-term memory

Indeed fully connected layers are not the only type of layer used in deep learning models. For data such as stock prices, speech, or text, the order of the observations must be considered, as there is correlation across observations. Feedforward layers cannot model data with time dependencies. Recurrent neural networks (RNNs) were developed to model sequential data by sharing weight matricies across time steps. These models struggled with the vanishing gradient problem, where the gradient of some parameters can become too small, preventing the parameter value from being updated. Additionally, RNNs suffer from the exploding gradient problem, where the gradient of some parameters become too large, resulting in very large

parameter updates, and in some cases numerical overflow. Long short term memory (LSTM) Hochreiter and Schmidhuber [1997] layers were developed to combat theses issues. An LSTM uses an input gate, output gate, forget gate, and memory cell to determine what information to preserve across time steps. Formally, the forward pass of the LSTM is defined by

$$\mathbf{i}_t = \sigma_g(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_u\mathbf{h}_{t-1} + \mathbf{b}_i), \tag{2.6}$$

$$\mathbf{o}_t = \sigma_g(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o), \tag{2.7}$$

$$\mathbf{f}_t = \sigma_g(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f), \tag{2.8}$$

$$\tilde{\mathbf{c}}_t = \sigma_c(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c), \tag{2.9}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \tag{2.10}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \sigma_c(\mathbf{c}_t), \tag{2.11}$$

where $\mathbf{x}_t \in \mathbb{R}^d$ is the input vector to the LSTM (with input dimensionality d), $\mathbf{i}_t \in (0,1)^h$ is the input gate's output vector, $\mathbf{o}_t \in (0,1)^h$ is the output gate's output vector, $\mathbf{f}_t \in (0,1)^h$ is the forget gate's output vector (using h hidden units), $\tilde{\mathbf{c}}_t \in (-1,1)^h$ is the input to the cell state, $\mathbf{c}_t \in \mathbb{R}^h$ is the cell state's output vector, σ_g denotes the sigmoid function, σ_c denotes the hyperbolic tangent function, $\odot$ denotes the element-wise product, and lastly, $\mathbf{h}_t \in (-1,1)^h$ is the hidden state's output vector (with dimensionality h). Figure 2.2 shows the structure of an LSTM cell,

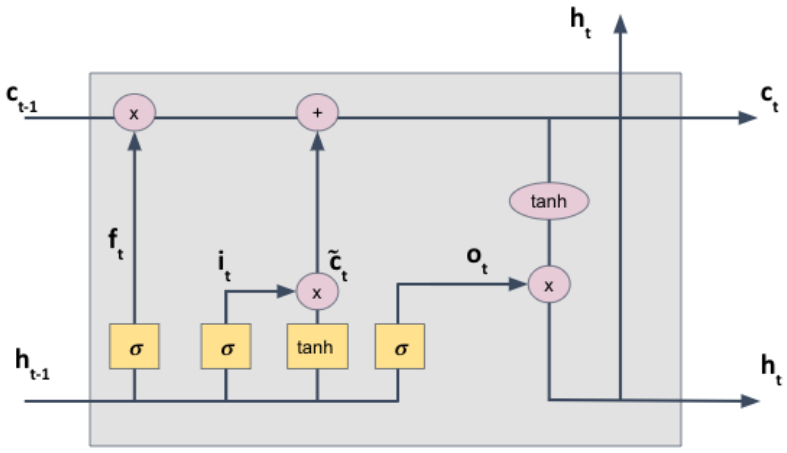


Figure 2.2: Structure of LSTM cell Hochreiter and Schmidhuber [1997]

including the input gate (Equation 2.6), output gate (Equation 2.7), and the forget gate (Equation 2.8), and outputting the cell state (Equation 2.10), and hidden state (Equation 2.11).

2.3.4 Sequence to Sequence Models

In the field of Natural Language Processing (NLP), Sequence to Sequence models (Seq2Seq) Sutskever et al. [2014] are commonly used for tasks such as Neural Machine Translation (NMT) Bahdanau et al. [2015], Cho et al. [2014]. Seq2Seq models consist of an encoder-decoder architecture, where the encoder takes a source sequence (typically a sentence) as input, and passes this into the decoder, which outputs a translated sequence. A Seq2Seq model typically uses two LSTMs, an encoder for the input data, and a decoder to generate a prediction at each time step. An example of this architecture can be seen in Figure 2.3. For English to French translation, the encoder takes as input an English sentence "The cat is black." The output of the encoder is passed into the decoder, and the model is trained to output the target French sentence "Le chat est noir." A Seq2Seq model computes a conditional probability of

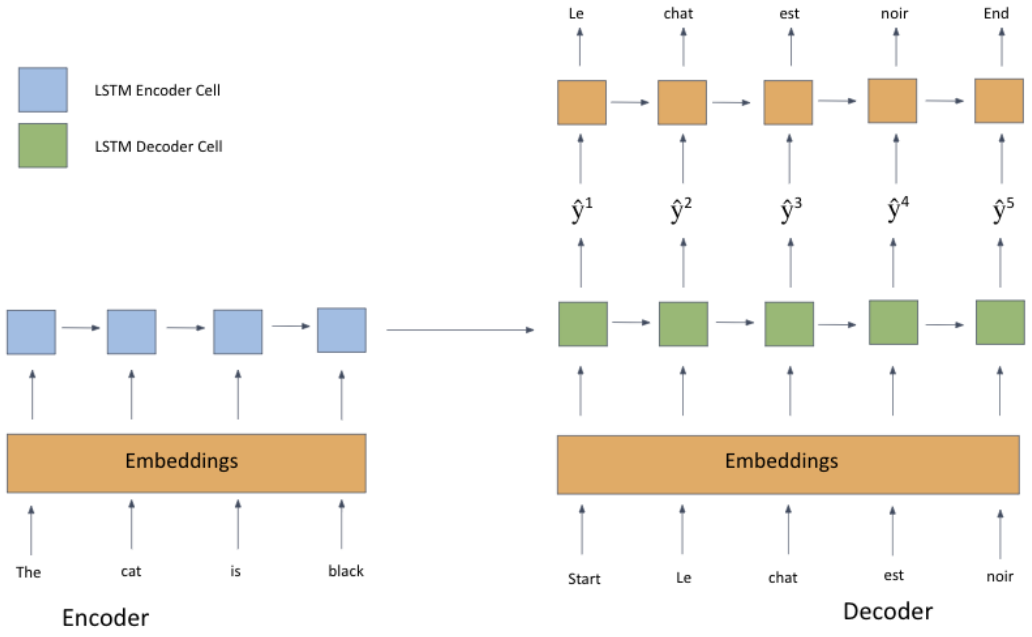


Figure 2.3: Seq2Seq model Sutskever et al. [2014] for Neural Machine Translation.

the sequence $y^1, ..., y^T$ with T time steps given the input X, i.e. $P(y^1, ..., y^T|X)$. In this example, the output of the decoder at each time step is a predicted word in the sequence.

2.3.5 Graph Convolutional Networks

Graph Convolutional Networks (GCNs) Kipf and Welling [2017] are an extension of neural networks to graph data. These models are commonly used for tasks such as graph classification, node classification, and link prediction. Generally, GCNs learn an embedding representation for each node in the graph, and update the embedding of each node by aggregating features of its neighbors. Different variants of aggregation are used to learn properties of the graph. The standard GCNs are not intended to be used on Knowledge Graphs, as they do not take into account the different types of links.

RGCNs Schlichtkrull et al. [2018] have recently been proposed to learn embeddings

and perform link prediction on Knowledge Graphs. For any given entity, RGCNs performs embedding updates by multiplying neighboring entity embeddings 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. Formally, for some entity v_i, the embedding representation $\mathbf{h}$ for some layer $l+1$ is given by

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\mathbf{W}_r^{(l)}\mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)}\right), \tag{2.12}$$

where $r \in \mathcal{R}$ is a relation from the set of relations, and $\mathcal{N}_i^r$ denotes the set of neighbors of entity i. The weight matrix learned for relation r is denoted $\mathbf{W}_r$, the weight matrix learned for self connections is denoted $\mathbf{W}_0$, and $c_{i,r}$ is a problem-specific normalizing constant.

This layer is used in the neural network to update embeddings for each entity. In order to perform link prediction, Schlichtkrull et al. [2018] use a DistMult layer Yang et al. [2015] as the succeeding layer, serving as a scoring function. For some triple (s,r,o) whose embeddings have been updated by the RGCN layer, the DistMult layer predicts the probability that this triple is a fact. Formally, the DistMult layer is defined by

$$f(s,r,o) = \mathbf{e}_s^T\mathbf{R}_r\mathbf{e}_o, \tag{2.13}$$

where $\mathbf{e}_s$ and $\mathbf{e}_o$ are the updated embeddings for the subject and object respectively, $\mathbf{R}_r$ is a diagonal parameter matrix for relation r, and f is the RGCN model. In this work, we use the term RGCN to describe a neural network with an RGCN layer and DistMult layer. The RGCN uses the cross entropy loss function, which is given by

$$\mathcal{L}_{RGCN} = -\frac{1}{(1+\omega)|\hat{\mathcal{E}}|} \sum_{(s,p,o,y)\in\mathcal{T}} ylog\big(f(s,p,o)\big) + (1-y)log\big(1-f(s,p,o)\big), \qquad (2.14)$$

where $\mathcal{T}$ is the set of all real (positive) and corrupted (negative) triples, ω is the number of negative triples, $|\hat{\mathcal{E}}|$ is the number of unique predicates, f is the function learned by the RGCN, and for positive triples, the label $y = 1$ for positive triples and $y = 0$ for negative triples.

2.4 Explanation Methods

Deep learning models are typically treated as a black box, where no explanation is given to the user to describe the why the model made a particular decision. This lack of transparency has hindered adoption of these models into real world settings. Recently, researchers have proposed algorithms, or explanation methods, to explain the predictions of these black box models. Explanation methods are typically applied to a model post hoc, and explanations are given to users of the deep learning system, practitioners implementing and debugging the models, or any anyone wanting an understanding of how the black box model makes decisions.

Indeed there are many approaches for producing post hoc explanations. Feature importance methods Lundberg and Lee [2017], Ribeiro et al. [2016, 2018] are commonly used on tabular datasets, where relevant input dimensions are identified and assigned a score to rank its importance relative to the other dimensions. For image data, saliency maps Simonyan et al. [2014], Springenberg et al. [2015], Bach et al. [2015], Selvaraju et al. [2016], Shrikumar et al. [2017, 2016], Zeiler and Fergus [2014], Smilkov et al. [2017], Sundararajan et al. [2017], Montavon et al. [2017] identify relevant pixels in the input image. Counterfactual explanations Wachter et al. [2017] give the smallest possible perturbation to the given input that will change the prediction

Explanation Method	Data type			
	Image	Tabular	Graph	Text
Lundberg and Lee [2017]	X	X	-	X
Ribeiro et al. [2016]	X	X	-	X
Ribeiro et al. [2018]	X	X	-	X
Simonyan et al. [2014]	X	-	-	-
Springenberg et al. [2015]	X	-	-	-
Bach et al. [2015]	X	-	-	-
Selvaraju et al. [2016]	X	-	-	-
Shrikumar et al. [2017]	X	-	-	-
Shrikumar et al. [2016]	X	-	-	-
Zeiler and Fergus [2014]	X	-	-	-
Smilkov et al. [2017]	X	-	-	-
Sundararajan et al. [2017]	X	-	-	-
Montavon et al. [2017]	X	-	-	-
Wachter et al. [2017]	-	X	-	-
Ming et al. [2019]	-	X	-	X
Chen et al. [2019]	X	-	-	-
Li et al. [2018]	X	X	-	-
Kang et al. [2019]	-	-	X	-
Ying et al. [2019]	-	-	X	-

Table 2.1: Breakdown of popular explanation methods for different types of data

to a desired target outcome. Counterfactual explanations are commonly used when users receiving an explanation want to know which attributes to change in order to be given a particular target outcome. Lastly, prototype explanations Chen et al. [2019], Li et al. [2018], Ming et al. [2019] learn a continuous vector that represents a "typical" training example, where explanations are given based on their relative distance to a prototype vector. Table 2.1 outlines all of the aforementioned explanation methods based on their applications. This thesis focuses on explanation methods designed to explain the predictions of RGCNs post hoc, ExplaiNE Kang et al. [2019], and GNNExplainer Ying et al. [2019], detailed below.

2.4.1 ExplaiNE

For some model with scoring function g, ExplaiNE Kang et al. [2019] computes the gradient of g with respect to each element of the adjacency matrix. In our case, the scoring function g is defined as the function learned by the RGCN model, and the adjacency matrix is a binary matrix that specifies which nodes are neighboring. The gradient quantifies then the change in score due to a small perturbation in the adjacency matrix, that is, how much will the score (i.e. RGCN predictions) change, if a link is added or removed between two given nodes. Given two nodes i, j serving as candidate predictions, and two nodes k, l serving as a candidate explanation, the score assigned to node pair k, l is given by

$$\frac{\partial g_{ij}}{\partial a_{kl}}(\mathbf{A}) = \nabla_{\mathbf{X}} g_{ij}(\mathbf{X}^*)^T \cdot \frac{\partial \mathbf{X}^*}{\partial a_{kl}}(\mathbf{A}), \tag{2.15}$$

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

2.4.2 GNNExplainer

GNNExplainer Ying et al. [2019] 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 given by

$$\min_{\mathbf{M}} - \sum_{c=1}^{C} \mathbb{1}[y = c] \, log P_{\Phi}(Y = y | \mathbf{A}_c \odot \sigma(\mathbf{M}), \mathbf{X}_c), \tag{2.16}$$

where Φ is the GNN model, $\mathbf{M}$ is a mask learned, $\odot$ denotes element-wise multiplication, $\mathbf{A}_c$ is the adjacency matrix, $\mathbf{X}_c$ is the feature matrix, and σ is the sigmoid

function. Lastly, P_Φ indicates the probability of nodes belonging to each of the C classes.

2.4.3 Datasets and metrics used for RGCN explanations evaluation

The original authors of GNNExplainer perform experiments on several synthetic datasets, along with the MUTAG Debnath et al. [1991] and REDDIT-BINARY Yanardag and Vishwanathan [2015] datasets for the task of node classification. For this task, the authors measure explanation quality using accuracy, that is, the average number of times the explanation method correctly predicted the correct edge in the ground truth explanation. This scoring metric is limited to the task of node classification, and cannot be used for tasks such as link prediction on Knowledge Graphs. At the time of writing this thesis, we are not aware of any papers benchmarking GNNExplainer on the task of link prediction on Knowledge Graphs.

The authors of ExplaiNE perform experiments on four datasets, Karate Zachary [1977], DBLP Tang et al. [2008], MovieLens Harper and Konstan [2016], and Games of Thrones [1] . These datasets do not include ground truth explanations, and the scoring metrics used rely on the assumption that good explanations for some node i are derived from first degree neighbors of i. Different scoring metrics are defined for several datasets, as assumptions made on the MovieLens dataset do not generalize to the DBLP, Karate, and Game of Thrones datasets.

There exists no standard datasets or scoring metrics to quantitatively compare the quality of explanations from explanation methods for link prediction on Knowledge Graphs using Graph Neural Networks. In this thesis, we choose to explain the predictions of an RGCN, as it can be used with multiple explanation methods without the need for any further adaptations. GNNExplainer is only defined for Graph Neural

[1] `https://github.com/mathbeveridge/asoiaf`

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.

In the next chapter, we justify the need for ground truth explanations.

Chapter 3

MOTIVATING THE NEED FOR GROUND TRUTH EXPLANATIONS

Prototype networks Li et al. [2018] provide explanations to users using a prototype vector; that is, a vector learned by the network representing a "typical" observation. In this chapter, we propose an approach that identifies relevant features in the input space used by the Prototype network. We find however that empirical evaluation of explanation quality is difficult, as there is no consistent way to determine if the predicted explanations are accurate without ground truth explanations. We include a discussion about developing methods for generating explanations, identifying when one explanation method is preferable to another, and the complications that arise when measuring explanation quality.

3.1 Introducing a Post Hoc Explanation Approach for Prototype Networks

The Prototype network architecture Li et al. [2018] combines an autoencoder with a prototype layer, where each observation in the training set is classified based on its distance to a prototype vector. The encoded input from the autoencoder is used as features for predictions downstream. The prototype vectors learned by this network are defined as typical observations in the training set, and, because they are learned in the same space as the encoded input, they can be mapped back into the original input space for visualization using the decoder. Explanations are given in the form of a most similar prototype vector. The specific architecture of this network allows us to further develop and improve the types of explanations generated post hoc.

In this chapter, we expand the type of explanations generated by the Prototype network to identify relevant features in the input space. Due to the architecture of this

network, the latent features learned by the model can be exploited to identify relevant input space features. We make use of the network's encoded input by randomly setting latent features to zero, and using the network's decoder to determine which input space values changed the most. Finally, this chapter allows us to open a general discussion about generating explanations, identifying when one explanation method is preferable to another, and the complications that arise when measuring explanation quality.

3.2 Proposed Approach: Leveraging Prototype Networks for Post Hoc Explanations

The encoder function f maps a p dimensional vector to a q dimensional vector where $p > q$. This encoded input contains relevant information for classification, as it is used as features downstream, and is using a lower dimensional representation of the input data. Identifying relevant information in the encoded latent space should provide further insight into how the model is making decisions. For some observation $\mathbf{x}$ we want an explanation for, we encode the input using the Prototype network's encoder f. We then make m copies of the encoded input $f(\mathbf{x})$, and apply m different masks element-wise. Each mask, denoted $\mathbf{m}_i$, is the same dimensions as the encoded input $f(\mathbf{x})$, where each element of a mask is assigned a 1 with 90% probability and a 0 with 10% probability. The element-wise product is then averaged across the m masks, given by

$$\hat{f}(\mathbf{x}) = \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}_i) \odot \mathbf{m}_i. \tag{3.1}$$

The result $\hat{f}(\mathbf{x})$ is then decoded by the Prototype network's decoder g for visualization, given by

$$\hat{g} = g(\hat{f}(\mathbf{x})). \tag{3.2}$$

To identify the relevant dimensions in the input space, the input is mapped through the encoder and then decoded, denoted $g(f(\mathbf{x}))$. We then compute the absolute difference between the decoded input and the decoded masked input given by

$$\mathbf{x}^* = |\hat{g} - g(f(\mathbf{x}))|, \tag{3.3}$$

where $\mathbf{x}^*$ gives the feature importance scores of $\mathbf{x}$ for each dimension. Here the absolute difference gives the features in the input space with the largest change. Code for this chapter is available online. [1]

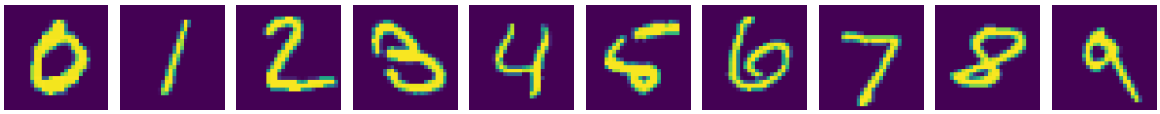
3.3 Experiments of Proposed Approach using Multiple Data Types

3.3.1 Image Data

With image data, we have the ability to visualize the explanation. We train a Prototype network on the MNIST dataset LeCun et al. [1998] with 3 encoding layers, 3 decoding layers, 1 prototype layer, and 1 fully connected layer. This model learns 10 prototype vectors (one for each class), achieving 99.1% accuracy on the test set.

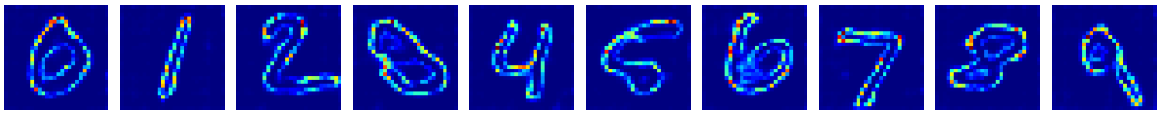
Figure 3.2 shows saliency maps of the proposed approach for each image in Figure 3.1. We can see that the proposed approach produces saliency maps that outline the digit in the original image. We perform the model parameter randomization and data randomization test Adebayo et al. [2018]. The model parameter randomization test generates saliency maps from a model with untrained, random parameters. The

[1] `https://github.com/halliwelln/prototype-explanations/`



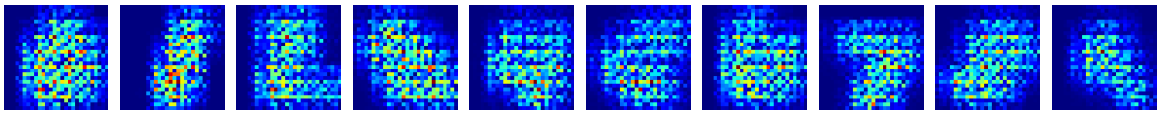
(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figure 3.1: MNIST Images



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

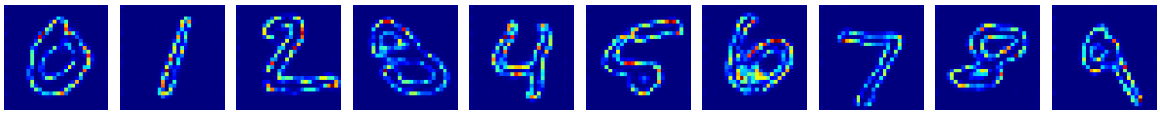
Figure 3.2: Saliency maps: Proposed approach



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

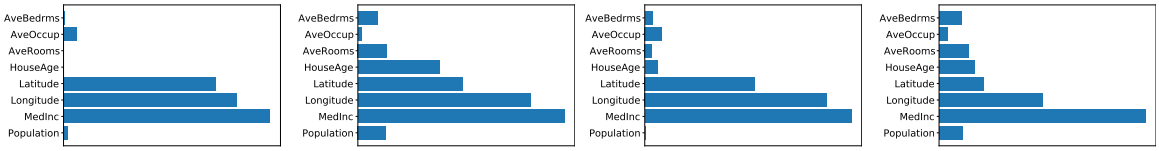
Figure 3.3: Saliency maps: Proposed approach-randomly initialized untrained network

resulting saliency maps should be random noise. The data randomization test trains a model where the training labels have been randomly shuffled. Similar to the model parameter randomization test, the resulting saliency maps should be random noise, and the end user should not be able to determine the object in the image. Figure 3.3 shows saliency maps from an untrained Prototype network with randomly initialized parameters (model parameter randomization test). Figure 3.4 shows saliency maps for a model trained on random labels (data randomization test). From these figures, we can see the proposed approach passes the model parameter randomization test but fails the data randomization test. In other words, the proposed approach to generating explanations is not providing insight into what the model has learned.



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4 (f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figure 3.4: Saliency maps: Proposed approach-network trained on randomly permuted labels



(a) Lime features: Observation 1 (b) Proposed features: Observation 1 (c) Lime features: Observation 2 (d) Proposed features: Observation 2

Figure 3.5: Explanations generated by Lime and proposed approach on California Housing dataset.

3.3.2 Tabular Data

We demonstrate our approach on a well known tabular dataset, the California Housing dataset Pace and Barry [1997]. Here, we are tasked with determining if houses should be sold above or below the median price. We train a Prototype network on the California Housing dataset with 2 encoding layers, and 2 decoding layers, 1 prototype layer, and 1 fully connected layer. This model learns 2 prototype vectors, achieving 84.2% accuracy on the test set.

Figure 3.5 compares relevant features identified by Lime Ribeiro et al. [2016] to our proposed approach for selected observations. For both observations, we can see that the top 3 dimensions with the highest attribution scores are the same for both explanation methods. Although both explanations are similar, they are not exactly equal. From these examples, which explanation method is actually displaying what

the model has learned? In other words, which explanation method is preferable to the other? These questions are difficult to answer without ground truth explanations to quantitatively compare against.

3.4 On the Development and Evaluation of Post Hoc Explanation Methods

From the experiments on tabular and image data, we found our approach produced what looked like faithful explanations on both types of data. After using several sanity checks Adebayo et al. [2018] on an image dataset, we were able to determine that this was not the case. For image data, we have the ability to visually verify any explanation generated in the input space. With tabular data, we do not have this luxury. Depending on the type of data used for experimentation, researchers can be mislead into thinking the explanations their model is generating are faithful because they are similar to a state-of-the-art method. With ground truth explanations, researchers would not have to rely on previous state-of-the-art explanation methods to determine if their approach is generating faithful explanations.

In general, this is a common problem in the field of XAI. When a new explanation method is proposed, researchers often show several "good looking" examples to display to the reader the capability of the proposed method. Comparisons against a state-of-the-art method typically involve a small number of cherry-picked examples to demonstrate the ability of an explanation method. This can be misleading. Indeed a small number of selected examples do not truly represent how the explanation method is performing on the entire test set. As we demonstrated on the tabular dataset, our proposed approach can compete with Lime on "selected" examples, however, this is not conclusive evidence that this explanation method is preferable to Lime. In order to accurately determine which explanation method is preferable, ground truth explanations are needed.

Defining ground truth explanations may be more difficult for different tasks, and different types of data. Additionally, there may be more than one way to explain a particular observation. Datasets with ground truth explanations must include all possible ways to explain each observation. Failing to include all possible ground truth explanations can unfairly penalize an explanation method for identifying a correct explanation not included in the ground truths.

There is existing work on qualitative evaluation of explanations. Poursabzi-Sangdeh et al. [2021] perform a user experiment to determine what makes a model interpretable. Jeyakumar et al. [2020] perform a user experiment to determine what style of explanation is preferred by users. Adebayo et al. [2020] develop a series of debugging tests, and include a user experiment to determine if users can identify defective models. Not much existing research focuses on quantitatively evaluating all test set explanations for quantitative comparisons across explanation methods. Relying on users to evaluate each explanation in the test set does not scale to large datasets, and cannot be performed on certain types of data (tabular data for example users shown an explanation would not know if its an accurate explanation or not). Additionally, users without a background in machine learning may not be able to determine a good explanation. For quantitative evaluations of explanations that scales to large datasets, scoring metrics must be defined that give an accurate representation of the explanation method's performance. Scoring metrics that measure explanation quality can be formally defined with ground truth explanations.

3.5 Concluding Remarks on the Empirical Evaluation of Post Hoc Explanations

In this chapter, we proposed a method to expand Prototype networks to identify relevant features in the input space. We compared selected examples against a state-of-the-art explanation method on tabular data and verify that the explanations are

similar. On image data however, our approach passed the model parameter randomization test but failed the data randomization test. It is common practice in the field of XAI to compare explanation methods using a few selected examples. This is not a thorough evaluation of explanation quality.

We discussed the development of explanation methods, identifying when one explanation method is preferable to another, and the complications that arise when measuring explanation quality. Much research in the field of XAI is devoted to developing new explanation methods. This chapter points out that more work should be devoted to evaluating the quality of explanation generated. Many of these issues can be solved with ground truth explanations. We recognize this can be difficult with tabular data. Research should be devoted to defining ground truth explanations for all domains in order to quantitatively evaluate explanations.

In the next chapter, we address this issue by proposing a method to construct datasets with ground truth explanations for link prediction on Knowledge Graphs. We also propose several scoring metrics, allowing researchers and practitioners to quantitatively compare explanations across different explanation methods for RGCNs.

Chapter 4

BENCHMARKING EXPLANATION METHODS FOR RGCN-BASED LINK PREDICTION WITH UNIQUE EXPLANATIONS

Practitioners and researchers typically apply multiple explanation methods to an RGCN post hoc and compare the quality of explanation from each method. Comparisons across explanation methods 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 chapter, 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.

4.1 Introduction to Explanation Generation for RGCN-based Link Prediction

Recently, there has been a push to explain the predictions of link prediction algorithms, creating the task of explainable link prediction. The term explanation is commonly defined as a statement that makes something clear. Throughout this thesis, 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.

State-of-the-art explanation methods such as ExplaiNE Kang et al. [2019], and GNNExplainer Ying et al. [2019] 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 chapter, 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 chapter is organized as follows: Section 4.2 provides an overview of the shortcomings of state-of-the-art explanation methods on the task of explainable link prediction along with our contributions. Section 4.3 describes a generic approach to generate datasets with ground truth explanations, and a metric for empirical evaluation. Section 4.4 applies this approach to construct two datasets, outlining the rules that define each dataset. Section 4.5 details the benchmark performed on the Royalty datasets, and reports the results. Section 4.6 discusses the limitations of the Royalty datasets. Lastly, Section 4.7 provides a summary of the contributions in this chapter. All the resources used and produced in this chapter are available online including the download link for the reasoner, code, and datasets. [1]

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

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 Kang et al. [2019]. 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. Ad-

ditionally, 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 in this chapter 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 include 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.

4.3 Generating Ground Truth Explanations for Evaluation

4.3.1 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 Horridge [2011] 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 Corby et al. [2012] 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 Jeyakumar et al. [2020]. This generic approach to generating ground truth explanations can be applied to many Knowledge Graphs and many sets of rules. In this chapter, 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.

4.3.2 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:

$$J(E, \hat{E}) = \frac{|E \cap \hat{E}|}{|E \cup \hat{E}|} = \frac{|E \cap \hat{E}|}{|E| + |\hat{E}| - |E \cap \hat{E}|}. \tag{4.1}$$

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 explana-

tion set. Additionally, the order of the explanations is not considered. Metrics like ROUGE-N Lin and Hovy [2003] or BLEU Papineni et al. [2002] 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

$$precision = \frac{tp}{tp + fp}, \tag{4.2}$$

$$recall = \frac{tp}{tp + fn}, \tag{4.3}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{4.4}$$

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

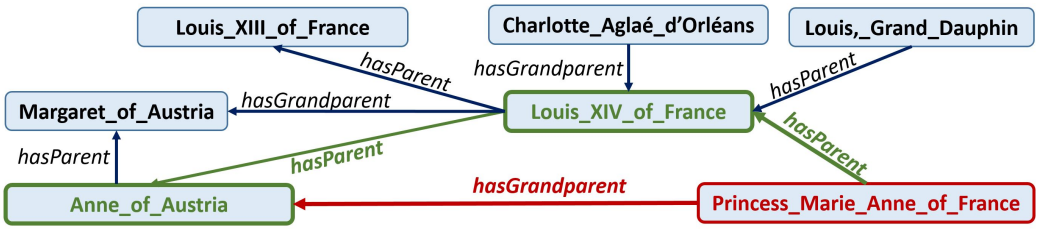


Figure 4.1: A triple (*Princess Marie Anne of France*, *hasGrandparent*, *Anne of Austria*) plotted in red with its explanation set in green {(*Princess Marie Anne of France*, *hasParent*, *Louis XIV of France*), (*Louis XIV of France*, *hasParent*, *Anne of Austria*)}, and neighboring triples.

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.

4.4 Extracting and Generating the Royalty Datasets

Applying the method of Section 4.3.1, we build two datasets (Royalty-20k and Royalty-30k), a collection of 20,080 and 30,734 triples respectively, containing royal family members from DBpedia Lehmann et al. [2015]. 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 4.1. 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*)}.

4.4.1 Royalty Datasets Rule Generation

In this chapter, 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 chapter, 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, for example, $hasSpouse(X, Y) \leftarrow hasSpouse(Y, X)$. 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 4.4.2. 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.

4.4.2 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

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 4.1: 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.

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 4.1 gives the details of each predicate. The "# Triples" column denotes the total number of triples in the dataset with that predicate. The "# 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 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.

4.5 Benchmark Explanation Methods on Royalty Datasets

4.5.1 Experiment Details

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 Kang et al. [2019] and GNNExplainer Ying et al. [2019] 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.

4.5.2 RGCN Link Prediction-Results

The first section of Table 4.2 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

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 4.2: 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.

hasSuccessor, and *hasPredecessor*.

The second section of Table 4.2 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.

4.5.3 Quantitative Evaluation of RGCN Link Prediction Explanations

GNNExplainer From Table 4.2, 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 4.2 reports the performance of ExplaiNE on the task of explainable link prediction. On both datasets, again 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*.

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

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.

4.5.4 Qualitative Evaluation of RGCN Link Prediction Explanations

Table 4.3 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

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*	27%
		hasParent	27%

Table 4.4: 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* 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 4.4 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.

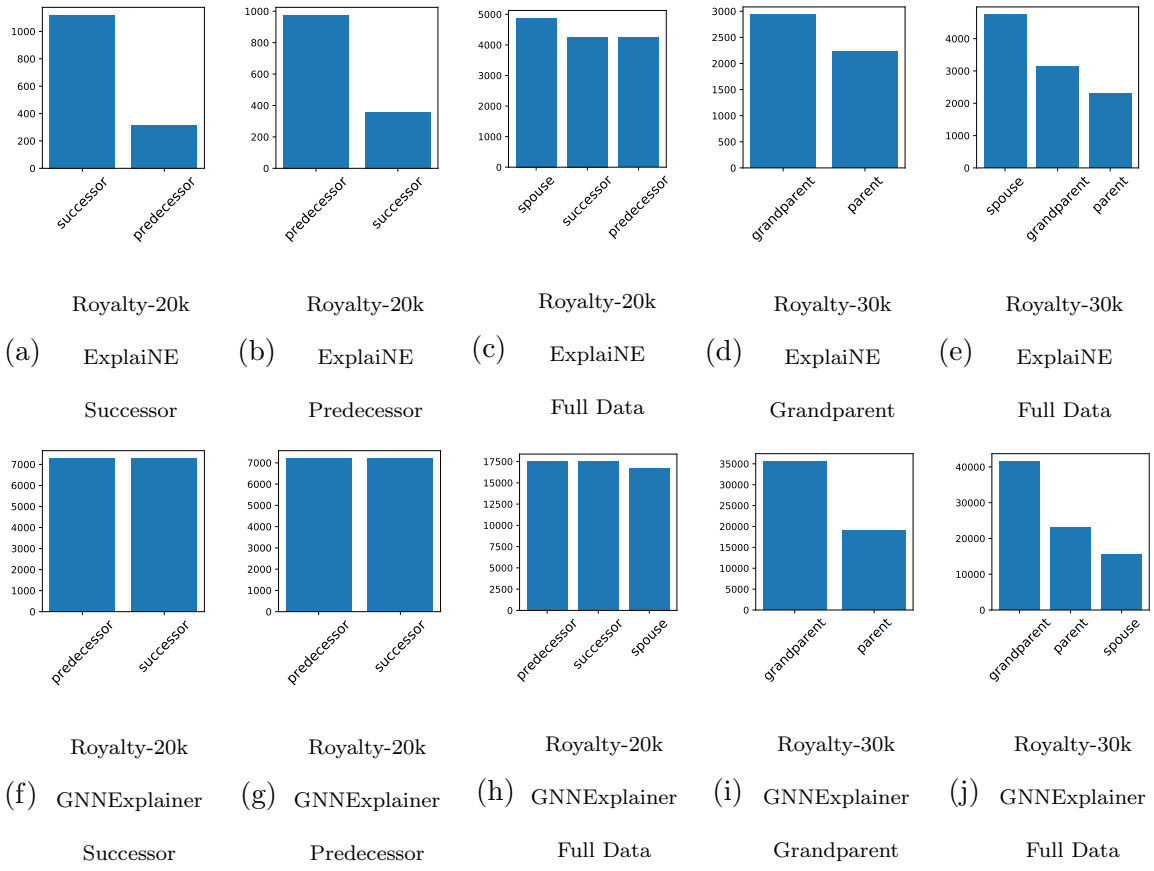


Figure 4.2: Predicate Frequency Count on Incorrectly Predicted Explanations. Note *hasSpouse* was omitted as only one predicate could be predicted (*hasSpouse*).

The first row of Figure 4.2 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 that 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 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 4.2 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.

4.5.5 Discussion of Royalty Benchmark Results

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 2.15 and 2.16, 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.

4.6 Limitations of Royalty Datasets

The Royalty-20k and Royalty-30k datasets provide a ground truth explanation for each observation in the training and test sets. The main drawback of these datasets is that they do not evaluate explanation method performance when there is more than way to explain an observation. That is, how do current state-of-the-art explanation methods perform when there are multiple ground truths to choose from. Additionally, the scoring metrics in this chapter were proposed only for use when there is one and only one ground truth explanation. These scoring metrics will not generalize to the case of multiple explanations, hence additional scoring metrics must be proposed to handle the case of multiple explanations.

4.7 Concluding Remarks on Royalty Datasets

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 chapter, we proposed a method, including two datasets (Royalty-20k, and Royalty-30k), to compare predicted and ground truth explanations. Furthermore, we proposed 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 benchmarked 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 to identify and quantify the different types of errors they make in terms of both data and semantics.

Chapter 5

BENCHMARKING EXPLANATION METHODS FOR RGCN-BASED LINK PREDICTION WITH MULTIPLE GROUND TRUTH EXPLANATIONS

The previous chapter showed how ExplaiNE and GNNExplainer perform when there was one and only one ground truth explanation to choose from. Indeed there can be multiple explanations for a given prediction in a KG. No dataset exists where observations have multiple ground truth explanations to compare against. Additionally, no standard scoring metrics exist to compare predicted explanations against multiple ground truth explanations. In this chapter, we introduce a method, including a dataset (FrenchRoyalty-200k), to benchmark explanation methods on the task of link prediction on KGs, when there are multiple explanations to consider. We conduct a user experiment, where users score each possible ground truth explanation based on their understanding of the explanation. We propose the use of several scoring metrics, using relevance weights derived from user scores for each predicted explanation. Lastly, we benchmark this dataset on state-of-the-art explanation methods for link prediction using the proposed scoring metrics.

5.1 Introduction to Multiple Explanations

In the previous chapter, we proposed several datasets with ground truth explanations, where each observation has one and only one ground truth. Ground truth explanations however can be non-unique. There can be multiple, logically correct ways to explain why a link could exist between two nodes. Consider an example where a model predicts the *hasChild* link between two entities Louis VII of France, and Agnes of France, i.e. (Louis VII, *hasChild*, Agnes of France). One way to explain

why this link could exist between these two entities is because Agnes of France is the child of Louis VII's spouse Adela of Champagne. This is not the only way to explain why the *hasChild* link exists between Louis VII and Agnes. Agnes could be the child of Louis VII because Agnes' grandparent, Louis VI is the parent of Louis VII. Both of these explanations are correct, it is unclear as to which explanation is optimal.

State-of-the-art explanation methods for link prediction on Knowledge Graphs have no common dataset or performance metrics to quantitatively evaluate explanation quality when there are multiple ways to explain the model's prediction. In this chapter, we propose a method, including a dataset (FrenchRoyalty-200k), to quantitatively evaluate explanation methods on the task of link prediction using Graph Neural Networks, when there are multiple explanations. This dataset includes all possible ground truth explanations for each triple, allowing for quantitative comparisons across every possible explanation. Additionally, we perform a user experiment, where users decide which explanations are optimal. Furthermore, we propose the use of several scoring metrics using these user scores as relevance weights for each predicted explanation. Lastly, we benchmark this dataset on state-of-the-art explanation methods using the proposed dataset and evaluation metrics.

5.2 Shortcomings of RGCN Explanation Methods and Contributions

Explanation quality The weakness of these explanation methods is the empirical evaluation of explanation quality. The authors of ExplaiNE acknowledge the difficulty in measuring the quality of explanation due to the lack of available datasets with ground truth explanations Kang et al. [2019]. Recall that ExplaiNE relies on the assumption that an 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 rec-

ommended movie, and the set of genres from the top 5 ranked explanations computed. A p-value is then computed to estimate the significance of the average. It is unknown how this evaluation method generalizes to tasks outside of movie recommendation. ExplaiNE has been previously benchmarked with 4 datasets: Karate, DBLP, MovieLens, and Game of Thrones networks. These datasets do not include ground truth explanations, and defining ground truth explanations for these networks is non-trivial. GNNExplainer has not been benchmarked on the task of explainable link prediction on Knowledge Graphs due to the lack of available datasets. Both GNNExplainer and ExplaiNE lack a common dataset and metric to evaluate explanation quality.

Multiple ground truths There can be multiple ways to explain why a link could exist between two nodes. Not all explanations are equally informative about the model's decision, some explanations can be arbitrarily more complicated than others. Explanation methods could generate an explanation when a more intuitive explanation could exist. Datasets containing only one ground truth explanation for each observation are insufficient to quantitatively evaluate explanation methods. Without considering all possible explanations, an explanation method could be incorrectly penalized for identifying a correct explanation not included in the ground truths. Therefore, a predicted explanation must be evaluated against all possible ground truth explanations. To our knowledge, there exists no dataset for link prediction on Knowledge Graphs where all possible ground truth explanations are included. Additionally, there are no standard quantitative metrics to measure the quality of explanations generated when there are multiple explanations to consider, making quantitative comparisons across explanations difficult.

Contributions Our contributions include a method to quantitatively evaluate explanation methods on the task of link prediction on Knowledge Graphs, when there are multiple ground truth explanations to consider. Additionally, we propose a dataset, FrenchRoyalty-200k, that includes every possible ground truth explanation for each observation. We perform a user experiment to determine which ground truth explanations are most intuitive. Furthermore, we propose the use of several performance metrics that score predicted explanations based on how intuitive users found the explanation, allowing for quantitative comparisons across explanation methods. Lastly, we benchmark state-of-the-art explanation methods using the proposed dataset and metrics.

5.3 Generating a User Evaluated Dataset with Multiple Ground Truth Explanations

5.3.1 Inference Traces as Explanations

In a Knowledge Graph, the available formal semantics allow us to view ground truth explanations as equivalent to computing justification for an entailment. We select an open-source semantic reasoner with rule-tracing capabilities Corby et al. [2012] to generate ground truth explanations for each defined rule, without needing manual annotations. 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 $l_c \leftarrow l_1 \wedge ... \wedge l_n$. 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.

Indeed this forms an intuitive explanation for graph data, a study shows that non-personalized feature-based explanations are efficient Tintarev and Masthoff [2012]. This generic approach to generating ground truth explanations can be applied to many Knowledge Graphs and many sets of rules. In this chapter, we focus on non-unique explanations, i.e. logical rules constructed to include all possible ground truth explanations. To our knowledge, this approach to generating non-unique ground truth explanations has not been applied to the task of explainable link prediction on Knowledge Graphs using Graph Neural Networks. All the resources used and produced in this chapter are available online including the download link for the reasoner, code, and dataset. [1]

5.3.2 Ensuring Completeness of Explanations

In this chapter, we focus on providing a dataset with non-unique explanations. We chose to describe family relations as no prior domain knowledge is needed. The explanation methods we want to evaluate provide their explanations as a set of triples that justify a prediction. In order to construct a dataset that includes all possible explanations for a given predicted triple, we first enumerated all possible paths

[1] `https://github.com/halliwelln/multiple-explanations/`

between the two nodes involved in this predicted triple. To exhaustively list all possible cases, we defined a small synthetic family graph systematically using all the possible types of family relations. This graph describes some individual Paul, and all family members within a 2-hop neighborhood, this includes aunts, grandparents, children, etc. This complete synthetic graph is then used to identify all possible paths with a maximum length of 2 linking the subject and objects of its triples. This graph is purposely kept small, to ensure each possible path can be verified manually. Each of these paths corresponds to one possible explanation as to why a link could exist between two given nodes e.g., Paul and Tom are brothers because Paul and Tom both have the same parent Jim. Indeed some of these paths can be turned into the antecedent of an inference rule for that type of triple e.g. $hasGrandparent(X, Z) \leftarrow hasParent(X, Y) \wedge hasParent(Y, Z)$. We define paths that can always be turned into the antecedent of an inference rule as *logical explanations*. In other words, these explanations are always true. Other paths do not always logically imply the targeted triple but still provide a good indication that could have triggered a human guess or a statistical prediction. For instance, $hasParent(X, Y) \wedge hasParent(Z, Y)$ could indicate X and Z are brothers or sisters or any combination of these relations. Without additional knowledge (e.g. the gender) the explanation is not always logically true. We define these paths as *partial explanations*. Some explanations will be more convincing than others to a user, especially among partial explanations. A score is needed for each possible explanation to distinguish good, intuitive explanations from bad, unintuitive ones.

5.3.3 Logical Derivation and Partial Explanation Rules

In this chapter, we focus on 6 family relationships: *hasSpouse*, with 3 possible explanations, *hasBrother*, with 7 possible explanations, *hasSister*, with 7 possible

explanations, *hasGrandparent*, with 6 possible explanations, and *hasChild*, and *hasParent* with 9 possible explanations. As we have two types of explanations (logical vs. partial), there are also have two types of rules (logical derivation vs. partial explanation). We define a *logical derivation rule* as one that is always true, and a *partial explanation rule* as one that is not always true without additional information, such as gender. The predicate of each triple used on the link prediction task is in the consequent of one or more of these rules. The associated explanation consists of the all possible triples that triggered each rule. The logical derivation rules trigger every time their antecedent is matched, and its corresponding triple and logically true explanation are generated. The partial explanation rules trigger only if the triple is already known (asserted or inferred by other rules) and are just adding alternative partial explanations, therefore preventing any false triples from being included in the graph. In addition, each rule (both logical derivation and partial explanation) associates a score to the explanations it generates. The score is defined at the per rule level, and is the same for all the explanations generated by that rule. The score captures how intuitive a given pattern of explanation is for a given type of predicted triple, detailed in the next section.

We apply all rules to the entire Knowledge Graph of the French Royalty families found in DBpedia Lehmann et al. [2015] to build the FrenchRoyalty-200k dataset. Included with each triple in the training and test sets are all possible explanations as to why a given link could exist between the two nodes. Figure 5.1 shows an example of a candidate triple along with several possible explanations. A candidate triple (*Louis VII of France*, *hasChild*, *Agnes of France*) plotted in red with its non-unique explanations in green: {(*Agnes of France*, *hasGrandparent*, *Louis VI of France*), (*Louis VII of France*, *hasParent*, *Louis VI of France*)}, {(*Adela of Champagne*, *hasChild*, *Agnes of France*), (*Louis VII of France*, *hasSpouse*, *Adela of Champagne*)}, and neighboring

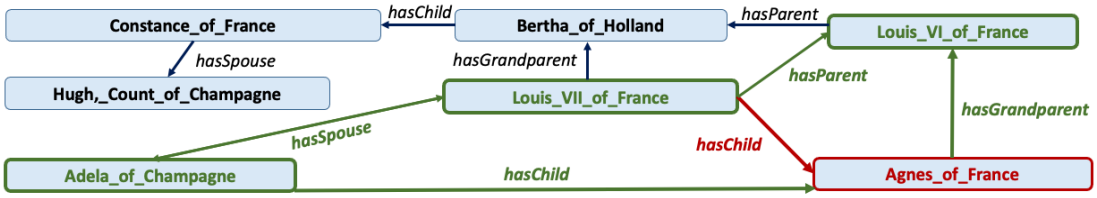


Figure 5.1: A candidate triple (*Louis VII of France*, *hasChild*, *Agnes of France*) plotted in red with its non-unique explanations in green: {(*Agnes of France*, *hasGrandparent*, *Louis VI of France*), (*Louis VII of France*, *hasParent*, *Louis VI of France*)}, {(*Adela of Champagne*, *hasChild*, *Agnes of France*), (*Louis VII of France*, *hasSpouse*, *Adela of Champagne*)}, and neighboring triples.

triples.

Table 5.1 outlines all rules defined in the FrenchRoyalty-200k dataset. Included in this table is the number of total triples for each predicate, the predicates used to define every possible explanation, the user score assigned to each predicate, and a column to indicate whether a rule is logically true or provides a partial explanation.

5.3.4 Users' Evaluation of Explanation Scores

Although ExplaiNE and GNNExplainer have many possible explanations to choose from, these explanations are not equal. Some explanations may be easier to understand than others. When benchmarking explanation methods with non-unique explanations, a scoring metric should assign a high score when an explanation method correctly predicts an intuitive explanation, and a low score when an unintuitive, overly complicated explanation is predicted.

We conducted a user experiment to score each possible explanation. This allows us to distinguish explanations that are intuitive from those that are not without relying on any prior assumptions. One could rely on the assumption that for example

Predicate	# Triples	Explanations	User Score	Explanation Type
hasBrother	6, 067	*hasSister*	0.8	Partial
		hasChild, *hasParent*	0.8	Partial
		hasGrandparent (x2)	0.3	Partial
		hasBrother, *hasSister*	0.6	Logical
		hasBrother, *hasBrother*	0.7	Logical
		hasSister, *hasSister*	0.7	Partial
		hasParent, *hasParent*	0.9	Partial
hasChild	52,399	*hasParent*	0.9	Logical
		hasChild, *hasSister*	0.7	Logical
		hasBrother, *hasChild*	0.7	Logical
		hasChild, *hasSpouse*	0.7	Logical
		hasGrandparent, *hasParent*	0.4	Partial
		hasChild, *hasGrandparent*	0.4	Partial
		hasBrother, *hasParent*	0.7	Logical
		hasParent, *hasSpouse*	0.7	Logical
		hasParent, *hasSister*	0.7	Logical
hasSpouse	31,984	*hasSpouse*	0.8	Logical
		hasChild,*hasParent*	0.5	Logical
		hasChild,*hasChild*	0.9	Logical

Table 5.1: FrenchRoyalty-200k dataset: Breakdown of all predicates each possible explanation, and its score given by users. # Triples column denotes the total number of triples with that predicate. User Score column gives the score assigned to each explanation by users. Explanation Type column denotes whether this explanation is a logical (always true) or only partial.

the shortest path, i.e., the explanation that uses the fewest number of predicates, is the most intuitive explanation. This assumption would fail when predicting the

Predicate	# Triples	Explanations	User Score	Explanation Type
hasSister	4,433	*hasBrother*	0.8	Partial
		hasChild, *hasParent*	0.8	Partial
		hasGrandparent (x2)	0.2	Partial
		hasBrother, *hasSister*	0.7	Logical
		hasBrother, *hasBrother*	0.7	Partial
		hasSister, *hasSister*	0.7	Logical
		hasParent, *hasParent*	0.9	Partial
hasParent	48,241	*hasChild*	0.9	Logical
		hasChild, *hasSister*	0.7	Logical
		hasBrother, *hasChild*	0.7	Logical
		hasChild, *hasSpouse*	0.7	Logical
		hasGrandparent, *hasParent*	0.3	Partial
		hasChild, *hasGrandparent*	0.3	Partial
		hasBrother, *hasParent*	0.7	Logical
		hasParent, *hasSpouse*	0.7	Logical
		hasParent, *hasSister*	0.7	Logical
hasGrandparent	61,333	*hasGrandparent*, *hasSister*	0.6	Logical
		hasChild, *hasParent*	0.9	Logical
		hasBrother, *hasGrandparent*	0.6	Logical
		hasParent,*hasParent*	0.9	Logical
		hasGrandparent, *hasSpouse*	0.7	Logical
		hasChild, *hasChild*	0.9	Logical

Table 5.1: FrenchRoyalty-200k dataset (continued)

hasGrandparent predicate, as there are no 1 hop paths, but many 2 hop paths. Relying on the shortest path would treat all 2 hop paths as equally intuitive, while the *hasParent/hasParent* path is by far the best explanation for *hasGrandparent*.

Using Google Survey, we conduct an experimental evaluation where for each predicate, users are shown all possible explanations on the Paul's Family graph, and asked to assign a score to each path based on if the explanation is intuitive. Users are

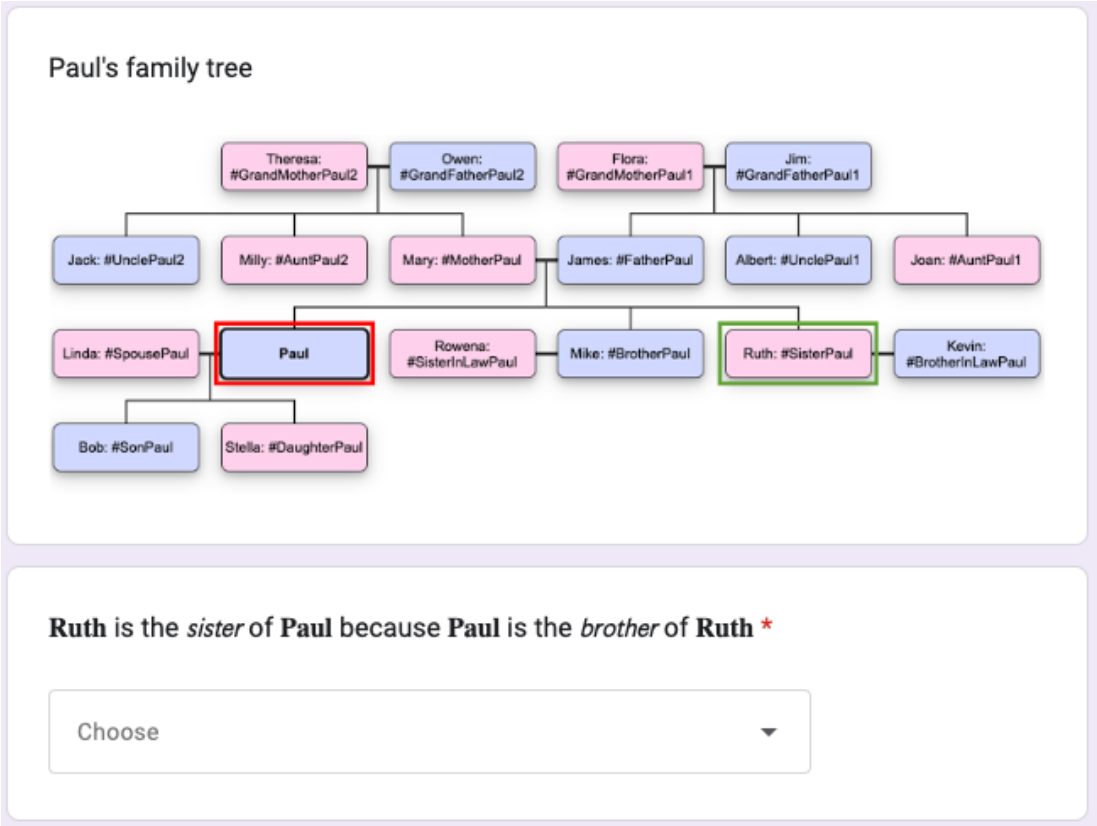


Figure 5.2: Example question from user survey on *hasSister* relation. Users are asked to determine how intuitive *hasBrother* is as an explanation.

given the following definition: "An explanation is considered intuitive if it is easily and immediately understood." For each predicate, and for each possible explanation, users are given an example of the predicate and its explanation used in a sentence. For example, for the *hasSister* predicate, one explanation uses the *hasParent*, and *hasChild* relations. Users are asked to score the following explanation: "Ruth is the *sister* of Paul because Mary is the *parent* of Paul, and Ruth is the *child* of Mary." Figure 5.2 shows an example *hasSister* question from the survey. In this example, users are asked to score how intuitive they find the *hasBrother* explanation.

Users scored each of these explanations on a five-point Likert scale: (4) Very intuitive, an explanation I could give or expect; (3) Intuitive; (2) Neither intuitive or

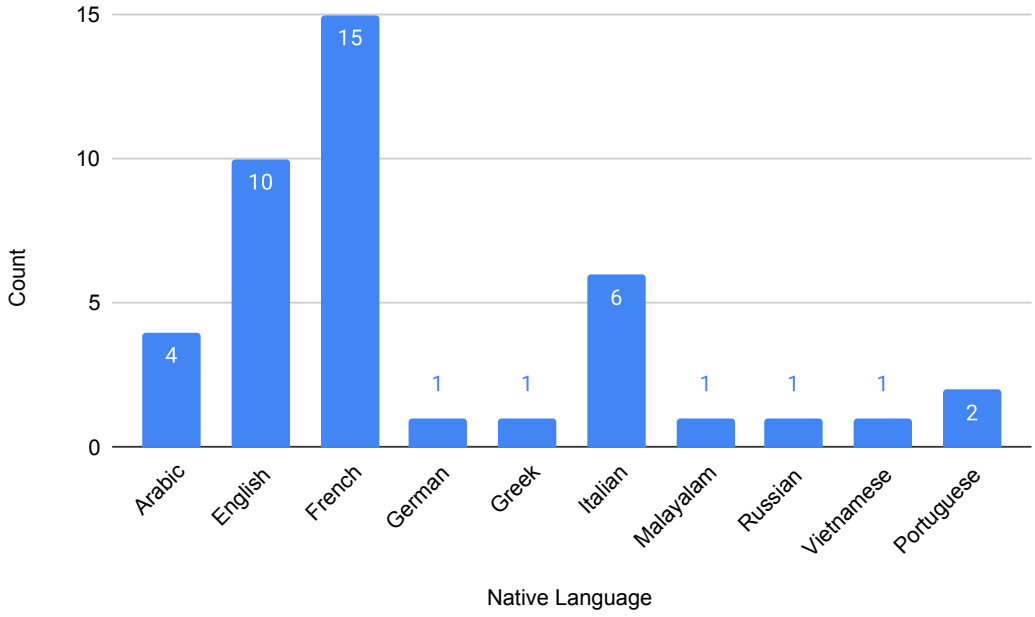


Figure 5.3: Native languages of user survey participants

unintuitive; (1) Unintuitive ; (0) Not intuitive at all, not an explanation I would give or expect.

In total, 42 users responded from 11 different nationalities, consisting of both computer science and non-computer science backgrounds. Figure 5.3 gives a breakdown of the native languages of survey participants. The most common native languages were French, English, and Italian. We normalized the average scores between 0 and 1 for each possible explanation, and round them to the nearest tenth. These user scores are used in the rules and in the benchmark, detailed below, to penalize unintuitive predicted explanations, and reward intuitive predicted explanations. User scores for each predicate and explanation can be found in Table 5.1.

This survey also showed that, even for humans, explanations can be difficult to define and assess. Users had difficulty deciding which explanations were intuitive. Even when presented equivalent explanations for two different predicates, for exam-

ple, using *hasBrother*, *hasSister* to explain a *hasBrother* link, and using *hasBrother*, *hasSister* to explain a *hasSister* link, users on average did not assign these explanations the same score. This lack of symmetry can be seen for multiple predicates in Table 5.1.

5.4 Evaluation of Multiple Ground Truth Explanations

5.4.1 Scoring Metrics for Multiple Explanations

To our knowledge, there is no standard evaluation metric to measure the quality of explanations generated by explanation methods when there are non-unique explanations available to predict. A standard evaluation metric is needed to identify when one explanation method is preferable to the other. The binary precision and recall could be used for this task, however, these metrics fail to account for the fact that some explanations can be more intuitive than others to users. Both metrics would give a score of 1 when a predicted explanation exactly matches a ground truth explanation. However, an explanation method could predict an unintuitive explanation, and receive the highest possible evaluation score, potentially misleading practitioners into thinking the predicted explanation is of high quality. Therefore, scoring metrics used for this task must compare a predicted explanation to all possible explanations, and account for the fact that explanations have different degrees of relevance. Ideally, a scoring metric for this task should assign a lower score to an unintuitive predicted explanation, and a higher score to an intuitive predicted explanation.

We propose to score explanation methods with non-unique explanations by adapting the generalized precision and generalized recall Kekäläinen and Järvelin [2002]. Originally proposed for document retrieval, generalized precision and generalized recall measure precision and recall based on the relevance score assigned to each re-

trieved document. Generalized precision is defined by the sum of relevance scores for each retrieved document divided by the number of retrieved documents. Generalized recall is defined by the sum of relevance scores for each retrieved document divided by the sum of relevance scores for all documents in the database.

We adapted these metrics in the context of link prediction on Knowledge Graphs. Formally, let t_i be a triple, $e_i = \{t_1, \ldots, t_n\}$ be one of the possible ground truth explanations for triple t_i. Let $\hat{e}_i \in \hat{E}_i$ be the predicted explanation for t_i, and E_i be defined as all possible explanations for t_i. Lastly, let $s(.)$ gives the relevance score (determined by the user experiment) for a given explanation. First, the best possible user score for an explanation is given by Equation 5.1:

$$s_i = \max_{e_i \in E_i} s(e_i). \tag{5.1}$$

The generalized precision between a predicted explanation and a ground truth explanation is given by Equation 5.2:

$$gp(\hat{e}_i, e_i) = \frac{|\hat{e}_i \cap e_i| \times s(e_i)}{|\hat{e}_i| \times s_i}. \tag{5.2}$$

Intuitively, it is a sum of the user scores of the triples shared by the prediction and the ground truth, divided by the number of triples in the prediction, and the largest possible user score. For a given triple t_i, we take the highest generalized precision across all of t_i's ground truth explanations, given by Equation 5.3:

$$gp(\hat{e}_i, E_i) = \max_{e_i \in E_i} gp(\hat{e}_i, e_i). \tag{5.3}$$

We compute the maximum generalized precision for each triple, and average across the dataset. For the set of all predicted explanations $\hat{E}$, and the set of all ground

truth explanation sets E, the generalized precision for an explanation method across the entire dataset is given by Equation 5.4:

$$GP(\hat{E}, E) = \frac{\sum_{\hat{e}_i \in \hat{E}, E_i \in E} gp(\hat{e}_i, E_i)}{|\hat{E}|}. \tag{5.4}$$

The generalized recall sums the relevance scores for each predicted explanation that exists in the ground truth explanations, divided by the number of triples in the ground truth explanation, and the largest possible user score, given by Equation 5.5:

$$gr(\hat{e}_i, e_i) = \frac{|\hat{e}_i \cap e_i| \times s(e_i)}{|e_i| \times s_i}. \tag{5.5}$$

Similar to generalized precision, we propose to compute a maximum generalized recall for each triple in the dataset (Equation 5.6) and average it across the dataset (Equation 5.7).

$$gr(\hat{e}_i, E_i) = \max_{e_i \in E_i} gr(\hat{e}_i, e_i), \tag{5.6}$$

$$GR(\hat{E}, E) = \frac{\sum_{\hat{e}_i \in \hat{E}, E_i \in E} gr(\hat{e}_i, E_i)}{|\hat{E}|}. \tag{5.7}$$

Note that we normalize the generalized precision and recall by the largest possible user score for each explanation to ensure they take values between 0 and 1.

We compute the generalized F_1 score, defined as the harmonic mean between the generalized precision and generalized recall. To ensure the recall and precision are computed on the same explanation, we compute them before we select the maximum (Equation 5.8) and average it (Equation 5.9)

$$gf_1(\hat{e}_i, E_i) = \max_{e_i \in E_i} \frac{2 \times gr(\hat{e}_i, e_i) \times gp(\hat{e}_i, e_i)}{gr(\hat{e}_i, e_i) + gp(\hat{e}_i, e_i)}, \tag{5.8}$$

$$GF_1(\hat{E}, E) = \frac{\sum_{\hat{e}_i \in \hat{E}, E_i \in E} gf_1(\hat{e}_i, E_i)}{|\hat{E}|}. \tag{5.9}$$

Finally we propose the use of the max-Jaccard metric to identify which explanation had the most in common with the predicted explanation. Formally, for triple t_i, we compute the Jaccard similarity between predicted explanation $\hat{e}_i$ with one of the possible ground truth explanation sets e_i, given by Equation 5.10:

$$j(e_i, \hat{e}_i) = \frac{|e_i \cap \hat{e}_i|}{|e_i \cup \hat{e}_i|} = \frac{|e_i \cap \hat{e}_i|}{|e_i| + |\hat{e}_i| - |e_i \cap \hat{e}_i|}. \tag{5.10}$$

We compute this Jaccard similarity across all possible explanations in set E_i for triple t_i (Equation 5.11) and average the result over the dataset (Equation 5.12):

$$mj(\hat{e}_i, E_i) = \max_{e_i \in E_i} j(\hat{e}_i, e_i), \tag{5.11}$$

$$MJ(\hat{E}, E) = \frac{\sum_{\hat{e}_i \in \hat{E}, E_i \in E} mj(\hat{e}_i, E_i)}{|\hat{E}|}. \tag{5.12}$$

The max-Jaccard compares a predicted explanation with all possible explanations available to choose from. Intuitively it identifies the ground truth explanation that shares a maximum number of triples with the predicted explanation, therefore indicating which explanation a method may have tried to predict.

We argue these metrics are sufficient to quantitatively compare explanation methods when there are multiple explanations to consider. The max-Jaccard score measures if the explanation method is able to accurately predict one of the possible explanations to choose from. The generalized precision and generalized recall measure if the predicted explanation was given a high intuitive score assigned by users.

Both metrics prevent an explanation method from only predicting low scored, unintuitive explanations, and receiving a high score. Lastly, the generalized F_1 provides an overview of performance on the generalized precision and recall.

5.4.2 Benchmark Setup and Protocol for Multiple Explanations

In this benchmark, we fix the number of dimensions to 10, the best performing in terms of accuracy from the set $\{3, 5, 10\}$. We use a learning rate of 0.01, the best performing from the set $\{0.01, 0.001, 0.0001\}$. Lastly, we train the RGCN on 1000 epochs for all rules, found to the best performing from the set $\{50, 100, 500, 1000, 2000\}$. We report the accuracy of the RGCN on the link prediction task. For each data subset and each explanation method, we report the generalized precision, generalized recall, generalized F_1, and max-Jaccard.

We train GNNExplainer using a learning rate of 0.001 for each rule. We use 20 iterations for each observation. 3-fold cross validation is performed for both explanation methods, and we report the results of the best performing fold.

5.4.3 Results and Discussion

Results per Subset We benchmark the FrenchRoyalty-200k dataset by splitting the full data into subsets where only one type of predictable predicate is included. The top half of Table 5.2 reports performance results of each predicate subset. For example, the Spouse subset included only triples in the training and test sets with the *hasSpouse* predicates, and their associated explanations.

First, the topmost row of Table 5.2 reports the results of the RGCN on the link prediction task. We observe the highest accuracy on the *hasSpouse* relation, and a drop in performance across the other predicates. We observe the lowest accuracy on the *hasChild* relation.

Models	Metrics	Statistics in separated subsets focused on one predicate						
		Spouse	Brother	Sister	Grandparent	Child	Parent	Full data
RGCN	Accuracy	0.903	0.877	0.825	0.787	0.767	0.805	0.81
GNN Explainer	Generalized Precision	0.261	0.366	0.281	0.17	0.137	0.123	0.11
	Generalized Recall	0.434	0.395	0.31	0.17	0.158	0.152	0.121
	Generalized F_1	0.318	0.376	0.291	0.17	0.144	0.133	0.114
	Max-Jaccard	0.275	0.372	0.373	0.137	0.166	0.161	0.11
ExplaiNE	Generalized Precision	**0.296**	**0.407**	**0.353**	**0.21**	**0.181**	**0.202**	**0.173**
	Generalized Recall	**0.546**	**0.458**	**0.459**	**0.21**	**0.223**	**0.243**	**0.2**
	Generalized F_1	**0.378**	**0.424**	**0.388**	**0.21**	**0.195**	**0.216**	**0.182**
	Max-Jaccard	**0.315**	**0.447**	**0.417**	**0.179**	**0.22**	**0.252**	**0.174**

Models	Metrics	Individual predicate statistics on the full dataset						
		Spouse	Brother	Sister	Grandparent	Child	Parent	Full data
RGCN	Accuracy	0.786	0.878	0.826	0.822	0.804	0.8	0.81
GNN Explainer	Generalized Precision	0.071	0.174	0.117	0.129	0.109	0.091	0.11
	Generalized Recall	0.106	0.192	0.142	0.129	0.125	0.102	0.121
	Generalized F_1	0.083	0.18	0.126	0.129	0.114	0.095	0.114
	Max-Jaccard	0.066	0.2	0.151	0.102	0.125	0.12	0.11
ExplaiNE	Generalized Precision	**0.138**	**0.25**	**0.194**	**0.177**	**0.166**	**0.182**	**0.173**
	Generalized Recall	**0.221**	**0.263**	**0.214**	**0.177**	**0.207**	**0.222**	**0.2**
	Generalized F_1	**0.165**	**0.253**	**0.2**	**0.177**	**0.18**	**0.195**	**0.182**
	Max-Jaccard	**0.133**	**0.27**	**0.237**	**0.145**	**0.187**	**0.225**	**0.174**

Table 5.2: Benchmark results on FrenchRoyalty-200k: Link prediction results for RGCN, and explanation evaluation for GNNExplainer and ExplaiNE. Highest scores per predicate denoted in bold.

Additionally, the top half of Table 5.2 reports the results of GNNExplainer on the task of explainable link prediction. We can see GNNExplainer performed the best on the *hasBrother* predicate explanation in terms of the generalized F_1 score. Note that the RGCN link prediction also performed well on the *hasBrother* predicate. We observe performance drops on the relations *hasChild* and *hasParent*, and on the full

dataset, with all predicates included. Indeed the *hasChild* and *hasParent* explanations follow a similar structure and definition of being logically inverse relations of each other.

The top half of Table 5.2 reports the results of ExplaiNE on the task of explainable link prediction. This method performed the best on the *hasBrother* and *hasSister* predicate subsets in terms of generalized F_1 score. We see the lowest performance on the full datset, followed by the *hasGrandparent* and *hasChild* predicate subsets. Across all metrics and predicate subsets, we find ExplaiNE outperformed GNNExplainer.

Full Data Results The bottom half of Table 5.2 further breaks down the results on the full dataset (Full data column of the top half table). We filter the results on the full data for each predicate and compare performance metrics to each predicate subset. For example, the Spouse column from the bottom half of Table 5.2 reports the benchmark performance of all input triples with the *hasSpouse* predicate from an RGCN trained on the full data. This RGCN is exposed to all possible predicates, whereas the Spouse column from the top half reports benchmark performance on an RGCN trained only on the input triples with the *hasSpouse* predicate.

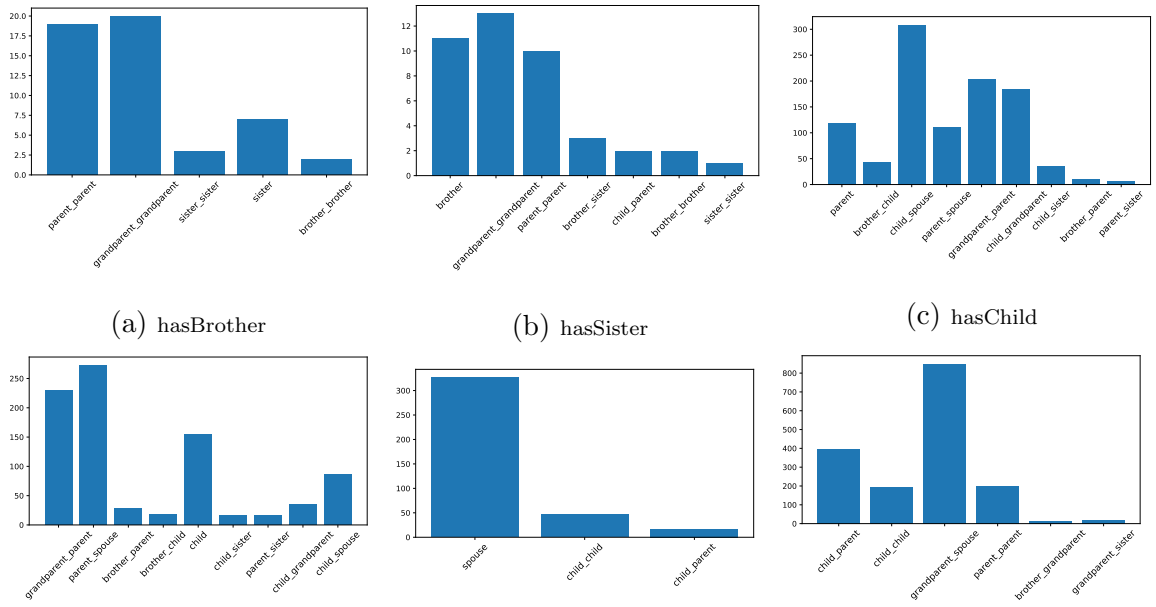
Comparing the two halves of Table 5.2, we can see the generalized precision, recall and F_1 scores generally decreased. These large changes across explanation performance metrics suggest that embeddings learned by the RGCN play a significant role. The RGCN trained on the *hasSpouse* subset is learning embeddings using only triples with *hasSpouse* and explanations containing *hasSpouse*, *hasChild*, and *hasParent*. In other words, the RGCN trained on this subset only has access to these predicates. The RGCN trained on the full dataset however has access to all predicates listed in Table 5.1. This could suggest, for instance, the embeddings from the full data is

Models	Rule	User Scores							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
GNN Explainer	Spouse	0	0	0	50	0	0	**276**	59
	Brother	0	21	0	0	0	2	10	**23**
	Sister	**19**	0	0	0	0	3	13	7
	Grandparent	0	0	0	0	61	504	0	**1104**
	Child	0	0	353	0	0	**585**	0	91
	Parent	0	192	0	0	0	**515**	0	152
	Full data	11	195	312	59	47	**1668**	301	1499
ExplaiNE	Spouse	0	0	0	17	0	0	**327**	47
	Brother	0	**20**	0	0	0	5	7	19
	Sister	**13**	0	0	0	0	6	**13**	10
	Grandparent	0	0	0	0	32	**850**	0	788
	Child	0	0	389	0	0	**516**	0	118
	Parent	0	264	0	0	0	**437**	0	154
	Full data	16	274	336	62	30	**1765**	267	1333

Table 5.3: Distributions of user scores amongst incomplete attempts. For example, of ExplainE's incorrect predictions on the *hasSpouse* predicate, ExplaiNE unsuccessfully attempted to predict an explanation with a user score of 0.8 on 327 observations.

incorporating additional, useless information into the embedding causing a drop in explanation metrics.

Error Analysis We define an incomplete attempt to be a predicted explanation where the max-Jaccard score across all possible explanations is less than 1. If two explanations have the same max-Jaccard score, we take the explanation with the highest user score. An incomplete attempt is considered to be a mistake made by the explanation method. Table 5.3 reports the distributions of user scores amongst the incomplete attempts of GNNExplainer and ExplaiNE for each predicate subset. On the *hasSpouse* subset, GNNExplainer unsuccessfully attempted to predict an explanation with a user score of 0.5 on 50 observations in the test set. From this table,



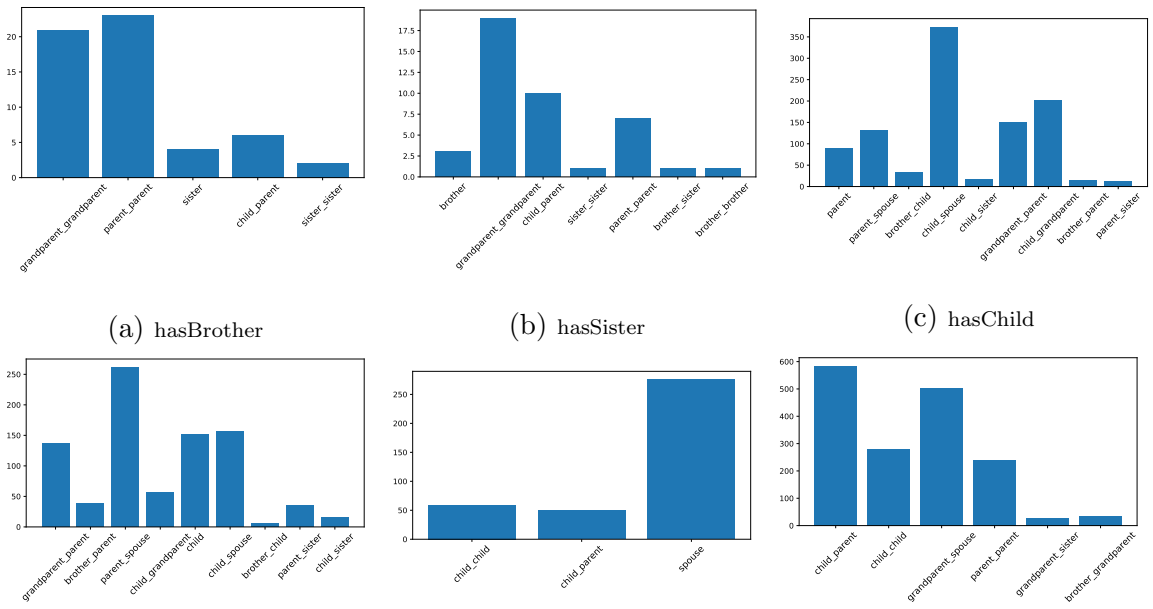
(a) hasBrother (b) hasSister (c) hasChild

(d) hasParent (e) hasSpouse (f) hasGrandparent

Figure 5.4: ExplaiNE FrenchRoyalty-200k: Most frequently predicted predicates amongst incomplete attempts

we can see both explanations methods attempted many times but failed to predict explanations with user scores of 0.7. Both explanation methods do not always attempt to predict explanations with the highest user scores (0.9). We recognize the imbalance of user scores, with 0.7 being the most common user score assigned to an explanation. Still, we bring to attention the fact that these explanation methods do not always try to predict the best possible explanation (those with the highest user scores).

Finally, the proposed method and dataset allows us to perform an error analysis on the most frequently predicted predicates amongst incomplete attempts. For instance, Figure 5.4e shows a histogram of ExplaiNE's incomplete attempts on the *hasSpouse* predicate. The most frequently predicated predicate was *hasSpouse*, accounting for 83% of incomplete attempts. As an example, for an input triple (Eadhild, *hasSpouse*,



(a) hasBrother (b) hasSister (c) hasChild

(d) hasParent (e) hasSpouse (f) hasGrandparent

Figure 5.5: GNNExplainer FrenchRoyalty-200k: Most frequently predicted predicates amongst incomplete attempts

Hugh the Great), and its ground truth explanations (Hugh the Great, *hasSpouse*, Eadhild), ExplaiNE predicted a first degree neighbor (Hugh the Great, *hasSpouse*, Hedwig of Saxony). This incorrect predicted explanation uses the *hasSpouse* predicate but in the wrong way. Similarly, Figure 5.5e shows a histogram of GNNExplainer's incomplete attempts on the *hasSpouse* predicate.

We can see the importance of the FrenchRoyalty-200k dataset from this benchmark, along with the method we use to construct it, and the metrics we provide. State-of-the-art explanation methods do not always give accurate explanations. Explanation methods must be evaluated with ground truth explanations and quantitative metrics that consider all possible explanations. Our method, dataset, and metrics allow researchers to do so, and to develop new explanation methods and quantitatively evaluate their explanations in a way they were previously unable to.

On all three datasets benchmarked so far in this thesis (Royalty-20k, Royalty-30k, and the FrenchRoyalty-200k), we observe that both ExplaiNE and GNNExplainer struggle to produce explanations that match the ground truth. These explanation methods are applied post hoc, thus it can be difficult to determine what is causing an error. In order to improve the quality of explanation produced by these explanation methods, one must first understand what is causing the error. Are the RGCN embeddings lacking the necessary information needed for GNNExplainer/ExplaiNE to produce an accurate explanation? Or are GNNExplainer/ExplaiNE to blame for an incorrect explanation?

5.5 Concluding Remarks on Multiple Explanation Benchmark

On the task of explainable link prediction, there is no standard dataset available where there are multiple ground truth explanations to choose from. Additionally, no standard method exists to generate datasets with all possible explanations. Furthermore, there is no standard evaluation metric to compare a predicted explanation with all possible ground truths. In this chapter, we proposed a method, including a dataset, FrenchRoyalty-200k, to compare predicted and ground truth explanations when there are multiple ground truths. We proposed the use of several evaluation metrics, leveraging the use of graded precision and recall for quantitative comparisons across explanation methods. Lastly, we benchmarked two state-of-the-art explanation methods, ExplaiNE and GNNExplainer using the proposed dataset and scoring metrics. This method can be used to generate other Knowledge Graphs with a variety of different domains, size, density, etc., to support the qualitative and quantitative evaluation of explanations for RGCN link prediction.

Chapter 6

IMPACT OF INJECTING GROUND TRUTH EXPLANATIONS INTO RGCN EMBEDDINGS ON EXPLANATION METHOD PERFORMANCE

As we saw in the previous chapters, benchmark results of state-of-the-art explanation methods showed the difficulties in predicting explanations. Performance metrics for both methods on all three Royalty datasets were low. In this chapter, we leverage prior knowledge to further constrain the loss function of RGCNs, by penalizing node embeddings far away from the node embeddings in their associated ground truth explanation. Empirical results show improved explanation prediction performance of state-of-the-art post hoc explanations methods for RGCNs, at the cost of predictive performance. Additionally, we quantify the different types of errors made both in terms of data and semantics.

6.1 Introduction to Explanation Aware RGCNs

In previous chapters, we proposed several datasets with ground truth explanations for link prediction on Knowledge Graphs, allowing for quantitative comparisons of predicted explanations. For state-of-the-art explanation methods such as ExplaiNE and GNNExplainer, initial benchmark results showed these methods do not always correctly predict ground truth explanations. Previous approaches to learning Knowledge Graph embeddings did not have access to ground truth explanations, hence do not incorporate information from explanations into the embedding.

In this chapter, we adapt RGCNs to incorporate prior knowledge from ground truth explanations into each embedding. This is done by constraining the cross entropy loss functions used by RGCNs. We compare several different explanation-

constrained loss functions to an RGCN using the standard binary cross entropy. Results show improved predicted explanation performance of post hoc explanation methods for RGCNs, at the cost of predictive performance. Additionally, we quantify the different types of errors made in terms of data and semantics.

6.2 Injecting Ground Truth Explanations into RGCN Embeddings

In this section, we outline several proposed approaches to train RGCNs with prior knowledge from ground truth explanations. We describe each approach using a penalty associated with the explanations and adapted to different datasets. Code for this chapter is available online. [1]

6.2.1 Constraining the Loss Function of RGCNs

In the context of Image classification, Rieger et al. [2020] show that interpretations are useful and that we can penalize explanations to align neural networks with prior knowledge. To do so, they constrain the loss functions of deep neural networks by introducing an explanation penalty term, which teaches the model to generate correct explanations. This additional constraint was shown to increase classification performance. The explanations generated by this approach however were not empirically evaluated. Without ground truth explanations, this paper relies on assumptions made by either manually annotating explanation labels, or rules to define correct explanations for image data. Indeed manual annotation is difficult with large datasets.

For link prediction on Knowledge Graphs, the standard RGCN optimizes a cross entropy loss function (Equation 2.14) to learn embeddings. In recent work, Halliwell et al. [2021c,b] used a standard RGCN in a benchmark of three datasets to determine the quality of explanations generated post hoc by GNNExplainer and ExplaiNE.

[1] `https://github.com/halliwelln/penalized-rgcn`

Until recently, benchmarks did not include ground truth for explanations, and the loss functions used by the standard RGCN do not include any constraints that account for them. This lack of constraints on the standard RGCN loss function causes subject and object embeddings in each triple to be mapped far away in the embedding space from the subject and object embeddings in its associated explanation. The Royalty datasets from Halliwell et al. [2021c,b] gives us the opportunity to train the predictors with the prior knowledge of ground truth explanations.

We propose a loss function for RGCNs to improve post hoc explanation method performance on the Royalty datasets. This is achieved by adding an explanation constraint that pushes subject and object embeddings from each input triple closer to subject and object embeddings in the input triple's explanation. This is captured by the penalty expressed in Equation 6.1 where, for some input triple $t_p = (s, p, o)$ and an explanation triple $t_j = (s_j, p_j, o_j)$, we propose an explanation aware constraint that can be added to the binary cross entropy used by RGCNs:

$$\begin{aligned}\mathcal{P}(t_p, t_j) = max(||Emb(s) - Emb(s_j)||_2, ||Emb(s) - Emb(o_j)||_2) \\ + max(||Emb(o) - Emb(s_j)||_2, ||Emb(o) - Emb(o_j)||_2).\end{aligned} \tag{6.1}$$

This penalty sums the maximum ℓ_2 distances between embedding $Emb(.)$ of the subjects and objects of the input triple t_p and an explanation triple t_j. Penalizing the maximum allows us to push the subject embedding $Emb(s)$ from the input triple closer to subject and object embeddings from its ground truth explanation.

The ℓ_2 maximum distance computations accounts for the fact that the direction of the links is an arbitrary modelling decision that should not impact the comparison of the embeddings of subjects and objects. Consider the case when the input triple $t_p = (John, hasParent, Tom)$, and its associated ground truth explanation $t_j = (Tom, hasChild, John)$. Simply summing the distance between subject and objects

gives $||Emb(John) - Emb(Tom)||_2 + ||Emb(Tom) - Emb(John)||_2 = 2 * ||Emb(John) - Emb(Tom)||_2$. If however, the direction of the predicate in the explanation changes, for example, if $t_j = (John, isChildof, Tom)$, the distance summation then becomes $||Emb(John) - Emb(John)||_2 + ||Emb(Tom) - Emb(Tom)||_2 = 0$. Certainly the triple pair $(John, hasParent, Tom)$, and $(Tom, hasChild, John)$ contains the same amount of information as $(John, hasParent, Tom)$, and $(John, isChildof, Tom)$, however, a simple summation over distances results in two times the distance or zero. In order to account for this ambiguous case, we compute the maximum between the subject distances and add this to the maximum between the object distances.

We can now augment the standard RGCN binary cross entropy loss with the penalty term in Equation 6.1 in several ways and to account for several types of prior knowledge from explanation ground truth.

6.2.2 Explanation Aware Loss Function for Unique Explanations

The Royalty-20k and Royalty-30k datasets Halliwell et al. [2021c] contain one and only one unique explanation per predicted triple, describe as the case of non-ambiguous explanations. We introduce the first loss function incorporating the penalty term from Equation 6.1 for non-ambiguous explanations. Formally, let $t_p \in \mathcal{T}^+$ be a positive triple in the form (s, p, o), let e_p be its explanation that contains a set of explanatory triples t_j for the prediction of t_p. The equation our proposed approach optimizes is given by

$$\mathcal{L}_{sum} = \mathcal{L}_{RGCN} + \quad \frac{1}{|\mathcal{T}^+|} \sum_{t_p \in \mathcal{T}^+} \quad \frac{1}{|e_p|} \sum_{t_j \in e_p} \mathcal{P}(t_p, t_j), \tag{6.2}$$

where $|.|$ denotes the cardinality, for example $|e_p|$ denotes the number of triples in e_p. Intuitively, Equation 6.2 takes a training set triple $t_p = (s, p, o)$, and its asso-

ciated explanation e_p, and applies an ℓ_2 penalty for subject and object embeddings in the explanation of t_p that are far away from t_p's subject and object in the embedding space. In other words, the subject and object embeddings found in each triple's ground truth explanation should be "similar" in the embedding space to the subject and object embeddings, as they are used to explain why a predicate exists between the triple's subject and object. Using the standard RGCN loss function, this relationship between a triple and its ground truth explanation may not be captured in the embedding space without the additional constraint from Equation 6.2. We apply this loss function to the Royalty-20k and Royalty-30k datasets, results are reported in the following section.

6.2.3 RGCN Loss Summing all Possible Explanations

The FrenchRoyalty-200k dataset Halliwell et al. [2021b] contains multiple explanations for each predicted triples, described as the case of non-unique, or ambiguous explanations. We introduce a loss function including a penalty term for these ambiguous explanations. Formally, let $E_p = \{e_1, ...e_l\}$ be the set of l explanations available for t_p and $e_i = \{(s_1, p_1, o_1), ..., (s_k, p_k, o_k)\}$ be i^{th} explanation for triple t_p. Explanation e_i contains a set of explanatory triples t_j for the prediction of t_p. The proposed loss function is given by

$$\mathcal{L}_{sum'} = \mathcal{L}_{RGCN} + \frac{1}{|\mathcal{T}^+|} \sum_{t_p \epsilon \mathcal{T}^+} \quad \frac{1}{|E_p|} \sum_{e_i \epsilon E_p} \quad \frac{1}{|e_i|} \sum_{t_j \epsilon e_i} \mathcal{P}(t_p, t_j). \tag{6.3}$$

Similarly, Equation 6.3 takes a training set triple $t_p = (s, p, o)$, and its associated explanations E_p, and applies an ℓ_2 penalty for subject and object embeddings from all explanations of t_p that are far away from t_p's subject and object in the embedding space. The subtle difference between this loss function and Equation 6.2 is that

$E_p = \{e_p\}$, that is, there is only one explanation available for Equation 6.2, hence the inner summation can be dropped. Equation 6.3 however must sum across all explanations available to t_p, hence is used for the FrenchRoyalty-200k dataset.

6.2.4 RGCN Loss Weighting each Possible Explanations

We also consider a loss function that weights the distance penalty term by the relevance score of each explanation. This approach again pushes the subject and object embeddings of t_p closer to the subject and object embeddings from all triples in E_p. However, this distance penalty term is weighted by the user score of each explanation in E_p, therefore making the embeddings in t_p more similar to the embeddings from explanations with high relevance scores. This equation is given by

$$\mathcal{L}_{weight} = \frac{1}{|\mathcal{T}^+|} \sum_{t_p \in \mathcal{T}^+} \quad \frac{1}{|E_p|} \sum_{e_i \in E_p} \quad \frac{1}{|e_i|} \sum_{t_j \in e_i} score(e_i) * \mathcal{P}(t_p, t_j), \tag{6.4}$$

$$\mathcal{L}_{weight} = \mathcal{L}_{weight} + \mathcal{L}_{RGCN}, \tag{6.5}$$

where $score(e_i)$ is the relevance score of e_i taking values between 0 and 1 of the explanation as provided by the FrenchRoyalty-200k dataset. Intuitively, large distances from embeddings in highly relevant explanations are given a larger penalty than large distance from embeddings in less relevant explanations. This loss function considers all triples in the ground truth explanation set E_p, but focuses on intuitive explanations. This loss function relies on the user assigned scores for each explanation included in the FrenchRoyalty-200k, and is thus limited only to applications on this dataset.

6.2.5 RGCN Loss Selecting the Highest Score

Lastly, we consider a loss function that penalizes subject and object embeddings in t_p that are far away from the subject and object embeddings of the best available explanation e_i, as determined by the given user relevance score. This equation is given by

$$\mathcal{L}_{max} = \frac{1}{|\mathcal{T}^+|} \sum_{t_p \in \mathcal{T}^+} \sum_{t_j \in e_i; score(e_i) = \max_{e \in E_p} score(e)} \frac{\mathcal{P}(t_p, t_j)}{|e_i|}, \tag{6.6}$$

$$\mathcal{L}_{max} = \mathcal{L}_{max} + \mathcal{L}_{RGCN}. \tag{6.7}$$

This loss function pushes the subject and object embeddings from t_p close to the subject and object embeddings in the best explanation e_i in the embedding space, making these embeddings more similar to each other than the standard RGCN. Embeddings from all other available ground truth explanations are not factored in. Similar to Equation 6.5, this loss function relies on the user assigned scores from the FrenchRoyalty-200k, hence this loss function is limited only to applications on this dataset.

6.3 Explanation Aware RGCN Benchmark Results and Evaluations

In this section, we evaluate the proposed loss functions on three datasets. The Royalty-20k dataset contains 3 types of predicates: *hasSpouse*, *hasSuccessor*, and *hasPredecessor*. The Royalty-30k dataset also contains 3 types of predicates including *hasSpouse*, *hasGrandparent*, and *hasParent*, where *hasParent* is only used to explain *hasGrandparent*. These datasets are used to evaluate explanation quality when there is one and only one explanation for each predicted triple. The FrenchRoyalty-200k

contains 6 types of predicates also based on family relations, *hasSpouse hasBrother*, *hasSister*, *hasGrandparent*, *hasChild*, and *hasParent*. Each predicted triple in this dataset includes all possible explanations, and is used to evaluate explanation quality when there are multiple to choose from. We compare all loss functions with a standard RGCN (using the loss function from Equation 2.14). We apply two state-of-the-art explanation methods, GNNExplainer Ying et al. [2019] (Equation 2.16), and ExplaiNE Kang et al. [2019] (Equation 2.15) to all RGCNs post hoc, and compare the quality of explanation generated by GNNExplainer and ExplaiNE.

For all experiments in this work, we fix the number of embedding dimensions to 10 as done in Halliwell et al. [2021b]. Additionally, for GNNExplainer, we use a learning rate of 0.001, and use 20 iterations for each observation on all datasets. Across all three datasets, we subset the data by each predicate, and report results on each subset. For example, on the Royalty-20k dataset, the Spouse column from Table 6.1 gives performance results on a subset of data using only *hasSpouse* triples and their associated explanations. Additionally, we report results on the full dataset, with all predicates included.

On the Royalty-20k and Royalty-30k datasets, predicted explanation performance is measured using the Jaccard score between each predicted and ground truth explanation. We also report precision, recall and F_1 scores, however, Halliwell et al. [2021c] recommend measuring explanation quality on these datasets using the Jaccard score. On the FrenchRoyalty-200k dataset, predicted explanation performance is measured using the Generalized Precision, Recall, and F_1 scores Kekäläinen and Järvelin [2002], along with the Max-Jaccard score Halliwell et al. [2021b].

		Royalty-20k Results				Royalty-30k Results		
Models	Metrics	Spouse	Successor	Predecessor	Full data	Spouse	Grandparent	Full data
RGCN	Accuracy	**0.737**	0.612	0.683	**0.77**	**0.737**	**0.654**	**0.687**
$\mathcal{L}_{sum}$	Accuracy	0.517	**0.989**	**0.717**	0.758	0.517	0.643	0.678
GNN Explainer								
	Precision	**0.657**	0.154	0.123	0.273	**0.657**	0.064	0.258
	Recall	**0.498**	0.151	0.121	0.22	**0.498**	0.089	0.297
with RGCN	F_1	**0.567**	0.153	0.122	0.251	**0.567**	0.074	0.276
	Jaccard	**0.34**	0.149	0.119	0.193	**0.34**	0.089	0.114
	Precision	**0.657**	**0.18**	**0.133**	**0.275**	**0.657**	**0.066**	**0.259**
	Recall	**0.498**	**0.176**	**0.13**	**0.234**	**0.498**	**0.092**	**0.298**
with $\mathcal{L}_{sum}$	F_1	**0.567**	**0.178**	**0.131**	**0.253**	**0.567**	**0.077**	**0.277**
	Jaccard	**0.34**	**0.171**	**0.128**	**0.194**	**0.34**	**0.092**	**0.154**
ExplaiNE								
	Precision	0.886	0.28	0.178	0.574	0.886	0.104	0.427
	Recall	0.668	0.272	0.176	0.493	0.668	0.139	0.471
with RGCN	F_1	0.762	0.276	0.177	0.531	0.762	0.119	0.448
	Jaccard	0.45	0.264	0.174	0.412	0.45	0.139	0.185
	Precision	**0.949**	**0.402**	**0.257**	**0.582**	**0.949**	**0.123**	**0.453**
	Recall	**0.712**	**0.391**	**0.251**	**0.501**	**0.712**	**0.164**	**0.506**
with $\mathcal{L}_{sum}$	F_1	**0.813**	**0.396**	**0.254**	**0.539**	**0.813**	**0.141**	**0.478**
	Jaccard	**0.474**	**0.38**	**0.245**	**0.419**	**0.474**	**0.164**	**0.286**

Table 6.1: Results on Royalty-20k, Royalty-30k datasets: Link prediction results for baseline RGCN and proposed loss functions, along with explanation evaluation for GNNExplainer and ExplaiNE. Highest scores per predicate denoted in bold.

6.3.1 Results with Non-Ambiguous Explanations

The top two rows of Table 6.1 report the link prediction results for the standard RGCN and the loss function in Equation 6.2 on the Royalty-20k dataset. We can see the standard RGCN outperformed the proposed approach on the full dataset, along with the *hasSpouse* subset. The proposed approach outperformed the standard RGCN on the *hasSuccessor* and *hasPredecessor* subsets.

Rows three and four of Table 6.1 report the results of GNNExplainer applied to a standard RGCN, and applied to the proposed RGCN in Equation 6.2 on the task of explainable link prediction. We observe the GNNExplainer applied to the proposed RGCN outperformed or matched the GNNExplainer applied to the baseline in terms of the Jaccard score on all subsets, and the full dataset.

Rows five and six of Table 6.1 report the results of ExplaiNE applied to a standard RGCN, and applied to the proposed RGCN in Equation 6.2. On the *hasSpouse*, *hasSuccessor* and *hasPredecessor* subsets, we find ExplaiNE when applied to the RGCN in Equation 6.2 improved all performance metrics. On the full dataset, we found using the proposed approach resulted in an improved Jaccard score.

The three rightmost columns of Table 6.1 report the performance metrics for the standard RGCN and the proposed approach on the Royalty-30k dataset. On the task of link prediction, we again find the proposed approach decreased the accuracy on all subsets, including the full dataset.

Rows three and four of Table 6.1 report the results of GNNExplainer applied to the baseline RGCN, and proposed RGCN on the task of explainable link prediction. We find equal or better performance across all metrics. The precision, recall, and F_1 score remain relatively unchanged on the full dataset.

Rows five and six of Table 6.1 report the results of ExplaiNE applied to the

baseline RGCN, and the proposed RGCN. Here we see improved performance on all metrics, and across all data subsets, including the full dataset. We observe a large increase in Jaccard score on the full dataset.

6.3.2 Results with Non-Unique Explanations

The top four rows of Table 6.2 report the link prediction results of the baseline and proposed methods. In general, the baseline RGCN from Equation 2.14 outperformed the proposed methods in terms of accuracy on the task of link prediction, with the exception of the *hasSpouse*, *hasSister*, and *hasChild* subsets.

Rows five through eight of Table 6.2 report the results of GNNExplainer applied the baseline RGCN, and the proposed approaches from Equations 6.3, 6.5, 6.7 on the task of explainable link prediction. Overall, we found all approaches had similar performance metrics, with two proposed approaches having a small increase in Max-Jaccard score on the full dataset.

Rows nine through twelve of Table 6.2 report the results of ExplaiNE applied to the baseline RGCN, and the proposed approaches on the task of explainable link prediction. We found the proposed approach improved performance on almost all metrics. Most notably, a large increase in Max-Jaccard score on the full dataset and *hasSister* subset.

6.4 Error Analysis: Quantitative Evaluation of Explanations

In this section, we evaluate the errors made by GNNExplainer and ExplaiNE, after being applied post hoc to the baseline RGCN, and the models using Equations 6.2, 6.3, and 6.5. In this work, we define an error to be a predicted explanation with a Jaccard score strictly less than 1 (for the Royalty-20k and Royalty-30k datasets), and a Max-Jaccard score strictly less than 1 for the FrenchRoyalty-200k dataset.

Models	Metrics	FrenchRoyalty-200k Results						
		Spouse	Brother	Sister	Grandparent	Child	Parent	Full data
$\mathcal{L}_{RGCN}$	Accuracy	0.935	**0.909**	0.853	**0.858**	0.792	**0.838**	**0.928**
$\mathcal{L}_{sum'}$	Accuracy	**0.973**	0.864	**0.999**	0.599	0.8	0.639	0.877
$\mathcal{L}_{max}$	Accuracy	0.966	0.75	0.824	0.648	0.793	0.697	0.878
$\mathcal{L}_{weight}$	Accuracy	0.966	0.909	0.971	0.615	**0.801**	0.706	0.897
GNN Explainer								
	G. Precision	**0.246**	0.323	**0.34**	0.162	0.142	**0.131**	0.109
with	G. Recall	**0.415**	0.333	0.353	0.162	0.167	**0.154**	**0.119**
$\mathcal{L}_{RGCN}$	G. F_1	**0.302**	0.327	0.344	0.162	0.15	**0.139**	0.112
	Max-Jaccard	**0.256**	**0.345**	0.299	0.128	0.16	**0.161**	0.109
	G. Precision	0.243	**0.324**	**0.34**	0.162	0.142	0.13	0.108
with	G. Recall	0.411	**0.335**	0.353	0.162	0.166	**0.154**	0.118
$\mathcal{L}_{sum'}$	G. F_1	0.299	0.328	0.344	0.162	0.14	0.138	0.111
	Max-Jaccard	0.254	**0.345**	0.299	0.128	0.161	0.16	0.109
	G. Precision	**0.246**	**0.324**	**0.34**	**0.164**	0.143	0.128	0.108
with	G. Recall	0.414	**0.335**	0.353	**0.164**	0.167	0.151	0.118
$\mathcal{L}_{max}$	G. F_1	**0.302**	0.328	0.344	**0.164**	0.151	0.136	0.112
	Max-Jaccard	0.255	**0.345**	0.299	**0.13**	0.161	0.159	**0.11**
	G. Precision	0.243	**0.324**	0.328	0.163	**0.144**	0.13	**0.11**
with	G. Recall	0.411	**0.335**	0.342	0.163	**0.168**	0.153	0.12
$\mathcal{L}_{weight}$	G. F_1	0.299	0.328	0.333	0.163	**0.152**	0.138	**0.113**
	Max-Jaccard	0.254	**0.345**	0.299	0.129	**0.162**	**0.161**	**0.11**
ExplaiNE								
	G. Precision	0.336	0.48	0.379	0.234	0.221	0.255	**0.192**
with	G. Recall	0.637	0.49	0.418	0.234	0.279	0.27	0.218
$\mathcal{L}_{RGCN}$	G. F_1	0.435	0.483	0.392	0.234	0.24	0.26	0.2
	Max-Jaccard	0.363	0.504	0.417	0.201	0.241	0.27	0.193
	G. Precision	**0.37**	**0.585**	**0.536**	**0.244**	**0.231**	0.247	0.188
with	G. Recall	**0.726**	**0.605**	**0.667**	**0.244**	**0.291**	0.258	**0.232**
$\mathcal{L}_{sum'}$	G. F_1	**0.488**	**0.591**	**0.58**	**0.244**	**0.251**	0.25	**0.203**
	Max-Jaccard	**0.41**	**0.598**	**0.539**	**0.211**	**0.246**	0.273	**0.209**
	G. Precision	0.338	0.433	0.487	0.224	0.224	**0.267**	0.177
with	G. Recall	0.66	0.471	0.592	0.224	0.283	**0.286**	0.213
$\mathcal{L}_{max}$	G. F_1	0.444	0.444	0.522	0.224	0.243	**0.273**	0.189
	Max-Jaccard	0.377	0.523	0.402	0.189	0.242	**0.303**	0.196
	G. Precision	0.351	0.538	0.485	0.232	0.227	0.263	0.189
with	G. Recall	0.69	0.567	0.629	0.232	0.287	0.275	0.23
$\mathcal{L}_{weight}$	G. F_1	0.463	0.547	0.533	0.232	0.247	0.267	**0.203**
	Max-Jaccard	0.39	0.557	0.407	0.196	0.243	0.295	0.208

Table 6.2: Results on FrenchRoyalty-200k: Link prediction results for baseline RGCN and proposed model, along with explanation evaluation for GNNExplainer and ExplaiNE. Highest scores in bold, and G. being an abbreviation for Generalized.

Most Frequently Predicated Predicate									
		ExplaiNE with $\mathcal{L}_{sum}$		ExplaiNE with $\mathcal{L}_{RGCN}$		GNNExplainer with $\mathcal{L}_{sum}$		GNNExplainer with $\mathcal{L}_{RGCN}$	
Dataset	Predicate	Most Frequent Predicate	% of Error	Most Frequent Predicate	% of Error	Most Frequent Predicate	% of Error	Most Frequent Predicate	% of Error
Royalty – 20*k*	*hasSpouse*	*hasSpouse*	100%	*hasSpouse*	100%	*hasSpouse*	100%	*hasSpouse*	100%
	hasSuccessor	*hasPredecessor*	94%	*hasPredecessor*	67%	*hasSuccessor*	52%	*hasPredecessor*	52%
	hasPredecessor	*hasPredecessor*	64%	*hasPredecessor*	55%	*hasPredecessor*	57%	*hasPredecessor*	51%
Royalty – 30*k*	*hasSpouse*	*hasSpouse*	100%	*hasSpouse*	100%	*hasSpouse*	100%	*hasSpouse*	100%
	hasGrandparent	*hasParent*	56%	*hasGrandparent*	55%	*hasGrandparent*	64%	*hasGrandparent*	64%
FrenchRoyalty 200*k*	*hasSpouse*	*hasSpouse*	92%	*hasSpouse*	84%	*hasSpouse*	51%	*hasSpouse*	50%
	hasBrother	*hasParent*	72%	*hasGrandparent*	53%	*hasGrandparent*	47%	*hasGrandparent*	45%
	hasSister	*hasParent*	60%	*hasParent*	53%	*hasGrandparent*	32%	*hasGrandparent*	32%
	hasGrandparent	*hasGrandparent*	44%	*hasGrandparent*	56%	*hasGrandparent*	57%	*hasGrandparent*	56%
	hasChild	*hasParent*	30%	*hasParent*	33%	*hasGrandparent*	41%	*hasGrandparent*	41%
	hasParent	*hasParent*	45%	*hasParent*	46%	*hasGrandparent*	37%	*hasGrandparent*	38%

Table 6.3: Most frequent predicate across incorrectly predicted explanations, along with the percentage of error by subset. Note $\mathcal{L}_{sum'}$ is used for FrenchRoyalty-200k.

6.4.1 Royalty-20k

The top row of Table 6.3 gives a breakdown of each explanation method's most frequent error by subset for the Royalty-20k dataset when applied to $\mathcal{L}_{RGCN}$ and $\mathcal{L}_{sum}$. Each row reports the most frequent predicate, and the percentage of errors this predicate occured in. For example, under the *hasSpouse* subset, the most common predicate across ExplaiNE's incorrectly predicted explanations (when applied to the RGCN in Equation 6.2) 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. This can occur on the *hasSpouse* subset, as under this subset, there is only one possible predicate to predict (*hasSpouse*).

On the Royalty-20k dataset, we can see on the *hasSuccessor* subset that the 94% of ExplaiNE with $\mathcal{L}_{sum}$ errors contained the *hasPredecessor* predicate. This type of error occurs when the subject and/or object in the predicted explanation are incorrect. We can deduce this due to the fact that on the *hasSuccessor* dataset, the RGCNs and explanation methods only observe two predicates, *hasSuccessor* and *hasPredecessor*. GNNExplainer when applied to both RGCNs however produce more uniform errors, where 52% of errors occurred by using the wrong subject and/or object, and the remaining errors occurred by identifying the wrong predicate. For GNNExplainer applied to both RGCNs, we observe a similar phenomenon on the *hasPredecessor* subset as well.

Note there are three types of explanation errors, one where the predicate in the predicted explanation is incorrect, one where the subject and/or object in the predicted explanation is incorrect, or both. From Tabe 6.3, we can see that ExplaiNE, when applied to the RGCN from $\mathcal{L}_{sum}$, has an increased number of errors using the wrong subject and object on the *hasSuccessor* subset. Recall each *hasSuccessor* pred-

Most Frequently Missing Predicate						
Dataset	Predicate	Ground Truth	ExplaiNE with $\mathcal{L}_{sum}$ % Missing	ExplaiNE with $\mathcal{L}_{RGCN}$ % Missing	GNNExplainer with $\mathcal{L}_{sum}$ % Missing	GNNExplainer with $\mathcal{L}_{RGCN}$ % Missing
Royalty – 20k	*hasSpouse*	*hasSpouse*	0%	0%	0%	0%
	hasSuccessor	*hasPredecessor*	6%	33%	52%	48%
	hasPredecessor	*hasSuccessor*	64%	55%	57%	49%
Royalty – 30k	*hasSpouse*	*hasSpouse*	0%	0%	0%	0%
	hasGrandparent	*hasParent*	44%	55%	64%	64%
		hasParent	44%	55%	64%	64%

Table 6.4: 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)

icate has an associated *hasPredecessor* ground truth. Here, the proposed approach produces more errors using the *hasPredecessor* predicate.

The first row of Table 6.4 reports the most frequently missing predicate from the explanation method's errors for the Royalty-20k dataset. 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, 6% of ExplaiNE's errors (when applied to $\mathcal{L}_{sum}$) did not contain *hasPredecessor*.

Lastly, Figures 6.1a and 6.1b show histograms of the most frequently predicted predicate amongst errors for ExplaiNE and GNNExplainer with $\mathcal{L}_{sum}$ on the Royalty-20k dataset. We can see GNNExplainer made a similar number of errors on the *hasPredecessor*, and *hasSuccessor* subsets, confirming the findings from Tables 6.3

and 6.4.

6.4.2 Royalty-30k

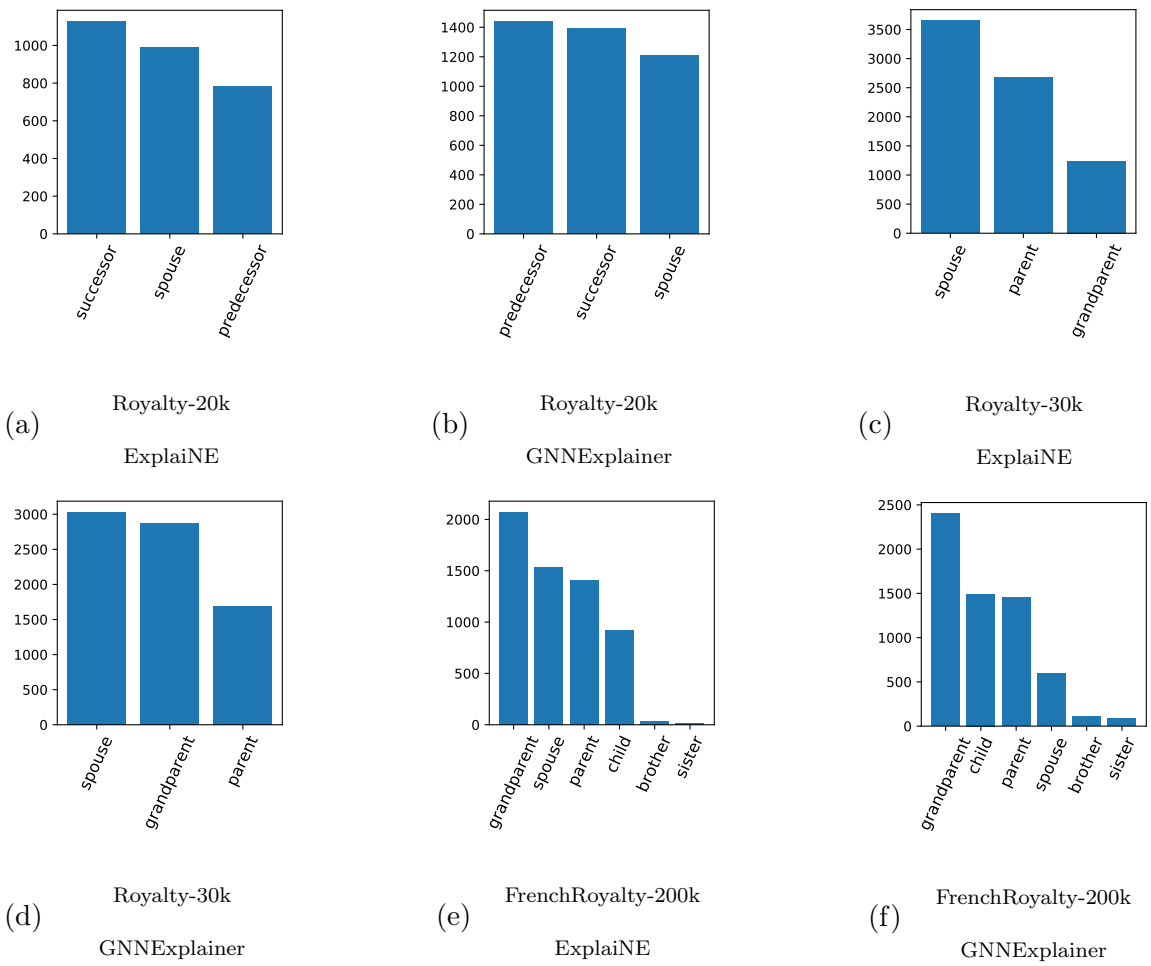
The second row of Table 6.3 gives a breakdown of each explanation method's most frequent error by subset for the Royalty-30k dataset when applied to $\mathcal{L}_{RGCN}$ and $\mathcal{L}_{sum}$. After applying ExplaiNE to $\mathcal{L}_{sum}$, we can see on the *hasGrandparent* subset, the most frequently predicted predicate was *hasParent*, accounting for 56% of errors. Conversely, for ExplaiNE with the baseline $\mathcal{L}_{RGCN}$, the most frequently predicted predicate is *hasGrandparent*. We can conclude from this that the explanation aware loss function $\mathcal{L}_{sum}$ changed the most frequent type of error made by ExplaiNE. Rather than predict the wrong predicate, the explanation aware loss instead produces errors using the correct predicate but wrong subject and/or objects.

The second row of Table 6.4 reports the most frequently missing predicate from each explanation method's errors for the Royalty-30k dataset. On the *hasGrandparent* subset, we find a decreased number of errors missing the *hasParent* explanation, consistent with Table 6.3. In general, we found GNNExplainer when applied to an explanation aware RGCN had minimal changes in errors metrics.

6.4.3 FrenchRoyalty-200k

The last row of Table 6.3 gives a breakdown of each explanation method's most frequent error by subset for the FrenchRoyalty-200k dataset when applied to $\mathcal{L}_{RGCN}$ and $\mathcal{L}_{sum'}$. On the *hasBrother* subset, we can see the errors produced by ExplaiNE with $\mathcal{L}_{sum'}$ results in errors using the *hasParent* predict, instead of *hasGrandparent* produced by the baseline.

Figures 6.1e and 6.1f show frequency counts of the most frequently predicted predicates amongst predictions made by $\mathcal{L}_{sum'}$ with a Max-Jaccard score less than 1. We



(a) Royalty-20k ExplaiNE (b) Royalty-20k GNNExplainer (c) Royalty-30k ExplaiNE

(d) Royalty-30k GNNExplainer (e) FrenchRoyalty-200k ExplaiNE (f) FrenchRoyalty-200k GNNExplainer

Figure 6.1: RGCN with $\mathcal{L}_{sum}$: Predicate Frequency Count on Incorrectly Predicted Explanations on each Full Dataset.

can see both ExplaiNE and GNNExplainer's most frequently predicte predicate is *hasGrandparent*. Additionally, both explanation methods least frequently predicted predicate amongst errors were *hasBrother*, and *hasSister*. We found ExplaiNE had difficulty predicting *hasSpouse* explanations, while GNNExplainer had fewer *hasSpouse* errors, and more errors with *hasChild* explanations. The number of errors made by GNNExplainer on the *hasParent* and *hasChild* subsets were nearly equal.

Table 6.5 reports the distributions of user scores amongst the errors of GNNEx-

Models	Rule	User Scores							
		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
GNN Explainer with $\mathcal{L}_{RGCN}$	Spouse	0	0	0	28	0	0	**214**	48
	Brother	0	**18**	0	0	0	8	4	13
	Sister	5	0	0	0	0	6	**16**	7
	Grandparent	0	0	0	0	35	419	0	**805**
	Child	0	0	165	0	0	**541**	0	69
	Parent	0	135	0	0	0	**464**	0	48
	Full data	8	150	212	52	24	**1315**	186	1122
GNN Explainer with $\mathcal{L}_{sum'}$	Spouse	0	0	0	28	0	0	**214**	48
	Brother	0	**18**	0	0	0	7	5	13
	Sister	5	0	0	0	0	6	**16**	7
	Grandparent	0	0	0	0	40	408	0	**812**
	Child	0	0	172	0	0	**534**	0	72
	Parent	0	133	0	0	0	**466**	0	47
	Full data	7	146	216	53	22	**1296**	188	1141
ExplaiNE with $\mathcal{L}_{RGCN}$	Spouse	0	0	0	11	0	0	**252**	29
	Brother	0	13	0	0	0	0	1	**19**
	Sister	4	0	0	0	0	2	**14**	10
	Grandparent	0	0	0	0	28	**711**	0	506
	Child	0	0	203	0	0	**456**	0	115
	Parent	0	142	0	0	0	**470**	0	33
	Full data	4	240	262	34	11	**1441**	208	862
ExplaiNE with $\mathcal{L}_{sum'}$	Spouse	0	0	0	8	0	0	**263**	22
	Brother	0	4	0	0	0	0	2	**21**
	Sister	0	0	0	0	0	0	**18**	7
	Grandparent	0	0	0	0	16	**753**	0	468
	Child	0	0	196	0	0	**467**	0	115
	Parent	0	194	0	0	0	**430**	0	27
	Full data	3	222	269	12	8	**1379**	263	835

Table 6.5: Distributions of user scores amongst errors for $\mathcal{L}_{sum'}$ relative to the $\mathcal{L}_{RGCN}$ on FrenchRoyalty-200k. Ex. of ExplainE with $\mathcal{L}_{sum'}$'s incorrect predictions on the *hasSpouse* predicate, ExplaiNE unsuccessfully attempted to predict an explanation with a user score of 0.8 on 263 observations.

plainer and ExplaiNE applied to $\mathcal{L}_{sum'}$ and the baseline $\mathcal{L}_{RGCN}$ for each predicate subset. For example, on the *hasSpouse* subset, ExplaiNE applied to $\mathcal{L}_{sum'}$ unsuccessfully attempted to predict an explanation with a user score of 0.8 on 263 observations in the test set. From this table, we can see both explanations methods attempted many times but failed to predict explanations with user scores of 0.7. Both explanation methods do not always attempt to predict explanations with the highest user scores (0.9). Indeed there is an imbalance of user scores, with 0.7 being the most common user score assigned to an explanation Halliwell et al. [2021b]. We note the fact that these explanation methods do not always try to predict the best possible explanation (those with the highest user scores). A similar phenomenon is observed in Halliwell et al. [2021b]. On the full dataset, we can see ExplaiNE with $\mathcal{L}_{sum'}$ made more errors with user scores of 0.8 than its respective baseline. Additionally, GNNExplainer $\mathcal{L}_{sum'}$ made more errors with user scores of 0.9 than its respective baseline.

6.5 Discussion of Explanation Aware RGCN Benchmark Results

On all three datasets, we found the proposed approaches matched or increased the Jaccard (or Max-Jaccard) scores on ExplaiNE when training on the full dataset with all predicates included. We found however, the baseline RGCN outperformed the proposed approach on the task of link prediction on the same datasets. From these experiments, we observe a trade off between black box model performance and explainability. Including prior information from ground truth explanations into the embeddings of RGCNs improves the quality of explanations generated by ExplaiNE and GNNExplainer. However, this comes at the cost of predictive power. Our approach allows practitioners and researchers to find a balance between predictive power and model explainability that the standard RGCN is unable to provide.

Additionally, we found our approach had the biggest impact on ExplaiNE's explanations, and a minimal impact on GNNExplainer's explanations. Understanding why the proposed approach had a larger impact on ExplaiNE's performance metrics than that of GNNExplainer would require a further understanding of what properties of the graph the embedding has learned. We leave this task for future work.

We recognize the difficulties in predicting explanations, even after making improvements, Jaccard (and Max-Jaccard) scores were still low. In fact, we found many of the Jaccard scores to be less than 0.5. Applying explanation methods post hoc to a black box model creates difficulties in diagnosing errors in predicted explanations, as there are many possible sources of error. When an explanation method produces an incorrectly predicted explanation, there are no available techniques to our knowledge that can identify if the explanation method is flawed, or if the error is due to a bad embedding that has not capturing the necessary information. Recent research has raised a similar concern, and that explanation methods for black boxes can be misleading Rudin et al. [2021], Rudin [2019], Laugel et al. [2019], Lakkaraju and Bastani [2020], Dimanov et al. [2020].

The lack of significant changes in performance metrics of GNNExplainer is likely due to the large number of parameters used by the model for each observation. Perturbations to the RGCN embedding are less influential on the predicted explanation, hence we can conclude GNNExplainer is less dependent on the RGCN embeddings for explanation predictions than ExplaiNE.

This work contributes to being able to identify where in the pipeline errors are caused. Injecting knowledge into the graph embedding shows GNNExplainer's errors are likely due to its parameters learned and not the RGCN embeddings, where as ExplaiNE's error are due to the embeddings.

We are aware that there are few Knowledge Graphs providing a ground truth for

explanations, however we wanted to evaluate the impact of such knowledge on different methods before investing resources in campaigns to manually annotate Knowledge Graphs with explanations. This work focuses on the case of supervised explanation prediction, where ground truth explanations are available. We provide a theoretical study of the behaviour of several explanation methods in the presence of explanation aware embeddings.

6.6 Concluding Remarks on Explanation Aware RGCNs

In this chapter, we applied the explanation-constrained loss function similar to that of Rieger et al. [2020] to RGCNs for link prediction on Knowledge Graphs. We added a penalty term for subject and object embeddings far away from the subject and object embeddings found in the ground truth explanation. We compared several different explanation-constrained loss functions to a baseline RGCN, and evaluate performance on three datasets with ground truth explanations. Results showed improved performance of post hoc explanation methods. We performed an error analysis on the Royalty datasets, quantifying errors in terms of both data and semantics.

Chapter 7

SEQUENCE TO SEQUENCE MODELS FOR EXPLAINING RGCN-BASED LINK PREDICTIONS

In the previous chapter, we saw that even with prior knowledge injected into the embeddings, current state-of-the-art explanation methods struggled to generate accurate explanations. In particular, when several RDF triples appear in the explanations. To cope with this issue, in this chapter, we use a Sequence to Sequence (Seq2Seq) model to generate explanations for RGCNs on the task of link prediction on KGs, no matter the number of triples in the explanations. In order to properly apply and evaluate this model, we convert RDF triples from the Royalty-30k dataset into sequences, thus generating a corpus of synthetic explanations. Experiment results show significant improvements in the quality of explanation compared to state-of-the-art methods. Furthermore, we propose an approach to construct false RDF triples to guide the Seq2Seq model during training. We observe a further performance increase when training with these valid synthetic counter-examples. Lastly, we propose several sanity checks to verify the robustness of the Seq2Seq model.

Benchmarks on the Royalty-20k, Royalty-30k, and FrenchRoyalty-200k have shown that many state-of-the-art explanation methods do not accurately predict ground truth explanations, in particular, these methods struggled to generate accurate explanations when there are more than one explanation in the ground truth.

In this chapter, we propose a Sequence-to-Sequence model to generate explanations for RGCNs on the task of link prediction on KGs. We benchmark this approach, along with the aforementioned state-of-the-art explanation methods, and find significant performance improvements with the proposed Seq2Seq model, specifically on

explanations that require more than one triple in the ground truth. Furthermore, we propose an approach to constructing false triples used during training to improve performance. Lastly, we propose several sanity checks to verify the robustness of the Seq2Seq model.

We propose to focus on Seq2Seq models to generate explanations for RGCNs. Indeed a Transformer could be considered in this work, however the main drawback of Transformers is that the architecture has to be very large in order to preserve memory across long sequences, at the cost of requiring large amounts of training data. When sequences are relatively short, for example, in the dataset used in this paper, a model built to handle long sequences is not necessary, and Seq2Seq models are competitive. To the best of our knowledge, Seq2Seq models have not been previously applied on Knowledge Graphs to learn to generate explanations.

7.0.1 Contributions

In this chapter, our first contribution is to propose the generation of a synthetic corpus explaining links from a Knowledge Graph. The sequences of the corpus include explained true links and example of false links. We then train a Seq2Seq model on this ground truth to predict and explain links at the same time, initializing a sequence with a candidate link and asking the model to complete that sequence. We demonstrate the performance of this approach against state-of-the-art methods, and perform several sanity checks to verify model robustness.

7.1 Knowledge Graphs and their explanations as a Corpus

In this section, we describe the process of converting a KG to a corpus to be used by NLP models, and a description of how counter-examples are constructed.

7.1.1 Generating a Synthetic Corpus from a KG

As shown in previous chapters, state-of-the-art explanation methods struggle to predict explanations with multiple triples, such as the *hasGrandparent* relation, defined as $hasGrandparent(X, Z) \leftarrow hasParent(X, Y) \wedge hasParent(Y, Z)$. Indeed Chapters 4, 5, and 6 showed that even if ExplaiNE can identify more than one pair of triples, the largest gradients when dealing with multiple triples were not, most of the time, in the ground truth explanation. ExplaiNE is almost always producing large gradients when triples are first degree neighbors, while for some predicates like *hasGrandparent*, one of the triples is not a first degree neighbor but a second degree neighbor. This behavior is confirmed in Table 7.1 by the low value of Jaccard similarity for ExplaiNE on the *hasGrandparent* predicate. Similarly, GNNExplainer's approach of learning a mask over the adjacency matrix struggles to learn a subgraph that included multiple triples of both first and second degree neighbors. In Chapters 4, we saw performance metrics drop on predicates like *hasGrandparent*. Similar to ExplaiNE, this behavior is confirmed in Table 7.1 by the low Jaccard metric for GNNExplainer on the *hasGrandparent* predicate.

Both ExplaiNE and GNNExplainer operate on the adjacency matrix of a KG. For a small KG such as the Roylaty-30k dataset, the adjacency matrix contains more than 395 million elements. For each observation, the number of parameters learned by GNNExplainer is equal to the size of the adjacency matrix. ExplaiNE computes the gradient of the scoring function for each element of the adjacency matrix. Operating on the adjacency matrix could be why these explanation methods struggled to produce accurate explanations when there are more than one triple in the ground truth.

We thus want to design a model which is not based on the adjacency matrix, while still being able to generate multiple triples as an output. As mentioned pre-

viously, Seq2Seq models are perfect candidates in this case, but then training sets have to be adapted to train a sequential model. We thus propose to transform the Royalty-30k KG into a synthetic corpus of logical sequences describing implications between triples and their explanations. We then use this corpus to learn a model to complete sequences, and predict missing triples together with their explanations. We consider each subject, object and predicate in the KG as a token to be used in a Seq2Seq model. In other words, each triple in the Royalty-30k KG is converted into a sequence, as well as its associated ground truth explanation, resulting in a training set of pairs of sequences. For example, the triple (*Louis VII*, *hasSpouse*, *Adela of Champagne*) becomes the sequence [*Louis VII*, *hasSpouse*, *Adela of Champagne*], and its associated ground truth explanation (*Adela of Champagne*, *hasSpouse*, *Louis VII*) becomes also a sequence starting with the truth value of the triple and its explanation [*Start*,$True$,*Adela of Champagne*, *hasSpouse*, *Louis VII*,*End*]. The $True$ token in the explanation represents whether or not (*Louis VII*, *hasSpouse*, *Adela of Champagne*) is a true or false triple. Converting the KG into a corpus allows us to apply sequential models to a Knowledge Graph in a controlled manner, such an example can be seen in Figure 7.1. Our Seq2Seq model functions as language translation model: it encodes the initial triples in a latent representation (the last blue LSTM Encoder Cell on Figure 7.1), which is then input to the decoder that "translates" the encoded triples into its explanation.

7.1.2 Adding Valid Counter-Examples to the Corpus

Recall a true triple is defined as one that is a known fact, such as (*Louis VII*, *hasSpouse*, *Adela of Champagne*). Similarly, a false triple is one that is known to not be a fact, such as (*Louis VII*, *hasGrandparent*, *Adela of Champagne*). In the original RGCN paper Schlichtkrull et al. [2018], a corresponding false triple is generated for

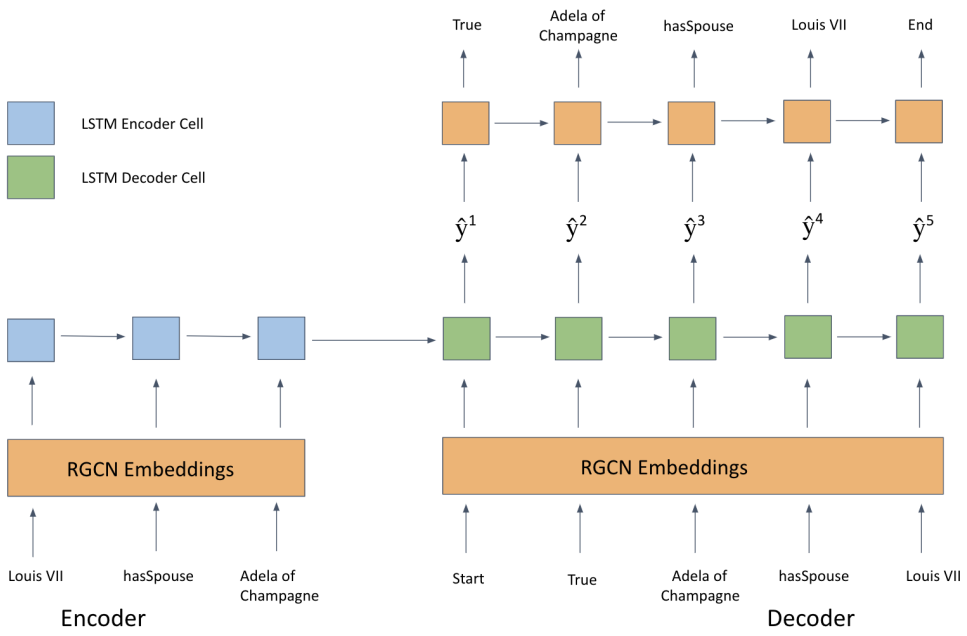


Figure 7.1: Seq2Seq: Generating Explanations on Positive Triples

each positive (true) triple in the training set. This is done by randomly selecting the subject or object of each positive triple, and replacing the selected entity with a randomly selected entity. One drawback to this approach is that randomly replacing one of the entities could, unfortunately, construct a triple that is true. Consequently, the RGCN would then be trained with some true triples that are wrongly seen as false.

In this work, we construct false triples to guide the Seq2Seq model during training. The Royalty-30k dataset includes explanations for each observation, and explanations are constructed using rules. Knowing that the only available predicates in the Royalty-30k dataset are *hasSpouse*, *hasParent*, and *hasGrandparent*, we can construct false triples to help training. For every *hasSpouse* triple, $hasSpouse(X, Y)$, we construct a new false triple $hasParent(X, Y)$ using the same subject and object. This creates a false triple because two spouses X and Y cannot be the parent of one another (we know there are no cases of this in this particular KG). For each true

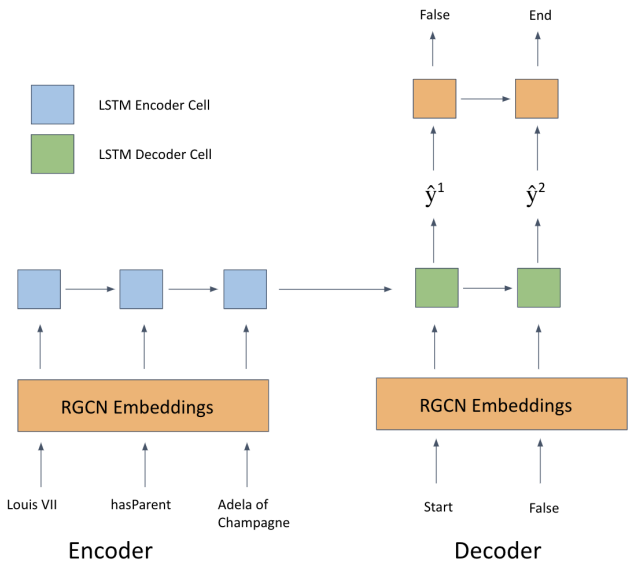


Figure 7.2: Seq2Seq: Training with counter-examples

triple in the dataset, we generate an associated false triple to be used in training. As an example, for some false triple (*Louis VII*, *hasParent*, *Adela of Champagne*) we generate the sequence starting with [*Louis VII*, *hasParent*, *Adela of Champagne*], and followed by [*Start*,*False*,*End*]. Note that no false triples are used during test time.

During training, when a false triple is passed through the encoder, the decoder output at the first time step is a *True* or *False* token. For positive triples, the Seq2Seq model outputs an explanation for why some predicate exists between the two entities in the candidate triple. For negative triples however, no explanation is generated by the decoder. If the model determines the candidate triple is false, only the *False* token is outputted by the decoder, and the generation of the decoded sequence is terminated. An example of this can be seen in Figure 7.2. The candidate triple (*Louis VII*, *hasParent*, *Adela of Champagne*) is a false triple, as we can see from Figure 7.1, that (*Louis VII*, *hasSpouse*, *Adela of Champagne*) is a true triple. Each subject, predicate, and object of the candidate triple is passed into the encoder, and

the decoder outputs a $False$ token, indicating the candidate triple is false, and the sequence generation ends.

7.2 Sequence to Sequence Models for Explaining Link Predictions in Knowledge Graphs

7.2.1 Task Description

Given a *subject* and *object*, the task is to predict the correct *predicate* that should exist between the two entities. Additionally, we want to identify which entities and relations in the graph are influencing the existence of that particular *predicate*, that is, we want to identify an explanation for why that *predicate* exists between the two entities. We are interested in a model that learns the structure of the relationship between a given *predicate* and its explanation, without relying on any prior assumptions about the structure. In other words, can we learn the fact that $hasGrandparent(X, Y) \leftarrow hasParent(X, P) \wedge hasParent(P, Y)$, using only observed data. Of course, a rule based link prediction approach could easily achieve the best performance on a rule-generated dataset such as the the Royalty datasets. However, the sole purpose of these datasets is to evaluate the quality of explanation produced from the explanation methods of embedding based link prediction approaches.

Formally, let $t_i \in X$ be a triple in the training/test set, let $e_i = \{t_1, \ldots, t_n\}$, be one of the possible ground truth explanations for t_i, where $e_i \in E_i$, and E_i is the set of all possible ground truth explanations for t_i. Let $\hat{e}_i = \{\hat{t}_1, \ldots, \hat{t}_n\}$, be a predicted explanations for a predicted triple $\hat{t}_i$. We define the prediction task as learning the function $f(\hat{t}_i) = (\hat{y}, \hat{e}_i)$ where $\hat{y} \in \{True, False\}$ indicates if $\hat{t}_i$ is predicted correctly, and $\hat{e}_i$ is the explanation of the predicted triple. For the Royalty-30k dataset,$e_i = E_i$, in other words, there is only one possible ground truth explanation for each triple t_i.

Recall a Seq2Seq model computes a conditional probability of the sequence $y^1, ..., y^T$ with T time steps given the input X, i.e. $P(y^1, ..., y^T|X)$. In our case, this could be modeled as computing the probability that some triple t_i is true, along with an explanation for t_i, i.e. $P(True, e_i|t_i)$.

7.2.2 Seq2Seq Architecture

The approach we propose to generate explanations is to adapt an NLP approach such as a Sequence to Sequence (Seq2Seq) model to produce explanations as synthetic sentences describing the Knowledge Graph. We propose the use of a Seq2Seq model that learns to generate explanations for RGCNs on the task of link prediction on Knowledge Graphs. This architecture takes a triple that we want to predict and explain as input (candidate triple), and the model is trained to output either the $True$ of $False$ token to determine if the candidate triple is a true of false triple, and either a subject, predicate, or object at each remaining time step. The resulting generated sequence serves as the explanation for why a predicate exists between the subject and object of the candidate triple.

Figure 7.1 gives an overview of the architecture. A candidate triple passed into the encoder t_i = (*Louis VII*, *hasSpouse*, *Adela of Champagne*). For any given triple passed into the encoder, a lookup operation is performed on the subject, predicate, and object to obtain the graph embedding. In this model, the graph embeddings are taken from a pre-trained RGCN, trained on the task of link prediction. The graph embeddings are then passed into the LSTM layer. The encoder, denoted Enc, takes a triple as input $Enc(t_i)$, and outputs the cell and hidden states, denoted $\mathbf{c}$ and $\mathbf{h}$, respectively. Additionally, the proposed model is asked to output a $True$ or $False$ in the first time step, to predict whether the input triple is a true or false triple.

The decoder of the Seq2Seq model is trained to generate the ground truth expla-

nation e_i associated with each input triple t_i. The decoder takes the ground truth explanation e_i as an input, along with the cell and hidden states $\mathbf{c}$ and $\mathbf{h}$ from the encoder. A lookup operation is performed for the subject, predicate, and object embeddings of the explanation, similar to the encoder. Additionally, at the first time step after the *Start* token, the Seq2seq model is trained to output whether the input triple is a true of false triple. If the input triple is true, the following time steps output the explanation for the input triple. Note during testing, only the *Start* token is passed as input to the decoder. We denote the decoder *Dec*, following the previous example, let e_i = (*Adela of Champagne*, *hasSpouse*, *Louis VII*), hence $Dec(e_i, \mathbf{h}, \mathbf{c}) = P(True, e_i|t_i)$.

For all experiments in this paper, we use a Bidirectional LSTM Schuster and Paliwal [1997] for the encoder and decoder. The main drawback to using an LSTM (and Bidirectional LSTM) is the ability to maintain memory across long sequences. On the Royalty-30k, the longest possible explanation occurs when explaining the *hasGrandparent* relation, which requires 2 *hasParent* triples. Therefore, the longest possible sequence the model has to predict is 9 tokens in total, including 6 for the two *hasParent* triples, 2 for the *Start* and *End* tokens, and 1 for the $True$ or $False$ token. An LSTM will not have memory issues with sequences of this length. Note, the Seq2Seq model does not perform updates to the graph embeddings. We freeze the graph embedding parameters and use pre-trained embeddings for best performance. This allows the Seq2Seq model to learn the structure of explanations.

Using a Seq2Seq model to learn to generate explanations has several advantages. First, a Seq2Seq model can learn to generate sequences of arbitrary length. The Royalty-30k dataset requires an explanation method to predict a set of triples, in some cases more than one. Second, the internal length of the recurrent neural network (here a bi-LSTM) is fixed and not linked to the size of the input or output data,

while it is the case for other architectures such as Convolutional Neural Networks, or Transformers, that are constrained by the size of the analysis window. Thus such models are really light in terms of resources required for training and inference.

ExplaiNE computes the gradient of the adjacency matrix with respect to the score. This approach does not learn a representation of the structure of explanations, instead computing a gradient that is used to rank explanations. GNNExpainer learns a mask over the adjacency matrix to identify a relevant subgraph. This explanation method is not able to learn the structure of explanations. A Seq2Seq model uses the information from the RGCN embeddings, and uses parameters from the Bidirectional LSTMs to learn the structure of explanations.

7.3 Evaluation of Seq2Seq Explanations

7.3.1 Protocol and Metrics for Seq2Seq Explanations

We compare the propose Seq2Seq model against the same explanation methods from previous chapters, ExplaiNE (Equation 2.15) and GNNExplainer (Equation 2.16). For both ExplaiNE and GNNExplainer, explanations are extracted using the top k scores, where $k = 1$ for *hasSpouse* triples, and $k = 2$ for *hasGrandparent* triples. We include results of a Seq2Seq model trained without using counter-examples. For all experiments, we set the number of RGCN dimensions to 200, all models are trained from 1500 to 2000 epochs, and the learning rate is 0.001. These are the best performing hyperparameters obtained by validation.

Regarding the performance metrics, we report the Jaccard similarity between the predicted and ground truth explanations (Equation 4.1), precision (Equation 4.2), recall (Equation 4.3) and F_1-Score (Equation 4.4). As mentioned in Chapter 4, a recall of 1 can be achieved by outputting the entire KG as a prediction, thus the

resulting F_1-Score can be biased. The accuracy of the RGCN counts the number of times on average the model predicts the correct relation between two entities. The Seq2Seq accuracy however determines how often on average the model predicts the correct $True$ or $False$ token in the first time step after the $Start$ token.

7.3.2 Seq2Seq Benchmark Results

The results of the experiment can be found in Table 7.1. Each predicate column gives the results of all performance metrics when trained only on triples with that given predicate. The top row gives the accuracy of the RGCN trained on the task of link prediction. The following rows give the results of each explanation method. The best Accuracy, F_1 and Jaccard scores per predicate are denoted in bold.

Chapters 4, 5, and 6 showed that GNNExplainer and ExplaiNE struggled to explain when there was more than one triple in the explanation. Using a different number of dimensions, we also find that both ExplaiNE and GNNExplainer performed best on the *hasSpouse* subset, and performed their worst on the *hasGrandparent* subset, where 2 triples are required to explain each candidate triple, plus one of the triple is of second degree.

Overall we see the Seq2Seq model outperformed both GNNExplainer and ExplaiNE in terms of Jaccard similarity on all subsets, and the full dataset. Additionally, we find large performance jumps on precision, recall, and F_1 score, with the exception of the F_1 score on the *hasSpouse* subset. For the Seq2Seq model trained without counter-examples, we see a small decrease in Jaccard similarity on the *hasSpouse* subset, and a slight increase in performance on the *hasGrandparent* subset. This is due to the way false triples are constructed. Whenever a *hasSpouse(X, Y)* triple is observed, a false *hasParent(X, Y)* triple is constructed using the same entities. For the *hasSpouse* subset, training is done using positive triples that use *hasSpouse*, and false triples that

Models	Metrics	Predicates		
		Spouse	Grandparent	Full data
RGCN	Accuracy	0.991	0.837	0.85
GNN Explainer	Precision	0.324	0.031	0.074
	Recall	0.355	0.042	0.081
	F_1	0.339	0.035	0.077
	Jaccard	0.171	0.042	0.075
ExplaiNE	Precision	0.761	0.067	0.192
	Recall	0.566	0.089	0.21
	F_1	**0.649**	0.076	0.201
	Jaccard	0.371	0.089	0.209
Seq2Seq with counter-examples	Accuracy	0.986	0.978	0.948
	Precision	0.473	0.364	0.404
	Recall	0.464	0.702	0.552
	F_1	0.468	0.479	**0.467**
	Jaccard	**0.455**	0.702	**0.551**
Seq2Seq without counter-examples	Accuracy	**1.0**	**1.0**	**1.0**
	Precision	0.454	0.369	0.298
	Recall	0.452	0.719	0.451
	F_1	0.453	**0.488**	0.359
	Jaccard	0.451	**0.719**	0.448

Table 7.1: Seq2Seq benchmark results on Royalty-30k dataset: Link prediction results for RGCN, and explanation evaluation for GNNExplainer and ExplaiNE. Best Accuracy, F_1, and Jaccard scores per predicate denoted in bold.

use *hasParent*. In other words, no positive triples use *hasParent*, hence the model is not learning a representation of this predicate from positive examples. The same can be said for the *hasGrandparent* subset, as false triples contain the *hasSpouse* predicate, but the model does not learn from positive *hasSpouse* predicates. Training with the counter-examples shows its largest impact on the full dataset, where the model is asked to generate explanations for *hasSpouse* and *hasGrandparent* triples. On the

full dataset, the model learns from positive and negative examples from all predicates in the dataset. We see a large increase in Jaccard score on the Seq2Seq trained with counter-examples compared to the Seq2Seq trained without counter-examples. Note the accuracy is the highest on the Seq2Seq without counter-examples, as every triple in the training set contains the $True$ token in the time step immediately after the *Start* token. Additionally, only true triples are evaluated in the test set, hence this model is able to always correctly predict the $True$ token.

We find the Seq2Seq performed the best in terms of Jaccard similarity on the *hasGrandparent* subset. From this we see the Seq2Seq model is able to generate high quality explanations when there are multiple triples in the ground truth explanation. We argue this is the strength of using a Seq2Seq model to generate explanations.

7.3.3 Sanity Checks for Model Robustness

Recent research in the field of XAI has shown that explanation methods do not always give accurate explanations, and can be misleading Rudin et al. [2021], Rudin [2019], Dimanov et al. [2020], Halliwell et al. [2021b]. On the task of image classification, some researchers have proposed a series of sanity checks Adebayo et al. [2018] for saliency map algorithms to test if the saliency map is truly dependent on the model parameters. Many well known saliency map algorithms failed these sanity checks.

When an explanation method generates an incorrect explanation, it can be difficult to understand where the error came from. When GNNExplainer or ExplaiNE make a predicted explanation error, it is unclear if this error is a result of bad RGCN embebddings that have not captured the structure of the graph, or if the explanation method is flawed. We propose 2 sanity checks to verify the explanations generated by the Seq2Seq model are truly dependent on the ground truth explanations, and the RGCN embeddings. We report the Accuracy, which is the number of times on average

the model is able to correctly predict a $True$ or $False$ token in the first time step. We also report precision, recall, F_1 and Jaccard scores for explanation evaluation of the Seq2Seq model trained with counter-examples.

Shuffled Explanations Test

The first sanity check we perform is to train the Seq2Seq model using shuffled explanations. Ground truth explanations are randomly assigned to each candidate triple. That is, each candidate triple passed into the encoder is asked to generate an explanation that is selected randomly from the set of all ground truth explanations. If a Seq2Seq model generating explanations can generate accurate explanations while trained on random explanations, the model has learned noise. This test is used to determine whether or not the Seq2Seq model has learned noise, and if there is truly a relationship between the input triples and their ground truth explanations. This test takes inspiration from the Data Randomization Test Adebayo et al. [2018].

Random Embeddings Test

The second sanity check we perform is to train the Seq2Seq model using random RGCN embeddings. As described in earlier sections, the proposed approach uses an RGCN embeddings to capture information about the graph. These embeddings are not updated during training, hence not learned by the Seq2Seq model. In this sanity check, we randomly initialize RGCN embeddings, freeze them, then train the Seq2Seq model. This test is used to determine if the Seq2Seq model is incorporating information from the graph into each explanation, or if the sequence model does not require graph embeddings in order to learn to generate explanations. This test takes inspiration from the Model Parameter Randomization Test Adebayo et al. [2018].

Models	Metrics	Predicates		
		Spouse	Grandparent	Full data
Shuffled Explanations	Accuracy	0.982	0.996	0.961
	Precision	0.0	0.0	0.0
	Recall	0.0	0.0	0.0
	F_1	0.0	0.0	0.0
	Jaccard	0.0	0.0	0.0
Random Embeddings	Accuracy	0.784	0.999	0.553
	Precision	0.0	0.004	0.0
	Recall	0.0	0.006	0.0
	F_1	0.0	0.005	0.0
	Jaccard	0.0	0.006	0.0

Table 7.2: Sanity Checks for Seq2Seq model with counter-examples

Sanity Checks Discussion

The results of the sanity checks can be found in Table 7.2. For the shuffled explanations test, we observe high accuracy, that is, the Seq2Seq model accurately predicts a $True$ of $False$ token as the first time step correctly. This makes intuitive sense as the Seq2Seq model parameters should be able to learn where the $True$ or $False$ tokens belong, but still should not be able to learn to generate explanations. We observe the precision, recall, F_1, and Jaccard scores are zero for all subsets. We can conclude the Seq2Seq model passed the shuffled explanations sanity check.

For the random embeddings test, we observe similar results to the shuffled explanations test. The model is able to accurately predict a $True$ or $False$ token as the first time step, but performs poorly on all explanation evaluation metrics. Again this makes intuitive sense, as the Seq2Seq parameters can learn where the $True$ and $False$ tokens belong, however, it cannot learn to generate explanations without information from the graph. We conclude the Seq2Seq model passed the random embeddings test.

7.3.4 Limitations

We recognize the limitations of the proposed Seq2Seq model that relies on the availability of a ground-truth. For instance, the construction of counter-examples requires knowledge of the rules used to generate the KG. Indeed this model relies on ground truth explanations which are not always available in real world settings. However, the purpose of this chapter is to demonstrate that Seq2Seq models can be used to generate explanations for RGCNs with competitive performance. We hope the ideas in this paper can help future researchers develop a Seq2Seq model that generates accurate RGCN explanations without using ground truth triples in training.

7.4 Concluding Remarks on Seq2Seq models for RGCN Explanations

In this chapter, we transformed selected aspects of a Knowledge Graph into a corpus of sequences to be used by recurrent neural models to predict and explain links. We proposed the use of a Seq2Seq model to learn to jointly generate predictions and explanations for RGCNs on the task of link prediction. Additionally, we proposed an approach to generate false triples in order to help train the Seq2Seq model. Empirical results show significant improvements over state-of-the-art explanation methods. Furthermore, we found the Seq2Seq model performed better on explanations with more than one triple in the ground truth explanation. Lastly, we propose several sanity checks to verify how sensitive the generated explanations are to the ground truth explanations and RGCN embeddings. We recognize that there are few KGs providing a ground truth explanations, however, with this chapter, we want to evaluate the ability to learn to generate explanations before investing resources in campaigns to manually or semi-automatically annotate KGs with such explanations. We showed it is possible to learn to predict and explain links in a KG as one task.

Chapter 8

CONCLUSION

8.1 Contributions

Deep learning models are too often treated as black boxes, where no insight is given as to why they make a prediction. Users of the deep learning model, including patients receiving a medical diagnosis, families applying for a bank loan, and machine learning practitioners debugging the model, want to know why the model has made a particular suggestion. Without an explanation as to why a result was obtained, users of the system will not trust it.

Recently, researchers have proposed explanation methods that explain to the user why a deep learning model makes a given decision. Without ground truth explanations, these explanation methods are not able to properly evaluate the quality of explanations returned to the user.

For current state-of-the-art explanation methods, there are no common datasets or scoring metrics available to quantitatively evaluate explanations. As a result, it is difficult to determine if an explanation method is producing accurate explanations, and when to prefer one method over another.

In this thesis, we address the lacking empirical evaluation of explanations from state-of-the-art explanation methods. Chapter 1 provides an introduction to the thesis, giving an overview of the need to explain and evaluate automated decisions, and outlines the contributions of this thesis. Chapter 2 gives an overview of the related work used in the thesis, including Knowledge Graphs, link prediction models, deep learning models, and state-of-the-art explanation methods. Chapter 3 motivates the

need for ground truth explanations, as explanation methods can produce inaccurate explanations and fool practitioners into thinking these inaccurate explanations are of high quality. In Chapter 4, we define two datasets, Royalty-20k, and Royalty-30k, with ground truth explanations, where each observation has one and only one ground truth. We propose the use of several scoring metrics, and benchmark two state-of-the-art explanation methods on these datasets. In Chapter 5, we define a dataset, FrenchRoyalty-200k, with ground truth explanations, where each observation includes all possible ground truth explanation. We propose the use of several scoring metrics derived from user scores calculated from a user experiment, and benchmark two state-of-the-art explanation methods on this dataset. In Chapter 6, we incorporate prior knowledge from ground truth explanations into RGCN embeddings. We find improved predicted explanation performance of post hoc explanation methods for RGCNs, despite a small decrease in classification performance. Lastly, Chapter 7 proposes a Seq2Seq model to generate RGCN explanations. Experimental results show significant performance improvements, notably on explanations that require more than one triple in the ground truth explanation. We verify the robustness of this model by proposing several sanity checks.

8.2 Perspectives

The work in this thesis provides several opportunities for future research. With multiple datasets including ground truth explanations, and several suitable scoring metrics, future researchers can now properly evaluate the explanations produced by their algorithms. One difficulty of applying an explanation method such as ExplaiNE or GNNExplainer post hoc to an RGCN is that it can be difficult to determine where explanation errors are coming from. That is, when a predicted explanation does not match the ground truth explanation, does this error come from the RGCN or

ExplaiNE/GNNExplainer? Has the RGCN learned a bad embedding representation, or is there an fundamental flaw with ExplaiNE or GNNExplainer? This research question relates to another open research question: when an RGCN is trained, what information is incorporated into the embedding? Indeed a practitioner or researcher may want to verify that the embedding captures the relationship between each triple and all possible ground truth explanations.

The Seq2Seq model proposed in Chapter 7 also provides opportunities for future work. The proposed model in the previous chapter uses pre-trained RGCN embeddings used in the sequence generating task. When training the RGCN embeddings with the Seq2Seq model, we observed poor explanation performance. We observed large performance improvements when using pre-trained RGCN embeddings. One possible extension of this model would be to identify why the model performs well with pre-trained RGCN embeddings, and find a way to adapt the model to learn RGCN embeddings.

The Seq2Seq model is tested only on the Royalty-30k dataset, where each observation has one and only one ground truth explanation. An interesting extension of this work would be to adapt this Seq2Seq model to the case of multiple explanations, where each observation has all possible ground truth explanations included. This would involve training the model on the FrenchRoyalty-200k dataset to output the best explanation. Additionally, this task may require incorporating the user assigned scores into the model's loss function, in order to assign a large penalty to low user scored explanations.

This approach uses counter-examples during training to help the sequence model learn. One interesting contribution to this approach could be to train the Seq2Seq model without counter-examples, while maintaining competitive performance. That is, can the architecture or loss function of the Seq2Seq model be adapted to have

similar performance metrics without having to train with counter-examples?

Lastly, a difficult but significant contribution could be made to the Seq2Seq model if the model could be learned to generate accurate explanations without using ground truth explanations. In other words, could the Seq2Seq model learn to generate accurate explanations without specifically feeding ground truth explanations into the decoder during training. This idea can be though of as an unsupervised approach, as the ground truth explanations exist in the dataset, but the model is not specifically trained on them. If this could be accomplished, the explanation method could then be applied to datasets from other domains where ground truth explanations are difficult to define, while still generating accurate explanations. The work completed in this thesis may provide the necessary tools for future researchers trying to answer these questions.

Bibliography

J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. In S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html.

J. Adebayo, M. Muelly, I. Liccardi, and B. Kim. Debugging tests for model explanations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/075b051ec3d22dac7b33f788da631fd4-Paper.pdf.

S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 2015. doi: 10.1371/journal.pone.0130140. URL https://doi.org/10.1371/journal.pone.0130140.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

M. Barati, Q. Bai, and Q. Liu. Mining semantic association rules from RDF data. *Knowl. Based Syst.*, 133:183–196, 2017. doi: 10.1016/j.knosys.2017.07.009. URL https://doi.org/10.1016/j.knosys.2017.07.009.

T. Berners-Lee. Linked data, 2006. URL https://www.w3.org/DesignIssues/LinkedData.html.

A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, 2013.

C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. Su. This looks like that: Deep learning for interpretable image recognition. In H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox,

and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/adf7ee2dcf142b0e11888e72b43fcb75-Abstract.html.

K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014. doi: 10.3115/v1/d14-1179. URL https://doi.org/10.3115/v1/d14-1179.

A. Choudhury and D. Gupta. A survey on medical diagnosis of diabetes using machine learning techniques. In *Recent developments in machine learning and data analytics*, pages 67–78. Springer, 2019.

O. Corby, A. Gaignard, C. Faron Zucker, and J. Montagnat. KGRAM Versatile Inference and Query Engine for the Web of Linked Data. In *IEEE/WIC/ACM Int. Conference on Web Intelligence*, Dec. 2012.

R. Cyganiak, D. Wood, and M. Lanthaler. RDF 1.1 Concepts and Abstract Syntax, Feb. 2014. URL http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

A. K. Debnath, R. L. L. de Compadre, G. Debnath, A. J. Shusterman, and C. Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34 2:786–97, 1991.

S. Dhankhad, E. Mohammed, and B. Far. Supervised machine learning algorithms for credit card fraudulent transaction detection: A comparative study. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 122–125, 2018. doi: 10.1109/IRI.2018.00025.

B. Dimanov, U. Bhatt, M. Jamnik, and A. Weller. You shouldn't trust me: Learning models which conceal unfairness from multiple explanation methods. In H. Espinoza, J. Hernández-Orallo, X. C. Chen, S. S. ÓhÉigeartaigh, X. Huang, M. Castillo-Effen, R. Mallah, and J. McDermid, editors, *Proceedings of the Workshop on Artificial Intelligence Safety, co-located with 34th AAAI Conference on Artificial Intelligence, SafeAI@AAAI 2020, New York City, NY, USA, February 7, 2020*, volume 2560 of *CEUR Workshop Proceedings*, pages 63–73. CEUR-WS.org, 2020. URL http://ceur-ws.org/Vol-2560/paper8.pdf.

J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, 2011. doi: 10.5555/1953048.2021068. URL https://dl.acm.org/doi/10.5555/1953048.2021068.

E. Dumitrescu, S. Hué, C. Hurlin, and S. Tokpavi. Machine learning for credit scoring: Improving logistic regression with non-linear decision-tree effects. *European*

Journal of Operational Research, 297(3):1178–1192, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2021.06.053. URL https://www.sciencedirect.com/science/article/pii/S0377221721005695.

L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730, 2015. doi: 10.1007/s00778-015-0394-1. URL https://doi.org/10.1007/s00778-015-0394-1.

L. A. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In D. Schwabe, V. A. F. Almeida, H. Glaser, R. Baeza-Yates, and S. B. Moon, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013*, pages 413–422. International World Wide Web Conferences Steering Committee / ACM, 2013. doi: 10.1145/2488388.2488425. URL https://doi.org/10.1145/2488388.2488425.

N. Halliwell. Evaluating Explanations of Relational Graph Convolutional Network Link Predictions on Knowledge Graphs. In *AAAI 2022 - 36th AAAI Conference on Artificial Intelligence*, Vancouver, Canada, Feb. 2022. URL https://hal.archives-ouvertes.fr/hal-03454121.

N. Halliwell, F. Gandon, and F. Lecue. A Simplified Benchmark for Non-ambiguous Explanations of Knowledge Graph Link Prediction using Relational Graph Convolutional Networks. International Semantic Web Conference, Oct. 2021a. URL https://hal.archives-ouvertes.fr/hal-03339562. Poster.

N. Halliwell, F. Gandon, and F. Lecue. User Scored Evaluation of Non-Unique Explanations for Relational Graph Convolutional Network Link Prediction on Knowledge Graphs. In *International Conference on Knowledge Capture*, Virtual Event, United States, Dec. 2021b. doi: 10.1145/3460210.3493557. URL https://hal.archives-ouvertes.fr/hal-03402766.

N. Halliwell, F. Gandon, and F. Lecue. Linked Data Ground Truth for Quantitative and Qualitative Evaluation of Explanations for Relational Graph Convolutional Network Link Prediction on Knowledge Graphs. In *International Conference on Web Intelligence and Intelligent Agent Technology*, Melbourne, Australia, Dec. 2021c. doi: 10.1145/3486622.3493921. URL https://hal.archives-ouvertes.fr/hal-03430113.

N. Halliwell, F. Gandon, and F. Lecue. A Simplified Benchmark for Ambiguous Explanations of Knowledge Graph Link Prediction using Relational Graph Convolutional Networks. 36th AAAI Conference on Artificial Intelligence, Feb. 2022a. URL https://hal.archives-ouvertes.fr/hal-03434544. Poster.

N. Halliwell, F. Gandon, and F. Lecue. Impact of Injecting Ground Truth Explanations on Relational Graph Convolutional Networks and their Explanation Methods for Link Prediction on Knowledge Graphs. In *WI-IAT 2022 - The*

21st IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Niagara Falls / Hybrid, Canada, Nov. 2022b. URL https://hal.archives-ouvertes.fr/hal-03771424.

N. Halliwell, F. Gandon, F. Lecue, and S. Villata. The Need for Empirical Evaluation of Explanation Quality. In *AAAI 2022 - Workshop on Explainable Agency in Artificial Intelligence*, Vancouver, Canada, Feb. 2022c. URL https://hal.archives-ouvertes.fr/hal-03591012.

F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, 2016. doi: 10.1145/2827872. URL https://doi.org/10.1145/2827872.

W. E. Henley and D. j. Hand. Construction of a k-nearest-neighbour credit-scoring system†. *IMA Journal of Management Mathematics*, 8(4):305–321, 10 1997. ISSN 1471-678X. doi: 10.1093/imaman/8.4.305. URL https://doi.org/10.1093/imaman/8.4.305.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, C. Gutierrez, J. E. L. Gayo, S. Kirrane, S. Neumaier, A. Polleres, et al. Knowledge graphs. *preprint arXiv:2003.02320*, 2020.

M. Horridge. *Justification based explanation in ontologies*. PhD thesis, University of Manchester, UK, 2011.

J. V. Jeyakumar, J. Noor, Y. Cheng, L. Garcia, and M. B. Srivastava. How can I explain this to you? an empirical study of deep neural network explanation methods. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems 33*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/2c29d89cc56cdb191c60db2f0bae796b-Abstract.html.

S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu. A survey on knowledge graphs: Representation, acquisition and applications. *CoRR*, abs/2002.00388, 2020.

B. Kang, J. Lijffijt, and T. D. Bie. Explaine: An approach for explaining network embedding-based link predictions. *CoRR*, abs/1904.12694, 2019.

J. Kekäläinen and K. Järvelin. Using graded relevance assessments in IR evaluation. *J. Assoc. Inf. Sci. Technol.*, 2002.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations, ICLR*, 2017.

H. Lakkaraju and O. Bastani. "how do I fool you?": Manipulating user trust via misleading black box explanations. In A. N. Markham, J. Powles, T. Walsh, and A. L. Washington, editors, *AIES '20: AAAI/ACM Conference on AI, Ethics, and Society, New York, NY, USA, February 7-8, 2020*, pages 79–85. ACM, 2020. doi: 10.1145/3375627.3375833.

T. Laugel, M. Lesot, C. Marsala, X. Renard, and M. Detyniecki. The dangers of post-hoc interpretability: Unjustified counterfactual explanations. In S. Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2801–2807. ijcai.org, 2019. doi: 10.24963/ijcai.2019/388. URL `https://doi.org/10.24963/ijcai.2019/388`.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the Institute of Radio Engineers*, 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2015.

S. Lessmann, B. Baesens, H.-V. Seow, and L. C. Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124–136, 2015. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2015.05.030. URL `https://www.sciencedirect.com/science/article/pii/S0377221715004208`.

O. Li, H. Liu, C. Chen, and C. Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018. URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17082`.

C. Lin and E. H. Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In M. A. Hearst and M. Ostendorf, editors, *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL*. The Association for Computational Linguistics, 2003.

S. M. Lundberg and S. Lee. A unified approach to interpreting model predictions. In I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html`.

Y. Ming, P. Xu, H. Qu, and L. Ren. Interpretable and steerable sequence learning via prototypes. In A. Teredesai, V. Kumar, Y. Li, R. Rosales, E. Terzi, and

G. Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*. ACM, 2019. doi: 10.1145/3292500.3330908. URL https://doi.org/10.1145/3292500.3330908.

G. Montavon, S. Lapuschkin, A. Binder, W. Samek, and K. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognit.*, 2017. doi: 10.1016/j.patcog.2016.11.008. URL https://doi.org/10.1016/j.patcog.2016.11.008.

S. H. Muggleton. Inverse entailment and progol. *New Gener. Comput.*, 13(3&4): 245–286, 1995. doi: 10.1007/BF03037227. URL https://doi.org/10.1007/BF03037227.

M. Nickel, L. Rosasco, and T. A. Poggio. Holographic embeddings of knowledge graphs. In D. Schuurmans and M. P. Wellman, editors, *Proc. 30th AAAI Conference on Artificial Intelligence*. AAAI Press, 2016.

S. Ott, C. Meilicke, and M. Samwald. SAFRAN: an interpretable, rule-based link prediction method outperforming embedding models. In D. Chen, J. Berant, A. McCallum, and S. Singh, editors, *3rd Conference on Automated Knowledge Base Construction, AKBC 2021, Virtual, October 4-8, 2021*, 2021. doi: 10.24432/C5MK57. URL https://doi.org/10.24432/C5MK57.

R. K. Pace and R. Barry. Sparse spatial autoregressions. *Statistics & Probability Letters*, 33:291, 1997.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Meeting of the Association for Computational Linguistics*. ACL, 2002.

C. Patrício, J. C. Neves, and L. F. Teixeira. Explainable deep learning methods in medical imaging diagnosis: A survey, 2022. URL https://arxiv.org/abs/2205.04766.

F. Poursabzi-Sangdeh, D. G. Goldstein, J. M. Hofman, J. W. Vaughan, and H. M. Wallach. Manipulating and measuring model interpretability. In Y. Kitamura, A. Quigley, K. Isbister, T. Igarashi, P. Bjørn, and S. M. Drucker, editors, *CHI '21: CHI Conference on Human Factors in Computing Systems*. ACM, 2021. doi: 10.1145/3411764.3445315. URL https://doi.org/10.1145/3411764.3445315.

D. Precup and Y. W. Teh, editors. *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia*, Proceedings of Machine Learning Research, 2017. PMLR. URL http://proceedings.mlr.press/v70/.

K. Randhawa, C. K. Loo, M. Seera, C. P. Lim, and A. K. Nandi. Credit card fraud detection using adaboost and majority voting. *IEEE Access*, 6:14277–14284, 2018. doi: 10.1109/ACCESS.2018.2806420.

M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016. doi: 10.1145/2939672.2939778. URL https://doi.org/10.1145/2939672.2939778.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1527–1535. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982.

L. Rieger, C. Singh, W. J. Murdoch, and B. Yu. Interpretations are useful: Penalizing explanations to align neural networks with prior knowledge. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8116–8126. PMLR, 2020.

C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x. URL https://doi.org/10.1038/s42256-019-0048-x.

C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *CoRR*, abs/2103.11251, 2021. URL https://arxiv.org/abs/2103.11251.

M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In A. Gangemi, R. Navigli, M. Vidal, P. Hitzler, R. Troncy, L. Hollink, A. Tordai, and M. Alam, editors, *European Semantic Web Conference, ESWC*, 2018.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093. URL https://doi.org/10.1109/78.650093.

R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016. URL http://arxiv.org/abs/1610.02391.

A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje. Not just a black box: Learning important features through propagating activation differences. *CoRR*, abs/1605.01713, 2016. URL http://arxiv.org/abs/1605.01713.

A. Shrikumar, P. Greenside, and A. Kundaje. Learning important features through propagating activation differences. In Precup and Teh [2017]. URL http://proceedings.mlr.press/v70/shrikumar17a.html.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Workshop Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6034.

D. Smilkov, N. Thorat, B. Kim, F. B. Viégas, and M. Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017. URL http://arxiv.org/abs/1706.03825.

J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, Workshop Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6806.

M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In Precup and Teh [2017]. URL http://proceedings.mlr.press/v70/sundararajan17a.html.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In Y. Li, B. Liu, and S. Sarawagi, editors, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 990–998. ACM, 2008. doi: 10.1145/1401890.1402008. URL https://doi.org/10.1145/1401890.1402008.

N. Tintarev and J. Masthoff. Evaluating the effectiveness of explanations for recommender systems. *User Modeling and User-Adapted Interaction*, 2012.

T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In M. Balcan and K. Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML*, 2016.

G. Varoquaux and V. Cheplygina. Machine learning for medical imaging: methodological failures and recommendations for the future. *NPJ digital medicine*, 5(1): 1–8, 2022.

S. Wachter, B. D. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *CoRR*, abs/1711.00399, 2017. URL http://arxiv.org/abs/1711.00399.

Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Trans. Knowl. Data Eng.*, 2017. doi: 10.1109/TKDE.2017.2754499.

Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In C. E. Brodley and P. Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI Press, 2014.

P. Yanardag and S. V. N. Vishwanathan. Deep graph kernels. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015.

B. Yang, W. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations, ICLR*, 2015.

Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In *Advances in Neural Information Processing Systems*, 2019.

W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977. ISSN 00917710.

M. D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In D. J. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference*, Lecture Notes in Computer Science. Springer, 2014. ISBN 978-3-319-10589-5. doi: 10.1007/978-3-319-10590-1_53. URL https://doi.org/10.1007/978-3-319-10590-1_53.