EXPLAINING LINK PREDICTION IN GRAPH NEURAL NETWORKS

EXPLAINING LINK PREDICTION IN GRAPH NEURAL NETWORKS

Thesis

to obtain the degree of Master in Computer Science with Specialization in Data Science and Technology at Delft University of Technology, to be publicly defended on 9th September, 2024.

by

Yuchuan FU
Born in Fugu, China.

Multimedia Computing Group, Faculty of Electrical Engineering, Mathematics and Computer Science (Faculteit Elektrotechniek, Wiskunde en Informatica), Delft University of Technology, Delft, The Netherlands.

Keywords: Graph Neural Networks, Explainability, Link Prediction

Contents

SUMMARY

In this thesis, we investigated the explainability of Graph Neural Networks (GNNs) within the context of link prediction tasks. While GNNs have demonstrated state-of-the-art performance in various applications due to their ability to capture complex structural relationships within graph data, their "black-box" nature presents significant challenges in terms of model interpretability. The focus of this thesis was to explore and enhance the explainability of GNNs specifically for link prediction, a task that is crucial for applications such as social network analysis, recommendation systems, etc.

We proposed a novel efficient extension of the Zorro explainer, originally developed for node classification, to handle the complexities of link prediction. It includes using top similar nodes for initialization and other techniques to accelerate the greedy search. Zorro finally returns binary node and feature masks as explanations for model predictions, which are more informative than continuous masks.

The thesis also included multiple evaluation metrics for different explainers. Apart from adopting the RDT-Fidelity and Sparsity metrics, we proposed a new metric of Edge-level RDT-Fidelity to help evaluate explainers with edge masks. These metrics provided a more refined understanding of the contribution of different graph components to the prediction task, addressing some of the shortcomings of traditional metrics. Our experiments on Cora and PubMed show the superiority of Zorro extension both in evaluation metrics and efficiency.

In conclusion, this thesis advances the state of research in the explainability of GNNs for link prediction by providing new methodologies and evaluation frameworks. Future work could further explore the integration of these explainability methods into more complex GNN architectures and their application to other types of graph-related tasks.

Acknowledgements

I would like to express my special thanks to all the people who have supported me throughout my master's studies and during my thesis work. First and foremost, I extend my sincere thanks to Elvin and Megha for giving the Graph Machine Learning course at TU Delft, which provided me with the opportunity to delve into the world of Graph Neural Networks. Additionally, I would give special thanks to Megha, who supported my master's thesis extensively during the past year. We talked a lot about the possibilities of my research topics and directions, and the ideas she gave were always enlightening and motivating, which helped me solve the problems I had during my work. Lastly, I would like to thank Yizhao Du from the Faculty of Architecture at TU Delft. The past two years spent together in the Netherlands and across Europe have been filled with joy and laughter, and will always be the memories of my life.

1
Introduction

Graph Neural Networks (GNNs) have achieved state-of-the-art performance in many applications due to their powerful ability to capture the complex structural relations between nodes and edges. Yet, as GNNs perform tasks in an inherent black-box nature, it complicates the interpretability of such models, limiting their use in cases where transparency is needed. While much research has been studied regarding node and graph classifications, little has focused on link prediction, which is also an essential part of the research community.

Notably, popular GNN explainers can be adapted to link prediction tasks [5], but few have been specifically designed to explain interactions in link predictions [35, 5], which limits the generalization of explainers. Explaining link prediction in GNNs is challenging compared to node and graph classification due to the three challenges about *defining explanations*, *complexity of finding explanations*, and *evaluating such explanations*.

1.1. Explanation definition

Traditionally, explanations of GNN models are defined as the node, edge, and feature masks over the original graph, either binary or continuous, representing the importance of their contribution to the existence of a link. In link prediction, some recent papers have explored new definitions.

For example, PaGE-Link [33] defines explanations as paths connecting a node pair in heterogeneous graphs, where edges and nodes have different meanings. A valid explanation path could be: "$u_1 \rightarrow i_3 \rightarrow u_2 \rightarrow i_1$", which means "user 1 and user 2 both bought item 3, and user2 bought item 1." While the paths provide human-interpretable explanations, they set strict requirements for the datasets used. PaGE-Link experiments on an augmented citation network and a synthetic recommendation dataset, both created with available ground truths.

One major limitation of this definition is that it requires existing paths between these nodes in the dataset in order to find a plausible explanation. If no paths connect these nodes within the dataset, no explanation can be found. In contrast, the traditional definition focuses more on structural and feature similarities to find explanations, which does not necessarily require there to be paths connecting the nodes to be explained. Additionally, evaluating such path-based explanations can be challenging and may also include the qualitative analysis of domain experts, which can be both time-consuming and expensive. Due to these considerations, we chose not to use this definition in our paper.

ILP-GNN [35] proposed a self-explainable GNN model for link prediction, where they defined the explanations as K important neighbors of the link. The model first finds K most important neighbors of the source and target nodes using node similarity and high-order structure similarity and then uses these K neighbors to learn pair-specific representations to predict links. Therefore, these neighbors naturally serve as the explanation for the existence of a link. However, this definition lacks interpretability since it only shows which neighbors are important and ignores the features of nodes. In addition, it also has the same problem of being difficult to evaluate.

Considering existing popular evaluation frameworks, ultimately we adopt the definitions from node and graph classification tasks, defining explanations as binary or continuous masks over nodes, edges, and features. This approach offers a comprehensive summary of explanations and allows us to effectively use current evaluation frameworks.

1.2. COMPLEXITY OF FINDING EXPLANATION

Explaining link prediction is significantly more complex than explaining node classification. In node classification, most explainers extract explanations from the K-hop computation subgraph centered around the node. However, link prediction involves both nodes in the prediction, which includes more neighbors and edges, adding considerable complexity. This is particularly challenging for greedy search algorithms like Zorro [8].

Directly applying Zorro to link prediction may not yield results efficiently due to the increased time complexity. Therefore, optimization is crucial. In the following sections, we experiment with various methods to address this challenge, focusing on optimizing complexity.

Unlike node classification, explaining link prediction requires considering the interactions between different nodes. In link prediction tasks, explanations must account for the relationship and interactions between pairs of nodes within the graph, based on both graph structure and node attributes [35].

To capture these interactions, we adopted the approach from [35] to extract top similar nodes by calculating the similarity between the neighbors of the source node and the target node. This similarity measure helps in understanding the influence of neighboring nodes on the prediction. Details about this approach will be introduced in later sections.

1.3. EVALUATING EXPLANATIONS

Intuitively, GNN explanations should faithfully explain the behaviors of GNN models [29]. Several benchmarking papers have proposed different frameworks to evaluate GNN explanations for node classification tasks [18, 2, 12, 19, 1]. Similar to the challenge faced in node classification tasks, evaluating explanations of link prediction can also be tricky due to the lack of ground truths. Even for synthetic datasets where the ground truths are available, the evaluation against such ground truths may be suboptimal due to several challenges, as proposed by [7]. For instance, GNN models may not use ground truth edges to make predictions, or multiple explanations could exist, making the evaluations unreliable.

Therefore, we focus on real-world datasets without ground truths. For these datasets, typical metrics include Fidelity, Sparsity, Stability, etc. Fidelity and sparsity metrics are among the most popular ones used in evaluation, yet they still have problems.

Fidelity metrics measure if the explanations are faithful to the model. Fidelity+ is defined as the difference in accuracy or prediction probability between the original predictions and the new predictions after removing the important elements. Fidelity- is defined as the difference when only keeping the important elements [29]. Intuitively, if the explanations extracted are indeed the important nodes/edges/features used by the model to make predictions, then Fidelity+ should be high, and Fidelity- should be low.

However, traditional fidelity metrics suffer from distribution shift problems. When subgraphs are removed while computing fidelity metrics, the resultant subgraphs might be Out Of Distribution (OOD), violating the assumption that training and test data come from the same distribution [34]. The RDT-Fidelity proposed by Zorro [8] does not suffer from this problem. RDT-Fidelity is grounded in rate-distortion theory, which measures the validity and stability of explanations in node classification tasks. Instead of directly removing features, it randomly perturbs the non-important features with random values sampled from the distribution of the associated features.

Apart from fidelity, sparsity is defined as the fraction of nodes/edges/features selected as important by explanation methods [29]. However, for continuous masks, a threshold for importance score must be decided to compute the sparsity value. Zorro improves upon this by introducing a more generalized version of sparsity, defined as the entropy over the mask distribution.

In conclusion, the challenges of evaluation metrics for link prediction explanations are similar to those in node classification tasks. In our evaluation framework, we aim to adopt the RDT-Fidelity and Sparsity metrics to evaluate the explanations in link prediction tasks better.

1.4. RESEARCH QUESTIONS AND CONTRIBUTIONS

Motivated by current research gaps, we propose and answer the following research questions:

RQ1: How do we define the explanation of a GNN for a link prediction task?

RQ2: How do we develop efficient explainers for link prediction?

RQ3: Which metrics can be used to evaluate the quality of explanations to achieve a well-rounded evaluation?

In the subsequent chapters, we address these questions by exploring various approaches to enhance the interpretability and efficiency of GNN explainers for link prediction tasks. The main contributions of this thesis are:

1. The development and optimization of Zorro's extension to link prediction tasks, where we tailored Zorro to handle the complexity and interactions inherent in link prediction. This approach involved incorporating a larger computation graph, initializing node masks with top similar nodes, effectively capturing interactions between source and target nodes, and optimizing the efficiency of the explanation.

2. The evaluations of the quality of explanations using a combination of RDT-Fidelity and sparsity metrics. Traditional fidelity metrics like Fidelity+ and Fidelity- were also considered, although they were less discriminative and faced distribution shift problems. We also proposed a new metric of Edge-level RDT-Fidelity metric to help evaluate explainers with edge masks.

These contributions advance the interpretability of GNNs in link prediction tasks, addressing a critical gap in current research.

1.5. ORGANIZATION

The thesis is organized as follows: First, Chapter 2 presents the background of Link Prediction and GNN. Chapter 3 then investigates the literature related to link prediction, explainability of GNN, and explainability of GNN in link prediction. Next, Chapter 4 introduces the GNN models we used for training, the baseline explainers, the evaluation metrics, and the model we developed. Chapter 5 presents the experiments we conducted and the result analysis. Finally, Chapter 6 concludes the whole thesis. Experiment codes are available at PriXAI/explainLinkPrediction.

2

Background

In this chapter, we present the background knowledge used in this topic, including the introduction about graph neural networks and link prediction. Section 2.1 introduces the general architecture of GNN models with two stages of *Aggregation* and *Transformation*. Section 2.2 explains how link prediction tasks are conducted in general.

2.1. Graph Neural Networks

Graph Neural Networks take graph structures and node or edge features of graphs as input and generate node representations by integrating the features of a node's neighbors across multiple hops through the processes of transformation and aggregation. Different from the traditional machine learning approach, GNNs can capture more complex relationships between nodes and edges, and such robust representations can be used for various tasks such as node classification, graph classification, and link prediction.

GNNs leverage the inherent connections and relationships between nodes and edges in a graph to learn meaningful representations, which can be used for various tasks such as node classification, graph classification, and link prediction. The architecture of a typical GNN involves two main stages: Aggregation and Transformation. These stages are applied iteratively to capture local and global structural information within the graph.

Let $G = (V, E)$ be a L-layer graph with each node having d-dimensional features, V represents the node set and E represents the edge set. $\mathbf{x}_v^{(\ell)}$ denotes the feature representation of node $v \in V$ in layer $\ell \in L$, $\mathcal{N}_v$ denotes the neighborhood nodes of node v. $\mathbf{x}_v^{(\ell)}$ is therefore computed by first aggregating the information from all neighbors' last layers' representations, and then a transformation operation is applied, for example, non-linear operations like ReLU.

2.1.1. Aggregation

In the aggregation stage, each node in the graph gathers information from its neighboring nodes. This process is similar to the message-passing mechanism, where each node receives messages from its neighbors and aggregates them to update its state. Specific aggregation functions vary across different GNN models, and they may include simple operations such as summing, averaging, or taking the maximum of the neighboring nodes' features [11, 9]. More complex aggregation functions may involve attention mechanisms [23] or other learnable functions that weigh the importance of different neighbors.

Formally, for a node v with neighbors $\mathcal{N}_v$, the aggregation process at layer ℓ can be represented as:

$$\mathbf{z}_v^{(\ell)} = \text{AGGREGATE}^{(\ell)}\left(\left\{\mathbf{x}_u^{(\ell-1)} \mid u \in \mathcal{N}_v \cup \{v\}\right\}\right) \tag{2.1}$$

2.1.2. TRANSFORMATION

After aggregation, the transformation stage applies a neural network layer, typically a fully connected layer followed by a non-linear activation function, to update the node's feature representation based on the aggregated information. The transformation stage allows the model to learn complex patterns and relationships in the graph data.

$$\mathbf{x}_v^{(\ell)} = \text{TRANSFORMATION}^{(\ell)}\left(\mathbf{z}_v^{(\ell)}\right) \tag{2.2}$$

These two stages are repeated for a predefined number of layers, allowing the GNN to learn progressively more abstract representations of the nodes in the graph.

2.2. LINK PREDICTION

Link Prediction (LP) is a fundamental task that involves predicting the existence of an edge between two nodes in a graph. Specifically, it predicts the probability of an unseen edge. LP has a wide range of applications, including social network analysis, recommendation systems, knowledge graph completion, etc [32].

Link prediction has been widely studied in the past few years, with methods generally categorized into similarity-based, probabilistic models, learning-based models, etc [13]. However, with the advent of deep learning, more advanced graph neural network models have been developed. It offers powerful tools to capture more complex relationships in the graph and produce better representations of nodes.

Link prediction models are typically composed of two key components: an Encoder and a Decoder.

2.2.1. ENCODER

The encoder is responsible for learning latent representations of the nodes (or edges) in the graph. This is typically achieved using a GNN, which aggregates and transforms the features of each node from its neighbors. The encoder outputs a vector for each node, which captures its structural and feature-based context within the graph.

2.2.2. DECODER

The decoder uses the node embeddings generated by the encoder to predict the existence of an edge between a pair of nodes. There are various approaches about how to design decoders depending on the task and the graph structure. Commonly used methods include cosine similarity and the inner product of node embeddings, which assess how closely related the two nodes are in the embedding space.

The models are trained to minimize the difference between the predicted and actual edges in the graph with various loss functions, where the binary cross-entropy loss is a popular choice. Overall, the combination of GNN-based encoders and decoders has significantly enhanced the field of link prediction, enabling more accurate solutions to be applied across different domains.

3
Literature Review

This chapter concludes the related work concerning link prediction, GNN, and explainability. The chapter is divided into three sections. Section 3.1 is a short survey about link prediction tasks; Section 3.2 introduces the general aspect of explainability in terms of GNN models, the methods used for explanations, the evaluation metrics, etc.; The final Section 3.3 summarizes the explainability papers specifically developed for link prediction tasks in GNNs.

3.1. Link Prediction

Link Prediction (LP) is a critical function in graph and network analysis, aimed at determining the likelihood of an edge existing between two nodes. LP can be applied in various practical applications across different fields: in finance for detecting fraudulent transactions, in social networking for predicting potential connections and friend suggestions, and in biology for discovering new medicinal insights through protein-protein interaction networks.

Graph Neural Networks (GNNs) have revolutionized LP as they leverage the structural and feature-specific data within graphs, leading to improved prediction capabilities. Key developments in this area include Graph Convolutional Networks (GCNs) by Kipf and Welling [11], which effectively utilize graph structures for predictive analysis. Following this, more sophisticated methods like Graph Attention Networks (GATs) by Veličković et al. [23] and GraphSAGE by Hamilton et al. [9] have emerged, enhancing the ability to understand node relationships and generalize to unseen nodes. Most advanced models initially designed for node classification can also be adapted to link prediction.

However, a significant challenge with these deep learning models is their "black-box" nature, leading to concerns about interpretability. While numerous explainability methods have been developed for GNNs, they mostly target node and graph classification tasks. Consequently, the explainability of LP in GNNs emerges as a promising yet underexplored research domain, presenting opportunities for further development.

3.2. EXPLAINABILITY OF GNNS

In this section, we introduce the explainers developed for explaining GNN models, as well as the evaluation metrics. Most existing papers focus on node classification and graph classification tasks.

3.2.1. EXPLAINERS

We focus on instance-level and post-hoc explainability methods. The explainability papers of GNNs can generally be divided into the following four categories: (1) perturbation-based methods, (2) gradient-based methods, (3) Decomposition, (4) Surrogate, and (5) counterfactual methods.

PERTURBATION-BASED METHODS

Perturbation-based explainers measure the discrepancies between model performances when the input data is perturbed. The motivation is to study the variations of model output concerning different input perturbations. If the explanations have learned important information, the predictions should ideally be stable against perturbations. Generally, perturbation-based methods find subgraphs as explanations.

GNNExplainer [28] learns a subgraph with a subset of features of the computation graph as the explanation, which is essential to the model prediction. It learns continuous soft masks over edges and features so that the mutual information between the prediction probabilities of the explanation subgraph G_s, X_s and the computation graph G_c, X_c is maximized. Its optimization framework is as follows:

$$\max_{G_S} MI(Y,(G_S,X_S)) = H(Y) - H(Y|G = G_S, X = X_S)$$

PGExplainer [15] is very similar to GNNExplainer, but it learns discrete masks for edges as explanations. Each edge is independently modeled as a Bernoulli distribution, with the parameters determined by an MLP. The MLP is optimized to maximize the mutual information between the explanation subgraph and the predictions of the original GNN model.

Zorro [8] finds important nodes and features as explanations by maximizing RDT-Fidelity. Unlike GNNExplainer, Zorro applies hard masks (binary masks) over nodes and features for better interpretability. The optimization process uses a greedy approach to iteratively add nodes or features that result in the highest increase in RDT-Fidelity, while RDT-Fidelity is computed as the expected validity of the perturbed input.

SubgraphX [30] employs Monte Carlo Tree Search to identify potential explanatory subgraphs in GNNs. It generates these subgraphs by progressively pruning the parent graph. To assess the importance of each subgraph, SubgraphX utilizes Shapley values. This approach combines advanced search techniques to quantify the significance of various subgraphs in GNN predictions.

GRADIENT-BASED METHODS

Gradient-based methods focus on utilizing gradients or the values of hidden feature maps to assess the significance of inputs, a popular technique first used for image and text data. They have also been widely explored in GNN explainability due to their straightforwardness. Gradient-based methods produce explanations as importance scores (masks) over input nodes/features/edges.

Given a node i and a trained GNN model $f(G, \mathbf{x}_i)$, Grad [21] extracts an explanation by assigning importance scores to the input features. The gradient score $S(\mathbf{x}_i)$ corresponding to the input feature identifies how important the feature is in making predictions. The explanation score is computed by taking gradients of f to input features $\mathbf{x}_i$:

$$S(\mathbf{x}_i) = \frac{\partial f}{\partial \mathbf{x}_i}$$

GradInput [21] extends Grad by applying element-wise products with the input node and edge features. The score is computed as follows:

$$S(\mathbf{x}_i) = \frac{\partial f}{\partial \mathbf{x}_i} \odot \mathbf{x}_i$$

However, these methods are often criticized sometimes insensitive to variations in input. For instance, if the model function is linear, the gradients remain constant regardless of the input. Integrated Gradients (IG) [22] solves this by accumulating gradients along a path between the input feature vector and a baseline vector. It interpolates a bunch of points between the input and the baseline, takes gradients with respect to each one of them, and averages the gradient results. IG argued that gradient methods should follow two axioms of sensitivity and implementation invariance, which many other methods do not meet. The explanation for node i is defined as the integrated gradient along the input $\mathbf{x}_i$ and baseline $\mathbf{x}'_i$ as follows. The baseline $\mathbf{x}'_i$ is often taken as all zeros or ones.

$$S(\mathbf{x}_i) = (\mathbf{x}_i - \mathbf{x}'_i) \times \int_{\alpha=0}^{1} \frac{\partial f(\mathbf{x}' + \alpha \times (\mathbf{x} - \mathbf{x}'))}{\partial \mathbf{x}_i} \, d\alpha$$

DECOMPOSITION

Decomposition methods explain model predictions by attributing the final output score to the individual input features. Unlike gradient-based methods, this is done through a backward layer-by-layer distribution of importance scores, starting from the output layer and moving towards the input layer. The scores of different parts of the input can be viewed as importance scores over edges and node features to the prediction, thus contributing to the explanations.

Deconvolution [31] employs deconvolution to perform a backward propagation of the original model. This technique emphasizes the most influential features or edges by tracing back through the network, resulting in a distribution of positive and negative importance scores across the node features and edges.

Layer-wise relevance propagation (LRP) [3] leverages a sequential backward propagation approach across all neural network layers. LRP first computes predictions using forward neural network propagation, storing the outputs of each layer. Then it proceeds in reverse, assigning relevance scores from the output back to the input layer, ultimately providing these scores at the input layer as the model's prediction explanation.

SURROGATE METHODS

Surrogate models work by constructing a local dataset containing a subset of neighboring nodes and their predictions, then fitting an interpretable model to extract the explanations as the explanations of the original model [29].

GraphLime [10] explains GNN predictions by focusing on the N-hop neighboring nodes, where N is the number of layers in the trained GNN. The local dataset is created from the neighboring nodes and their predictions. To generate explanations, a nonlinear surrogate model called HSIC (Hilbert-Schmidt Independence Criterion) Lasso [27] is used to fit this local dataset. The most important features identified by HSIC Lasso are then selected to explain the predictions, and these features are considered as the explanation for the original GNN prediction.

PGM-Explainer [24] builds a probabilistic graphical model to produce explanations for GNNs, which is an interpretable Bayesian network used to approximate the predictions. It has three steps: Data Generation, Variable Selection, and Structure Learning.

PGM-Explainer builds the local datasets by randomly perturbing the features of several nodes within the computational graph and records whether a node's features were perturbed and the resulting influence on the GNN's predictions. Different from other methods of directly including neighbors, it contains node variables. It then applies the Grow-Shrink (GS) algorithm [16] to eliminate less important variables, narrowing down the dataset to the most influential ones. Finally, PGM-Explainer uses an interpretable Bayesian network to fit the local dataset and explain the predictions.

PGM-Explainer provides explanations in terms of node masks but ignores the edges. Additionally, it can provide explanations based on conditional probabilities, which is not common in other explainers.

COUNTERFACTUAL METHODS

Counterfactual explainers aim to identify necessary changes to the original input graph that most significantly impact the model's prediction. Unlike other methods, counterfactual explainers can add/delete edges or change feature values instead of just selecting a subgraph. The explanations are therefore counterfactual graphs/subgraphs.

CF-GNNExplainer [14] iteratively removes edges from the adjacency matrix to perturb the graph, and returns the perturbation with the smallest number of deletions as the counterfactual explanation. The final loss function is a combination of the prediction accuracy of the counterfactual graph and the distance between the counterfactual graph and the original graph.

In addition to the counterfactual property, RC-Explainer [4] also optimizes for the robustness of counterfactuals to noise. It generates the explanations by modeling the common decision logic of GNNs on similar input graphs.

3.2.2. EVALUATION METRICS

Though many explainers have been proposed in the past few years, the explainability of GNN models lacks a unified and comprehensive framework for comparisons and evaluations. Several benchmark papers have been published [2, 12, 18, 19, 1], which separately provide frameworks with diverse evaluation metrics for explanations. Table 3.1 and 3.2 summarize the tasks and evaluation metrics used in these benchmark papers. NC, GC, and GR in the table stand for Node Classification, Graph Classification, and Graph Regression.

Paper	Year	Task	Metrics with Ground Truth
Evaluating Explainability [1]	2023	NC, GC	Accuracy, Faithfulness, Stability, Fairness: counterfactual fairness, group fairness
GNNX-BENCH [12]	2023	NC, GC	Accuracy
GraphFramEx [2]	2022	NC	Accuracy
BAGEL [18]	2022	NC, GC	Plausibility
Evaluating Attribution [19]	2020	NC, GC, GR	Accuracy: AUROC(classification), Kendall's tau(regression) Consistency, Faithfulness, Stability

Table 3.1: Summary of benchmark papers (Part 1).

Paper	Year	Metrics without Ground Truth
Evaluating Explainability [1]	2023	-
GNNX-BENCH [12]	2023	Stability, Sufficiency (Fidelity) Necessity: 1 - sufficiency/fidelity Reproducibility, Feasibility, Size
GraphFramEx [2]	2022	Fidelity (Sufficiency: Fid-, Necessity: Fid+) Characterization score, Efficiency
BAGEL [18]	2022	Faithfulness: RDT-Fidelity, Comprehensiveness, Sufficiency, Sparsity, Correctness
Evaluating Attribution [19]	2020	-

Table 3.2: Summary of benchmark papers (Part 2).

EVALUATING ATTRIBUTION

Apart from the traditional accuracy, Sanchez-Lengeling et al. [19] proposed *Consistency*, *Faithfulness*, and *Stability* to evaluate. ***Consistency*** quantifies the variability in attribution accuracy under different hyperparameters. ***Faithfulness*** measures the change in explanation accuracy when the model performance degenerates. ***Stability*** assesses the change in explanation accuracy when perturbing the test samples

Though they were all computed against explanation accuracy, the idea has been widely explored in other papers for evaluating without ground truth. Evaluating against ground truth suffers from several pitfalls as stated by [7]. For example, GNN models may not use the ground truth to make predictions, but we are comparing our explanations with them. Also, multiple explanations could lead to the same prediction and all are valid explanations. Therefore, only comparing with one ground truth is not rigorous enough.

EVALUATING EXPLAINABILITY

To solve this problem, Agarwal et al. [1] proposed a new method of generating synthetic datasets that avoids such misconduct. The ShapeGGen generator generates new graphs by combining subgraphs containing any given motif and additional nodes. Meanwhile, the explanations generated are deeply grounded in how the motifs were used, thus making sure to be prone to the pitfalls from [7]. But unfortunately, the paper only provides node and graph classification datasets, and can not be used for link prediction.

Agarwal proposed an extension of ***Faithfulness*** by comparing the prediction probabilities of only keeping top-k features by an explanation with the original one. Let f be the trained GNN model, G be the original graph, and G_s be the masked subgraph by only keeping the original values of top-k features identified by the explanation, KL be the KL divergence, then the graph explanation unfaithfulness is computed as:

$$\text{Unfaithfulness} = 1 - \exp^{-KL(f(G)||f(G_s))}$$

An explanation is considered stable if the explanations for a graph and its perturbed graph (inject infinitely small perturbations to node features and edges) are similar. The paper then phrased ***Stability*** as the cosine distance of predicted explanation masks from the original and its perturbed graph, meanwhile bounded by a graph difference check. Let the perturbed graph be G', M_G and M'_G be the explanation masks for G and G'. The graph explanation instability is defined as:

$$\text{instability} = \max D(M_G, M'_G), \forall G' \in \beta(G)$$

where D is the cosine distance, β is a δ-radius ball around G for which the model behavior is the same.

Additionally, the paper proposed two fairness metrics: *Counterfactual Fairness Mismatch* and *Group Fairness Mismatch* to measure fairness.

Counterfactual Fairness Mismatch measures the difference between the explanations by the original graph and its counterfactual graph. ***Group Fairness Mismatch*** is defined as the absolute difference between the statistical parity of the predictions of a set of K graphs using the original and important features identified by the explanation.

Apart from the above papers, others [18, 2, 12]focus on evaluating datasets without ground truth explanations. We are more interested in cases like this as they are more inclined to real-world scenarios where we do not have access to the ground truth explanations.

BAGEL

BAGEL [18] approached ***Faithfulness*** as the ability of explanations to approximate the model's behavior. Faithfulness is measured by *RDT-Fidelity*, *Sufficiency*, and *Comprehensiveness.*

BAGEL applied ***RDT-Fidelity***, which was first proposed in [8]. RDT-Fidelity differs from the original Fidelity in that it randomizes the features not selected in the explanations instead of using all zeros to replace them. Higher RDT-Fidelity means the explanations have high predictive power under input perturbations. The details about RDT-Fidelity will be stressed in later chapters, as we include it as our evaluation metric.

Additionally, BAGEL introduced ***Sparsity*** as the entropy over the explanation mask distribution. A lower sparsity implies a near-uniform feature attribution and consequently lower interpretability. The paper also proposed ***Correctness*** to measure the explanations' ability to detect externally injected decoys (fake edges) that alter model decisions.

GRAPHFRAMEX

GraphFramEx [2] considers three different "user needs" in GNN explainability and proposed associated metrics. It extends the idea of ***Fidelity*** [29] with Phenomenon and Model focus. The fidelity is computed against node labels with phenomenon focus and prediction probability with model focus.

Phenomenon

$$fid_+ = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{G_{C\setminus S}} = y_i)\right|$$

$$fid_- = \frac{1}{N}\sum_{i=1}^{N}\left|\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{G_S} = y_i)\right|$$

Model

$$fid_+ = 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(\hat{y}_i^{G_{C\setminus S}} = \hat{y}_i)$$

$$fid_- = 1 - \frac{1}{N}\sum_{i=1}^{N}\mathbb{1}(\hat{y}_i^{G_S} = \hat{y}_i)$$

fid_+ measures the ***Necessity*** of the explanation: if the model prediction changes after the removal of the explanation, then the explanation is necessary. fid_- measures the ***Sufficiency*** of the explanation: whether the explanation alone can result in the same prediction as before. The paper further combines these two metrics as a ***Characterization Score***, the harmonic mean of fid_+ and fid_-:

$$\text{charact score} = \frac{(w_+ + w_-) \times fid_+ \times (1 - fid_-)}{w_+ \times (1 - fid_-) + w_- \times fid_+}$$

The characterization score allows a balance between sufficiency and necessity when evaluating the explanations. ***Efficiency*** is also mentioned to provide a trade-off between characterization and computation time.

GNNX-BENCH

GNNX-BENCH [12] provided an in-depth benchmarking for perturbation-based explainers. Apart from seven factual explainers, it also included four counterfactual ones.

GNNX-BENCH computes ***Stability*** as the Jaccard similarity between the set of edges in the two explanations by the original and perturbed graph.

Apart from the traditional ***Sufficiency (Fidelity)*** and ***Necessity***, the paper also proposed several new metrics. ***Size*** is the number of edges in factual explanations and the number of edges perturbed to change the label in counterfactual explanations.

Similar to ***Necessity*** where the model measures whether the model prediction accuracy decreases when the explanation subgraph is removed, ***Reproducibility*** measures if the model is retrained on a subset of the residual graph, can it recover the predictions on the remaining unused residual graphs?

Feasibility is specially designed for counterfactual explanations, which evaluates how feasible it is to achieve a specific counterfactual outcome. For example, in the protein dataset, the explanation provided must be a valid molecule. This metric is calculated on a protein dataset by the statistical significance of the differences in the number of connected components observed in the test set versus those in their respective counterfactual explanations.

Though many metrics have been proposed, most of them are still evaluated against node/graph classification. There is still little research on benchmarking against link prediction tasks.

3.3. EXPLAINABILITY OF GNNS IN LINK PREDICTION

The specific papers proposed for explaining link predictions are rather few. We summarize the up-to-date papers on this topic.

3.3.1. ILP-GNN

Zhu et al. [35] proposed an Interpretable Link Prediction based on GNN (ILP-GNN) for link prediction. Unlike post-hoc methods, the model is inherently self-explainable and can generate accurate predictions and explanations at the same time. It defines the explanation for link prediction as the most important K neighbors of both nodes from an edge as explanation neighborhoods for the link between them. These neighbors have learned pair-specific representations for links from this node to other nodes, and thus can be used for explanations.

METHOD

For each node pair (v_i, v_j), ILP-GNN searches for K interpretable neighbors of v_i that are similar to v_j using node and high-order structure similarity. Node similarity is measured by the hidden representations of two nodes encoded by MLP. For high-order structure similarity, graph diffusion is used to compute the closeness of nodes between the neighbors of v_i to v_j based on graph structure.

These neighbors can represent common interests or features between v_i and v_j and are aggregated to learn pair-specific representation by the Explanation Enhancement module. Since the predictions of links are based on the identified K neighbors, these neighbors can act as explanations.

METRICS

As for evaluations, for synthetic datasets where ground truths are available, it uses precision@2 and precision@1 for the ranked neighbor lists. When there are no ground truths, it compares the fidelity score of $\Delta AUC\%$ (the LP performance drop) for each pair of nodes when important neighbors of the pair of nodes are removed. It also includes a case study to justify their method.

The evaluation part is somewhat limited compared with the complex models they designed. Also, as their definition of explanations differs from traditional masks over feature/node/edge, it is difficult to conduct a comparison with other explainers.

3.3.2. PaGE-Link

PaGE-Link [33] focuses on explaining link predictions in heterogeneous graphs. It defines paths connecting two nodes as the explanation for the existence of a link.

Method

PaGE-Link finds the optimal sets of paths between nodes as explanations. It first utilizes k-core pruning to eliminate low-degree nodes and reduces graph size for scalability. Then it applies path-enforcing mask learning to select important edges from the k-core-pruned computation graph. This step is optimized over two conditions: the masked subgraph should provide enough information for predicting the missing link, and the defined score functions over edges and paths should help find short paths with low-degree nodes. Important edges should be essential for model predictions and also form meaningful paths.

Dataset

The paper created two synthetic datasets of citation and recommendation, where different edges represent different types of relations with different nodes. Due to these special heterogeneous graphs, PaGE-Link offers explanations in human-readable format, as the paths between two nodes have actual meanings. For example, in the citation graph, the explanation of a link between user u_1 and item i_1 could be: user1 bought item2, and item2 shares attribute1 as item1, which is a path of $u_1 \to i_2 \to a_1 \to i_1$.

Metrics

PaGE-Link was evaluated both quantitatively and qualitatively. Since the original datasets are created based on path rules, it compares the edge masks learned while selecting paths with the ground truth to compute the ROC-AUC metric. For qualitative analysis, it includes case studies on the citation graph and a survey for human evaluation.

3.3.3. Evaluating LP Explanations

Borile et al. [5] is the first to evaluate link prediction explanations for GNN models systematically. It defines explanations as masks over node features and edges. It employs both synthetic datasets like Stochastic Block Models (SBM) and Watts-Strogatz, which offer ground truth for explanations, and real-world datasets such as Cora, PubMed, and DDI.

Models & Explainers

For GNN models, the paper utilizes VGAE and GIN as encoders, inner product, and cosine similarity as decoders to train the link prediction task. It explores several explainers, including GNNExplainer, which uses a mask on edges and node features; Integrated Gradients, assigning scores to edges and node features; Deconvolution, highlighting activated features or edges; and Layer-wise Relevance Propagation, which uses a linear approximation of neuron scores. All these explainers are extensions, not specifically designed for link prediction.

METRICS

As for evaluation metrics, for datasets with ground truth, metrics of sensitivity(true positive rate) and specificity (true negative rate) are used. For real-world datasets without ground truth, the paper proposes novel metrics of the area under feature and edge insertion/deletion curves. These compare explanations against a random baseline by manipulating features and edges based on their importance.

However, the paper also has limitations when dealing with ground truth data. For example, in the SBM dataset, when computing the metrics for each edge to be explained, it uses all edges within the same block as true positives (ground truth), which has negative effects on the results. Overall, this paper provides an important framework for evaluating explanations in link prediction with novel evaluation metrics.

Apart from the above new methods and four explainers used in [5], many other classical explainers can also be adapted to link prediction tasks [25, 26, 30].

3.4. CONCLUSION

In this chapter, we reviewed relevant papers in the field of link prediction, explainability of GNN models, and explainability in link prediction tasks. We started by discussing different approaches to link prediction, followed by an in-depth look at how different types of explainers work for GNNs. We also included benchmark papers to illustrate the varieties of evaluation metrics. Towards the end, we focused on specific methods developed for explaining link predictions in GNNs. This chapter sets a solid foundation and context for our research, highlighting the key concepts and current research trends in the field.

4
Methods

In this chapter, we introduce the methods and approaches employed in our study to develop and evaluate explainers for link prediction in Graph Neural Networks. Section 4.1 introduces the GCN model we train for our link prediction tasks. Section 4.2 presents the baseline explainers used in our experiments. We discuss how these explainers are extended and adapted to the task of link prediction. Section 4.3 summarizes the evaluation metrics we use to assess the quality and effectiveness of the explanations. Section 4.4 shows the core content of this thesis. Here, we elaborate on the ideas and methods explored to develop an efficient extension of Zorro for link prediction. This section covers subgraph reduction techniques, node mask importance investigation, and the extensions of Zorro Baseline and Zorro with different optimization strategies. It shows how we finally arrived at an efficient version of Zorro.

4.1. GNN Models

Graph Convolutional Networks (GCNs) have emerged as a powerful tool for various graph-based tasks, including node classification, graph classification, and link prediction. In the context of link prediction, GCNs are particularly effective due to their ability to capture both local node features and the global structure of the graph.

4.1.1. Architecture of GCN

A GCN model typically consists of several graph convolutional layers, each designed to aggregate information from a node's neighbors to learn a more comprehensive representation. This core operation can be described as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right) \tag{4.1}$$

where:

- $H^{(l)}$ is the matrix of node features at layer l,

- $\tilde{A} = A + I$ is the adjacency matrix of the graph with added self-loops, and I is the identity matrix,
- $\tilde{D}$ is the degree matrix corresponding to $\tilde{A}$, and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$
- $W^{(l)}$ is a layer-specific trainable weight matrix,
- σ is an activation function, such as ReLU.

4.1.2. LINK PREDICTION WITH GCNS

For link prediction, the task is to predict the probability of an edge existing between two nodes in the graph. We use GCN as the encoder, and the inner product as decoder as the final model.

4.2. BASELINE EXPLAINERS

For our baseline explainers, we selected methods from various categories of instance-level explanations as classified by [29]. Specifically, we chose GNNExplainer from the perturbation-based method, Integrated Gradients (IG) from the gradient-based method, Deconvolution from decomposition, and PGM-Explainer from the surrogate method as baseline explainers to extract explanations. The specific types of masks used by these explainers are shown in Table 4.1.

	Node Mask	**Feature Mask**	**Edge Mask**	**Mask Type**
GNNExplainer	0	1	1	Continuous
Integrated Gradients	0	1	1	Continuous
Deconvolution	0	1	1	Continuous
PGM-Explainer	1	0	0	Binary

Table 4.1: Comparison between baseline explainers.

4.2.1. GNNEXPLAINER

GNNExplainer [28] identifies a subgraph and a subset of features from the computation graph that are crucial for the model's prediction. It achieves this by learning continuous soft masks over edges and features, aiming to maximize the mutual information between the prediction probabilities of the explanation subgraph G_s, X_s and those of the computation graph G_c, X_c.

The extension of GNNExplainer to link prediction tasks is rather straightforward. It provides explanations for link prediction and graph classification with no change to its optimization algorithm. When predicting a link, GNNExplainer learns two masks for both endpoints of the link, which is equivalent to learning the masks over the larger computation graph induced by the source and target nodes.

With GNNExplainer, we only generate the feature masks for all nodes (not specific for each node). Since the above methods do not produce node masks, we convert their edge masks into node masks by equally distributing the importance to both sides of the edges. With GNNExplainer, we generate feature masks for all nodes collectively rather than for each node individually.

4.2.2. Integrated Gradients

Integrated Gradients (IG) is an axiomatic attribution method designed to explain model predictions in terms of input features. Compared with traditional gradient-based methods, it satisfies sensitivity and implementation invariance. IG works by interpolating a bunch of points between the input and a baseline, taking gradients to each one of them, and averaging the gradient results. In the context of link prediction, IG assigns positive and negative explanation scores to each link and node feature based on the model's sensitivity to these inputs [5].

4.2.3. Deconvolution

Deconvolution [31] is a saliency method that performs a backward propagation of the original model using a deconvolution operation. Initially introduced to explain convolutional neural networks in image classification, it highlights the most activated features or edges. Similar to IG, it also produces positive and negative importance scores for node features and edges, emphasizing those that are most influential in the model's predictions.

4.2.4. PGM-Explainer.

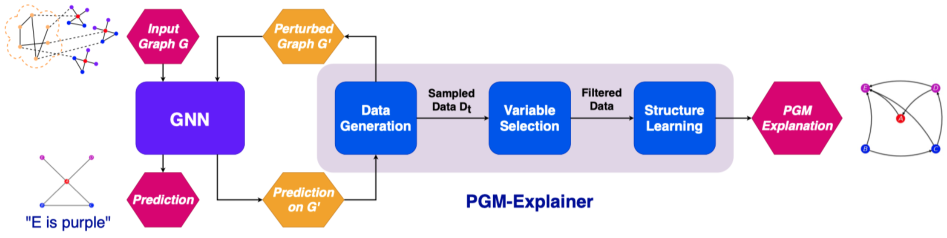


Figure 4.1: The architecture of PGM-Explainer from [24]

PGM-Explainer [24] is a surrogate model-based explainer designed to provide interpretable explanations for GNN predictions. It constructs a probabilistic graphical model (PGM) to approximate the behavior of the original GNN, and the conditional probabilities serve as explanations. It also generates node masks through the Markov blanket of the target node. Figure 4.1 shows the architecture of PGM-Explainer.

The local dataset of PGM-Explainer is constructed by randomly perturbing the node features. Within the k-hop computation graph, it randomly perturbs the node features of random nodes. For each node in the computation graph, it records a random variable indicating whether the features of the node are perturbed (perturbations are conducted by setting node features to the mean value across all nodes) and its influence on the predictions of GNN. By repeating the above procedures multiple times (100 in our case), a local dataset is obtained.

With the local dataset built, it first reduces the size of the local dataset by selecting the top dependent variables (the Markov blanket) via the Grow-Shrink (GS) algorithm. However, due to the complexity of this algorithm, PGM-Explainer uses pairwise dependence tests to get the Markov blanket.

The extension of PGM-Explainer to link prediction is a bit more complicated than others, which requires a modification of the approach to generate the Markov blanket of the link. We extend PGM-Explainer for link prediction as follows:

1. Every time PGM-Explainer randomly perturbs the node features of several random nodes within the larger computational graph that is derived from both the source and target nodes of an edge. Then for any node in the computational graph, PGM-Explainer records a random variable indicating whether its features are perturbed and its influence on the GNN predictions. Repeat the above steps to obtain a local dataset.

2. Change the process of including neighbors and evaluate the independent test. Instead of iterating over nodes in the computation graph, we now iterate over all the edges.

Originally, PGM-Explainer fits an interpretable Bayesian network to get the conditional probabilities. However, this information is never used in the evaluation phase in the original paper. Evaluating conditional probabilities can be complicated and often lacks ground truths. Therefore, we do not use such information. Instead, we focus solely on the first stage of PGM-Explainer which computes node masks. The performance of these baseline explainers will be included in later sections.

4.2.5. CONCLUSION

Since GNNExplainer, IG, and Deconvolution do not produce node masks, we convert their edge masks into node masks by equally distributing the importance to both nodes connected by the edges for later evaluations. For explainers with negative importance values, we only keep the positive scores.

Theoretically, GNNExplainer would perform well due to its optimization over subgraphs, IG and Deconvolution may perform poorly since they tend to ignore the graph structure, potentially generating unconnected explanations.

4.3. EVALUATION METRICS

We define our explanations of link prediction as a subset of features, and neighboring nodes and edges that are crucial for the correct prediction of the existence of a link. With this definition, we can easily adapt the evaluation metrics from node classification tasks. In this section, we introduce the evaluation metrics we use for our experiments, including feature-level and edge-level RDT-Fidelity, Sparsity, and traditional fidelity metrics.

4.3.1. RDT-FIDELITY

FEATURE-LEVEL

RDT-Fidelity was first proposed in [8]. The key idea of this metric is that given the important nodes and features as an explanation, the explanation alone should produce a stable and valid prediction that is the same as the original prediction. On top of this, we do not set the non-selected elements to be 0 as 0 could have special meanings that can act as noises. Instead, we randomize the features not selected by the explanation by assigning them random values from the same feature. RDF-Fidelity measures the effectiveness of a retrieved explanation.

Given the extracted explanation subgraph as $S = \{V_s, F_s\}$ where V_s represents selected nodes and F_s represents selected features. Let $M(S)$ be the binary mask over the computation graph that each element $M_{i,j} = 1$ if and only if the ith node and jth feature are included in V_s and F_s. The perturbed input Y_s will then be:

$$Y_s = X \odot M(S) + Z \odot (\mathbb{1} - M(S))$$

RDT-Fidelity represents the expected validity of the perturbed input Y_s. Unlike the node classification task in Zorro, we induce the k-hop subgraph of both nodes as the computation graph in link prediction. For each edge, we first get the prediction from the whole computation graph as the ground truth, then the explanation subgraph is perturbed 100 times to compare new predictions with the ground truth. Final RDT-Fidelity is calculated as the correct samples divided by 100.

Edge-level

Similar to the original RDT-Fidelity where non-important features are perturbed, we developed a new approach to perturb non-important edges to evaluate the performance of explainers with edge masks. The original RDT-Fidelity does not account for edge masks, therefore this metric could provide a more balanced comparison across different explainers.

Inspired by the original RDT-Fidelity and the Graph Rate-Distortion (GRDE) Explanations by [6], we proposed a new metric of edge-perturbed RDT-Fidelity. In the GRDE approach, the edges and features are perturbed upon the adjacency matrix and feature matrix respectively. In contrast, we directly perturb the edge masks and treat them as the input edge weights in the model. This approach avoids the possibility of creating new edges that do not exist in the original graph and allows us to focus on the edges within the subgraph, ensuring that our evaluation remains focused and relevant to the original structure.

Edge-level RDT-Fidelity defaultly selects all nodes in the computation graph and perturbs the edge and feature importance masks. We define S_E as the edge masks extracted by the explainers, Z as the random noise we inject into the edge masks, then the perturbed edge importance I is calculated as:

$$I = S_E + (\mathbb{1} - S_E) \odot Z$$

The perturbed edge weights, together with the perturbed feature matrix from RDT-Fidelity, were treated as the new input of the model. The edge masks were perturbed 100 times and edge-level RDT-Fidelity was thereafter computed. We have experimented with different types of noise in Table 4.2 and report the performance in the next chapter.

Noise Type	Description
None	Acts as a baseline, where the noise is all 0s.
bernoulli_whole	Sample from a Bernoulli distribution learned over the adjacency matrix of the complete graph.
bernoulli_computation	Sample from a Bernoulli distribution learned over the adjacency matrix of the computation graph.
bernoulli_1/2	Sample from a Bernoulli distribution with a 1/2 probability.
KDE fitted on the mask	Use KDE (Kernel Density Estimation) to approximate the distribution of edge masks and then randomly sample from this distribution.
KDE fitted on 1 - mask	Use KDE (Kernel Density Estimation) to approximate the 1 - edge mask distribution and then randomly sample from this distribution.

Table 4.2: Descriptions of Different Noise Types Used in Experiments

4.3.2. SPARSITY

Sparsity is defined as the entropy over the mask distribution, which measures the size of an explanation. Good explanations should be sparse since information is stored in fewer bits, thus providing explanations with compact features and nodes. We compute the sparsity for the node mask, edge mask, and feature mask. In cases where the node mask is not available but the edge mask is available, we evenly distribute the edge masks to the two connecting nodes.

4.3.3. FIDELITY+, FIDELITY-

Apart from the new metrics added, we also include the traditional Fidelity metrics from [17]: Fidelity+ and Fidelity-. Both metrics have variants of label and probability that represent the change in label and prediction probability.

Fidelity+ is defined as the prediction change by removing important nodes/edges/features. Higher fidelity+ means the prediction is significantly influenced by the removal of nodes, thus better explanations.

$$\text{Fidelity}^{+acc} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{1-m_i} = y_i)\right)$$

$$\text{Fidelity}^{+prob} = \frac{1}{N}\sum_{i=1}^{N}\left(f(G_i)_{y_i} - f(G_i^{1-m_i})_{y_i}\right)$$

Fidelity- is defined as the prediction change by keeping important input features and removing unimportant features. Lower Fidelity- means the explanation alone can already lead to the same prediction and, therefore better explanation.

$$\text{Fidelity}^{-acc} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{m_i} = y_i)\right)$$

$$\text{Fidelity}^{-prob} = \frac{1}{N}\sum_{i=1}^{N}\left(f(G_i)_{y_i} - f(G_i^{m_i})_{y_i}\right)$$

4.4. Model Development

In this section, we focus on developing an extension of the Zorro explainer for link prediction tasks. Traditional explainers like GNNExplainer, IG, and Deconvolution rely on edge masks, which can be less informative than node masks [8]. Moreover, the continuous nature of their masks also makes it harder to interpret the results [8]. Therefore, we decided to develop extensions of Zorro for link prediction. We introduce the journey of ideas explored to develop an efficient version of Zorro for link prediction, including subgraph reduction, node mask importance investigation, and the extensions of Zorro Baseline and Zorro. We will report the experiment results in the next chapter.

4.4.1. Subgraph Reduction

Our initial idea to improve the efficiency of Zorro involves reducing the computation subgraph when extracting explanations. By refining the subgraph to a smaller scope, we can decrease the greedy search time during the optimization process for Zorro.

Inspired by [32], the key takeaway from this paper is to use node labels to obtain the structural information of the computation graph. This information can be used to filter out nodes that are too far from the center nodes, serving as a pre-filtering stage before the actual experiment. Additionally, we adopted the idea from [35], which combines node similarity and high-order similarity to identify various K important neighbors of a node.

We define the reduced subgraph as a reduced subgraph that only includes nodes similar to the source/target node. Let (s, t) be the edge to be explained, G_c be the K-hop computation graph of the edge, N_s, N_t be the neighbors of s and t respectively. G_r be the reduced subgraph we aim to induce by only including the top similar nodes. The pseudocode of the algorithm is shown in Table 4.3 and the specific steps are described as follows:

1. Compute the embedding similarity of the neighbors of the source node with the target node and vice versa.

2. Merge two lists of sorted similarity scores to create a list of total top similar nodes.

3. Include only the top similar nodes and reduce the computation graph to retain only the edges connected to the selected important nodes.

4.4.2. Upper limit of Subgraph Reduction

As an extension of the subgraph reduction idea, we want to explore the upper limit of performance if we reduce the graph by different proportions. Therefore, we design another experiment to test the cap of metrics. This experiment also verifies our approach's effectiveness in identifying top similar nodes, highlighting the importance of node masks. Details about this experiment are introduced in the next chapter.

```
Algorithm 1: Subgraph Reduction for Link Prediction
Input:  source node $s$, target node $t$, embeddings $E$, number of top neighbors $k$
Output: reduced computation graph $G_r$
1: $N_s \leftarrow$ set of neighbors of source node $s$
2: $N_t \leftarrow$ set of neighbors of target node $t$
3: Compute similarity $Sim_s(v,t) = \frac{E(v)\cdot E(t)}{|E(v)||E(t)|}$ for each $v \in N_s$
4: Compute similarity $Sim_t(v,s) = \frac{E(v)\cdot E(s)}{|E(v)||E(s)|}$ for each $v \in N_t$
5: $Sim_{\text{union}} \leftarrow Sim_s \cup Sim_t$
6: $N_{\text{sorted}} \leftarrow$ list of $v$ in $Sim_{\text{union}}$ sorted by similarity in descending order
7: $N_{\text{important}} \leftarrow$ top $k$ elements of $N_{\text{sorted}}$
8: Reduce the computation graph to include only edges containing nodes in $N_{\text{important}}$
9: return reduced computation graph $G_r$
```

Table 4.3: Pseudocode for Subgraph Reduction for Link Prediction.

4.4.3. Extension of Zorro Baseline

Inherited from the idea of the reduced subgraph, we use the top similarity method to identify the most important nodes contributing to link predictions. The Zorro Baseline model is developed by including all features in the explanations for simplicity, and gradually adding nodes to the explanations sequentially based on the merged sorted list of similarity scores calculated from the embeddings above. Optimization stops once a predefined threshold is achieved. During this process, we ensure that only nodes that increase fidelity are added. Special cases include:

1. If the empty explanation can achieve our target RDT-Fidelity, then any explanation is considered valid.
2. If the reachable fidelity is smaller than the target RDT-Fidelity, return all nodes and features as explanations.
3. If all nodes and features are selected and the target RDT-Fidelity is still not achieved, return the selected nodes and features.

4.4.4. Extension of Zorro

The original Zorro method greedily adds nodes or features that maximize RDT-Fidelity to the selected elements until the target RDT-Fidelity is achieved. However, extending this approach to link prediction poses several challenges, mainly due to time constraints and the greedy nature of Zorro's optimization: (1) Excessive calculations: The method computes the sorted list for each node and feature when explaining each edge, leading to inefficiency; (2) Minimal improvements: The improvements are minor if we only add one node or feature during each calculation, and it can take significant time to achieve optimal performance.

To address these challenges, we have made several improvements to the original Zorro method. Next, we introduce these enhancements and describe the experiments conducted to verify their effectiveness in the next chapter. These improvements aim to optimize the efficiency of Zorro while maintaining the same level of performance in explaining link prediction tasks.

1. Base Improvement: Separation of computing the sorted list. In the new extension, we first separate the process of computing the sorted list of contributions for features and nodes as a pre-filter step, especially for edges with high numbers of neighbors. This would make the experiments across different settings more efficient.

We precompute the sorted list for each element and its improvement in fidelity for later reuse. Compared with the original approach that computes the tailored list for each edge on the fly, pre-computing generates the same list of features across all the edges, which sacrifices a certain drop in accuracy but significantly improves the time to achieve optimization.

Dataset	Feature Dimension	Time/Hrs
Cora	1433	3+
PubMed	500	Around 3

Table 4.4: Time of pre-computing the sorted list.

Table 4.4 shows the pre-computing time for the Cora and PubMed datasets. Cora has fewer neighbors but high feature dimension, while PubMed lower feature dimension but high degree of neighbors on the opposite. Therefore, the computation time is roughly the same.

2. Further Improvements: Different strategies. By studying the process of the original Zorro method, we noticed the most time-consuming part is the process of searching for the optimal node or feature greedily, once at a time. Therefore, we came up with the idea to initialize the node masks with the node masks from the Zorro Baseline before searching. We introduce the experiments with different strategies (options in Table 4.5) to verify the feasibility of different techniques for improvements. The overall performance will be reported in the next chapter.

We ensure that during the optimization of Zorro, any element that does not negatively impact the RDT-Fidelity is added. Elements that result in a negative improvement are removed. This approach helps maintain efficiency by avoiding the addition of negatively contributing elements that can slow down the process.

Table 4.6 shows the greedy optimization of Zorro for link prediction. Through a series of experiments and comparisons, we have demonstrated the effectiveness of our techniques in improving the efficiency of Zorro, and we summarize the improvements as follows. Details and the analysis of experiments are reported in the next chapter.

1. Separate the process of computing the sorted list of Zorro.
2. Use Zorro Baseline to initialize node masks before greedy search.
3. Reduce the node greediness to expedite the search.

Option	Description
Option 1	Add nodes and features at the same time. Add 2 nodes at once / add 10 features at once. Search Greediness: 30. This is the baseline approach.
Option 2	Add node masks precomputed from Zorro baseline as the important nodes, then only add features with Zorro.
Option 3	Keep node masks precomputed as the initial selected nodes, then add nodes and features at the same time.
Option 4	Based on Option 3, reduce the greediness to fasten speed. Node selection greediness: 10. Feature selection greediness: 30.
Option 5	Based on Option 4, add a constraint of a max of 100 optimization steps.

Table 4.5: Description of the 5 options for improving Zorro's optimization.

4. Add a constraint of a maximum optimization step to be 100.

Algorithm 1: ZORRO for Link Prediction(n, τ)

Input: node n, threshold τ

Output: explanation, i.e. node mask V_s & feature mask F_s

```
1:  $V_n \leftarrow$ set of vertices in $G(n)$
2:  $F_p \leftarrow$ set of node features
3:  $V_r = V_p, F_r = F_p, V_s = \emptyset, F_s = \emptyset, k = 0$
4:  $R_{V_p} \leftarrow$ list of $v \in V_p$ sorted by $\mathscr{F}(\{v\}, F_p)$
5:  $R_{F_p} \leftarrow$ list of $f \in F_p$ sorted by $\mathscr{F}(V_p, \{f\})$
6:  Initialize $V_s$ with Zorro Baseline
7:  Add maximal element to $V_s$ or $F_s$
8:  while $\mathscr{F}(V_s, F_s) \leq \tau$ and $k \leq 100$ do
9:      $\tilde{V}_s = V_s \cup \mathrm{top}_2(\arg\max_{v \in \mathrm{top}_{10}(V_r)} \mathscr{F}(\{v\} \cup V_s, F_s))$
10:     $\tilde{F}_s = F_s \cup \mathrm{top}_{10}(\arg\max_{f \in \mathrm{top}_{30}(F_r)} \mathscr{F}(V_s, \{f\} \cup F_s))$
11:     if $\mathscr{F}(\tilde{V}_s, F_s) \leq \mathscr{F}(V_s, \tilde{F}_s)$ then
12:         if $\mathscr{F}(V_s, \tilde{F}_s) \geq \mathscr{F}(V_s, F_s)$ then
13:             $F_r = F_r \setminus \{f\}, F_s = \tilde{F}_s$
14:         else
15:             $F_r = F_r \setminus \{f\}$
16:     else
17:         if $\mathscr{F}(\tilde{V}_s, F_s) \geq \mathscr{F}(V_s, F_s)$ then
18:             $V_r = V_r \setminus \{v\}, V_s = \tilde{V}_s$
19:         else
20:             $V_r = V_r \setminus \{v\}$
21:     $k = k + 1$
22: return $\{V_s, F_s\}$
```

Table 4.6: Algorithm of Zorro for Link Prediction.

5 Experiments

This chapter contains one of the core contributions of this thesis. We introduce our experiments in detail and answer the three main research questions we proposed initially. Section 5.1 describes the datasets selected for our experiments and the rationale for choosing them. Section 5.2 introduces the experiment setup, and Section 5.3 presents the experiment results, including the experiment with edge-level RDT-Fidelity, the experiment with subgraph reduction, the experiment to verify the importance of node masks and our extension methods of Zorro and Zorro Baseline. These experiments are used to answer the research questions we proposed. In Section 5.4, we include a detailed analysis of the performances of different explainers and a qualitative case study analysis.

5.1. Dataset Selection

In this section, we introduce the datasets selected for our experiments. We focus on real-world datasets without predefined ground truth explanations but also include a discussion about why we do not choose datasets with ground truths.

Dataset without Ground Truth

For datasets without ground truths, we focus on Cora and PubMed [20]. Both are real-world citation network datasets that were originally designed for node classification but are also be widely used in link prediction to predict the existence of links between nodes.

In the graph datasets, a node represents a publication and an edge represents a citation relationship between the two nodes. Cora contains 2,708 nodes classified into 7 classes with 5,429 edges connecting the nodes, and PubMed has 19,717 nodes classified into 3 classes and 44,338 edges. Node features for both datasets are bag-of-words representations.

Dataset with Ground Truth

Synthetic datasets often have varying definitions of the ground truth for an explanation. For instance, the Stochastic Block Model (SBM) is a graph generation model based on the idea that each node belongs to a specific community, and edges between nodes are more common within than between communities. In the evaluation against link prediction explanations for SBM datasets in [5], an edge is considered present if two nodes belong to the same block. Therefore, the ground truths are determined by all edges within the same block as the source or target node. However, computing accuracy against such information may be inappropriate due to the excessive number of ground truths.

Datasets with pre-defined ground truths may not be reliable for providing meaningful ground truth labels. These ground truths can be misleading and suffer from the pitfalls mentioned in [7]. Consequently, we decided to exclude these datasets from our experiments and focus solely on datasets without ground truths.

5.2. EXPERIMENT SETUP

Dataset. For the Cora and PubMed datasets, we use an 80/20 split for train and test data. All results are reported in terms of their performance on the edges in the test set.

Model. For the encoder, we chose a 2-layer GCN with hidden sizes of 128 and 64, and the decoder is the inner product between two hidden representations. We trained the model with a total of 200 epochs on the Cora dataset for link prediction. The final AUC is 0.7977, and the accuracy is 0.7033 on the test set.

Explainer. We chose 4 explainers as our baseline explainers: GNNExplainer, Integrated Gradients, Deconvolution, and PGMExplainer. Table 4.1 shows the type of masks each explainer generates. For our explainers, we run Zorro Baseline and Zorro.

Evaluation Metrics. In our experiments, we use evaluation metrics of RDF-Fidelity, Sparsity, Fidelity+, and Fidelity-. Note that edge masks are not used in the calculation of RDT-Fidelity. Therefore, for explainers without node masks but edge masks, we convert the edge masks into node masks for calculation. We also experimented with Edge-level RDT-Fidelity but did not include it in our final framework.

Implementation Details. We randomly sample 100 edges to be explained. Notably, our focus is solely on explaining true positive edges, which means the final number of edges analyzed may be fewer than 100. A random seed is set in the selection of edges to be explained to ensure the reproducibility and consistency of the comparison between different explainers.

GNNExplainer does not produce node masks, therefore all nodes are selected in its explanation. The explainer only produces feature masks of shape [1, num_features]. For explainers that do not produce node masks but have edge masks, we evenly distribute the edge masks to nodes of both ends and use the converted ones as the final node masks. For explainers with negative values (IG, Deconvolution), we only keep the positive ones and set the negative masks to 0 in calculating RDT-Fidelity, sparsity.

5.3. EXPERIMENT RESULTS

In this section, we summarize the experiments conducted so far. These experiments consist of a variety of methodologies and approaches aimed at addressing the key research questions we proposed. Specifically, we explored: 1) edge-level RDT-Fidelity in evaluations, 2) subgraph reduction to reduce the computation complexity of link prediction, 3) node mask importance investigation, and 4) the extensions of Zorro Baseline and Zorro. Each experiment is designed to focus on different aspects of our research, providing a comprehensive view of the challenges and potential solutions in explaining link predictions using graph neural networks.

5.3.1. EDGE-LEVEL RDT-FIDELITY

In this experiment, we investigated the application of edge-level RDT-Fidelity in evaluating GNN explanations and now discuss the reliability of this metric. We introduced various types of noise, including Bernoulli and KDE, into the edge weights to perturb the importance of edges. Similar to the original RDT-Fidelity, we perturbed the edge importance 100 times and averaged the results to obtain the final score. The performances are reported in Table 5.1 and 5.2.

	bernoulli, whole	**bernoulli, computation**	**bernoulli, 1/2**	**kde, fitted on mask**	**kde, fitted on 1 - mask**
GNNExplainer	0.9072	0.9034	0.8960	0.9043	0.9019
IG	0.5418	0.5425	0.5761	0.5449	0.6075
Deconvolution	0.6241	0.6144	0.6048	0.6165	0.6141

Table 5.1: Edge-level RDT-Fidelity under different types of noise on Cora dataset (Part 1).

	None	**Random**	**All**
GNNExplainer	0.9077	0.8658	0.8998
IG	0.5419	0.5798	0.6082
Deconvolution	0.6237	0.5825	0.6087

Table 5.2: Edge-level RDT-Fidelity under different types of noise on Cora dataset (Part 2).

Apart from these noises, we also included three baselines of ’None’, ’Random’, and ’All’. The ‘None’ baseline involved using only the original edge masks as edge weights, while ‘Random’ assigned random importance weights to edges, and ‘All’ used all edges within the computation graph.

Edge Masks vs. Random Baseline When comparing the performance of the explainers using edge masks without perturbations to the random baseline, we observed a performance drop for GNNExplainer and Deconvolution. This suggests that the edges identified by these explainers do contribute positively to the model’s performance, though the contribution might be relatively small as the differences are small, especially in the context of the Cora dataset. For IG, the edge masks perform worse even than the random baseline, which means IG may not be able to extract edge masks correctly.

Using Important Edges vs. All Edges Interestingly, using only the most important edges resulted in better performance than using all edges for GNNExplainer and Deconvolution. This improvement could be due to noise reduction, as focusing on the most important edges likely reduces the potential introduction of irrelevant information.

Noise Perturbation Results Across all explainers, different types of noise (e.g., Bernoulli, KDE) produced results similar to the baseline 'None' of not introducing noise at all and only using edge masks. This consistency indicates that the edge masks extracted by these explainers are stable regardless of how the non-important edges are perturbed, the model's performance remains largely unchanged.

One key limitation of this approach is related to the GNN models employed. Since the method involves perturbing edge masks and assessing the impact on performance, it requires models that can utilize edge weights as inputs. However, many advanced GNN models do not inherently support edge weights. Also, since in later sections we focus on the Zorro model that produces node masks instead of edge masks, we do not include this metric in the later stages of our evaluations.

5.3.2. SUBGRAPH REDUCTION

To improve the efficiency of Zorro, we experimented with the subgraph reduction technique. This technique aims to reduce the computation subgraph size by only including the most important and relevant nodes and edges. We compute the list of top similar nodes based on the similarities of the neighbors of the source node to the target node and vice versa. By focusing on a smaller subset, we can potentially enhance the speed and performance.

The operation of getting the reduced subgraph can be inserted as an independent module in other explainers, for example, PGMExplainer and Zorro, etc. We have experimented with subgraph reduction for PGMExplainer. We define the number of top similar nodes to be top 50%, 20, 15, 10, 5 for Cora, and top 50%, 40, 20, 10 for PubMed respectively to measure the performance change. The results on Cora are shown in Table 5.3 and 5.4, and PubMed in Table 5.5 and 5.6.

	RDT-Fidelity ↑	**Node Sparsity ↓**	**Explanation Time/s ↓**
Full Graph	0.7773	2.7728	118.24
Reduced Subgraph, 50%	0.7378	2.0109	114.46
Reduced Subgraph, 20	0.7510	1.8670	102.81
Reduced Subgraph, 15	0.7294	1.6714	96.97
Reduced Subgraph, 10	0.7013	1.3148	93.70
Reduced Subgraph, 5	0.6425	0.7619	88.82

Table 5.3: Experiments of reduced subgraphs on PGMExplainer on the Cora dataset (Part 1).

For the **Cora dataset**, as the number of top nodes decreases, the RDT-Fidelity would also decrease, which indicates that the faithfulness of the model reduces as we remove more nodes. The node sparsity improves as the graph is reduced, meaning fewer nodes are used in the explanation, which simplifies the model and presents more compact explanations. There are also improvements in the computation time, but since the original computation is quite fast (excluding the second phase of PGMExplainer to fit the

	Fidelity+ label ↑	Fidelity+ prob ↑	Fidelity- label ↓	Fidelity- prob ↓
Full Graph	0.0323	0.1659	0.1720	0.2780
Reduced Subgraph, 50%	0.0430	0.1332	0.0968	0.3114
Reduced Subgraph, 20	0.0538	0.1339	0.0860	0.3036
Reduced Subgraph, 15	0.0753	0.1325	0.0968	0.3006
Reduced Subgraph, 10	0.0968	0.1026	0.1183	0.3200
Reduced Subgraph, 5	0.0215	0.0586	0.0645	0.3394

Table 5.4: Experiments of reduced subgraphs on PGMExplainer on the Cora dataset (Part 2).

Bayesian network, which is time-consuming, and the Cora is a simple dataset), the differences are relatively minor.

	RDT-Fidelity ↑	Node Sparsity ↓	Explanation Time/s ↓
Full Graph	0.8188	3.3096	1879.04
Reduced Subgraph, 50%	0.7885	2.9291	2135.59
Reduced Subgraph, 40	0.7487	2.2150	1350.84
Reduced Subgraph, 20	0.7045	1.6894	1049.34
Reduced Subgraph, 10	0.6940	1.1705	821.13

Table 5.5: Experiments of reduced subgraphs on PGMExplainer on the PubMed dataset (Part 1).

	Fidelity+ label ↑	Fidelity+ prob ↑	Fidelity- label ↓	Fidelity- prob ↓
Full Graph	0.0000	0.1070	0.0957	0.3094
Reduced Subgraph, 50%	0.0106	0.0580	0.0745	0.3504
Reduced Subgraph, 40	0.0213	0.0522	0.0851	0.3613
Reduced Subgraph, 20	0.0106	0.0337	0.0957	0.3668
Reduced Subgraph, 10	0.0106	0.0231	0.0851	0.3708

Table 5.6: Experiments of reduced subgraphs on PGMExplainer on the PubMed dataset (Part 2).

As for the **PubMed dataset**, which has a high number of edges and neighbors, the reduction in computation time is more significant. The computation time decreases significantly when the subgraph is reduced. One outlier is the increase in time when using the top 50% of nodes, likely due to the additional time needed to reduce the graph. However, when the number of top similar nodes is reduced to 40 or below, the RDT-Fidelity decreases drastically.

The reduction would bring better node sparsity and computation time in both cases, but worse RDT-Fidelity and traditional fidelity metrics. This shows a trade-off between the number of top nodes to include and the final performance. Though we did not directly apply this strategy due to its drop in fidelity scores, we extended this idea of extracting the top similar nodes in the experiment to verify the importance of node masks and the optimization of greedy searching of Zorro.

5.3.3. UPPER LIMIT OF SUBGRAPH REDUCTION.

We designed another experiment to compute the upper-performance limit achievable with reduced subgraphs. This experiment also aimed to verify the significance of node masks when all features are selected. This was achieved by incorporating the previously described subgraph reduction technique.

In this experiment, we use the same method to obtain the list of top similar nodes for the edge to be explained. We then select the top 10% to 50% of these nodes and their associated edges, respectively, to form a reduced computation subgraph. We measure the maximum performance (RDT-Fidelity, Fidelity+, and Fidelity-) achievable with this reduced graph. Additionally, we randomly select the same number of nodes within the reduced computation graph as a baseline for comparison.

Metric	**Number of Nodes**	**10%**	**20%**	**30%**	**40%**	**50%**
RDT-Fidelity ↑	Top	0.7088	0.8190	0.8711	0.9117	0.9330
	Random	0.6387	0.6994	0.7605	0.8078	0.8549
Fidelity+ label ↑	Top	0.1075	0.2043	0.2796	0.4086	0.4409
	Random	0.0215	0.0323	0.0538	0.0860	0.1290
Fidelity+ prob ↑	Top	0.1219	0.2348	0.3140	0.3580	0.3826
	Random	0.0408	0.0974	0.1525	0.2012	0.2496
Fidelity- label ↓	Top	0.0753	0.1183	0.1075	0.0108	0.0000
	Random	0.0430	0.0430	0.0323	0.0860	0.0860
Fidelity- prob ↓	Top	0.3246	0.2453	0.1872	0.1274	0.0829
	Random	0.3593	0.3390	0.3009	0.2694	0.2168
Node Sparsity ↓	-	1.389	2.0372	2.4309	2.7128	2.9557

Table 5.7: Fidelity metrics on Cora dataset with different percentages of nodes (Top vs. Random).

By making these comparisons, we aim to investigate whether the top similar nodes can effectively represent important information when explaining the predictions compared to random nodes. If the top similar nodes prove to be more effective, we can conclude that extracting these nodes can serve as a replacement step for identifying important node masks. This approach can also help justify the optimal number of nodes to include when reducing the subgraph, balancing performance, and computation time. The results on the Cora and PubMed datasets are reported in Table 5.7 and 5.8.

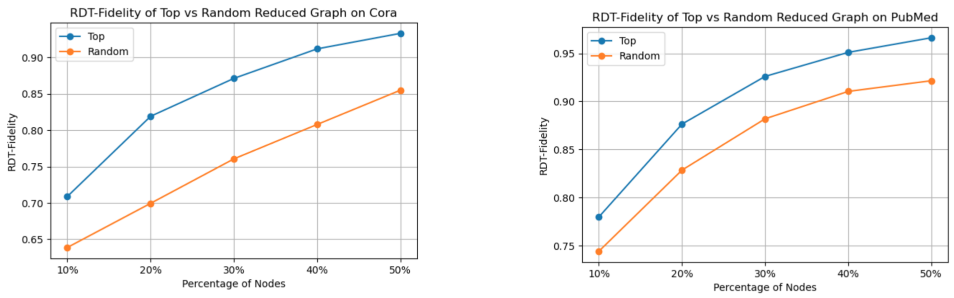
The tables illustrate the performance under a different proportion of top similar nodes. Each value in the Top and Random line represents the maximum value that can be achieved under the reduced subgraph with the associated number of nodes.

In both datasets, we observe that RDT-Fidelity improves as the number of retrieved top similar nodes increases. The top similar nodes consistently outperform the random nodes across all percentages, indicating their importance in maintaining model fidelity. It verifies that when all the features are selected, the strategy of extracting the top similar nodes can act as a surrogate method to find important node masks.

Figure 5.1 illustrates the differences in RDT-Fidelity between Top and Random nodes on both the Cora and PubMed datasets. The differences are more significant on Cora compared to PubMed, suggesting that features may contribute more to the predictions in the PubMed dataset since we select all features.

Metric	Number of Nodes	10%	20%	30%	40%	50%
RDT-Fidelity ↑	Top	0.7798	0.8764	0.9260	0.9510	0.9662
	Random	0.7441	0.8285	0.8819	0.9104	0.9215
Fidelity+ label ↑	Top	0.0638	0.1702	0.2872	0.3085	0.3404
	Random	0.0213	0.0532	0.0638	0.0532	0.0532
Fidelity+ prob ↑	Top	0.0773	0.1579	0.2283	0.2779	0.3170
	Random	0.0401	0.0839	0.1180	0.1611	0.2076
Fidelity- label ↓	Top	0.1277	0.0426	0.0532	0.0319	0.0213
	Random	0.1170	0.0851	0.0638	0.0638	0.0638
Fidelity- prob ↓	Top	0.3439	0.2809	0.2163	0.1605	0.1170
	Random	0.3660	0.3391	0.2960	0.2423	0.1928
Node Sparsity ↓	-	2.5683	3.2598	3.6333	3.9262	4.1636

Table 5.8: Fidelity metrics on PubMed dataset with different percentages of nodes (Top vs. Random).



(a) RDT-Fidelity Comparison for Cora

(b) RDT-Fidelity Comparison for PubMed

Figure 5.1: RDT-Fidelity Comparisons for Cora and PubMed datasets

Regarding the traditional fidelity metrics, we focus on Fidelity+ and Fidelity- probabilities as they indicate a reduction in prediction confidence. The values for top similar nodes are consistently higher than those for random nodes, suggesting that top nodes are more informative and reliable.

We observe that reducing the graph results in a decrease in the maximum achievable fidelity score, which is not desirable if we aim for high scores. Additionally, keeping a high proportion of nodes in the reduced graph does not significantly improve efficiency. Consequently, we decided not to adopt this graph reduction approach. Instead, we focus on identifying important node masks to improve our optimization of Zorro.

5.3.4. EXTENSION OF ZORRO BASELINE

Our experiments for extensions begin with the Zorro Baseline. We experimented with different target values of RDT-Fidelity for the Zorro Baseline. This section outlines our findings and justifies the choice of the optimal threshold. The performances are shown in Table 5.9 and 5.10 for Cora and Table 5.11 and 5.12 for PubMed.

By setting a higher target RDT-Fidelity for both datasets, we can achieve the highest

	RDT-Fidelity ↑	**Feature Sparsity ↓**	**Edge Sparsity ↓**	**Node Sparsity ↓**	**Explanation Time/s ↓**
GNNExplainer	0.8463	5.5002	3.7031	2.8024	302.94
Zorro Baseline, 0.85	0.8898	7.2675	-	1.9915	184.51
Zorro Baseline, 0.90	0.9257	7.2675	-	2.1318	204.59
Zorro Baseline, 0.95	**0.9602**	**7.2675**	**-**	**2.2697**	**261.69**
Zorro Baseline, 0.98	0.9728	7.2675	-	2.3736	271.49

Table 5.9: Explainability results of Zorro Baseline on the Cora dataset with GCN (Part 1).

	Fidelity+ Label ↑	**Fidelity+ Prob ↑**	**Fidelity- Label ↓**	**Fidelity- Prob ↓**
GNNExplainer	-0.0108	0.3539	-0.0108	-0.0570
Zorro Baseline, 0.85	0.1290	0.1994	0.0108	0.2667
Zorro Baseline, 0.90	0.1613	0.2329	0.0108	0.2345
Zorro Baseline, 0.95	**0.2258**	**0.2813**	**0.0108**	**0.1907**
Zorro Baseline, 0.98	0.2473	0.3048	0.0108	0.1644

Table 5.10: Explainability results of Zorro Baseline on the Cora dataset with GCN (Part 2).

	RDT-Fidelity ↑	**Feature Sparsity ↓**	**Edge Sparsity ↓**	**Node Sparsity ↓**	**Explanation Time/s ↓**
GNNExplainer	0.8400	5.8658	4.9373	3.7371	1552.23
Zorro Baseline, 0.90	0.9239	6.2146	-	2.1155	473.69
Zorro Baseline, 0.95	**0.9502**	**6.2146**	**-**	**2.3613**	**815.20**
Zorro Baseline, 0.98	0.9652	6.2146	-	2.5318	969.20

Table 5.11: Explainability results of Zorro Baseline on the PubMed dataset with GCN (Part 1).

	Fidelity+ Label ↑	**Fidelity+ Prob ↑**	**Fidelity- Label ↓**	**Fidelity- Prob ↓**
GNNExplainer	0.0000	0.3688	0.0213	0.0665
Zorro Baseline, 0.90	0.0851	0.1057	0.0532	0.3368
Zorro Baseline, 0.95	**0.0851**	**0.1169**	**0.0213**	**0.3279**
Zorro Baseline, 0.98	0.0745	0.1253	0.0106	0.3222

Table 5.12: Explainability results of Zorro Baseline on the PubMed dataset with GCN (Part 2).

performance in both RDT-Fidelity and traditional fidelity metrics. Node sparsity would also increase, but it is much better than GNNExplainer even if we use a threshold of 0.98. This is because we use all the features. There is a trade-off between the fidelity metrics and the sparsity and computation time, this is especially evident for PubMed: choosing a threshold of 0.95 and 0.98 over 0.90 almost doubles the computation time. We decide to use 0.95 as the threshold for the Zorro Baseline. The Zorro Baseline model will be used as a first-stage model to make Zorro more efficient in the next section.

5.3.5. EXTENSION OF ZORRO

In this section, we report the performance of Zorro in link prediction under different optimization strategies. We experimented with various options outlined in Table 4.5 and now compare them to demonstrate the improvement of each approach. The results are presented in Table 5.13 and Table 5.14 for the Cora dataset. For simplicity, we conducted our experiments only on Cora to justify the effectiveness of our methods.

	RDT-Fidelity ↑	Feature Sparsity ↓	Edge Sparsity ↓	Node Sparsity ↓	Explanation Time/s ↓
GNNExplainer	0.8463	5.5002	3.7031	2.8024	302.94
Zorro Baseline, 0.95	0.9602	7.2675	-	2.2697	261.69
(1) Zorro, 0.7, option 1	0.7437	3.4426	-	1.8638	2453.28
(2) Zorro, 0.7, option 2	0.7254	4.8766	-	1.6251	1580.92
(3) Zorro, 0.7, option 3	0.7437	3.0486	-	2.327	1248.79
(4) Zorro, 0.9, option 3	0.9048	4.2894	-	2.879	3830.47
(5) Zorro, 0.9, option 4	0.9062	4.3995	-	2.8858	2864.35
(6) Zorro, 0.9, option 5	0.8975	4.3394	-	2.8829	2602.65
(7) Zorro, 0.95, option 4	0.9541	4.8488	-	3.0366	3506.47
(8) Zorro, 0.95, option 5	**0.9373**	**4.7575**	**-**	**3.0343**	**2936.95**

Table 5.13: Explainability results of Zorro on the Cora dataset with GCN (Part 1).

	Fidelity+ Label ↑	Fidelity+ Prob ↑	Fidelity- Label ↓	Fidelity- Prob ↓
GNNExplainer	-0.0108	0.3539	-0.0108	-0.0570
Zorro Baseline, 0.95	0.2258	0.2813	0.0108	0.1907
(1) Zorro, 0.7, option 1	0.129	0.3524	0.0968	0.3406
(2) Zorro, 0.7, option 2	0.129	0.3068	0.0645	0.3467
(3) Zorro, 0.7, option 3	0.0968	0.3553	0.0538	0.3434
(4) Zorro, 0.9, option 3	0.086	0.3544	0.0000	0.2659
(5) Zorro, 0.9, option 4	0.0968	0.3562	0.0215	0.2596
(6) Zorro, 0.9, option 5	0.0968	0.3575	0.0323	0.2667
(7) Zorro, 0.95, option 4	0.0753	0.3541	0.0108	0.22
(8) Zorro, 0.95, option 5	**0.0645**	**0.3568**	**0.0215**	**0.2292**

Table 5.14: Explainability results of Zorro on the Cora dataset with GCN (Part 2).

Initializing baseline with fixed node masks. Option 1 serves as the baseline where we start with no initial nodes or features and iteratively add 2 nodes or 10 features at a time with a greediness of 30. In option 2, we initialize and fix the node masks using the important nodes extracted from Zorro Baseline. These node masks are derived by adding nodes according to the top similar nodes list. We then allow the addition of features only.

When comparing the results of (1) and (2), we observe that option 2 results in lower node sparsity, lower RDT-Fidelity, and significantly higher feature sparsity. The issue with option 2 lies in its approach of fixing the selected nodes, which may result in missing potentially important nodes, and subsequently, it continues adding features to achieve the target RDT-Fidelity. Due to these shortcomings, we discard option 2 and proceed with option 3. This rationale also justifies why we did not adopt the reduced subgraph approach.

Initializing baseline with flexible node masks. Option 3 is based on option 2, where we allow the addition of both nodes and features after the initialization from Zorro Baseline. Comparing the results of (1) and (3), we notice that Zorro can achieve the same level of RDT-Fidelity with nearly 50% of the original computation time. While the node sparsity is higher when initializing with node masks, fewer features are used in this case. However, as the target RDT-Fidelity increases from 0.7 to 0.9, the explanation time also increases significantly even for option 3. Therefore, if option 1 were used, the time would be expected to be much higher.

Reducing node greediness. Originally, the same level of greediness was applied to both nodes and features. Option 4 now reduces the node greediness to expedite searching based on option 3. As shown in (4) and (5) in the results, when the threshold set to be 0.9, reducing the node greediness from 30 to 10 achieves a 25.22% reduction in search time while maintaining the same level of performance. This indicates the effectiveness of lowering node greediness. Even though we search with more nodes, we still add 2 at a time. Therefore, setting a refined scope is beneficial.

Limiting maximum optimization steps. Option 5 imposes a constraint of a maximum of 100 optimization steps based on option 4, which helps prevent excessive computation time in certain cases. Comparing the results of (7) and (8), under the target RDT-Fidelity of 0.95, there is a slight decrease in RDT-Fidelity from 0.9541 to 0.9373. However, the explanation time is reduced by 19.39% from 3,506.47 to 2,936.95 seconds. Node sparsity remains nearly the same, and option 5 shows better feature sparsity.

Under the target RDT-Fidelity of 0.9, results from (5) and (6) show that the reduction in explanation time is rather limited, decreasing from 2,864.35 to 2,602.65 seconds, with similar performance in metrics. Therefore, limiting optimization steps is more beneficial when a high target threshold is set, as more steps are needed to achieve optimal.

5.4. RESULT ANALYSIS

In this section, we present the final performances of all explainers in Table 5.15 and 5.16 for Cora and Table 5.17 and 5.18 for PubMed. We next analyze the results to compare different explainers. Apart from the baseline explainers, Zorro Baseline, and Zorro, we also include the performance of random explanations in Table 5.7 and 5.8 with the similar level of node sparsity with GNNExplainer and Zorro. It would be 40% of all nodes for Cora and 10% for PubMed, which means we randomly select this proportion of nodes in each computation graph to act as the explanation node masks. Note that in this baseline we are selecting all features, therefore we focus on the node sparsity and fidelity metrics.

	RDT-Fidelity ↑	**Feature Sparsity ↓**	**Edge Sparsity ↓**	**Node Sparsity ↓**	**Explanation Time/s ↓**
GNNExplainer	0.8463	5.5002	3.7031	2.8024	302.94
IG	0.6082	4.1736	3.0121	2.3973	167.03
Deconvolution	0.6111	6.4488	3.0563	2.4191	336.73
PGMExplainer	0.7773	-	-	2.7728	122.70
Zorro Baseline, 0.95	0.9602	7.2675	-	2.2697	261.69
Zorro, 0.95, option 5	0.9373	4.7575	-	3.0343	2936.95
Random 40% nodes	0.8078	7.2675	-	2.7128	-

Table 5.15: Explainability results on the Cora dataset with GCN (Part 1).

	Fidelity+ Label ↑	**Fidelity+ Prob ↑**	**Fidelity- Label ↓**	**Fidelity- Prob ↓**
GNNExplainer	-0.0108	0.3539	-0.0108	-0.0570
IG	-0.0108	0.3602	-0.0108	0.3602
Deconvolution	-0.0108	0.3525	-0.0108	0.3532
PGMExplainer	0.0323	0.1659	0.1720	0.2780
Zorro Baseline, 0.95	0.2258	0.2813	0.0108	0.1907
Zorro, 0.95, option 5	0.0645	0.3568	0.0215	0.2292
Random 40% nodes	0.0860	0.2012	0.0860	0.2694

Table 5.16: Explainability results on the Cora dataset with GCN (Part 2).

	RDT-Fidelity ↑	**Feature Sparsity ↓**	**Edge Sparsity ↓**	**Node Sparsity ↓**	**Explanation Time/s ↓**
GNNExplainer	0.8400	5.8658	4.9373	3.7371	1552.23
IG	0.6726	4.3435	3.9270	2.999	609.86
Deconvolution	0.6723	5.4119	3.9902	3.0346	866.74
PGMExplainer	0.8188	-	-	3.3096	1879.04
Zorro Baseline, 0.95	0.9502	6.2146	-	2.3613	815.20
Zorro, 0.95, option 5	0.9457	3.5243	-	2.8746	3335.44
Random 10% nodes	0.7441	6.2146	-	2.5683	-

Table 5.17: Explainability results on the PubMed dataset with GCN (Part 1).

Baseline explainers. Among the four baselines, we observe that gradient-based approaches of IG and Deconvolution get very low RDT-Fidelity compared with other soft-masking baselines like PGMExplainer and GNNExplainer. Lower RDT-Fidelity means that the explanations extracted are unstable and subject to prediction changes when we perturb the unimportant features and nodes. This suggests that GNN-specific explainers are superior to directly adapting other methods. Although IG has lower feature sparsity and both IG and Deconvolution show lower edge sparsity and node sparsity, these should not be considered merits when the explanation is unreliable.

Furthermore, across all four traditional fidelity metrics, IG and Deconvolution per-

	Fidelity+ Label ↑	**Fidelity+ Prob ↑**	**Fidelity- Label ↓**	**Fidelity- Prob ↓**
GNNExplainer	0.0000	0.3688	0.0213	0.0665
IG	0.0000	0.3743	0.0000	0.3743
Deconvolution	0.0000	0.3683	0.0000	0.3684
PGMExplainer	0.0000	0.1070	0.0957	0.3094
Zorro Baseline, 0.95	0.0851	0.1169	0.0213	0.3279
Zorro, 0.95, option 5	0.1596	0.3484	0.1170	0.3468
Random 10% nodes	0.0213	0.0401	0.1170	0.3660

Table 5.18: Explainability results on the PubMed dataset with GCN (Part 2).

form similarly, expect for Fidelity- Prob. This indicates that traditional fidelity metrics are not discriminative enough in evaluating explainers.

For both datasets, GNNExplainer outperforms all other 3 explainers in RDT-Fidelity, as its explanations are based on the computation subgraph, which better preserves the important information in a structured manner. However, the higher feature, edge, and node sparsity suggest that GNNExplainer uses a larger proportion of elements to explain the predictions, and the explanations may not be compact enough. PGMExplainer only returns node masks and by default selects all features and edges, therefore only the comparison between node sparsity is possible.

Zorro and Zorro Baseline. Note that RDT-Fidelity measures the stability of the explanations. Since Zorro is specially designed to optimize RDT-Fidelity, we expect to observe high RDT-Fidelity for Zorro. In both datasets, Zorro Baseline and Zorro explainers show the highest RDT-Fidelity scores compared to other explainers. This indicates that Zorro's approach is more effective at maintaining faithfulness to the model's predictions. High RDT-Fidelity indicates that the explanations are more stable when we randomly perturb the unimportant elements. However, Zorro takes the most time to explain due to its greedy nature.

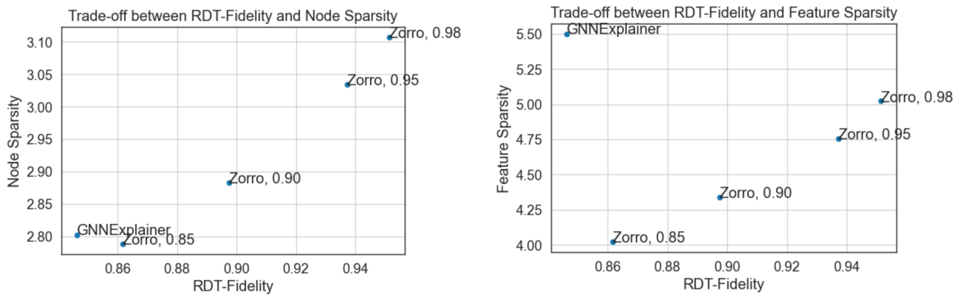
For the Cora dataset, Zorro Baseline has the lowest node sparsity but high feature sparsity since it uses all the features. While considering the balance between different metrics, Zorro achieves the best performance with RDT-Fidelity and feature sparsity, while it has slightly higher node sparsity but remains competitive. Despite the longer explanation time for Zorro, its performance in fidelity metrics highlights its effectiveness in providing accurate, robust, and stable explanations.

Comparing the Zorro Baseline and Random Baseline, where in both cases all features are selected, Zorro Baseline significantly outperforms the random one in terms of RDT-Fidelity and node sparsity, as well as Fidelity+ and Fidelity- metrics. This highlights Zorro Baseline's ability to extract important nodes while keeping the sparsity low.

Trade-off of RDT-Fidelity and Sparsity for Zorro. Since Zorro is a greedy optimization process, there exists a target RDT-Fidelity we aim to achieve, and therefore a trade-off between RDT-Fidelity and Sparsity metrics. Often the case, these two metrics are combined to evaluate the performance of explainers. Implementing all the above-mentioned enhancements in Zorro, we report the performance of Zorro under different thresholds and also GNNExplainer on the Cora dataset in Table 5.19, and provide a scatter plot of Figure 5.2 for the results.

	RDT-Fidelity ↑	**Feature Sparsity ↓**	**Edge Sparsity ↓**	**Node Sparsity ↓**	**Explanation Time/s ↓**
GNNExplainer	0.8463	5.5002	3.7031	2.8024	302.94
Zorro, 0.85	0.8616	4.0220	-	2.7883	2438.33
Zorro, 0.90	0.8975	4.3394	-	2.8829	2602.65
Zorro, 0.95	0.9373	4.7575	-	3.0343	2936.95
Zorro, 0.98	0.9514	5.0231	-	3.1075	3945.86

Table 5.19: Zorro performance under different thresholds.



(a) Node Sparsity (b) Feature Sparsity

Figure 5.2: Trade-off between RDT-Fidelity and Sparsity Metrics.

In both scatter plots, we aim to achieve a point in the lower right section, representing high RDT-Fidelity and low node and feature sparsity. We observe that higher thresholds in Zorro would result in higher RDT-Fidelity but at the cost of increased node and feature sparsity. This indicates that while the explanations become more faithful to the model's original predictions, they also become more complex, using more nodes and features. It would also take more time to search for explanations.

The specific choice of threshold in Zorro should balance between the desired fidelity, the complexity of the explanation, and the searching time. Lower thresholds offer more compact explanations but with lower fidelity, while higher thresholds provide better fidelity at the cost of complexity. Compared with GNNExplainer, Zorro consistently has lower feature sparsity. However, when the threshold is above 0.9, Zorro's node sparsity becomes higher. Combining these performances, we conclude a target RDT-Fidelity of 0.9 would achieve a nice balance across these metrics. However, a threshold of 0.95 can also be chosen if we aim for high RDT-Fidelity.

5.5. CASE STUDY: ANALYZING GNNEXPLAINER AND ZORRO.

Apart from the quantitative analysis, we randomly sample several edges, extract the explanations using GNNExplainer and Zorro, and visualize the results to provide a qualitative analysis in Figure 5.3 and 5.4.

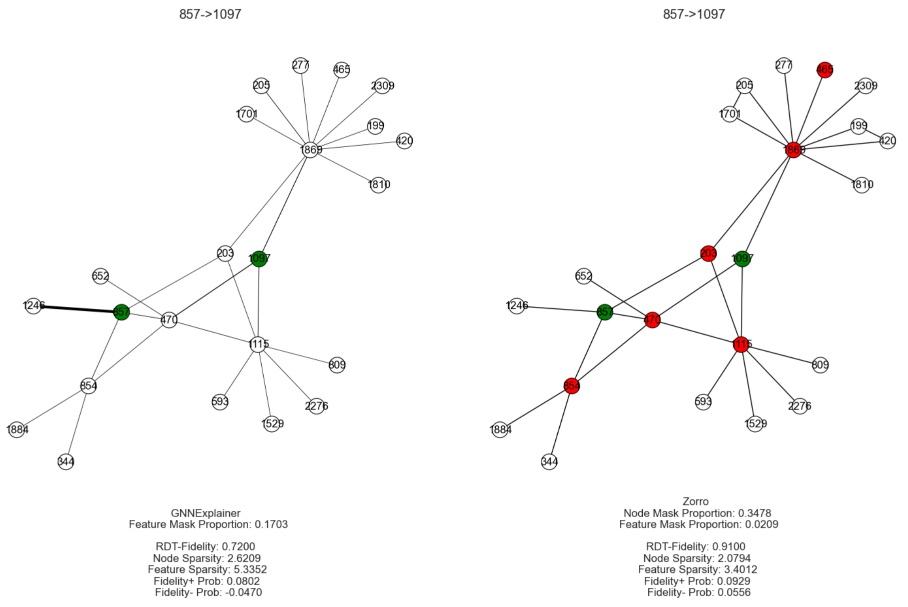


Figure 5.3: The explanation of edge (857, 1097) in Cora dataset.

For GNNExplainer, we visualize the edge masks according to their continuous importance values, with the bold edges representing the more important ones. However, determining which edges are the most critical is challenging. For example, in this case, only one edge is significantly more important than the others.

In contrast, Zorro uses hard node masks, allowing us to directly visualize the important nodes essential in the explanation. In terms of evaluation metrics, Zorro has higher RDT-Fidelity, lower node sparsity, and significantly lower feature sparsity. Zorro utilizes only 2.1% of features, compared to GNNExplainer's 17%. Therefore, Zorro outperforms GNNExplainer in this case.

However, Zorro does not always act better than GNNExplainer. For example, in another case of Figure 5.4, though Zorro has better RDT-Fidelity, it selects all nodes in the computation graph in the explanations. This occurs because Zorro operates on features and nodes, if including nodes results in a greater improvement in RDT-Fidelity compared to features, then these nodes are selected. This is more likely to happen when the threshold we set is high, causing Zorro to include more nodes and features. Additionally, considering the number of edges in the computation graph, it is less likely for GNNExplainer to select all the edges. Therefore, GNNExplainer can sometimes provide better visualizations.

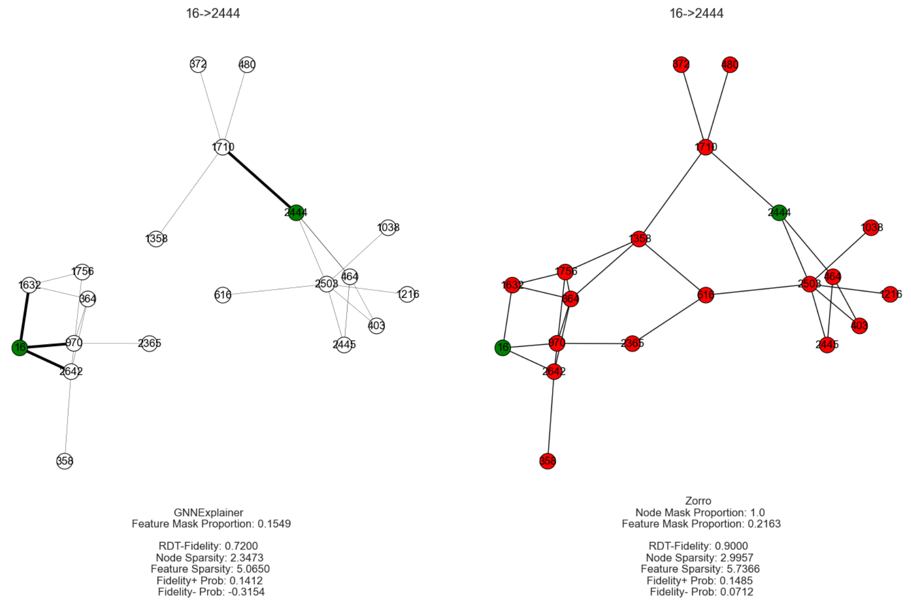


Figure 5.4: The explanation of edge (16, 2444) in Cora dataset.

6 CONCLUSION

In this chapter, we summarize our paper and answer the research question we proposed. Moreover, we conclude with possible future directions.

6.1. THESIS SUMMARY

In this thesis, we focused on the explainability of Graph Neural Networks (GNNs) in the context of link prediction tasks. To address this, we first developed an efficient extension of the Zorro explainer to accommodate the unique challenges of link prediction. This included methods such as using top similar nodes for initialization, resulting in more informative binary node and feature masks as explanations.

Additionally, we introduced and evaluated new metrics, particularly RDT-Fidelity, Edge-level RDT-Fidelity, and Sparsity to better assess explainers. Experiments on Cora and PubMed demonstrated the effectiveness and efficiency of our Zorro extension.

In summary, this work advances GNN explainability for link prediction by providing new methods and evaluation metrics, with potential for future application in more complex GNN models and other graph-related tasks.

6.2. ANSWER TO RESEARCH QUESTIONS

In this section, we conclude the experiments and address the research questions we proposed at the beginning with our results.

RQ 1: How do we define the explanation of a GNN for a link prediction task? We define the explanations in link prediction as the masks over node, edge, and features. In Zorro, we only use binary node and feature masks for better interpretability. Our experiments also indicate that edges may contribute little to the predictions of links when the graph dataset is relatively simple and does not contain many edges.

RQ 2: How do we develop efficient explainers for link prediction? To develop tailor-made explainers for link prediction, it is crucial to consider the unique nature of this task, which involves predicting relationships between pairs of nodes rather than individual node predictions. Faced with the challenges of *Complexity* mentioned in the introduction chapter, we developed an extension of Zorro by incorporating a larger computation graph during the explanation process.

To optimize the efficiency of Zorro, we developed several techniques and verified their effectiveness. One notable improvement is the initialization of the node masks with those from the Zorro Baseline before beginning the greedy search. These top nodes are selected from a similarity list computed between the neighbors of the source node and the target node and vice versa, thereby incorporating the interactions between neighbors into the explanation process.

The results show that Zorro can achieve high RDT-Fidelity scores and low node and feature sparsity values, representing its ability to extract faithful, stable, and sparse explanations compared with other methods. However, there exists a trade-off about which threshold to choose during optimization, which would have some influence on the balance between metrics.

RQ 3: Which metrics can be used to evaluate the quality of explanations to achieve a well-rounded evaluation? Evaluating the quality of explanations in link prediction requires a combination of metrics to ensure a balanced and fair assessment. Based on our results, we conclude that RDT-Fidelity and Sparsity metrics are more effective in evaluations.

Traditional fidelity metrics like Fidelity+ and Fidelity+ also work in some way. Higher fidelity+ means the prediction is significantly influenced by the removal of nodes, and lower Fidelity- means the explanation alone can already lead to the same prediction and, therefore better explanation. However, these metrics are less discriminative in evaluating explainers and can be difficult to interpret. Additionally, they may face distribution shift problems as stated by [34]. Nonetheless, they can act as supplements to other metrics.

RDT-Fidelity measures how well the explanation preserves the original prediction when unimportant elements are perturbed. It is a strong indicator of the explanation's reliability, relevance, and stability since we perturb the features 100 times for each edge to be explained. Additionally, we proposed the edge-level RDT-Fidelity, where we could evaluate explainers with edge masks. Node, edge, and feature sparsity metrics provide insights into the compactness of the explanation. While higher sparsity can indicate a more concise explanation, it should not come at the cost of lower fidelity. The balance between sparsity and fidelity is crucial, as demonstrated by the trade-offs observed in our experiments with different explainers.

Combining RDT-Fidelity with sparsity metrics offers a more detailed evaluation of an explainer's performance, ensuring that explanations are both reliable and concise. It allows for a more comprehensive assessment of different explainers, leading to more informed decisions in selecting the best methods for explaining link predictions.

6.3. FUTURE WORK

Apart from the work we have done, several directions for future exploration could further enhance the methods and address existing limitations.

6.3.1. ADVANCED NODE SIMILARITY TECHNIQUES

The method for selecting top similar nodes could be enhanced with more complex techniques. The current version of calculating similarity between neighbors is based on the embeddings. Some more advanced ones could be considered. For example, the high-order structure similarity proposed in [35].

6.3.2. BEYOND TRUE POSITIVE EXPLANATIONS

Our current approach focused only on addressing explanations for true positive predictions, but future work could expand this to include explanations for false positives, false negatives, and true negatives. Understanding why a model incorrectly predicts the existence of a link can be as valuable as understanding correct predictions. This broader focus would provide a more comprehensive view of the model's behavior and could uncover potential biases or weaknesses in the model that might be missed when focusing only on true positives.

6.3.3. NEGATIVE EXPLANATION SCORES

In some cases, explainers may produce negative scores, indicating features or nodes that decrease the likelihood of a correct prediction. Our current approach ignores the negative scores and assigns them to zero. However, they may provide critical insights into which graph aspects harm the model performance. Future work could explore how to better interpret and utilize these negative attributions.

These directions for future research will help fill the research gap in the explainability of GNNs, leading to more robust and interpretable models for link prediction tasks.

Bibliography

[1] Chirag Agarwal, Owen Queen, Himabindu Lakkaraju, and Marinka Zitnik. "Evaluating explainability for graph neural networks". In: *Scientific Data* 10.1 (2023), p. 144.

[2] Kenza Amara, Rex Ying, Zitao Zhang, Zhihao Han, Yinan Shan, Ulrik Brandes, Sebastian Schemm, and Ce Zhang. "Graphframex: Towards systematic evaluation of explainability methods for graph neural networks". In: *arXiv preprint arXiv:2206.09677* (2022).

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140.

[4] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. "Robust counterfactual explanations on graph neural networks". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 5644–5655.

[5] Claudio Borile, Alan Perotti, and André Panisson. "Evaluating Link Prediction Explanations for Graph Neural Networks". In: *World Conference on Explainable Artificial Intelligence*. Springer. 2023, pp. 382–401.

[6] Niklas Breustedt, Paolo Climaco, Jochen Garcke, Jan Hamaekers, Gitta Kutyniok, Dirk A Lorenz, Rick Oerder, and Chirag Varun Shukla. "On the Interplay of Subset Selection and Informed Graph Neural Networks". In: *arXiv preprint arXiv:2306.10066* (2023).

[7] Lukas Faber, Amin K. Moghaddam, and Roger Wattenhofer. "When comparing to ground truth is wrong: On evaluating gnn explanation methods". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 332–341.

[8] Thorben Funke, Megha Khosla, Mandeep Rathee, and Avishek Anand. "Zorro: Valid, sparse, and stable explanations in graph neural networks". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Advances in neural information processing systems* 30 (2017).

[10] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. "Graphlime: Local interpretable model explanations for graph neural networks". In: *IEEE Transactions on Knowledge and Data Engineering* 35.7 (2022), pp. 6968–6972.

[11] Thomas N Kipf and Max Welling. "Semi-supervised classification with graph convolutional networks". In: *arXiv preprint arXiv:1609.02907* (2016).

[12] Mert Kosan, Samidha Verma, Burouj Armgaan, Khushbu Pahwa, Ambuj Singh, Sourav Medya, and Sayan Ranu. "GNNX-BENCH: Unravelling the Utility of Perturbation-based GNN Explainers through In-depth Benchmarking". In: *arXiv preprint arXiv:2310.01794* (2023).

[13] Ajay Kumar, Shashank Sheshar Singh, Kuldeep Singh, and Bhaskar Biswas. "Link prediction techniques, applications, and performance: A survey". In: *Physica A: Statistical Mechanics and its Applications* 553 (2020), p. 124289.

[14] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. "Cf-gnnexplainer: Counterfactual explanations for graph neural networks". In: *International Conference on Artificial Intelligence and Statistics.* PMLR. 2022, pp. 4499–4511.

[15] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. "Parameterized explainer for graph neural network". In: *Advances in neural information processing systems* 33 (2020), pp. 19620–19631.

[16] Dimitris Margaritis and Sebastian Thrun. "Bayesian network induction via local neighborhoods". In: *Advances in neural information processing systems* 12 (1999).

[17] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. "Explainability methods for graph convolutional neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 2019, pp. 10772–10781.

[18] Mandeep Rathee, Thorben Funke, Avishek Anand, and Megha Khosla. "BAGEL: A Benchmark for Assessing Graph Neural Network Explanations". In: *arXiv preprint arXiv:2206.13983* (2022).

[19] Benjamin Sanchez-Lengeling, Jennifer Wei, Brian Lee, Emily Reif, Peter Wang, Wesley Qian, Kevin McCloskey, Lucy Colwell, and Alexander Wiltschko. "Evaluating attribution for graph neural networks". In: *Advances in neural information processing systems* 33 (2020), pp. 5898–5910.

[20] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. "Collective classification in network data". In: *AI magazine* 29.3 (2008), pp. 93–93.

[21] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps". In: *arXiv preprint arXiv:1312.6034* (2013).

[22] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International conference on machine learning.* PMLR. 2017, pp. 3319–3328.

[23] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. "Graph attention networks". In: *stat* 1050.20 (2017), pp. 10–48550.

[24] Minh Vu and My T Thai. "Pgm-explainer: Probabilistic graphical model explanations for graph neural networks". In: *Advances in neural information processing systems* 33 (2020), pp. 12225–12235.

[25] Zhen Wang, Bo Zong, and Huan Sun. "Modeling context pair interaction for pairwise tasks on graphs". In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 851–859.

[26] Yaochen Xie, Sumeet Katariya, Xianfeng Tang, Edward Huang, Nikhil Rao, Karthik Subbian, and Shuiwang Ji. "Task-agnostic graph explanations". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 12027–12039.

[27] Makoto Yamada, Wittawat Jitkrittum, Leonid Sigal, Eric P Xing, and Masashi Sugiyama. "High-dimensional feature selection by feature-wise kernelized lasso". In: *Neural computation* 26.1 (2014), pp. 185–207.

[28] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. "Gnnexplainer: Generating explanations for graph neural networks". In: *Advances in neural information processing systems* 32 (2019).

[29] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. "Explainability in graph neural networks: A taxonomic survey". In: *IEEE transactions on pattern analysis and machine intelligence* 45.5 (2022), pp. 5782–5799.

[30] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. "On explainability of graph neural networks via subgraph explorations". In: *International conference on machine learning*. PMLR. 2021, pp. 12241–12252.

[31] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer. 2014, pp. 818–833.

[32] Muhan Zhang and Yixin Chen. "Link prediction based on graph neural networks". In: *Advances in neural information processing systems* 31 (2018).

[33] Shichang Zhang, Jiani Zhang, Xiang Song, Soji Adeshina, Da Zheng, Christos Faloutsos, and Yizhou Sun. "PaGE-Link: Path-based graph neural network explanation for heterogeneous link prediction". In: *Proceedings of the ACM Web Conference 2023*. 2023, pp. 3784–3793.

[34] Xu Zheng, Farhad Shirani, Tianchun Wang, Wei Cheng, Zhuomin Chen, Haifeng Chen, Hua Wei, and Dongsheng Luo. "Towards robust fidelity for evaluating explainability of graph neural networks". In: *arXiv preprint arXiv:2310.01820* (2023).

[35] Huaisheng Zhu, Dongsheng Luo, Xianfeng Tang, Junjie Xu, Hui Liu, and Suhang Wang. "Self-Explainable Graph Neural Networks for Link Prediction". In: *arXiv preprint arXiv:2305.12578* (2023).