Link Explanation for Heterogeneous Graphs

Abhijit Gupta, advised by Rex Ying

December 15, 2022

Abstract

Graph Neural Networks (GNNs) utilize message passing to operate directly on graph structured data, and have achieved state-of-the-art performance in node classification, link prediction, and graph classification tasks. In order to build trust, promote transparency, and facilitate real-world applications, GNNs must be made explainable. But while models such as GNNExplainer and SubgraphX explain node and graph classification tasks, link explanation, particularly on heterogeneous graphs, remains underexplored. In this work, we propose a new explanation format for link prediction, where only a subset of immediate neighbors of the target edge are selected. We test four explanation methods on the Facebook, IMDB, and LastFM heterogeneous graph datasets. Each dataset has distinct properties that encourage a robust explanation model. We sample explanations of varying sparsity and measure the characterization score to assess a candidate explanation's necessity and sufficiency. We report characterization scores for 1-node, 5-node, and 10-node explanations. We make key changes to the GNNExplainer loss function and to the SubgraphX filtering method that yield significant improvements in explanation quality. Overall, we find our modified SubgraphX method outperforms existing baselines by 11% and our modified GNNExplainer outperforms existing baselines by 21% on average. The embedding baseline remains a fast and viable approach for small explanations on high degree graphs, while GNNExplainer produces the strongest explanations across all other configurations. Additionally, we make open-source contributions to the PyTorch Geometric library to allow for future extensions. Altogether, our work is the first exploration of heterogeneous link explanation and lays the foundation for future explanation approaches.

Contents

1 Introduction 4

2 Related Work 5
2.1 Graph Neural Networks . . . 5
2.2 Heterogeneous Graphs . . . 5
2.3 Explainability for GNNs . . . 6
2.4 Explanation Metrics . . . 8

3 Methods 9
3.1 Link Explanation Format . . . 9
3.2 Experimental Design . . . 10
3.3 Link Prediction . . . 10
3.4 Link Explanation . . . 11
3.4.1 Random Baseline . . . 11
3.4.2 Embedding Baseline . . . 11
3.4.3 GNNExplainer . . . 12
3.4.4 SubgraphX . . . 13
3.5 Thresholding and Evaluation . . . 14

4 Experiments 14
4.1 Datasets . . . 14
4.1.1 Facebook Ego Dataset . . . 15
4.1.2 IMDB Dataset . . . 15
4.1.3 LastFM Dataset . . . 15
4.2 Link Prediction . . . 16
4.3 Link Explanation . . . 17
4.3.1 Baselines . . . 17
4.3.2 GNNExplainer Variants . . . 19
4.3.3 SubgraphX Variants . . . 21
4.4 Analysis . . . 23
4.5 Open-Source Contributions . . . 25

5 Conclusion 25

1 Introduction

Recently, deep learning techniques have achieved strong performance on several artificial intelligence tasks, from natural language generation to image and speech recognition. Graph data emerges in many contexts (social networks, molecules, knowledge graphs) but creates unique challenges for traditional ML methods. Graph Neural Networks (GNNs) are a type of neural network that operate directly on the graph structure to perform node and graph classification, link prediction, and more [1]. In order to facilitate real-world applications, machine learning models must be made explainable, either during initial computation or post hoc. Just as graph structured data poses challenges to traditional deep learning methods, new explainability methods are required to interpret GNN outputs [2]. However, the explanation of GNN link prediction, particularly on heterogeneous graphs, is still underexplored.

In this work, we extend existing GNN explanation methods to the heterogeneous link explanation task. Explanations are restricted to immediate neighbors for increased interpretability and real-world use cases. We focus specifically on model-output structural explanations using instance-level perturbation methods. In particular, we examine GNNExplainer [3] and SubgraphX [4], proposing modifications to better adapt to our specific experiments. We evaluate explanations on the Facebook Ego [5], IMDB [6], and LastFM [7] datasets using fidelity metrics for varying explanation sparsity. On the Facebook and IMDB datasets, GNNExplainer and SubgraphX yield significantly better explanations than the baselines at all sparsity levels, with GNNExplainer slightly outperforming SubgraphX. On the LastFM dataset, the embedding baseline performs comparably to SubgraphX and slightly worse than GNNExplainer, but outperforms both methods for one-node explanations. Additionally, we make open-source contributions to PyTorch Geometric to allow for future extension. Altogether, our work is the first exploration of heterogeneous link explanation and encourages new explanation methods specifically designed to leverage heterogeneous graph metapaths.

2 Related Work

2.1 Graph Neural Networks

Let G denote a graph with edges E and nodes V, with nodes having d-dimensional features $X = \{x_1, x_2, \ldots, x_n\}$, $x_i \in \mathbf{R}^d$. GNN models uses features X and edges E to learn various tasks, including node classification/regression, link prediction, and graph classification. A GNN contains multiple stacked layers, each separated into a "Message", "Aggregation", and "Update" step [8]. This format allows for message passing, where features of neighboring nodes are aggregated into the central node [9]. Convolutional GNNs generalize the convolution operation from grid data to graph data, using either spectral or spatial methods. Two of the most impactful and effective methods are the Graph Convolutional Network (GCN) [10] and GraphSAGE [11]. These and other methods are often evaluated on citation, bioinformatics, and social network datasets [12, 13].

2.2 Heterogeneous Graphs

Heterogeneous graphs contain multiple types of node and/or edges, adding a layer of complexity beyond homogeneous graph learning. Formally, $G = (V, E, R, A)$, with the additional attributes R and A corresponding to node and edge type mappings respectively. Each heterogeneous graph consists of multiple node and/or edge type mappings, such that $|R| + |A| > 2$. Heterogeneous data is very common in real-world settings, including knowledge graphs, e-commerce (user-product interactions), and social networks. Moreover, homogeneous GNNs have been adapted to support heterogeneous graphs by iterating over all node/edge types, and specialized models such as HAN [14] and RGCN [15] explicitly leverage heterogeneous graph meta-paths to aggregate information. Figure 1 shows the multiple node types and meta-paths in the heterogeneous IMDB dataset. Meanwhile, there have been no major works explaining heterogeneous graph tasks, either extending homogeneous methods or leveraging heterogeneous graph attributes.

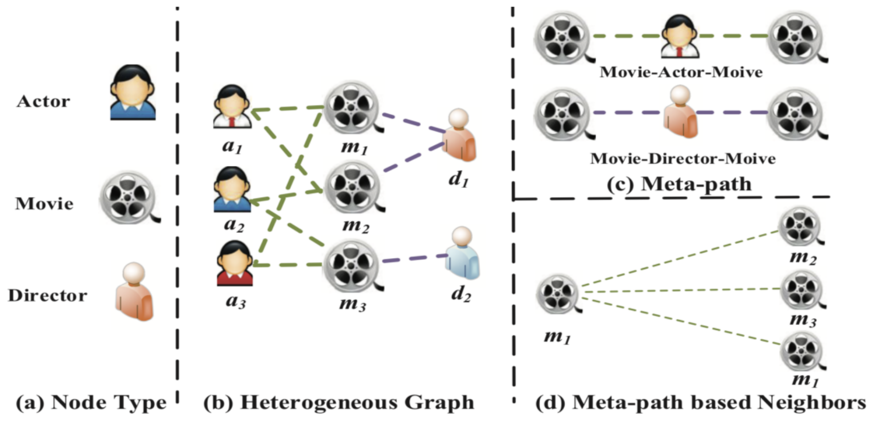


Figure 1: Heterogeneous IMDB Graph Meta Paths, adapted from Wang et. al [14]

2.3 Explainability for GNNs

Explainability builds trust, promotes transparency and fairness, and can improve human-in-the-loop performance. Several unique explanation methods have been developed to demystify black-box graph neural networks. Figure 2 helps categorize several promising GNN explanation methods. Most recent work has gone into instance-level explanation, which provides input-dependent predictions for each input graph. Gradient and decomposition based approaches requires access to the prediction model backwards computation while perturbation and surrogate models only require the prediction model and output [2].

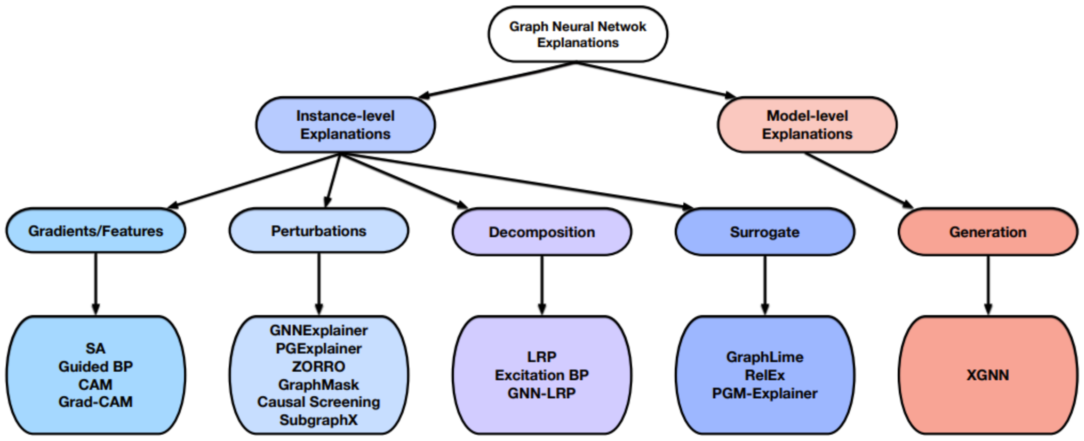


Figure 2: Taxonomy of GNN Explanation Methods, adapted from Yuan et. al [2]

We focus on instance-level perturbation explanations, as these are model agnostic and can be applied to any prediction GNN architecture. Explanations from these models can take a combination of two forms. First, structural explanations explain a target prediction by a subset of the surrounding neighborhood. A k-layer GNN aggregates information from a node's k-hop neighbors, so the k-hop neighborhood is traditionally used to construct the explanation. The left panel of figure 3 shows how the red node can be explained by the green subgraph in its neighborhood. Alternatively, an explanation can identify relevant features of the neighborhood nodes or edges that contribute to the model prediction. These attributes can be common across the entire neighborhood or specific to certain neighbors. Explanation methods can also jointly output a structural and feature explanation.

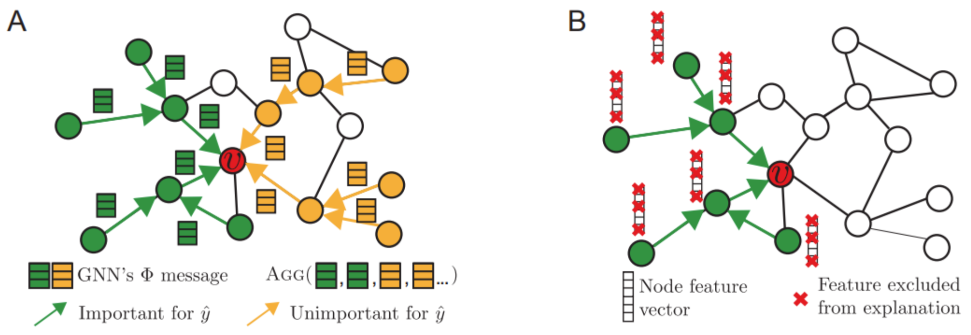


Figure 3: (A) Structural and (B) Feature explanation format, adapted from Ying et. al [3]

Among permutation instance-level explanations, GNNExplainer [3], PGExplainer [16] and SubgraphX [4] are some of the most popular approaches. GNNExplainer and PGExplainer initializes edge and node feature masks that define a smaller explanation subgraph. The masks are optimized to maximize the mutual information between the explanation and original graph. PGExplainer extends GNNExplainer by adopting a deep neural network to parameterize the generation process of explanations. Meanwhile SubgraphX uses Monte Carlo Tree Search to explore potential subgraphs and Shapley values to select the optimal explanation. GraphLIME is a surrogate method that extends the LIME algorithm to graph structured data [17]. A recent systematic evaluation on real-world node and graph explanation tasks found no single method dominates the others on all evaluation criteria [18].

2.4 Explanation Metrics

GNN explanation models must balance several goals including fidelity, sparsity, stability, and accuracy [2]. Several metrics have been proposed for each goal individually [19, 20], but many explanation models utilize their own metrics. GraphFramEx proposes a unified set of quantitative metrics to measure explanation quality. They introduce a distinction between model explanations that explain a model's (potentially incorrect) prediction and phenomenon explanations that explain the ground truth classification. Explanations are evaluated as sufficient if they individually result in the correct prediction, and necessary if their exclusion causes the model to change output. Fidelity+ and Fidelity- capture these qualities, and the characterization score combines both measurements [18].

$$fid_{+}^{prob} = \frac{1}{N}\sum_{i=1}^{N} |f(G_C)_{\hat{y}_i} - f(G_{C\setminus S})_{\hat{y}_i}| \tag{1}$$

$$fid_{-}^{prob} = \frac{1}{N}\sum_{i=1}^{N} |f(G_C)_{\hat{y}_i} - f(G_S)_{\hat{y}_i}| \tag{2}$$

$$charact = \frac{(w_+ + w_-) \times fid_+ \times (1 - fid_-)}{w_+ \times (1 - fid_-) + w_- \times fid_+} \tag{3}$$

where N is the number of explanation targets, $f(G_C)_{\hat{y}_i}$ is the original model prediction, $f(S)_{\hat{y}_i}$ is the explanation subset model prediction, $f(G \setminus S)_{\hat{y}_i}$ is the graph except the explanation subset model prediction, and w_+ and w_- are the respective weights of Fidelity+ and Fidelity- in the Characterization score.

While characterization score does not handle factors such as the out-of-distribution problem [21] and hindsight bias [22], it is a strong first step for model comparison. Finally, sparsity measures the size of the explanation in relative or absolute terms. The combination of a graph dataset, prediction model, explanation models, and metrics define an experiment.

3 Methods

3.1 Link Explanation Format

In this work, we focus specifically on structural explanations of link prediction using instance-level perturbation methods. While previous literature has used the k-hop neighborhood subgraphs to construct node and graph explanations, we adapt this format for link explanation specifically.

With an eye towards real-world applications and interpretability, we restrict explanations to the immediate neighbors of either the target edge endpoint, as opposed to the union of their k-hop neighborhoods. The resulting explanations, a set of direct neighbors, can be represented as a list, unlike the previous subgraphs, which can only be understood visually. This change also drastically shrinks the set of potential explanations and reduces inference time.

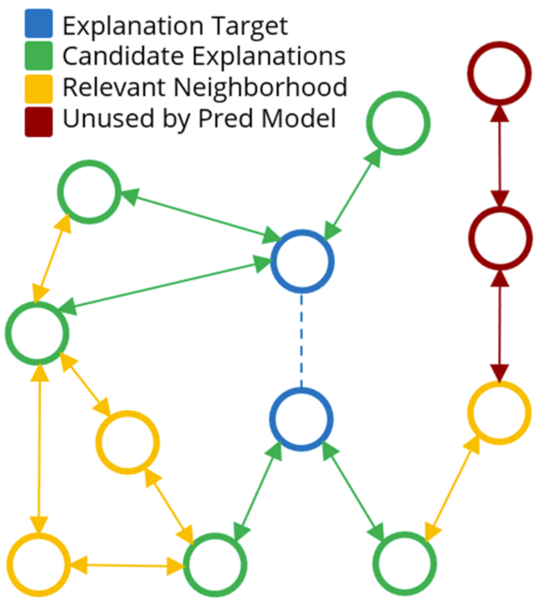


Figure 4: Proposed Link Explanation Format

In Figure 4, we have a homogeneous graph and seek to explain the link between the two blue nodes. Only the green nodes (direct neighbors) can contribute to any proposed explanation. The yellow nodes and edges are relevant to a 2-layer GNN prediction and are always included in any explanation subgraph. Finally, the red nodes are beyond the 2-hop neighborhood and do not contribute to the explanation. This link explanation format is easily extended to the heterogeneous case by allowing direct neighbors of any edge and node type.

3.2 Experimental Design

The goal of the experiment is to evaluate models for explaining heterogeneous graph link predictions. Towards this end, we use the following experimental setup.

1. Obtain dataset. If homogeneous, add node/edge types and transform to heterogeneous. Apply any feature engineering or preprocessing. Separate into train/val/test splits.

2. Train a link prediction model using a GNN Encoder/Decoder model. Obtain model predictions on previously unseen test edges. Filter out invalid predictions.

3. For each link prediction, apply the explanation models, which returns instance-specific weights for each direct neighbor. Sort the candidate nodes by the explanation weight.

4. Increment from a one-node explanation to a N-node explanation, measuring the characterization score. Each subsequent explanation builds upon the previous.

5. Average the characterization scores across all link predictions. Compare characterization score as a function of sparsity for each explanation methods.

Our pipeline aims to cleanly separate between dataset processing, link prediction, multiple link explanation methods, thresholding, and evaluation. This makes the project extensible, allowing for additional datasets and explanation models in the future. The following subsections expand upon this high-level methodology.

3.3 Link Prediction

We use a standard GNN Encoder/Decoder architecture for heterogeneous graph link prediction. This model uses message passing layers to learn node embeddings and applies another function to compare two node embeddings and return a similarity score. We use both GraphSAGE and GCN to learn useful node embeddings from the input graph. Future work can be done to implement heterogeneous graph specific models such as HAN and MAGNN that better incorporate intra- and inter-metapath aggregations [6, 14]. The similarity function can be as simple as a dot product between the two embeddings or a multi-layer perceptron.

To train the link prediction model, we first construct a train/val/test dataset split by withholding certain positive edges in the input graph. Negative edges are sampled and added to each batch to balance the model input. We use cross entropy loss to optimize the binary classification and an Adam optimizer with regularization to perform gradient descent. We use the validation dataset to avoid overfitting while preserving the test dataset for evaluation. Link prediction is evaluated by both accuracy and ROC AUC.

3.4 Link Explanation

We implement two baselines and two GNN explanation methods for the heterogeneous link explanation task. Each model takes as input a target edge, its k-hop neighborhood, and a prediction model, and outputs a weight for each direct neighbor, corresponding to that neighbor's importance in the link prediction. Some link predictions are filtered to satisfy assumptions of the explanation models and metrics, these are described in Section 4.2.

3.4.1 Random Baseline

The random baseline is primarily a sanity check and returns a random weight for each neighbor to the target edge.

3.4.2 Embedding Baseline

The embedding baseline suggests explanations based on the similarity between an explanation node and its non-adjacent target node. If both target nodes are adjacent, we take the maximum similarity to the two link endpoints. Essentially, the embedding baseline attempts to find examples of similar links in the neighborhood to explain the target link. Note for heterogeneous graphs, this method requires all node embeddings to have the same dimension.

3.4.3 GNNExplainer

We implement GNNExplainer [3] and make small adjustments for the heterogeneous link prediction task and our link explanation format. Although GNNExplainer optimizes for both structural and feature explanations, we restrict our analysis to just the structural. GNNExplainer attempts to maximize the mutual information between the ground truth and the explanation subgraph $G_S \subseteq G_C$.

$$\max_{G_S} MI(Y, G_S) = H(Y) - H(Y|G = G_S) \tag{4}$$

Where $H(Y)$ is the entropy. As $H(Y)$ is constant with respect to G_S, this is equivalent to minimizing the conditional entropy $H(Y|G = G_S)$. Expanding out,

$$H(Y|G = G_S) = -\mathbf{E}_{Y|G_S}[\log P_\Phi(Y|G = G_S)] \tag{5}$$

Where P_Φ is the output of the prediction model. GNNExplainer utilizes continuous relaxation to learn a soft adjacency mask $M \in \mathbb{R}^{n\times n}$ on the adjacency matrix A_C, replacing the discrete subgraph G_S with the continuous matrix $A_C \odot \sigma(M)$. The objective, which is optimized via gradient descent, becomes

$$\min_M -\sum_{c=1}^{C} \mathbf{1}[y = c] \log P_\Phi(Y = y|G = A_c \odot \sigma(M)) \tag{6}$$

Where there are C classification labels. Additional terms are integrated to penalize large explanation size and encourage approximately discrete masks. The final objective is

$$\min_M \Big[-H(Y|G = A_C \odot \sigma(M)) + \alpha \sum_{e_i \in M} e_i + \beta \cdot \text{CrossEntropy}(M) \Big] \tag{7}$$

In the GNNExplainer paper, $\alpha = 0.005$, $\beta = 1$, and the objective is minimized with an Adam optimizer with a learning rate of 0.001 for 1000 epochs.

3.4.4 SubgraphX

SubgraphX explains GNN predictions by efficiently testing multiple explanations, measuring subgraph importance with Shapley values [4]. Unlike GNNExplainer, which uses a continuous approximation, SubgraphX does not use gradient descent to learn its objective function, instead using Monte Carlo tree search (MCTS) to identify and explore promising coalitions. The tree structure is necessary to ensure that explanation nodes create a connected component. However, in the link explanation format where only immediate neighbors are considered, this is not necessary. To reduce inference time on par with GNNExplainer, we remove MCTS and employ the greedy algorithm to construct explanations.

SubgraphX simplifies to measuring subgraph importance using Shapley values for each direct neighbor of the target edge. The Shapley value of a subgraph G_i can be computed as

$$\phi(G_i) = \sum_{S \subseteq P \setminus G_i} \frac{|S|!(|P| - |S| - 1)!}{|P|!}(f(S \cup G_i) - f(S)) \tag{8}$$

The obtained Shapley value $\phi(G_i)$ considers all possible coalitions and measures the impact of the inclusion of G_i in the hypothetical explanation. GNNs are often sensitive to structural changes, and so node inclusion/exclusion is modeled by perturbing the node features of removed nodes to 0. Figure 5 visualizes the computation of the Shapley value of coalition $G_i = \{2, 3, 4, 5\}$, with $S \subseteq \{1, 6\}$.

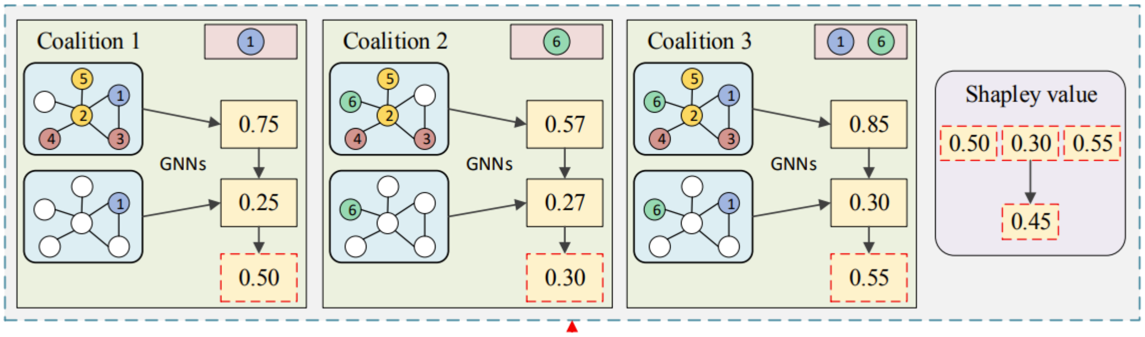


Figure 5: Illustration of Shapley value of $G_i = \{2, 3, 4, 5\}$, adapted from Yuan et. al [4]

In practice, the summation over all possible sets S in Equation (8) becomes computationally expensive as the neighborhood size increases. SubgraphX addresses this issue with Monte-Carlo sampling, using $T = 100$ samples per Shapley value computation.

$$\phi(G_i) = \frac{1}{T}\sum_{i=1}^{T}(f(S_i \cup G_i) - f(S_i)) \tag{9}$$

Compared to GNNExplainer, SubgraphX still has significantly higher inference time. In our experimental setup, G_i only ever includes a single node. Given this simplification, we find that reducing samples to $T = 5$ improves inference time with almost no performance hit.

3.5 Thresholding and Evaluation

Each of the link explanation models described above return a weight for each direct neighbor of either endpoint of the target edge. Instead of using a fixed threshold to select nodes into the explanation, we take the Top K nodes. This has one major advantage of standardizing the explanation size across multiple methods, allowing for a fairer comparison. We report results for explanations with 1, 5, and 10 included nodes. We also compute sparsity by dividing K by the neighborhood size, and compare results as a function of varying sparsity. We measure the Characterization Score, described in Section 2.4. We use $w_+ = 0.2$ and $w_- = 0.8$ to skew the score towards measuring sufficiency.

4 Experiments

4.1 Datasets

As stated earlier, a primary goal of this project is to extend existing explanation methods to heterogeneous graphs. Simultaneously, we seek to maintain homogeneous graph support by internally representing homogeneous graphs as single-type heterogeneous graphs. We consider three datasets in our experiments, one homogeneous and two heterogeneous.

4.1.1 Facebook Ego Dataset

The homogeneous Facebook Ego dataset was collected in 2012 from survey participants using the Facebook app [5]. It contains $4,167$ individuals and $89,162$ friendship edges in 10 connected components. Nodes are equipped with $1,406$ binary features denoting if a particular property (ex. ”Went to Yale” or ”Democrat”) is true. The Facebook dataset is converted to a heterogeneous dataset with one node type (”person”) and one edge type (”person-to-person”) to integrate seamlessly alongside the remaining heterogeneous datasets.

4.1.2 IMDB Dataset

The heterogeneous IMDB dataset is a subset of movies, actors, and directors scraped from the online IMDB database about movies and television programs [6]. This dataset contains $4,278$ movies, $2,081$ directors, and $5,257$ actors. Each object has a $3,066$-dimension bag-of-words feature representation. Each movie has an edge to exactly one director who directed the movie. There are $11,546$ edges connecting movies and actors. Compared to the Facebook dataset, there are almost 10 times fewer edges with more nodes. This sparser graph helps explore a distinct setting for link explanation.

4.1.3 LastFM Dataset

The heterogeneous LastFM dataset is a small subset of the users and artists social network on the LastFM music website [7]. The datasett consists of $1,892$ users, $17,632$ artists, $25,434$ friendship edges between users and $58,486$ follow edges between users and artists. Unlike the IMDB dataset, the LastFM dataset has multiple types of edges and is considerably more dense. As a result, when predicting and explaining ”user-to-artist” edges, the 2-hop neighborhood is considerably larger (incorporates user→user→user, user→user→artist, user→artist→user, artist→user→artist, and artist→user→user paths). Each unique dataset helps assess the applicability of the multiple link explanation methods evaluated.

Note: We also considered the DBLP dataset, a citation network scraped from the DBLP computer science bibliography website [23]. Ultimately, DBLP was omitted due to the prediction model primarily learning from the node features as opposed to the surrounding graph structure. Future work is needed to address scenarios such as this.

4.2 Link Prediction

Following the methodology described in Section 3.3, we train three separate link prediction models for the Facebook, IMDB, and LastFM datasets. The Facebook link prediction model uses a two-layer GCN (128, 32 hidden nodes) to learn the node embeddings, followed by two-dense layers (32, 1 hidden nodes) to learn a similarity score between two node embeddings. A final sigmoid activation constrains the link prediction to $[0, 1]$. The overall model has 186K parameters and is trained with an Adam optimizer with 1×10^{-3} learning rate and 2×10^{-3} weight decay for 200 epochs. It achieves an 81% accuracy and 89% ROC AUC on the balanced test set of unseen target edges. We prioritize improvements to the link explanation methods, so further tuning and optimizations could significantly improve these results.

The IMDB and LastFM link prediction models both use a two-layer GraphSAGE GNN (128, 32 hidden nodes) to learn the node embeddings, and dot-product followed by sigmoid activation to compute the link probability. Due to the differences in the input dimensionality, the IMDB model has 3.2M parameters while the LastFM model has only 50K parameters. Further work can be done to add dense layers before message passing. Both models are trained with an Adam optimizer with 1×10^{-3} learning rate and 2×10^{-5} weight decay for 50 epochs. The IMDB model reaches 71% accuracy and 77% ROC AUC while the LastFM model reaches 70% accuracy and 74% ROC AUC. While the accuracy is somewhat low, explanations are only computed for link predictions where the initial prediction without neighbors is $< 25\%$, the final prediction with all neighbors is $> 75\%$, and there are between 5 and 100 total neighbors. This helps ensure high data quality in the link explanation step.

4.3 Link Explanation

After applying our link prediction models on a subset of the three datasets (due to time constraint) and removing invalid target edges, we are left with 203 link predictions examples from the Facebook dataset, 385 from the IMDB dataset, and 218 from the LastFM dataset.

4.3.1 Baselines

We begin by applying the four methods described in Section 3.4 - the random baseline, embedding baseline, GNNExplainer, and SubgraphX. For each explanation method, we iterate over each link prediction and compute explanations of varying sparsity. The table below shows the mean characterization score for the 1-node, 5-node, and 10-node explanation across all three datasets and all four methods.

	Facebook (n=203)			IMDB (n=385)			LastFM (n=218)		
	K=1	5	10	1	5	10	1	5	10
Random	0.045	0.12	0.20	0.098	0.28	0.33	0.032	0.074	0.11
Embedding	**0.060**	**0.15**	**0.25**	**0.22**	**0.49**	**0.56**	**0.29**	**0.53**	0.68
GNNExplainer	0.048	0.13	0.22	0.075	0.32	0.35	0.005	0.032	0.10
SubgraphX	0.051	0.13	0.23	0.16	0.46	0.54	0.25	0.52	**0.70**

Table 1: Mean characterization score for Top-K explanations. Best methods bolded.

Across all three datasets, the embedding baseline performs best or second-best. Results are closest on the Facebook dataset while GNNExplainer performs very poorly on IMDB and LastFM. GNNExplainer even performs below random on the LastFM dataset. SubgraphX does better than GNNExplainer but does not outperform the embedding baseline.

Figures from each dataset are included on the next page. The left panel is a continuous plot of characterization score as a function of sparsity percentage. The right panel is a visual representation of the table with additional error bars. Given these results, we investigate if small changes can improve the performance of the GNNExplainer and SubgraphX models.

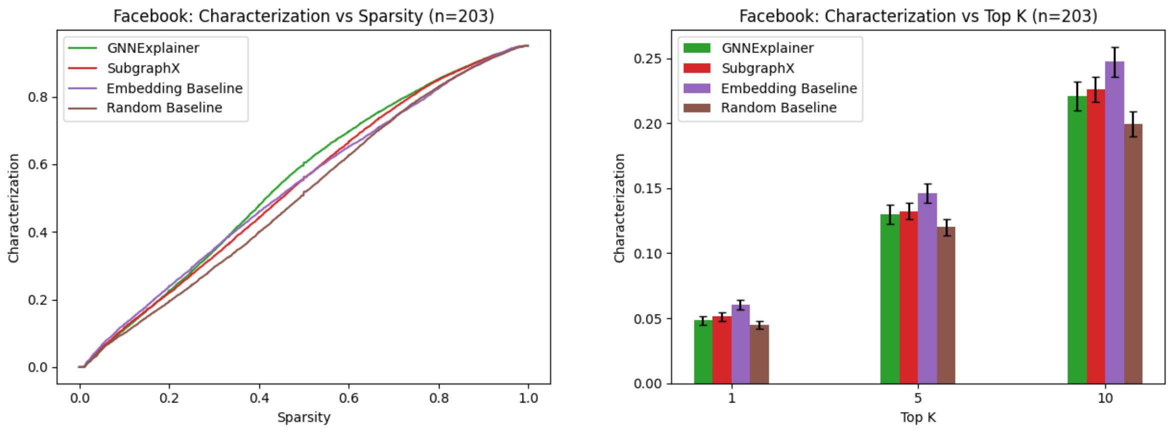


Figure 6: Characterization score by continuous and discrete sparsity on the Facebook dataset

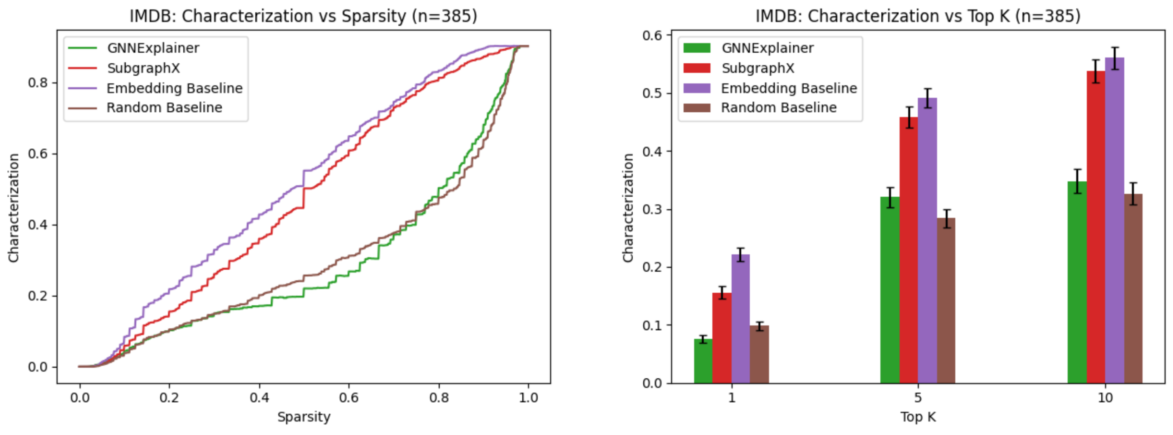


Figure 7: Characterization score by continuous and discrete sparsity on the IMDB dataset

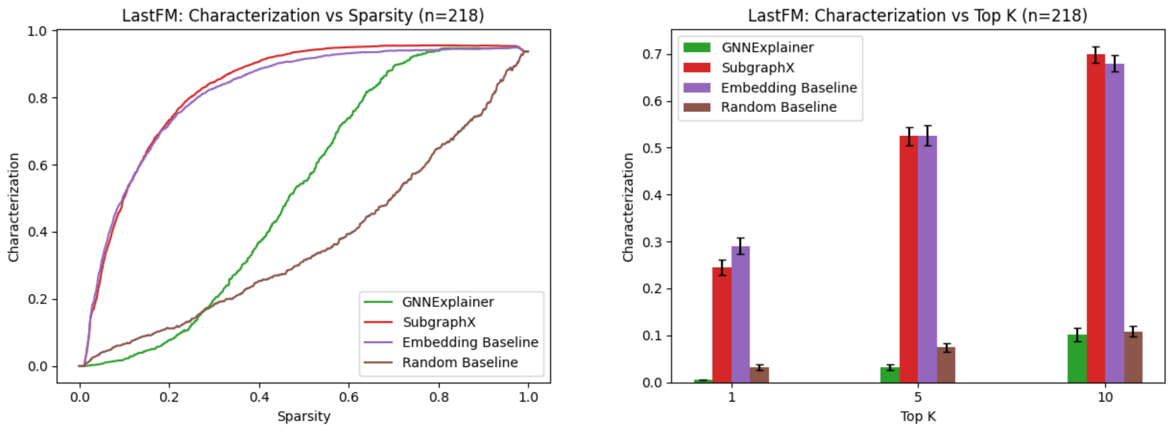


Figure 8: Characterization score by continuous and discrete sparsity on the LastFM dataset

4.3.2 GNNExplainer Variants

As discussed in Section 3.4.3, GNNExplainer uses a composite loss function to simultaneously optimize for high mutual information, low explanation size, and high mask cross entropy. We propose two changes to this implementation.

1. The loss function includes an $\alpha \sum_{e_i \in M} e_i$ term to encourage sparse explanations. This term is dependent on the absolute size of the explanation and can have varying relative importance in small, medium, and large neighborhoods. We therefore replace sum aggregation with mean aggregation to stay invariant to neighborhood size. Even with this change, we find the model sometimes suggests an empty or full subgraph. We propose optimizing for a 50% sparsity explanation, using mean squared error.

$$\alpha \cdot \sum_{e_i \in M} e_i \implies \alpha \cdot \left(\left(\frac{1}{|M|} \sum_{e_i \in M} e_i \right) - 0.5 \right)^2 \tag{10}$$

2. In the original GNNExplainer paper, the synthetic dataset used had a single explanation of fixed size, prompting the loss function to encourage nearly discrete masks. However, in our experiments, we seek a complete ordering of the neighborhood importance. Encouraging discrete masks actually worsens the output by making similar nodes harder to distinguish. In our modified Edge GNNExplainer, we subtract the cross entropy term to encourage continuous masks.

$$\beta \cdot \text{CrossEntropy}(M) \implies -\beta \cdot \text{CrossEntropy}(M) \tag{11}$$

We label the combination of these changes the Edge GNNExplainer method, and compare it against the Random and GNNExplainer methods. Table 2 summarizes the results. Overall, the Edge GNNExplainer significantly outperforms the previous two methods, achieving 51% better characterization score on the Facebook dataset, a 3-fold improvement on the IMDB dataset, and a 20-fold improvement on the LastFM dataset.

	Facebook (n=203)			IMDB (n=385)			LastFM (n=218)		
	K=1	5	10	1	5	10	1	5	10
Random	0.045	0.12	0.20	0.098	0.28	0.33	0.032	0.074	0.11
GNNExplainer	0.048	0.13	0.22	0.075	0.32	0.35	0.005	0.032	0.10
Edge GNNExplainer	**0.076**	**0.19**	**0.33**	**0.35**	**0.65**	**0.73**	**0.20**	**0.54**	**0.74**

Table 2: Mean characterization score for Top-K explanations. Best methods bolded.

The following figures reinforce these results. Not only does Edge GNNExplainer perform better for small explanations, but across the entire sparsity domain. On the IMDB dataset, a one-node explanation from Edge GNNExplainer is approximately as good as a 10-node explanation from the previous methods. On the LastFM dataset, Edge GNNExplainer is able to reach essentially complete characterization with a 40% neighborhood sparsity.

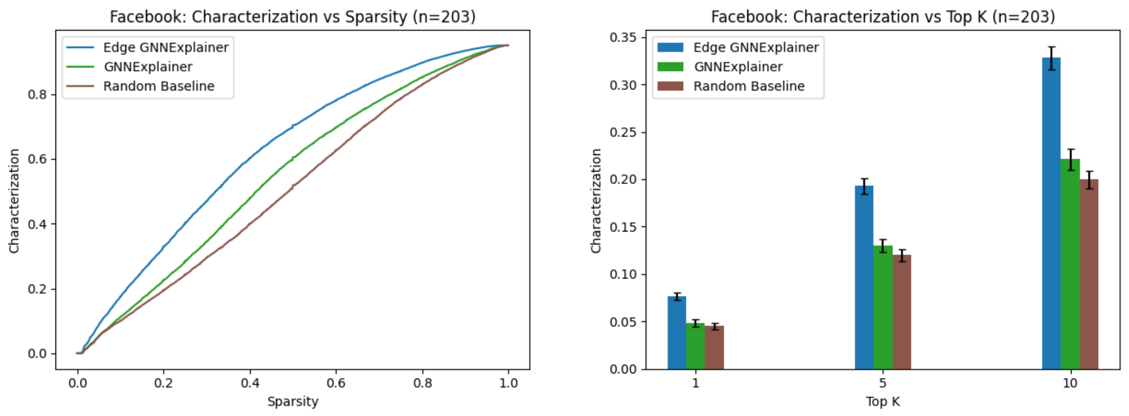


Figure 9: Characterization score by continuous and discrete sparsity on the FB dataset

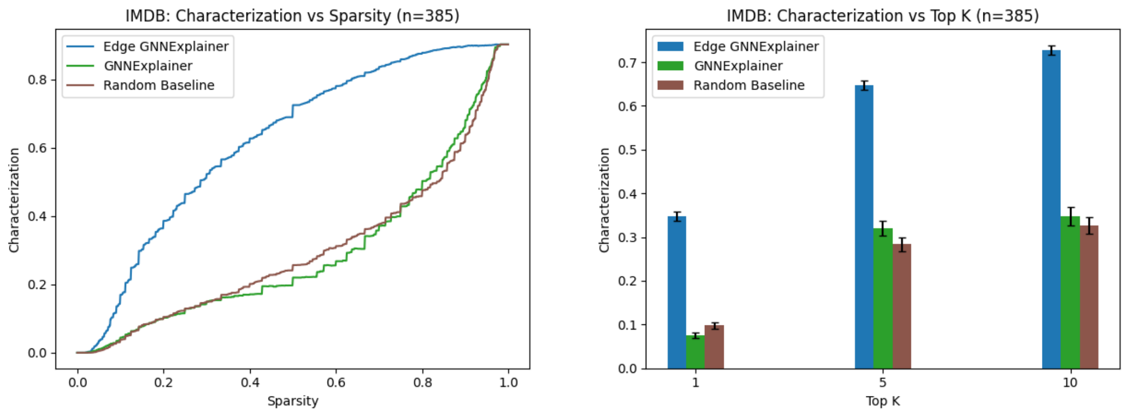
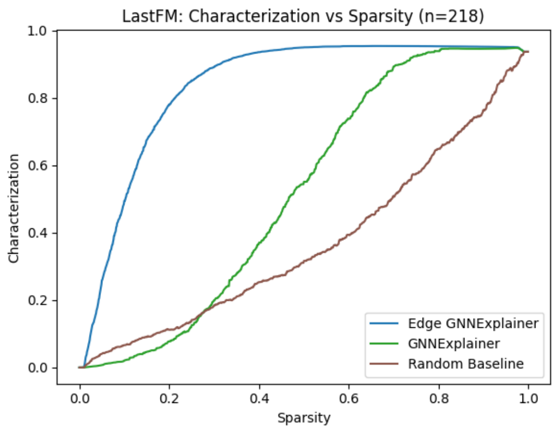


Figure 10: Characterization score by continuous and discrete sparsity on the IMDB dataset



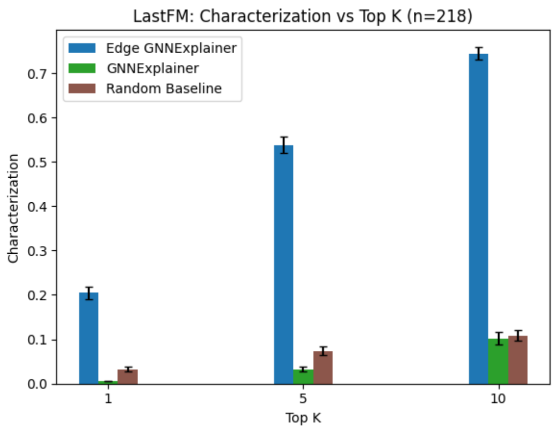


Figure 11: Characterization score by continuous and discrete sparsity on the LastFM dataset

4.3.3 SubgraphX Variants

We propose one key change to the original SubgraphX algorithm to improve performance on this link explanation task. As a consequence of the one-hop neighborhood explanation format, every candidate explanation node is adjacent to at least one of the target edge endpoints. Then, instead of masking a node by setting its features to 0, we can mask the node by removing its connection to the target edge. This better isolates the impact of the candidate explanation node on the link prediction and empirically leads to improved metrics.

	Facebook (n=203)			IMDB (n=385)			LastFM (n=218)		
	K=1	5	10	1	5	10	1	5	10
Random	0.045	0.12	0.20	0.098	0.28	0.33	0.032	0.074	0.11
SubgraphX	0.051	0.13	0.23	0.16	0.46	0.54	**0.25**	**0.52**	**0.70**
Edge SubgraphX	**0.066**	**0.17**	**0.30**	**0.33**	**0.59**	**0.68**	0.21	0.49	0.69

Table 3: Mean characterization score for Top-K explanations. Best methods bolded.

As demonstrated in Table 3, Edge SubgraphX outperforms SubgraphX by 30% on the Facebook dataset and by 53% on the IMDB dataset. While Edge SubgraphX actually performs slightly worse (8%) than SubgraphX on the LastFM dataset, this is the only result of the three that is not statistically significant (see figures on next page).

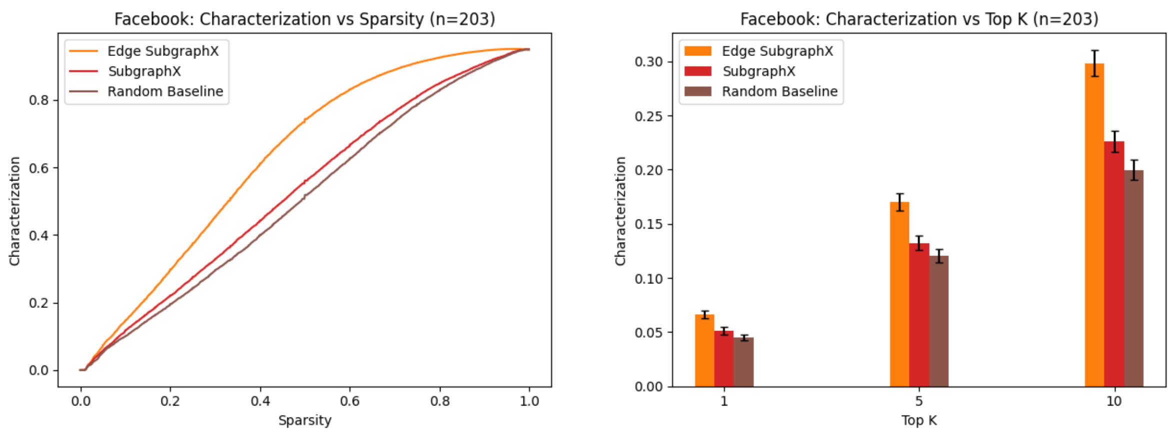


Figure 12: Characterization score by continuous and discrete sparsity on the FB dataset

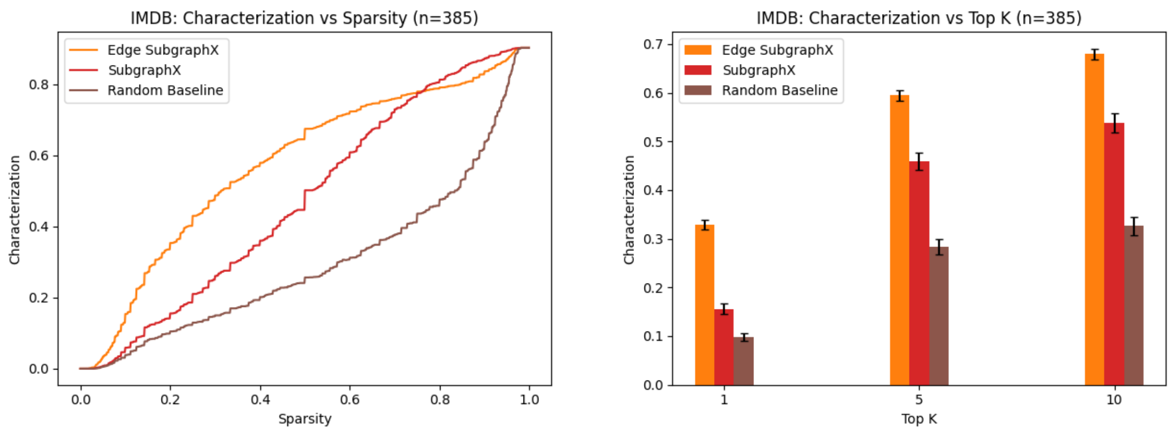


Figure 13: Characterization score by continuous and discrete sparsity on the IMDB dataset

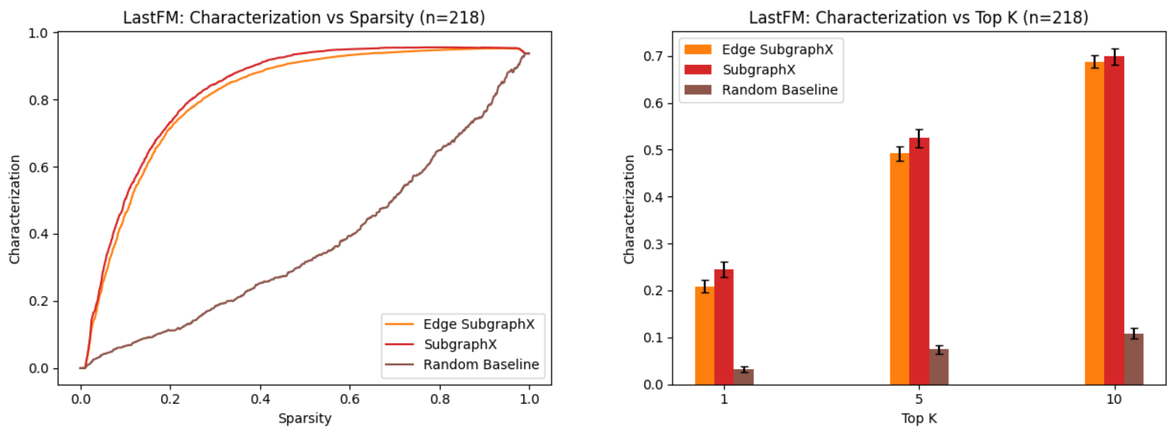


Figure 14: Characterization score by continuous and discrete sparsity on the LastFM dataset

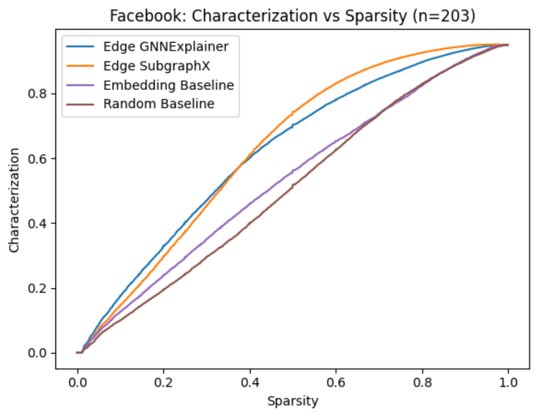
4.4 Analysis

Our final results are presented below:

	Facebook (n=203)			IMDB (n=385)			LastFM (n=218)		
	K=1	5	10	1	5	10	1	5	10
Random	0.045	0.12	0.20	0.098	0.28	0.33	0.032	0.074	0.11
Embedding	0.060	0.15	0.25	0.22	0.49	0.56	**0.29**	0.53	0.68
Edge SubgraphX	0.066	0.17	0.30	0.33	0.59	0.68	0.21	0.49	0.69
Edge GNNExplainer	**0.076**	**0.19**	**0.33**	**0.35**	**0.65**	**0.73**	0.20	**0.54**	**0.74**

Table 4: Mean characterization score for Top-K explanations. Best methods bolded.

While before the embedding baseline outperformed GNNExplainer and SubgraphX, we now see Edge SubgraphX and Edge GNNExplainer generally outperforming the embedding baseline. Edge SubgraphX outperforms the embedding baseline by 11% on average across the three datasets, while Edge GNNExplainer outperforms the embedding baseline by 21% and Edge SubgraphX by 9%. The embedding baseline outperforms both Edge SubgraphX and Edge GNNExplainer on the 1-node explanation for the LastFM dataset. Otherwise, Edge GNNExplainer is the best model for all datasets and explanation sizes.



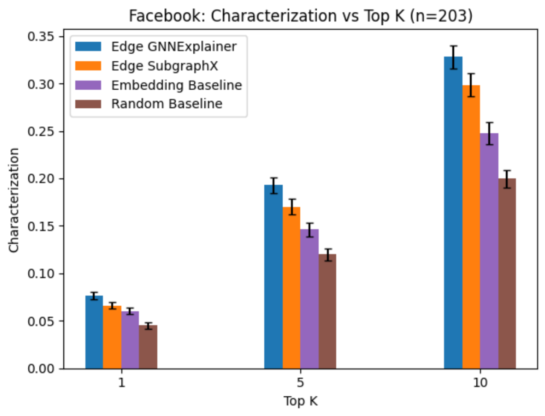


Figure 15: Characterization score by continuous and discrete sparsity on the FB dataset

From Figure 15, we see Edge GNNExplainer and Edge SubgraphX clearly separate from the two baselines, with Edge GNNExplainer performing better for smaller explanations, and Edge SubgraphX performing better for larger explanations.

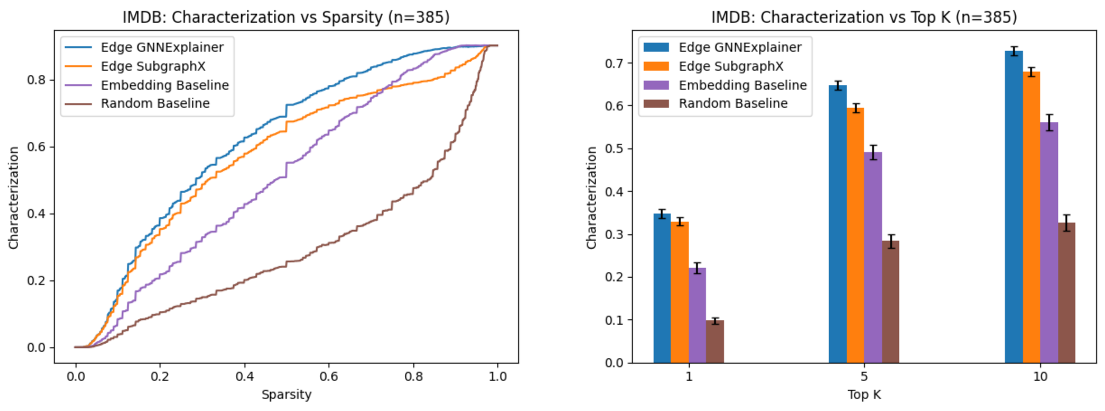


Figure 16: Characterization score by continuous and discrete sparsity on the IMDB dataset

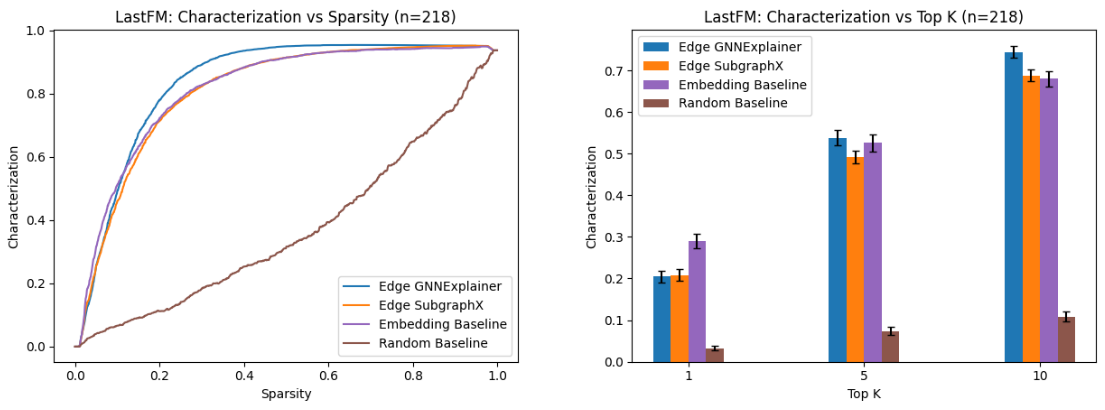


Figure 17: Characterization score by continuous and discrete sparsity on the LastFM dataset

Figure 17 shows some distinct results compared to the Facebook and IMDB dataset results. From the left panel, the Edge SubgraphX and embedding baseline perform almost identically across the sparsity domain. The embedding baseline performs slightly better at first, likely due to noise in Edge SubgraphX when estimating the Shapley value of dozens of candidate explanation nodes. Recall the LastFM dataset has the largest 1-hop and 2-hop neighborhoods surrounding the target edges, accentuating this effect. For medium explanations, Edge GNNExplainer seems to perform best, likely due to it's optimization method accounting for correlated explanations and improving upon the greedy approach employed by the embedding baseline and Edge SubgraphX. Likely due to noise in the Edge GNNExplainer output, the embedding baseline outperforms Edge GNNExplainer for the 1-node explanation.

4.5 Open-Source Contributions

In addition to implementing heterogeneous link explanation methods for this experiment, we made key contributions to the open-source PyTorch Geometric library to allow for future extension [24]. In particular, we contributed to the following pull requests:

1. GNNExplainer Migration (#5967): This PR transferred the legacy GNNExplainer implementation to the new Explanation classes.

2. GNNExplainer Edge Level Task (#6056): This PR implemented edge target explanations using the GNNExplainer class from #5967 and created corresponding test cases.

3. Heterogeneous Explanations (#6091): This PR extended the Explanation framework to support heterogeneous explanations by defining and testing a new HeteroExplanation class and extending several helper functions.

5 Conclusion

In this project, we implemented the first application of GNN explanation methods to heterogeneous link explanation. We proposed a new link explanation format where only a subset of immediate neighbors of the target edge are selected. We tested four explanation methods on the Facebook, IMDB, and LastFM heterogeneous datasets. Key changes to the GNNExplainer and SubgraphX methods yielded significant improvements in explanation quality. Using characterization scores as a function of explanation sparsity, we found that our modified SubgraphX outperformed existing baselines by 11% and our modified GNNExplainer outperformed baselines by 21%. We found the embedding baseline remains a fast and viable approach for high degree graph datasets. Overall, GNNExplainer produced the strongest explanations across dataset and explanation sparsity. We made key contributions to the open-source PyTorch Geometric library and are excited by the potential for new explanation methods to build upon this, potentially leveraging heterogeneous graph meta-paths.

References

[1] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[2] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–19, 2022.

[3] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in neural information processing systems*, volume 32. Curran Associates, Inc., 2019.

[4] 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*, pages 12241–12252. PMLR, 2021.

[5] Jure Leskovec and Julian Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

[6] Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of The Web Conference 2020*, pages 2331–2341, 2020.

[7] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In *Proceedings of the fifth ACM conference on Recommender systems*, pages 387–388, 2011.

[8] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[10] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, 30, 2017.

[12] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[13] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1365–1374, 2015.

[14] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pages 2022–2032, 2019.

[15] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer, 2018.

[16] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in Neural Information Processing Systems*, 33:19620–19631, 2020.

[17] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[18] 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. *arXiv preprint arXiv:2206.09677*, 2022.

[19] 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*, pages 10772–10781, 2019.

[20] 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. *Advances in neural information processing systems*, 33:5898–5910, 2020.

[21] Xiang Wang, An Zhang, Xia Hu, Fuli Feng, Xiangnan He, Tat-Seng Chua, et al. Deconfounding to explanation evaluation in graph neural networks. *arXiv preprint arXiv:2201.08802*, 2022.

[22] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting gnns for nlp with differentiable edge masking. *arXiv preprint arXiv:2010.00577*, 2020.

[23] Ming Ji, Yizhou Sun, Marina Danilevsky, Jiawei Han, and Jing Gao. Graph regularized transductive classification on heterogeneous information networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 570–586. Springer, 2010.

[24] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.