Explaining Graph Neural Network using Link Prediction

Rashida Bharmal[1], Saswat Rabindranath Swain[1], and Nikhil Jha[1]

Universität Paderborn,
Warburger Str. 100, 33098 Paderborn, Germany
https://www.uni-paderborn.de/

Abstract. In this study, we applied a graph neural network to link prediction using the IMDB dataset[1], a heterogeneous graph. In particular, we used the GraphSAGE approach to train our model and computed the Mean Square Error to check for loss. To make our model explainable, we made use of the CAPTUM Explainer, using which were able able to find the feature importance of our edge labels. We were able to visualise our results and we came up with explaination for it. Our findings offer a deeper knowledge of the graph by shedding light on the importance of certain features for link prediction tasks. With a clearer knowledge of the impact of the graph structure on the model's decision-making process, our findings make the significance of these features for link prediction tasks more clear and hence make the model more explained.

Keywords: Heterogeneous Graph · Graph Neural Network · Link Prediction.

1 Introduction

Link prediction [5] in graphs has found numerous practical uses in a variety of fields, including social network analysis, recommender systems, and biological networks. The capacity to foresee missing or possible connections within a graph facilitates the identification of hidden linkages, the generation of pertinent recommendations, and a greater comprehension of the dynamics and structure of the network. Researchers have developed a number of graph neural network (GNN) algorithms to address this problem, utilizing the power of deep learning to identify complex connections and patterns in graph-structured data.

As a heterogeneous graph [2] is represented by the IMDB dataset, our project's main focus was on link prediction utilizing this dataset. Heterogeneous graphs, in contrast to homogeneous graphs, in which nodes and edges belong to the same entity type, capture rich semantic information and relationships. The IMDB dataset consists of interactions between movies, with nodes standing in for movies, actors, and directors, and edges for movie events. Traditional supervised learning methods are challenged by the dataset's lack of explicit labels and features.

We used the GraphSAGE approach, a well-liked GNN framework created for semi-supervised learning on large-scale graphs, to complete this challenge. By combining data from nodes' local surroundings, GraphSAGE enables us to learn node representations, enabling accurate prediction even in the lack of explicit attributes. Positive and negative node pairings, with positive pairs denoting connections that already exist and negative pairs denoting connections that don't yet exist, were used to train our model. We estimated the mean square error to compute for the losses of our training model.

We performed a thorough investigation of the graph structure and its effect on link prediction in order to improve the interpretability of our model. We concentrated specifically on making our GNN explainable. We looked at many explainer algorithms and chose the CAPTUM Explainer [3] which fit our requirement. This explainer was best suited for edge prediction. We mainly calculated the feature importance of the features between a variety of edge labels and also visualised them to get a better understanding. In the end, we also generated the explaination subgraph and complement subgraph to get a final overview of our explaination model in terms of the whole graph network.

While offering insightful information about the interpretability and decision-making process of the GraphSAGE model for link prediction tasks, our analysis and findings help to clarify the underlying dynamics and structural features of the IMDB dataset.

2 Data Analysis

A useful tool for researching link prediction tasks in a heterogeneous graph context is the IMDB dataset. IMDB is a heterogeneous graph containing three types of entities - movies, actors, and directors. The movies are divided into three classes (action, comedy, drama) according to their genre.

Here is a better overview of the dataset:
1. The dataset contains the following entities: movies (4,278 nodes), actors (5,257 nodes), and directors (2,081 nodes).
2. Our main focus was to check for edge 'movie to director' and analyse data for that aspect.
3. We deleted the unnecessary nodes and edges from the dataset to refine our dataset and reduce execution time for the model.
4. We kept an 80,10,10 split on the training, validation and test datasets.

We successfully trained a model to anticipate missing links in the graph using the GraphSAGE technique. More information on this is provided in the next section.

Overall, our analysis emphasizes how crucial it is to take into account the local aspects of the network structure when evaluating and analyzing link prediction models. The knowledge collected from this work helps to increase performance and interpretability in link prediction challenges by fostering a better un-

derstanding of the IMDB dataset and the GraphSAGE model's decision-making process.

3 GNN Training and Evaluation

The IMDB dataset has undergone preprocessing in order to be used with GNNs. The dataset is divided into training, validation, and testing sets after unwanted edges are eliminated. In order to evaluate the model's performance properly, negative edges are also produced. The foundation for the later GNN implementation was laid by these first actions.

Our GNN model's architecture is created in the following way: We have a class GNNEncoder which represents the encoder model of the GNN. It includes two SAGEConv layers, which let the model to recognize and take advantage of complicated relationships found in the graph. We have another class EdgeDecoder which acts as the decoder module of the GNN. It retrieves our edge level index and concatenates the required features. The GNN model is turned into a heterogeneous variation to support various node types. Additionally, a classifier is built into the model to estimate the likelihood of user-artist interactions by using dot-product computations between node embeddings.

The model's parameters are iteratively updated to improve its predictive power using the Adam optimizer and binary cross-entropy loss function. Multiple epochs are used during the training phase to give the model time to gradually improve. Throughout this procedure, the loss is closely watched as a gauge of how well the model is being trained.

4 Explanation of GNN

To make our explainable we used CAPTUM explainer. This is a popular approach for emphasising the relevance of input features in model predictions. It calculates the cumulative gradients of the model's output in relation to the features of the input over a predetermined baseline input.

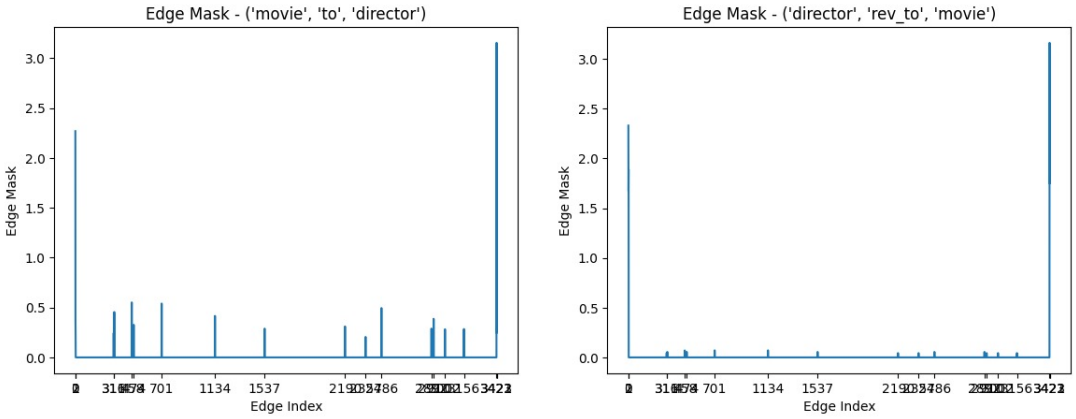
You may examine the behaviour of the model and comprehend how various features affect predictions by utilising the CAPTUM library and the Integrated Gradients algorithm. It improves the interpretability of the model's predictions by shedding light on the significance and influence of input features on the model's judgements.

We specifically chose this explainer because this fits well with our model requirement which is Edge Prediction. We setup the model for the configuration of the explainer and in the explanation we gave the edge label and edge label indexes as inputs and generated explanations based on it.

The type of explanation we generated was feature importance. It establishes which particular feature are important to the specified edge labels. An example representation is shown below.

We did this for multiple edge labels in varying sets. The set of edge labels for which we generated the feature importance are [486, 237], [0, 1, 2, 3421,

3422, 3423] and [678, 999, 82, 12]. We also visualised the edge masks which were generated while calculating the feature importance (see figure 1). We found that even though we mentioned specific edge labels for our feature importance, other edge indexes were also being added to our edge dictionary. For example while generating the feature importance for the label set [0,1,2,3421,3422,3423], we see that edge indexes other than the ones mentioned are also included in the edge dictionary. This means that the indexes not mentioned in the training also have a significance in the explanation model. We also found the same case for the other label sets.



(a) Movie to Director

(b) Director rev-to Movie

Fig. 1: Edge Masks

Looking at the graph of the feature importance we can see the influence of different directors and movies over the edge labels. For all the labels we see that the feature director has a bigger importance than the feature movie. Looking at the figure2 we can say that director comes in 9 of the top 10 mentioned feature importance for labels [486, 237]. Similarly, we see that director comes 14 of the top 20 mentioned feature importance for labels [0, 1, 2, 3421, 3422, 3423](see figure3). Same goes for edge labels [678, 999, 82, 12] where we see director coming in 13 of the top 15 (see figure4) mentioned feature importance. Hence, we can infer at least in the case of top feature importance of these particular edge labels, director holds a larger importance than movies.

Finally, we generated the explanation subgraph[4] for our model. It contains the following data as shown in figure5a. The explanation subgraph returns the induced subgraph, in which all nodes and edges with zero attribution are masked out.

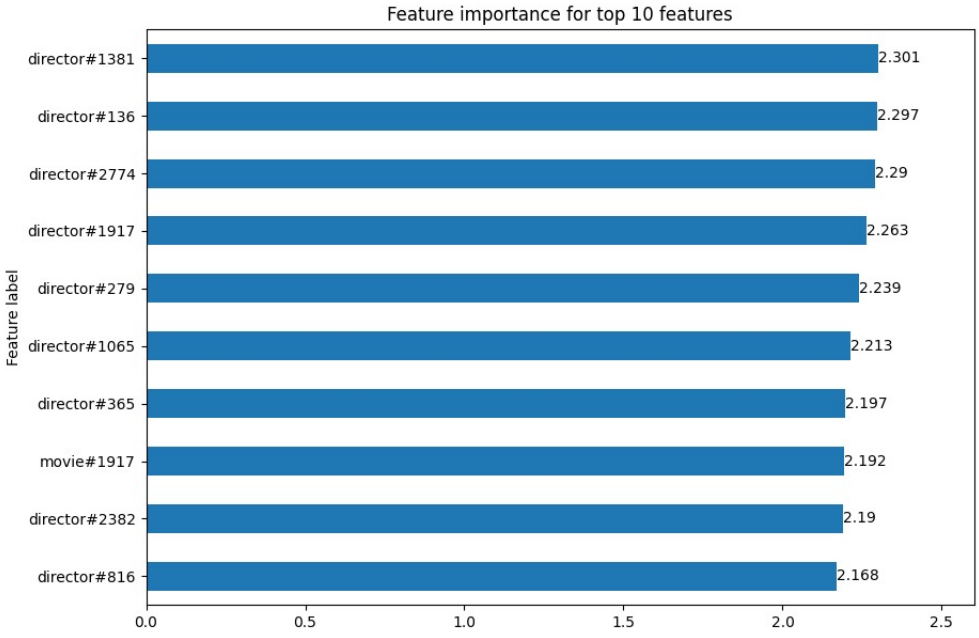


Fig. 2: Feature importance for top 10 features

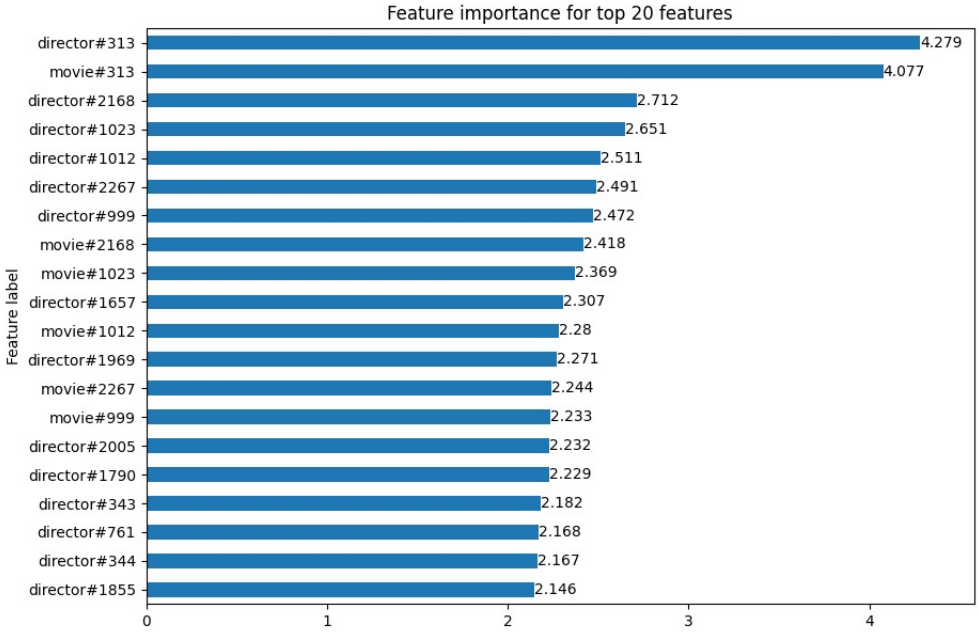


Fig. 3: Feature importance for top 20 features

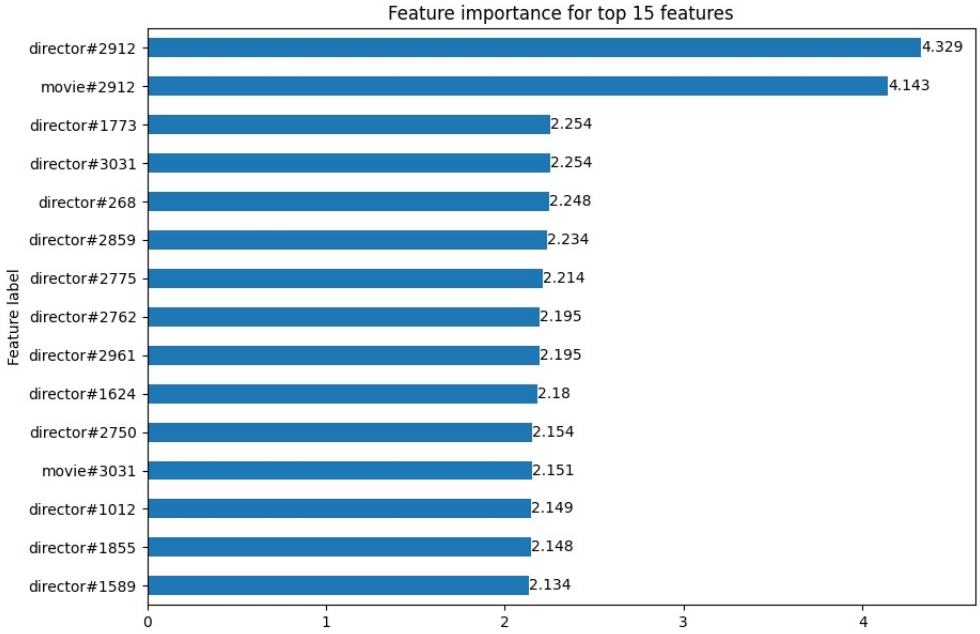


Fig. 4: Feature importance for top 15 features

By looking at the figure we can say that to generate our explanation model , 10 node of movies and 4 nodes of directors and 10 edges of each type were involved in the explanation.

Similarly, by analysing the complement subgraph [4] we see the that the opposite holds true as well. The complement subgraph returns the induced subgraph, in which all nodes and edges with any attribution are masked out. Meaning 4268 nodes of movies and 2077 nodes of director remains unused. Same holds true for 3414 edges(see figure 5b).

5 Conclusion

In this project, we selected a heterogeneous dataset to initially apply a graph neural network[6]. We accomplished this by utilizing the graphSAGE method for link prediction. After successfully training the model, our focus shifted towards making the graph neural network explainable. To achieve explainability, we used the captum explainer to to make our edge predictions explainable. We were able to successfully do so by calculating the feature importance of different sets of edge labels and visualising them. We saw which features were more important to the edges and analysed our results accordingly. Finally we checked the explanation and complement subgraph data to check for what data was used in our

```
HeteroExplanation(
  prediction=[3424],
  target=[3424],
  index=[4],
  edge_label_index=[2, 3424],
  movie={
    node_mask=[10, 3066],
    x=[10, 3066],
  },
  director={
    node_mask=[4, 3066],
    x=[4, 3066],
  },
  (movie, to, director)={
    edge_mask=[10],
    edge_index=[2, 10],
  },
  (director, rev_to, movie)={
    edge_mask=[10],
    edge_index=[2, 10],
  }
)
```

(a) Explanation Subgraph

```
HeteroExplanation(
  prediction=[3424],
  target=[3424],
  index=[4],
  edge_label_index=[2, 3424],
  movie={
    node_mask=[4268, 3066],
    x=[4268, 3066],
  },
  director={
    node_mask=[2077, 3066],
    x=[2077, 3066],
  },
  (movie, to, director)={
    edge_mask=[3414],
    edge_index=[2, 3414],
  },
  (director, rev_to, movie)={
    edge_mask=[3414],
    edge_index=[2, 3414],
  }
)
```

(b) Complement Subgraph

Fig. 5: Subgraph Details

explanation model. These findings led us to conclude that these methods are valuable parameters for establishing explainability in graph neural networks.

6 Contributions of team members

We divided the project tasks according to the expertise of individual group members. Saswat was responsible for implementing the GraphSAGE method for link prediction, ensuring accurate predictions using the IMDB dataset. Rashida meanwhile assisted Saswat and was responsible for setting up and maintaining the git repository enabling deliverables for reproducing the results. Nikhil on the other hand, prepared the overall comprehensive report, synthesizing the results.

References

1. 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*. ACM, apr 2020.
2. Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs, 2021.
3. PyTorch Geometric Contributors. PyTorch Geometric CaptumExplainer API Documentation. `https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.explain.algorithm.CaptumExplainer.html#torch_geometric.explain.algorithm.CaptumExplainer`, Accessed 2023.
4. PyTorch Geometric Contributors. PyTorch Geometric Explainability Documentation. `https://pytorch-geometric.readthedocs.io/en/latest/modules/explain.html`, Accessed 2023.
5. Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.
6. Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks, 2020.