Explainable AI - Mini Project

Harshit Purohit[1][4012382]

Universität Paderborn, Universität Paderborn, Warburger Str. 100, 33098 Paderborn

Abstract. This project focuses on enhancing the explainability of Graph Neural Networks (GNNs) for link prediction using a heterogeneous MovieLens graph. GNNs have proven effective in capturing complex relationships in graph-structured data for link prediction tasks. However, their lack of interpretability raises concerns in high-stakes domains. We address this by developing an interpretable GNN model that provides explanations for individual predicted links. The methodology involves creating a heterogeneous graph from the MovieLens dataset, training a GNN on this graph, and evaluating its link prediction performance. To achieve explainability, we employ the Captum explainer technique to gain insights into the model's prediction process for a given user–movie link. The results demonstrate the effectiveness of the proposed approach, with the GNN model achieving a validation AUC of approximately 0.95. By utilizing the Captum explainer, we provide clear explanations for the link predictions made by the GNN, thereby contributing to the interpretability of the model. This research highlights the potential for developing more transparent and trustworthy GNN models for recommender systems by explaining predictions in terms of influential nodes and relationships.

Keywords: Graph Neural Networks · Explainability · Link Prediction · Heterogeneous Graph · MovieLens · Captum Explainer

1 Introduction

Graph Neural Networks (GNNs) have emerged as powerful models for capturing complex relationships in graph-structured data, making them well-suited for link prediction tasks. In the context of movie recommendation systems, link prediction can help identify which user–movie pairs are likely to interact (e.g. a user liking or rating a movie). Despite their strong performance, a key limitation of GNNs is the lack of interpretability – it is often unclear why a GNN predicts a certain link. This opacity can undermine user trust, especially in domains where recommendations have significant consequences. Ensuring transparency and the ability to explain predictions is therefore crucial for the adoption of GNN-based recommender systems. In this project, we aim to enhance the interpretability of GNNs for link prediction on the heterogeneous MovieLens dataset. The MovieLens data contains users, movies, and rich metadata (genres, ratings, etc.), naturally forming a heterogeneous graph of users and movies connected by rating edges. We first construct a heterogeneous graph from the MovieLens

dataset and train a GNN model for link prediction (i.e. predicting user–movie links). The overall workflow is illustrated in Figure 1, which outlines the steps: building the graph from raw data, training the GNN, and then applying an explainability technique to the trained model. In particular, we employ Captum, a model interpretability library, to explain the GNN's predictions for specific links. Captum's explainer provides feature attribution scores that highlight the most influential nodes and edges that led the GNN to predict a given link.

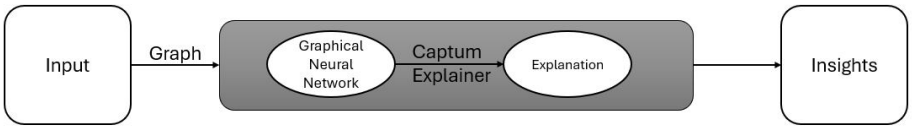

Fig. 1. Project workflow – The pipeline of the approach. A heterogeneous graph is extracted from the MovieLens dataset and used to train a GNN for link prediction. The trained GNN's predictions are then explained using Captum, which identifies important graph components (nodes/edges) contributing to a given link prediction, yielding human-interpretable insights.

By analyzing these explanations, we gain a deeper understanding of why the GNN predicts certain user–movie links, thus addressing the interpretability challenge. The ability to explain predictions can help validate the model's behavior and provide actionable insights – for example, understanding which user or movie attributes drive a recommendation.

2 Data Analysis

We first analyze the MovieLens heterogeneous graph to understand its composition and key features. The MovieLens latest-small dataset contains 610 unique users and 9742 movies with rating interactions. We represent this data as a bipartite graph with users and movies as nodes, and a directed edge user -> movie for each rating. Table 1 summarizes the graph's characteristics. There are 10,334 nodes in total (610 users + 9724 movies present in ratings) and 100,836 edges, which corresponds to the total number of ratings in the dataset. The average node degree is about 19.5, meaning the typical user has rated 20 movies (and similarly, movies have 20 ratings on average).

We performed exploratory data analysis on the movie attributes to uncover meaningful patterns. In particular, each movie is associated with one or more genres (e.g. Comedy, Drama, Thriller). We one-hot encoded the genre information into 20 distinct genre features (such as Action, Animation, Children, etc.) for each movie.

Metric	Value
Number of nodes (users + movies)	10,334
Number of edges (ratings)	100,836
Average node degree	19.52

Fig. 2. Key characteristics of the MovieLens graph dataset

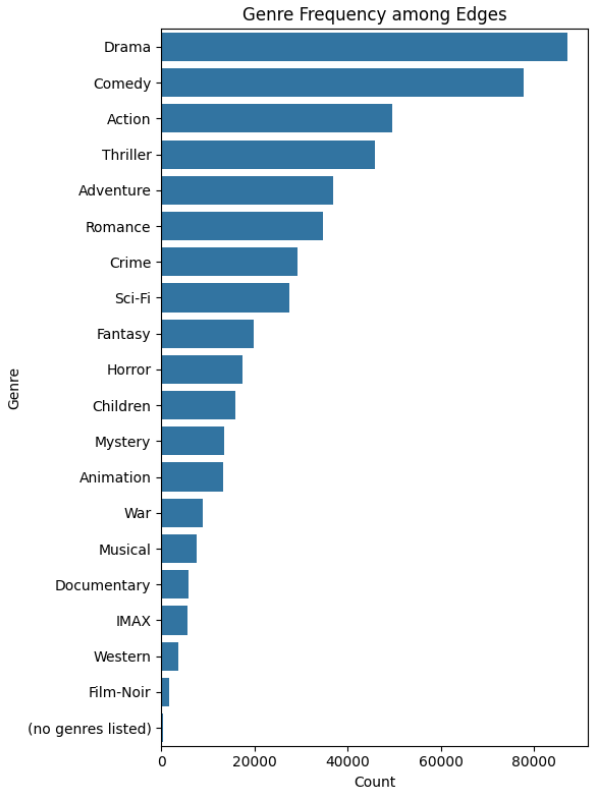

Fig. 3. Genre Distribution

Drama is the most common genre, followed by Comedy. Other prevalent genres include Thriller and Action, whereas categories like Western or Film-Noir are relatively rare. This long-tail distribution of genres indicates that a few genres are very frequent in the movie dataset, while many genres are less common.

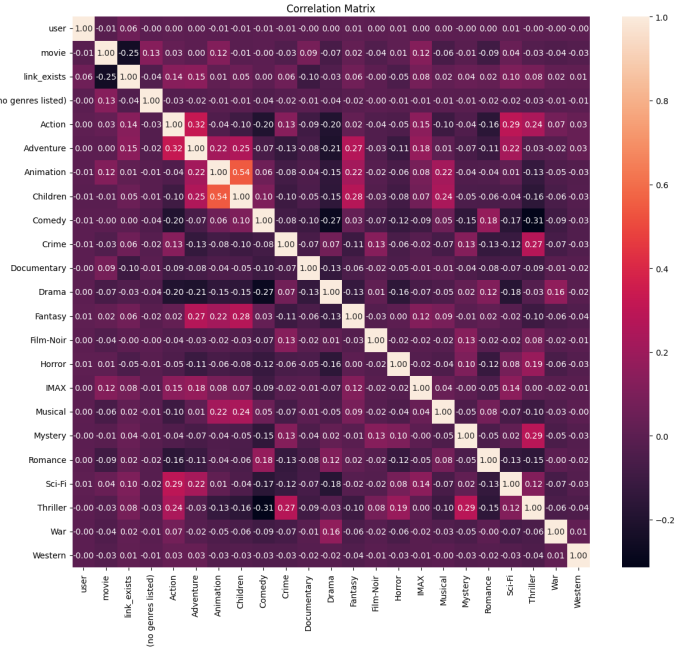

Fig. 4. Correlation Analysis

Figure 4 is a matrix that shows pairwise correlations between movie genres, IDs, and the target link (user–movie interaction). Most correlations are weak (near zero), though genres like Animation, Sci-Fi, and IMAX exhibit modest positive associations with link existence, suggesting diverse but subtle feature influences on link prediction.

3 Model Training and Evaluation

Having constructed the heterogeneous graph, we trained a GNN model for the link-prediction task. We utilized `PyTorch Geometric` to implement our GNN, taking advantage of its built-in support for heterogeneous graphs. The node features included 20-dimensional genre vectors for movies and a one-hot encoding (ID embedding) for users. We first mapped each user and movie ID to a contiguous index range to facilitate tensor operations. The resulting graph was stored

in a `HeteroData` object, containing separate node-feature matrices for users and movies and an edge-index list for user–movie interactions.

Our GNN architecture consists of a GraphSAGE-based encoder and a simple link-prediction decoder. The encoder uses `SAGEConv` layers to learn user and movie embeddings by aggregating information from their neighbors. Because the graph is heterogeneous (two node types, one relation type), we applied the `to_hetero` utility to convert the GraphSAGE encoder into a heterogeneous model that learns distinct transformations for user and movie nodes. The link-prediction decoder is a bilinear function:

$$\hat{y}_{u,m} \;=\; \sigma\big(\mathbf{h}_u^\top \mathbf{W}\, \mathbf{h}_m\big),$$

where $\mathbf{h}_u$ and $\mathbf{h}_m$ are the user and movie embeddings and σ is the sigmoid activation, yielding the probability of a link (rating) between u and m.

During training, existing user–movie pairs were treated as positive examples, and an equal number of non-interacting pairs were sampled as negatives. To scale to large graphs, we adopted neighbor sampling via `LinkNeighborLoader`: for each batch of edges, we sample a fixed number of neighbors per node to form subgraphs, thereby reducing memory footprint while preserving local structure.

We optimized our model with the binary cross-entropy loss with logits

$$\mathcal{L} = -\frac{1}{N} \sum_{(u,m)} \Big[y_{u,m}\, \log \hat{y}_{u,m} + (1 - y_{u,m})\, \log\big(1 - \hat{y}_{u,m}\big) \Big]$$

and the Adam optimizer. Training ran for several epochs until convergence of the training loss.

For evaluation, we held out a validation set of user–movie links unseen during training and computed the Area Under the ROC Curve (AUC). Our GNN achieved a validation AUC of approximately 0.95, well above the random-guess baseline of 0.5, indicating its strong ability to distinguish present versus absent links. Moreover, performance plateaued after a certain number of epochs, suggesting minimal overfitting.

In summary, the trained GNN exhibits excellent predictive performance on link prediction, providing a solid foundation for the subsequent generation of explanations.

4 Model Explanation

After training the GNN, we focus on explaining its predictions. We use Captum's graph explainer functionality to interpret the model's decision on a specific example: a particular user–movie link prediction. Our goal is to identify which parts of the graph (nodes or edges) most influenced the GNN in predicting that a given user would interact with a given movie. This kind of local explanation answers the question "Why did the model predict this link?"

We selected the test pair

$$(u^*, v^*) \;=\; (User480,\; Movie1220),$$

which the GNN predicted as likely to form a link. Using `CaptumExplainer` from PyTorch Geometric's explainability toolkit, we computed attribution scores for every feature in the computation subgraph. Here, "features" comprise the nodes and edges in the 1-hop neighborhood of the target link. Intuitively, Captum assesses how toggling each component on or off changes the model's raw output for (u^*, v^*). The result is an importance score for every neighbor node and connecting edge.

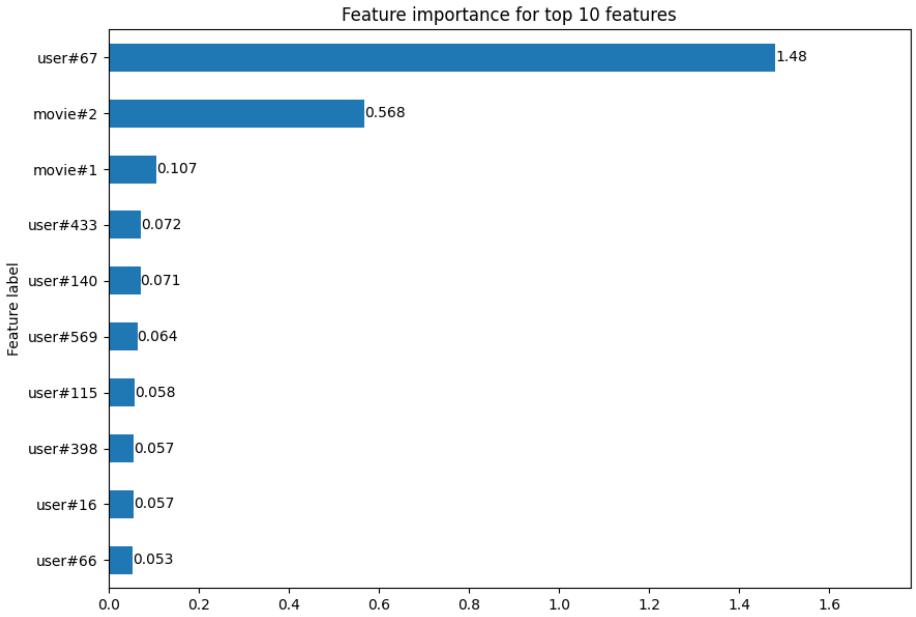

Fig. 5. Feature Importance - top ten

Feature Importance. Captum returns a ranked list of graph components by attribution. Figure 5 shows the top-10 features. We found:

- **User #67** had the highest importance, indicating that this neighbor user strongly influenced the prediction for User 480 → Movie 1220. This suggests similar tastes or overlapping ratings.
- **Movie #2** was second, implying a content-based signal: Movie 2 connects both to User 480 and to Movie 1220's neighborhood (e.g., shared genres or audiences).
- Other top features included several user and movie nodes forming the local context around User 480 and Movie 1220.

Such attributions reveal that the model leverages both social proof (similar users) and content signals (similar movies).

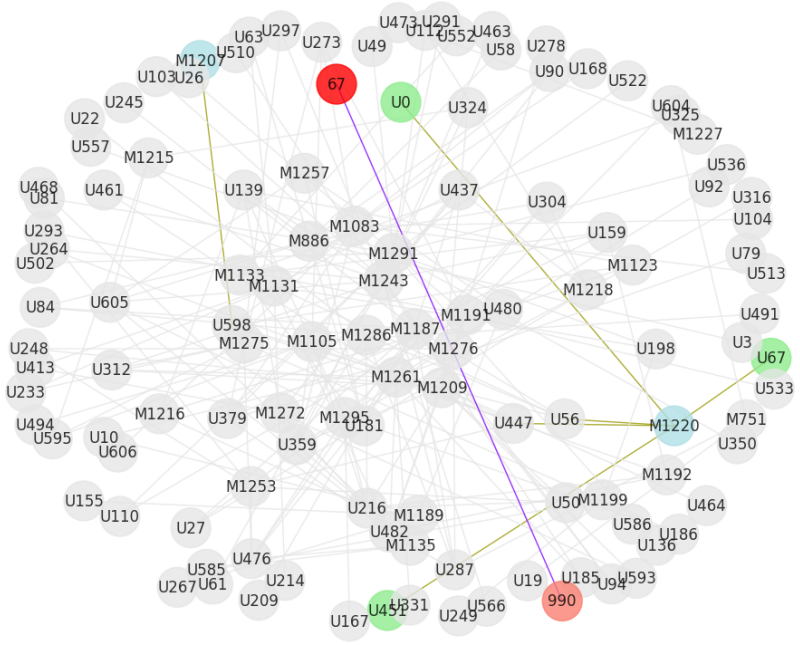

Fig. 6. Induced Subgraph Visualization

Induced Subgraph Visualization. The figure 5 shows the induced subgraph around the target user–movie link after masking low-importance connections. Only the most influential nodes (green users, turquoise movies) and edges remain, illustrating how specific neighbors bridge the prediction between the red-highlighted user 67 and the orange-highlighted movie 990.

A host of other Explanatory visualizations have been made in the code section.

5 Conclusion

In conclusion, we developed an explainable GNN model for the MovieLens movie recommendation graph, achieving high link prediction accuracy while providing human-interpretable explanations for its predictions. The GNN effectively learned user preference patterns (reaching a validation AUC of 0.95), and by leveraging the Captum explainer, we could unravel the model's decision-making process. The explainer highlighted which users and movies in the graph were most influential for a given recommendation, and we visualized these relationships through induced subgraph plots. The ability to explain why a user is recommended a movie is a significant step toward transparent and accountable AI in recommender systems. Our results show that the GNN's predictions can be explained in terms of intuitive factors like similar users or related movies, which enhances trust in the model's outputs. This work demonstrates the importance of combining strong predictive performance with explainability in graph-based models. For future research, one could explore explaining predictions at a broader level (e.g., global feature importance across many predictions) or applying more sophisticated explanation techniques (such as GNNExplainer or attention mechanisms) to the heterogeneous graph scenario. Nonetheless, our integrated GNN + Captum framework provides a valuable template for building recommender systems that are not only accurate but also interpretable, helping users and domain experts to understand and trust the recommendations generated.

References

1. Hamilton, W.L., et al. GraphSAGE: Inductive Representation Learning on Large Graphs. NeurIPS (2017).
2. Ying, Z., et al. GNNExplainer: Generating Explanations for Graph Neural Networks. NeurIPS (2019).
3. Chen, M., et al. Link Prediction Based on Graph Neural Networks. NeurIPS (2018).
4. Ruan, C., et al. Explainability in Graph Neural Networks: A Taxonomic Survey. arXiv (2021)