Enhancing Link Prediction in Real-World Networks Using Graph Coloring Techniques

LEEN MASA'DEH
dept. of Computer Science
JUST University
Irbid, Jordan
lhmasadeh24@cit.just.edu.jo

Amani Krieshan
dept. of Computer Science
JUST University
Irbid, Jordan
aakrieshan23@cit.just.edu.jo

Dr. Dana ElRushaidat
dept. of Computer Science
JUST University
Irbid, Jordan
dmelrushaidat@just.edu.jo

I. Abstract

Link prediction is essential for understanding relationships in real-world networks, such as social and biological systems. However, current methods like Graph Neural Networks (GNNs) face challenges in handling networks with diverse and frequently changing connections. This research introduces a new approach that combines graph coloring techniques with GNNs to improve link prediction accuracy.

The study uses the Musae GitHub Social Network dataset, where nodes represent developers, and edges represent their follower relationships. Graph coloring assigns unique labels to neighboring nodes, helping the model better handle the complexity of the network. A hybrid model integrates these graph coloring labels with GNNs, achieving an accuracy of 0.8496.

This method will be evaluated against baseline models using metrics such as precision, recall, and F1-score, along with its ability to scale in dynamic networks. The goal is to enhance the classification of developers, gain deeper collaboration insights, and advance fields like machine learning and social network analysis.

II. Introduction

In real-world complex networks, like social networks, recommendation systems, and biological systems, link prediction is a crucial task. The complexities of dynamic, large-scale, and heterogeneous networks are frequently too much for traditional link prediction techniques like matrix factorisation or similarity-based approaches to handle. The dynamic interactions and intricate topologies present in these networks are usually not captured by current techniques 0 [1].

By learning node embeddings that represent the underlying structure of networks, Graph Neural Networks (GNNs) have demonstrated potential in enhancing link prediction in order to address these difficulties. Nevertheless, GNNs have drawbacks when used to solve certain combinatorial problems, like graph colouring, which involves giving neighbouring nodes in a graph different "colours." In the context of graph colouring problems, GNNs frequently fall short of classic heuristics such as greedy algorithms, despite their successful application to other tasks [2].

Furthermore, heterophilic networks—where linked nodes should be given distinct colours—are difficult for current approaches to handle. Because GNNs are usually built to function in homophilic networks—where comparable nodes are more likely to be connected—this problem occurs. The efficiency of GNNs in link prediction tasks is hampered in heterophilic settings by the difficulty of guaranteeing unique node representations (or colours) [**?**]).

Link prediction models' scalability, accuracy, and flexibility could be improved by combining graph colouring approaches with GNNs, particularly for dynamic networks that undergo changes over time. This method can enhance the prediction of future links by more properly depicting the relationships between entities in growing networks by using graph colouring to avoid conflicts between neighbouring nodes [3].

In summary, there is an urgent need to close the gap between GNN-based architectures and graph colouring techniques in link prediction, particularly in dynamic, diverse, and real-world networks. In order to increase link prediction accuracy and address combinatorial optimisation issues that emerge in intricate network structures, this study will investigate how these two approaches function in concert.

This document explores Link Prediction in Real-World Networks Using Graph Coloring Techniques based on feature extraction using GNN,XGboost,RF. In Section 3, there is a literature review and related work to identify the gap. Section 4 details the methodology, including data collection, cleaning, and analysis. Section 5 presents exper- imental results of our framework, interprets the findings, and discusses implications. Section 6 summarizes conclusions and suggests applications. This structured approach ensures clarity and coherence in presenting the study's methodology, findings, and implications.

III. Literature Review

Graph Neural Networks (GNNs) have proven to be successful in link prediction tasks in recent studies. By learning node embeddings that represent the network's structure, GNNs increase the precision of linked node prediction. This strategy frequently performs better than conventional techniques in a variety of circumstances [4]. The two most popular varieties of GNNs are Graph Attention Networks (GATs) and Graph Convolutional Networks (GCNs). According to [5], these topologies improve link prediction by taking into account both the characteristics of nodes and the connections among them. In heterophilic networks, however, when the nodes that are connected have diverse characteristics rather than being identical, GNNs perform poorly. In these situations, traditional approaches perform better since they frequently presume that linked nodes are identical (homophily).One solution proposed in recent studies is to use graph coloring techniques. These techniques ensure that adjacent nodes have distinct representations, which helps the model make better predictions in such scenarios [6]. Another important gap in current research is that most GNN models are designed for static networks. These models don't adapt well to dynamic networks, where nodes and edges change over time. Many real-world networks, like social media or biological networks, are dynamic, and there is little research on how to incorporate dynamic link prediction. Using graph coloring in dynamic networks could help by adjusting node representations as the network evolves, improving the model's ability to predict links in changing networks.

This research will combine GNNs with graph coloring to improve link prediction in heterophilic and dynamic networks. The proposed approach will be tested on a dynamic network (GitHub social network) to show how it can adapt to changes over time and handle networks with diverse node features.

IV. Methodology

Our approach integrates advanced deep learning models, such as Graph Neural Networks (GNNs), with traditional machine learning (ML) techniques, including Random Forest (RF) and XGBoost, to enhance link prediction in real-world networks. We utilize the "Musae GitHub Social Network" dataset, which represents connections between GitHub developers.

The process begins with cleaning and processing the dataset, focusing on its edges (representing follower relationships), nodes (containing developer details like location and repositories starred), and the target variable (indicating whether a developer is a web developer or a machine learning developer).

GNNs are leveraged to learn patterns from the graph, generating embeddings that represent the network's structure and features. These embeddings are then used as input for RF and XGBoost models, which identify relationships in the data. By combining these models, we harness the strengths of both deep learning and traditional ML techniques.

A. Pipeline

We utilized the "Musae GitHub Social Network" dataset, where nodes represent GitHub developers and edges represent follower relationships. The dataset was divided into training (60%),testing (20%), and validation (20%) subsets. The training data was further split for optimization and evaluation of the models.

To ensure optimal performance, the data was carefully processed to balance the target variable.

GNNs were employed to create embeddings that captured both the structural and feature-based information of the graph. These embeddings were subsequently used as input for RF and XGBoost models to predict links between nodes.

B. Dataset

For this project, we will use the "Musae GitHub Social Network" [7] dataset available on Kaggle, created by László Rozemberczki,This dataset represents a large social network of GitHub developers, collected in June 2019 via the public GitHub API. The nodes in the network are GitHub developers who have starred at least 10 repositories, and the edges represent follower relationships between these developers. The dataset provides detailed vertex features based on attributes like the user's location, repositories starred, employer, and email address.

1) Data Description

IV-B1.1 Edge List :

Represents the follower relationships between GitHub developers.

IV-B1.2 Node Features :

Includes information like location, repositories starred, employer, and email address of the developers.

IV-B1.3 Target Variable :

A binary classification indicating whether the developer is primarily a web developer or a machine learning developer, derived from their job title.

2) Data preprocessing

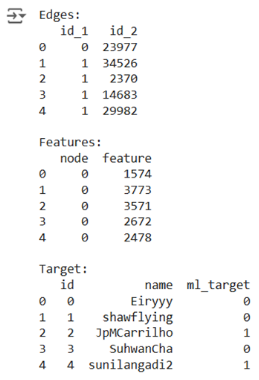
First, we begin with the preprocessing stage, which involves importing the dataset from Kaggle. For the edges (id1, id2), we rename the columns to source and target to better represent the connections in the graph. For the features file, we utilize the node number and corresponding feature number to organize the data.

In the target file of the MUSAE-Github Social Network dataset, the values 0 and 1 represent binary classification labels for the nodes (GitHub users). Their meanings are as follows:

0: Indicates that the node does not belong to a specific class or target group (e.g., not part of a particular community or does not use a certain programming language). 1: Indicates that the node belongs to the specified class or target group (e.g., part of a particular community or uses a certain programming language).

Initially, the training dataset is partitioned into four segments, and feature extraction is performed using advanced graph-based deep learning models such as Graph Neural Networks (GNNs). These models are trained on the "Musae GitHub Social Network" dataset to capture intricate patterns from the graph structure and node attributes.

The extracted embeddings are then used as input for traditional machine learning classifiers, including Random Forest (RF) and XGBoost, each trained to predict the likelihood of



```
Edges:
   id_1   id_2
0     0  23977
1     1  34526
2     1   2370
3     1  14683
4     1  29982

Features:
   node  feature
0     0     1574
1     0     3773
2     0     3571
3     0     2672
4     0     2478

Target:
   id          name  ml_target
0   0        Eiryyy          0
1   1    shawflying          0
2   2    JpMCarrilho         1
3   3      SuhwanCha         0
4   4    sunilangadi2        1
```

Fig. 1. Details of the dataset before rename column

connections between nodes. To ensure robustness and improve accuracy, a voting mechanism aggregates the predictions from all models, with the final outcome determined by majority vote.

3) Models

IV-B3.1 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate directly on graph-structured data. They excel at learning node, edge, and graph-level representations by iteratively aggregating and propagating information from neighboring nodes. GNNs have been widely used in tasks such as link prediction, node classification, and graph classification, particularly in domains like social networks, molecular chemistry, and recommendation systems [8].

IV-B3.2 Random Forest (RF)

Random Forest is an ensemble learning method that builds multiple decision trees during training and combines their outputs to improve prediction accuracy and reduce overfitting. Each tree is trained on a random subset of the data, and the final prediction is determined by averaging (regression) or voting (classification). RF is robust, easy to use, and effective for both small and large datasets. [9]

IV-B3.3 Extreme Gradient Boosting(XGBoost)

XGBoost is an optimized implementation of the gradient boosting algorithm, designed for speed and performance. It builds decision trees sequentially, where each tree corrects the errors of the previous one. XGBoost incorporates features like regularization, handling missing values, and parallel processing, making it a popular choice in machine learning competitions [10].

finally using NetworkX by integrating edge and node feature data, enabling the visualization and analysis of the graph's structure. Node features are assigned specific values from the dataset to enhance the graph representation. To focus on a manageable portion of the graph, a subset of 200 random nodes is selected, forming a subgraph for detailed exploration. A greedy coloring algorithm, employing the "largest_first" strategy, is applied to this subgraph to assign distinct colors to nodes based on their connectivity. Predefined colors, such as red, green, blue, and yellow, represent different groups, and each node is visually distinguished according to its assigned group. A spring layout is utilized to arrange the nodes in a visually intuitive manner. This visualization effectively highlights node groupings and connectivity, providing a clear representation of the graph's underlying structure and relationships.

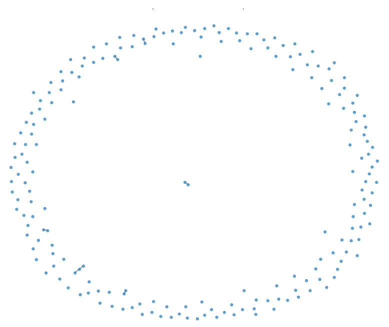


Fig. 2. The image shows a subgraph created by randomly selecting 200 nodes . The graph is visualized with nodes of size 20, alpha transparency of 0.7, and a figure size of 10x8 inches. The title "Sample Visualization of the Graph" is displayed, showcasing a portion of the graph's structure for analysis.

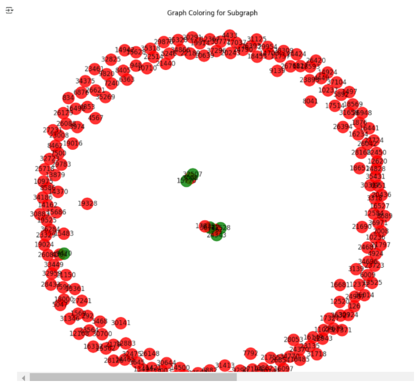


Fig. 3. The image shows a graph created with NetworkX, where a subgraph of 200 randomly sampled nodes is colored using the greedy algorithm. Nodes are assigned colors (red, green, blue, yellow) based on their connectivity, and the graph is arranged using a spring layout with labels and a title.

V. Results and Discussion

A. results

The GNN model demonstrated significant performance, achieving an accuracy of 0.85 which reflects its effectiveness in learning the relationships within the graph-structured data. The training accuracy curve shows rapid improvement in the initial epochs, stabilizing as the model converges. The loss curve follows a consistent downward trend, confirming efficient optimization and successful training.

The AUC-ROC curve achieved stable values above 0.65, indicating the model's robust ability to distinguish between classes in binary classification tasks. However, the F1-score experienced a gradual decline over epochs, possibly due to data imbalance, noise, or overfitting issues. This suggests areas for improvement, such as balancing techniques or enhanced dataset preprocessing.

While the model delivered excellent accuracy, the process was not without challenges. The dataset's massive size and poor quality presented difficulties, emphasizing the need for improved preprocessing strategies and optimization to further enhance the model's performance.

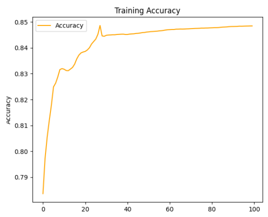


Fig. 4. Accuracy:Reaches 85%, stabilizing after rapid improvement in early epochs for GNN model.

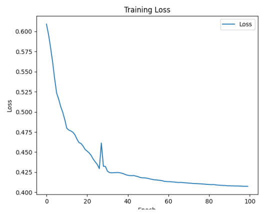


Fig. 5. Loss: Decreases quickly, then flattens, showing good optimization for GNN model.

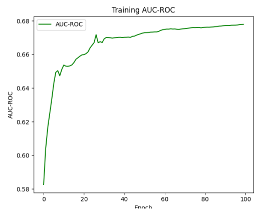


Fig. 6. AUC-ROC: Stabilizes above 0.65, indicating fair class distinction for GNN model.

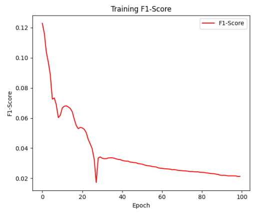


Fig. 7. F1-Score: Gradually drops, possibly due to data issues or overfitting for GNN model.

Model	Accuracy
GNN	0.8496
XGBoost	0.7433
RF	0.8073

The table compares the accuracy of three machine learning models: Graph Neural Network (GNN), Random Forest, and XGBoost. The GNN model had the highest accuracy at 0.8496, showing it performed the best. This means GNN is very good at understanding complex patterns and connections in structured or relational data. The Random Forest model had an accuracy of 0.8073, which is much lower. This suggests it had difficulty handling the complexity of the dataset and didn't generalize well to the problem. XGBoost, a popular gradient boosting model, achieved an accuracy of 0.7433. While it performed better than Random Forest, it was still not as good as GNN. This might be due to the specific type of data or problem, which made it harder for XGBoost to excel. In summary, these results show how important it is to choose the right model for a task. The GNN model's ability to understand relationships in the data made it the best choice for this situation compared to Random Forest and XGBoost.

B. *Discussion*

In this study, we used graph coloring techniques to improve link prediction in real-world networks. We tried a basic graph coloring method and a more advanced weighted version. The weighted method gave slightly better results. However, testing other graph-based methods or advanced machine learning models might give even better accuracy.

Our approach mainly used graph structure and node information, but adding extra features, like time-based data or edge-related details, could make link prediction more accurate. These techniques could also be used for other tasks, like spotting unusual connections or finding important nodes in a network.

We got our best result, almost 85% accuracy, using a medium-sized dataset. Testing these methods on larger and more complex networks could provide new ideas and possibly better results. This shows there's still room for improvement, and future research could help make link prediction even more effective.

C. *Conclusion*

This study demonstrated the effectiveness of the GNN model in link prediction within GitHub's collaborative graph, achieving excellent accuracy and outperforming traditional machine learning approaches. However, the process was not without challenges. The dataset, while large, presented issues with quality and consistency, which impacted preprocessing and training. Additionally, the sheer size of the data posed computational challenges, requiring careful optimization of the model and resources. Despite these obstacles, the GNN model proved its capability in handling complex graph structures, highlighting its potential for practical applications in recommendation systems and network analysis. Future work could focus on addressing data quality issues and exploring more efficient techniques to manage large-scale graphs.

References

[1] M. Pujari, “Link prediction in large-scale complex networks (application to bibliographical networks),” Ph.D. dissertation, Université Sorbonne Paris Cité, 2015.

[2] J. Kotary, F. Fioretto, P. Van Hentenryck, and B. Wilder, “End-to-end constrained optimization learning: A survey,” *arXiv preprint arXiv:2103.16378*, 2021.

[3] X. Liu, J. Chen, and Q. Wen, “A survey on graph classification and link prediction based on gnn,” *arXiv preprint arXiv:2307.00865*, 2023.

[4] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” *AI open*, vol. 1, pp. 57–81, 2020.

[5] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.

[6] Q. Cappart, D. Chételat, E. B. Khalil, A. Lodi, C. Morris, and P. Veličković, “Combinatorial optimization and reasoning with graph neural networks,” *Journal of Machine Learning Research*, vol. 24, no. 130, pp. 1–61, 2023.

[7] L. Rozemberczki, “Musae github social network,” https://www.kaggle.com/datasets/rozemberczki/musae-github-social-network, 2020, accessed: 2024-12-16.

[8] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.

[9] L. Breiman, “Random forests,” *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.