Graph Analysis: Link prediction and Node Classification in CORA citation network

Vikram Sahai Saxena
Rutgers University - New Brunswick
email

Sai Laxman Jagarlamudi
Rutgers University - New Brunswick
email

Vishwas Gowdihalli Mahalingappa
Rutgers University - New Brunswick
email

ABSTRACT

Abstract. Graphs have become increasingly important in various fields, leading to the development of predictive tasks on graphs. Among the methods used, graph embeddings have been found to be effective for a wide range of prediction tasks such as content recommendation and computer-aided drug design. This project aims to explore different methods of performing node classification and link prediction using the CORA graph dataset. Specifically, we compare the performance of GraphSAGE and GCN models on this dataset, and also examine the mathematical foundations of these networks.

KEYWORDS

Logistic Regression, GrphSAGE, GCN, Embedding, Node Classification, Link Prediction

1 INTRODUCTION

Graphs are a valuable source of information and predicting their behavior requires effective mathematical methods. Node embeddings provide a useful tool for prediction tasks on graphs, capturing high-dimensional information such as neighborhood nodes and degree, and converting them into one-dimensional vector representations. This simplifies a variety of prediction tasks, including node classification and link prediction.

Real-world complex networks can be represented as graphs, with their interactions between entities represented as edges. These networks are dynamic, with nodes and edges being added or removed over time. Studying the trend of how these graphs evolve has many advantages, particularly in predicting future associations between entities and classifying entities into categories.

Link prediction and node classification are challenging problems in complex graphs, with applications in various fields such as online social networks, knowledge graphs, biological networks, and citation networks. In the case of link prediction, the objective is to predict future connections between entities in a given graph, while node classification involves classifying entities into different categories.

The CORA citation dataset, which contains 2708 nodes representing scientific publications classified into one of seven classes and 5429 edges representing links, is used in this study. Several models using GraphSAGE and GCN node embeddings are implemented to predict the class of nodes and the existence of links between two nodes. The performance of each model is evaluated to determine which algorithm is most appropriate for obtaining the best results for link prediction and node classification tasks.

2 RELATED WORK

Semi-supervised learning on graphs is a popular research area, and several different approaches have been used to tackle it. The two main categories of these approaches are Laplacian regularization and graph embedding-based approaches.

One approach is to represent labeled and unlabeled data as vertices in a graph, with the weights of the vertices between them signifying the level of correlation between them. Gaussian random field can then be used to formalize the problem.

Another approach is to use manifold regularization to incorporate labeled and unlabeled data to develop a general use trainer. The theoretical basis for this method is rooted in Reproducing Kernel Hilbert spaces. Using them, the model could handle unlabeled data effectively.Deep semi-supervised embeddings applied to deep multi-layer neural network architectures either on every layer or simply at the output layer of the neural network is another notable approach. This approach yields competitive results compared to other semi-supervised techniques discussed above.

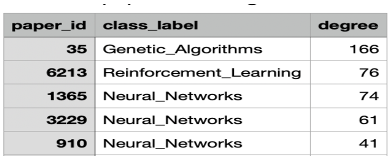
Newer models tend to focus more on models built on top of the skip-gram model. The skip-gram model was notable for not utilizing matrix multiplication while generating its embeddings, which allowed it to train up to 100 billion words in a day.Lastly, algorithms like DeepWalk and LINE sample random walks made on a graph to generate embeddings for each node of the graph. These algorithms have been shown to be effective in several applications.

3 PRELIMINARIES

Exploratory Data Analysis: Based on our analysis of the CORA citation dataset, we have discovered that the dataset is not a connected graph. We have identified the top five categories with the highest degrees, which are illustrated in Figure 1. Additionally, we have determined that there are a total of 3563 cliques in the dataset, with the maximum clique size being 5. There are nine cliques of size 5, all of which are in either the Reinforcement Learning or Neural Networks categories.To provide a better understanding of the CORA citation dataset, we have included visualizations of the dataset in Figure 3.

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

We have also discovered that the "Neural Networks" category has the highest number of scientific publications, followed by "Probabilistic Methods," which has the next highest number of scientific papers. This information is presented visually in Figure 2.



paper_id	class_label	degree
35	Genetic_Algorithms	166
6213	Reinforcement_Learning	76
1365	Neural_Networks	74
3229	Neural_Networks	61
910	Neural_Networks	41

Figure 1: Scientific paper with highest degrees

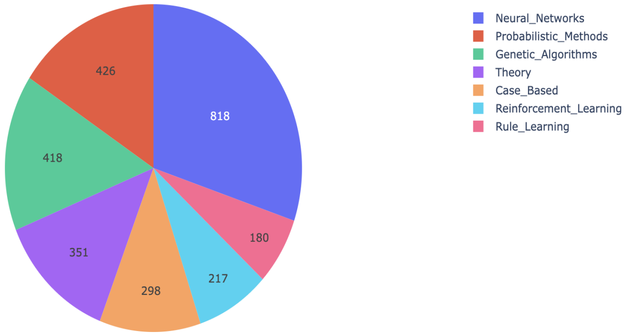


Figure 2: Counts of paper in each class

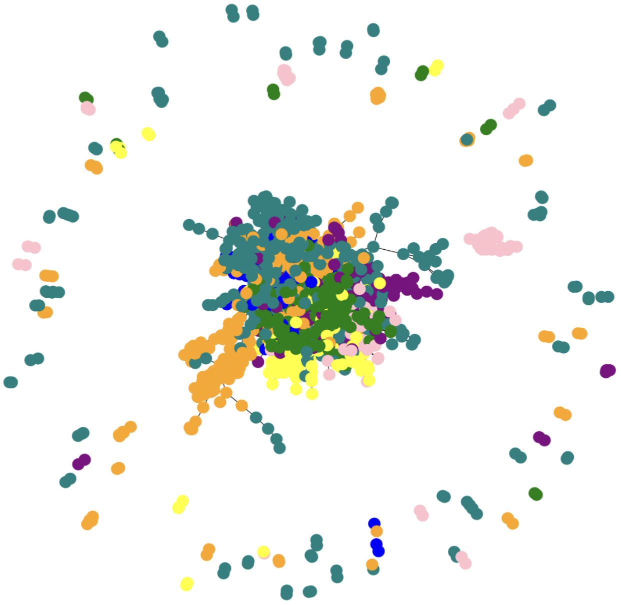


Figure 3: Plot of Scientific papers as nodes and citations as links.

4 PROBLEM FORMALIZATION

4.1 Link Prediction

Link prediction is a technique used to predict whether a link exists between two nodes in a graph. In this approach, each node in the graph is first converted into an embedding. An embedding is a low-dimensional vector representation of the node, where each element in the vector captures some aspect of the node's properties. The size of the embedding vector is determined by a parameter called 1Xdim, which is set by the user.

For the GraphSAGE link prediction model, the embeddings of two nodes that we want to predict a link for are concatenated using the multiply operator. This operation results in a vector that contains the element-wise multiplication of the two node embeddings. This concatenated vector is then fed into a neural network, which produces a score representing the likelihood of a link existing between the two nodes.

The neural network used in the GraphSAGE model is trained using a dataset of known links in the graph. During training, the network learns to assign higher scores to pairs of nodes that are likely to be linked, and lower scores to pairs that are unlikely to be linked. This process of training the network involves adjusting the weights of the neural network such that it can accurately predict the existence of links between nodes.

4.2 Node Classification

Node classification is a common task in graph analysis where the goal is to predict the label of a given node in the graph. To solve this problem, one approach is to convert each node of the graph into a low-dimensional vector, commonly known as node embeddings. The node embeddings capture the structural information of the graph in a compact and meaningful way.

Once the node embeddings are generated, they can be used as inputs to a conventional neural network. In this case, the labels of the nodes (in one-hot vector form) are used as outputs. The neural network can be trained using a supervised learning approach, where the objective is to minimize the cross-entropy loss between the predicted labels and the true labels.

The final output layer of the neural network typically consists of a softmax layer, which maps the output of the network to a probability distribution over the possible classes. This allows us to make predictions for previously unseen nodes by feeding their embeddings into the trained neural network and obtaining the predicted class probabilities.

One popular method for generating node embeddings is GraphSAGE (Graph Sampling and Aggregation), which is a general framework for inductive representation learning on graphs. GraphSAGE generates node embeddings by aggregating the features of a node's local neighborhood. The resulting embeddings capture the structural properties of the graph in the vicinity of each node, making them well-suited for node classification tasks.

5 THE PROPOSED MODEL

5.1 Logistic Regression

In order to predict links and classify nodes in a graph, a common approach is to use Node2Vec embeddings and logistic regression.

Node2Vec generates embeddings by performing random walks over the graph, where nodes that are close together in the graph remain close together in the embedding space. Using a binary operator on the embeddings of the source and target nodes, we can create edge embeddings for both positive and negative samples.

We can then train a logistic regression classifier using these embeddings to predict whether an edge should exist between two nodes. By combining the positive and negative edge embeddings, we can accurately classify nodes and predict links within the graph. This approach leverages the power of both Node2Vec and logistic regression to achieve accurate and efficient predictions in complex graph structures.

5.2 Graph convolutional network (GCN)

A graph convolutional network (GCN) is a type of neural network that is designed to operate on graphs efficiently. Unlike traditional convolutional neural networks, which operate on regular grids of pixels, GCNs can handle irregularly structured data represented as graphs.

The GCN algorithm obtains vector representations of each node in a graph by performing a simple forward propagation on inputs consisting of adjacency and feature matrices for each node. These node embeddings can then be used to make predictions about the class of each node by sending them to a Keras dense layer with softmax activation.

For link prediction, the node embeddings are concatenated using a simple sum to create a link embedding, which is then sent to a dense layer to produce predictions about the presence or absence of a link between nodes. Overall, GCNs offer a powerful and flexible tool for analyzing complex, structured data represented as graphs.

5.3 graphSAGE

GraphSAGE is an inductive semi-supervised learning framework that enhances over the Graph Convolutional Network (GCN) by considering not only a node's neighbors but also its neighbors' neighbors to compute the node representation. To avoid unnecessary computational overhead, the framework randomly samples a fixed number of neighbors and neighbors of neighbors for a fixed number of iterations.

Our model consists of a single layer of a 32x32 GraphSageModel, which takes in the sampled inputs aggregated by the generator. This model generates vectorized inputs and outputs that can be fed into a single dense layer of a neural network. The outputs from the dense layer are then passed through a softmax output layer, enabling multi-class classification.

For link prediction, we determine the link score by taking the dot product between the source node vector and the destination node vector.

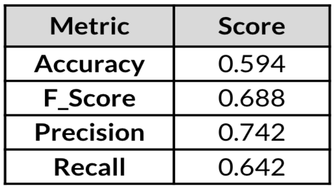
6 EXPERIMENTS

Our models have been assessed using four well-known metrics: accuracy, precision, recall, and f-score. The outcomes for several models are shown in the table below.

6.1 Link Prediction

We used a total of three different models—Logistic Regression, Graph Convolutional Network, and graphSAGE to perform link prediction in order to achieve benchmarks. The Logistic Regression model served as the baseline owing to its simplicity.

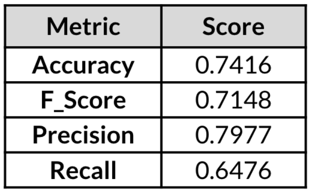
6.1.1 Logistic Regression. The Node2Vec technique was used to create embeddings that were fed into the simple logistic regression model. These are the outcomes that were attained:



Metric	Score
Accuracy	0.594
F_Score	0.688
Precision	0.742
Recall	0.642

Figure 4: Metric results for logistic regression link prediction

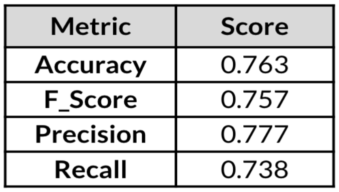
6.1.2 GCN. To obtain link embeddings, we produced vectors of length 16 for every node and concatenated them together. Then, these embeddings were input to a dense layer to make the link prediction.Forward propagation is used by the Graph Convolutional Network to create vector representations of the graph. The results obtained for this model:



Metric	Score
Accuracy	0.7416
F_Score	0.7148
Precision	0.7977
Recall	0.6476

Figure 5: Metric results for GCN link prediction

6.1.3 GraphSage. With a kernel size of 16 x 16, Graph convolutional network uses a single hidden layer. The result is h, a representation of a node. We created a dot product between the source node vector and the destination node vector for the prediction. These results reflect the degree to which a link between the source and destination nodes is present. Different kernel sizes and hop distances were tested. The metrics for the ideal grid size of 16 × 16 with a hop distance of 2 are as follows. The outcomes are a local optimum, although this testing was not complete.



Metric	Score
Accuracy	0.763
F_Score	0.757
Precision	0.777
Recall	0.738

Figure 6: Metric results for GraphSAGE link prediction

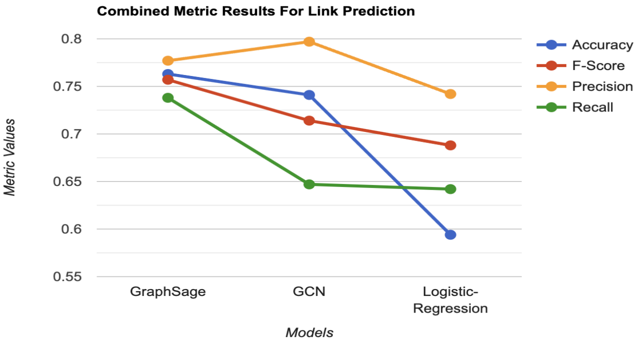
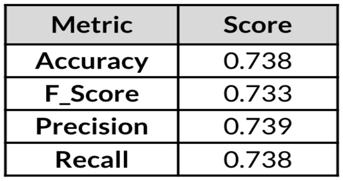


Figure 7: Combined metric results for link prediction

6.2 Node Classification

Utilizing three distinct models—Logistic Regression, Graph Convolutional Network, and graphSAGE—we attempted the node classification in order to achieve benchmarks. The Logistic Regression model served as the baseline owing to its simplicity.

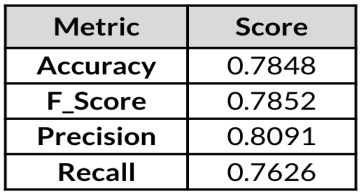
6.2.1 Logistic Regression. Logistic Regression: The Node2Vec algorithm was used to produce the embeddings that were fed into the simple logistic regression model. These are the outcomes that were attained:



Metric	Score
Accuracy	0.738
F_Score	0.733
Precision	0.739
Recall	0.738

Figure 8: Metric results for logistic regression node classification

6.2.2 GCN. Forward propagation is used by the Graph Convolutional Network to create vector representations of the graph. Following a single hidden dense layer and a softmax output layer, these representations are then used as input for multiclass classification. Following are the outcomes of the algorithm:

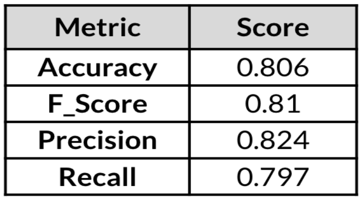


Metric	Score
Accuracy	0.7848
F_Score	0.7852
Precision	0.8091
Recall	0.7626

Figure 9: Metric results for GCN node classification

6.2.3 GraphSage. GraphSage. A Graph convolutional network with a single hidden layer and a 32 x 32 kernel size was employed. A softmax layer was applied to the results to categorise them.

Ten samples were counted for the first hop and five samples for the second hop, respectively. We experimented with a range of different counts on this one in hopes of improving or degrading it. This wasn't noticed, though, and the outcomes were all extremely identical to those listed below. We think this is because the dataset is rather tiny, and changing the count will have a bigger effect when working with much bigger datasets.



Metric	Score
Accuracy	0.806
F_Score	0.81
Precision	0.824
Recall	0.797

Figure 10: Metric results for GraphSAGE node classification

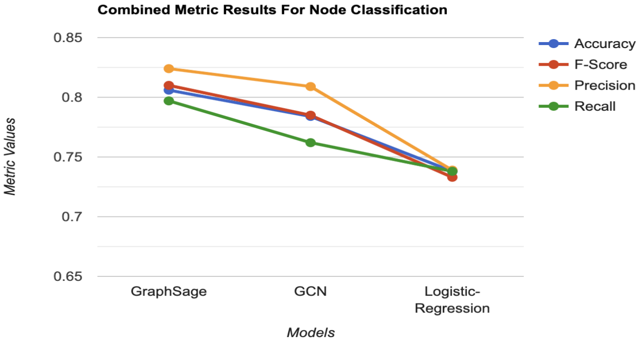


Figure 11: Combined metric results for node classification

7 CONCLUSIONS AND FUTURE WORK

This project utilized the CORA citation dataset, consisting of 2708 nodes representing scientific publications classified into one of seven classes, and 5429 edges representing links between the nodes. We implemented several models that utilized GraphSAGE and GCN node embeddings to predict the class of a node and the existence of a link between two nodes. We evaluated each model using metrics such as accuracy, f-score, precision, recall, and AUC, to determine which algorithms were most appropriate for our link prediction and node classification problems. Based on our evaluation, we found that the GraphSAGE algorithm outperformed other methods, including the GCN algorithm and baseline models, for both link prediction and node classification. In the future, We propose to create a more advanced ensemble model of GraphSAGE, GCN and other effective models to improve the current results. We plan to improve feature engineering by creating a robust feature transformation and selection methods to get better node or link embeddings. Additionally, we plan to conduct more extensive hyper parameter tuning to fine-tune the accuracy of our models. We also intend to balance the samples in each class to improve the quality of our data.

8 ACKNOWLEDGEMENT

We appreciate the expertise and materials provided by Professor Yongfeng Zhang and teaching assistants from Rutgers University faculty. Without cooperation from everyone involved in our team and assistance from the resources cited, this project could not have been finished. We are extremely grateful for the faculty's assistance and the learning opportunities they have given us.