CP-318 Project-1: Predicting Links in Social Networks - Report

Kartik Soni, Manya Verma, Yash Gupta

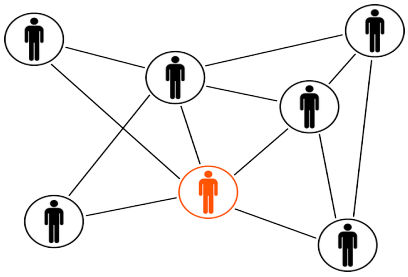
October 2, 2023

1 Introduction

Pairwise relationships are widespread in our world, and networks are a powerful tool for representing and studying them. However, real-world networks often contain missing edges, which can introduce biases into analysis and make it difficult to predict future connections.

This project aims to develop a model to predict whether edges exist between pairs of nodes in a network, even when some edges are missing. The training data for the model is a partial crawl of the Twitter social network, and the test data is a list of **2000** edges, of which **1000** are authentic and **1000** are fake.

The successful development of a model to predict missing edges in networks would have many applications. For example, such a model could be used to improve the accuracy of network analysis algorithms, to predict future connections in social networks, and to identify potential threats in networks such as the spread of diseases or cyberattacks.



2 About the Dataset

One of the initial challenges encountered was the loading of the dataset. Usually, we use the Pandas library to load the dataset, but in this particular instance, the dataset's size (**500MB**) posed a significant hurdle. Loading it through Pandas resulted in a cumbersome **7-8 minute** wait just to access the data. The experiments and model training were conducted on a computer with the following system configuration: **Intel i5 8th Gen, 8GB RAM**

The training dataset is a CSV file called **train.csv**. It contains **20,000 rows**, each of which represents a directed edge in the Twitter social network. The first column of the file contains the source node ID, and the rest of the columns contain the target node ID.

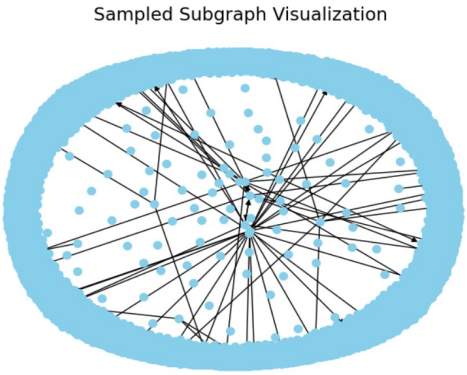
The training dataset is a large and complex dataset, with nearly **2,812,696** unique entries and a staggering **15,489,158 edges**. This made it difficult to load the dataset using the usual Pandas library. Instead, we used the CSV reader library from Python to open the file directly into a 2D list. We then used the NetworkX library to create the nodes and edges from the list.

3 Data Preprocessing

In this section, we detail the steps involved in preprocessing the dataset to prepare it for further analysis.

3.1 Subset Creation

To facilitate our analysis, we created a subset of the dataset. The subset consists of 10,000 source nodes randomly selected from the entire set of source nodes in the dataset. The below image is a further small subset of around 1000 nodes just for visualizing:



3.2 Positive and Negative Samples

For training purposes, we generated positive and negative samples. Positive samples comprise **10,000** randomly selected edges from the data set. Creating negative samples is crucial to train a robust link prediction model. As we didn't want to generate negative samples randomly. Instead, we aimed for **intelligent sampling (10,000) to make these negative samples** more realistic. In a real-world scenario, it's common for a node to have some connections (source node list) but not be directly connected to all nodes within that list.

4 Feature Extraction

In the context of our project, feature engineering plays a crucial role since the initial dataset primarily consists of source nodes and their respective followers. To train our machine learning models effectively, we need to derive meaningful features from this limited information.

4.1 Features Extracted

1. **Common Neighbors**:

$$\text{Common Neighbors}(A, B) = |\Gamma(A) \cap \Gamma(B)|$$

Common neighbours represent the number of nodes shared between two nodes in a network.

2. **Jaccard Coefficient**:

$$\text{Jaccard Coefficient}(A, B) = \frac{|\Gamma(A) \cap \Gamma(B)|}{|\Gamma(A) \cup \Gamma(B)|}$$

The Jaccard coefficient quantifies the similarity between two sets by measuring the intersection over the union.

3. **Adamic Adar Coefficient**:

$$\text{AAC}(A, B) = \sum_{u \in \Gamma(A) \cap \Gamma(B)} \frac{1}{\log(\text{degree}(u) + 1)}$$

This coefficient measures the similarity between nodes based on their shared neighbours. It assigns higher importance to shared neighbours with lower degrees, indicating a more significant connection.

4. **Resource Allocation Index**:

$$\text{RAI}(A, B) = \sum_{u \in \Gamma(A) \cap \Gamma(B)} \frac{1}{\text{degree}(u)}$$

The resource allocation index evaluates the extent to which two nodes allocate resources to common neighbours. It highlights the importance of shared neighbours in establishing a connection.

5. **Total Followers**:

This feature captures the total number of followers for a given node. It reflects the node's popularity or influence within the network.

6. **Friends Measure**:

$$\text{Friends Measure}(A, B) = \frac{2|\Gamma(A) \cap \Gamma(B)|}{\text{degree}(A) + \text{degree}(B)}$$

Friends measure assesses the similarity between two nodes by considering the ratio of shared friends to the total number of friends.

7. **In-Degree**:

$$\text{In-Degree}(v) = \sum_{u} A_{u,v}$$

In-degree refers to the number of incoming connections or links that a node (or user) receives from other nodes in the network.

8. **Out-Degree**:

$$\text{Out-Degree}(v) = \sum_{u} A_{v,u}$$

Out-degree, on the other hand, represents the number of outgoing connections or links from a node to other nodes in the network.

9. **Edge Rank**: It is a custom feature introduced to further enhance the feature set. It considers factors like the recency of interaction between nodes. nodes

5 Model Selection and Training

To tackle the link prediction problem, we explored several machine learning and deep learning models known for their effectiveness in such tasks. These models include:

- **Logistic Regression**: It assumes linear relationships between features and target variables. Our experiments revealed that with a limited set of features, the results were less promising. However, with the full feature set, it achieved an impressive **91.3%** accuracy on our training data.

Logistic Regression Model Accuracy: 0.9130263157894737

	precision	recall	f1-score	support
0	0.88	0.96	0.92	3821
1	0.96	0.86	0.91	3779
accuracy			0.91	7600
macro avg	0.92	0.91	0.91	7600
weighted avg	0.92	0.91	0.91	7600

Unfortunately, when applied to the Kaggle dataset, its performance dropped significantly to **85%**. This decline underscored the model's struggle to capture the complex non-linear patterns inherent in social networks, making it less suitable for accurate link prediction.

- **Ensemble Learning methods**: We explored two ensemble learning methods, **Random Forest and XGBoost**, for link prediction. Random Forest initially demonstrated strong performance during training, achieving an accuracy of approximately **89.6%**. However, when applied to the Kaggle dataset, its accuracy dropped to **86.3%**. This decline indicated the risk of overfitting, where the model learned the training data too well and struggled to generalize to unseen data.

Random Forest Model Accuracy: 0.8953947368421052

	precision	recall	f1-score	support
0	0.90	0.89	0.90	3821
1	0.89	0.90	0.90	3779
accuracy			0.90	7600
macro avg	0.90	0.90	0.90	7600
weighted avg	0.90	0.90	0.90	7600

XGBoost, an efficient gradient-boosting algorithm, was also leveraged to address the link prediction challenge. It was chosen for its ability to handle the weak learner problem effectively. To optimize the performance of **XGBoost**, we utilized **RandomizedSearchCV**, a hyperparameter tuning technique.

- **Final Approach: Deep Neural Network**

  Our ultimate strategy for tackling the link prediction challenge was deploying **Deep Neural Networks**. Here's the rationale that guided our choice:

  - Capturing Complex Relationships
  - Autonomous Feature Learning
  - Scalability on larger datasets

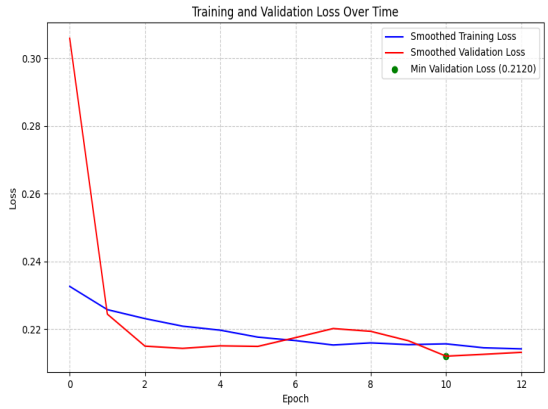
  Model Architecture: The model comprises several dense hidden layers, each followed by batch normalization. The activation function used for the hidden layers is **ReLU** which is computationally efficient and helps mitigate the vanishing gradient problem. The output layer consists of a single neuron with a **sigmoid** activation function, producing a probability score for link prediction.

  Model Summary:

```
Total params: 48,257
Trainable params: 47,297
Non-trainable params: 960
Number of epochs: 50
Batch size: 64
Optimizer: Adam with learning rate decay
```

6 Results and Conclusions

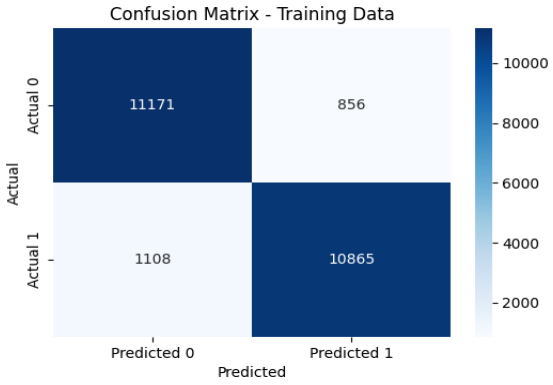
The deep neural network model underwent extensive training and validation over 50 epochs. We monitored its performance using both training and validation datasets. We can see clearly from the below graph how Training loss and Validation loss converge to minimum value after a few epochs, which shows that our model is well-trained and prevents overfitting.



Our deep neural network model exhibited outstanding performance in the link prediction task. It not only achieved a high accuracy of **92.7%** on the training set and an accuracy of **90.2%** on the Kaggle test set but also showcased remarkable predictive capabilities.

Neural Network Model Accuracy on Validation Data: 0.916833
Classification Report on Validation Data:

	precision	recall	f1-score	support
0	0.90	0.93	0.92	2973
1	0.93	0.90	0.92	3027
accuracy			0.92	6000
macro avg	0.92	0.92	0.92	6000
weighted avg	0.92	0.92	0.92	6000



Due to time and computational constraints, our exploration was limited. Future work could include:

1. Exploring advanced neural architectures like **Graph Convolutional Networks (GCNs)**
2. Conducting **Hyperparameter Tuning** to optimize learning rates, batch sizes, and layer configurations.
3. **Handling Class Imbalance** through techniques like oversampling, undersampling, or generating synthetic samples.

In conclusion, this project has been an invaluable learning experience.

References

[1] William Cukierski, Benjamin Hamner, Bo Yang, *Graph-based Features for Supervised Link Prediction*, Journal/Conference, Year.

[2] Michael Fire, Lena Tenenboim-Chekina, Rami Puzis, Ofrit Lesser, Lior Rokach, Yuval Elovici, *Computationally Efficient Link Prediction in a Variety of Social Networks*, Journal/Conference, Year.

[3] Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, Yuval Elovici, *Link Prediction in Social Networks using Computationally Efficient Topological Features*, Journal/Conference, Year.

[4] Rohan Gore, *Link Prediction in Social Networks*, Medium, URL: https://medium.com/@gorerohan15/link-prediction-in-social-networks-599e6d9bed9b

[5] Prateek Joshi, *A Guide to Link Prediction: How to Predict Your Future Connections on Facebook*, Medium, URL: https://medium.com/analytics-vidhya/a-guide-to-link-prediction-how-to-predict-your-future-connections-on-facebook-292c0b71697d