# Link Prediction in an Actor Co-Occurrence Network

Annabelle Luo
MSc DSBA 2022-2023
ruijie.luo@student-cs.fr

Konstantina Tsilifoni
MSc DSBA 2022-2023
konstantina.tsilifoni@student-cs.fr

Namrata Tadanki
MSc DSBA 2022-2023
namrata.tadanki@student-cs.fr

Yasmina Hobeika
MSc DSBA 2022-2023
yasmina.hobeika@student-cs.fr

Team Name: **Net-WorkingOurWaytotheTop**

## 1 INTRODUCTION

In this project, we aim to predict missing links in an actor co-occurrence network. The dataset contains information about actors where each node in the network represents an actor and edges between two nodes represents co-occurrence on the same Wikipedia page.

The dataset consists of three files: the training set, test set, and a node_information file. The training set includes 10k labeled node pairs, with 1 indicating the presence of an edge between two nodes, and 0 indicating the absence of an edge. The test set includes 3498 node pairs for which the presence of an edge needs to be predicted. The node_information file includes 932 features for each node in the network, representing an encoding of the textual information of each actor's Wikipedia page.

We aim to perform the prediction using graph-theoretical and node feature information.

## 2 GRAPH INFORMATION

Some characteristics of the training network are

1. Number of nodes: 3597
2. Number of edges: 5248
3. Minimum degree: 1
4. Maximum degree: 361

We also observed that there were some self-loops in the network

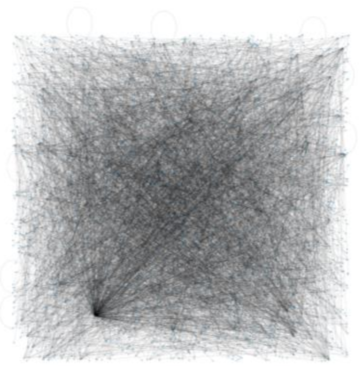

Fig.1. Graph of actor co-occurrence network

## 3 FEATURE ENGINEERING

### 3.1 USING NODE INFORMATION

We computed different similarity metrics based on node information as features to train the models to perform link prediction. These similarity metrics are

1. Cosine similarity - Cosine similarity measures the cosine of the angle between two vectors in high-dimensional space. It ranges from -1 to 1, where 1 means the two vectors are identical and -1 means they are completely dissimilar.

2. Euclidean distance - Euclidean distance measures the distance between two points in space. It is the straight-line distance between two points and is calculated as the square root of the sum of the squared differences between the coordinates of the two points.

3. Hamming distance - Hamming distance measures the number of positions at which the corresponding symbols in two strings of equal length are different. It is used to measure the difference between two binary strings or feature vectors.

4. Manhattan distance - Manhattan distance measures the sum of the absolute differences between the coordinates of two points in space.

### 3.2 USING GRAPH INFORMATION

As with the approach above, similarity metrics based on the graph information were computed as features. These similarity metrics are

1. Degree centrality of source node and target node - Degree centrality is a measure of the importance of a node in a graph based on the number of edges connected to it. Nodes with high degree centrality are highly connected to other nodes in the graph and are considered important.

2. Preferential attachment - Preferential attachment suggests that nodes in a network tend to link to other nodes that already have many links. So, nodes that are already highly connected are more likely to receive additional connections in the future.

3. Jaccard similarity - Jaccard similarity is a measure of the similarity between two sets of nodes based on the proportion of common nodes they share. The higher the Jaccard similarity, the more likely it is that a link will exist between the source and target nodes.

### 3.3 USING NODE EMBEDDINGS

Lastly, we used Node2Vec to generate low-dimensional embeddings of the nodes of the graph that capture the structural properties of the graph. It works by generating multiple random walks and using a skip-gram model to learn the embeddings for each node.

## 4 MODELS IMPLEMENTED

### 4.1 LOGISTIC REGRESSION

We have developed and trained two logistic regression models to predict labels for edges in a graph. The first model employs logistic regression with an L2 penalty and a 'saga' solver. It is trained on a combination of node and graph features. The second model trains a logistic regression model with node embeddings generated by Node2Vec for the nodes in each edge. The final predictions were made by performing a logical OR operation on the predictions of the two models.

Upon evaluation, we observed that the first model is performing well, while the second model is only predicting 800 instances as 1 with a Kaggle accuracy of 52%. This led to a conclusion that we have a balanced test data in the public test set. Therefore, we have used a logical OR operation to predict an edge as 1 if it is predicted by either model.

### 4.2 XGBOOST

We also trained an XGBoost model on the node and graph features and embedded features, as with the approach above. The model parameters were defined to have random state of 42, evaluated metric of log loss, gamma value of 0.7, learning rate of 0.1, maximum depth of 5, minimum child weight of 8, and 50 estimators.

### 4.3 UNSUPERVISED LEARNING

Unsupervised link prediction is used to predict the existence of links between nodes in a network based solely on their features or topological structure. In our case, unlike supervised link prediction models described above, where the labeled data is used to train the model, with the unsupervised link prediction we will not utilize any labels and will rely only on the available information about the network. Unsupervised learning for link prediction typically involves training a model to learn the underlying patterns in the network without using any explicit supervision. One approach to unsupervised link prediction is to use similarity measures between nodes based on their feature vectors or structural properties.

In our first approach we use the embeddings for the nodes in the network. We compute the similarity between the embeddings of pairs of nodes in the test set, and rank them according to this similarity. The idea is that pairs of nodes that are more similar in their embedding space are more likely to form a link in the network. More specifically, we use the cosine similarity, the euclidean distance and the pearson distance between the feature vectors of two nodes. All these similarities are averaged out and used as a final measure of similarity. The same metrics are then computed for the test set and the pairs of nodes are ranked according to their similarity score, and the top-ranked pairs are used as predictions for new links.

In our second approach, we use an MLPRegressor which has 3 hidden layers with 128, 32 and 8 units respectively. The activation function is ReLU, the solver is Adam, and the maximum number of iterations is set to 200. The model is trained using the features of the training set and a label vector of ones with the same length as the training set. Therefore, we use all the pairs of nodes in the training data as positive examples, assuming that they are all connected. This allows the model to learn the underlying patterns and similarities among the nodes that are indicative of links. In other words, the goal is to learn a function that maps pairs of nodes to a similarity score, with the assumption that pairs of nodes that have a high similarity score are more likely to be connected in the network. The unsupervised learning approach allows us to learn these patterns without any prior knowledge of the network structure and can potentially discover new connections that were not previously known. The predicted value in this case is not the similarity score itself, but rather a score that indicates how likely it is that a pair of nodes in the graph are connected. This score can be interpreted as a measure of similarity between the two nodes, where higher scores indicate greater similarity (i.e., greater likelihood of a connection). After training the model, the similarity scores are computed for the test set, the pairs of nodes in the test set are then ranked by similarity scores, and the top-ranked pairs are selected as predictions for new links.

## 5 MODEL COMPARISON & EVALUATION

### 5.1 EVALUATION METRICS

To evaluate the models, the available data is split in train, validation, and test sets. Each model is first built and trained using the training features and labels. Then, to evaluate the model's performance, the Area Under the Receiver Operating Characteristic Curve (ROC AUC) is calculated from the predicted probabilities, and a Receiver Operating Characteristic Curve (ROC curve) is plotted to visualize the results. The ROC curve provides a way to evaluate the performance of a binary classifier at different decision thresholds. In addition, the precision, recall, and F1 score are calculated and used to evaluate the performance of the model.

Furthermore, a classification report is printed to summarize the precision, recall, F1 score, and support for each class in the validation data to obtain a more thorough understanding of the strong and weak spots of the model. Following that, feature importance is observed (if possible), to drop variables that are not adding value to the model and possibly contribute to overfitting. Lastly, since the task at hand refers to binary classification we are trying to find the optimal threshold for the classifier that maximizes accuracy in the validation set and we use that threshold to update our classification model. At this stage, predictions were made on the real test set and submissions were made on Kaggle to have a baseline on overall performance.

### 5.2 RESULTS

| **Model** | **Train Accuracy (%)** | **Validation Accuracy (%)** | **Test Accuracy (Kaggle) (%)** |
|---|---|---|---|
| **Logistic Regression** | **71.15** | **76.97** | **76.63** |
| XGBoost | 73.96 | 77.74 | 74.78 |
| Unsupervised MLP | | 61.72 | 50.33 |
| Unsupervised 3 similarities approach | | 71.43 | 47.76 |

The best prediction was made by the Logistic Regression model which was trained on graph features and node embeddings and whose results were logically combined.

## 6 OTHER ATTEMPTS

We also tried other ideas to try and improve the quality of link prediction.

### 6.1 PRINCIPAL COMPONENT ANALYSIS

Since the node information has 932 columns, we attempted to perform Principal Component Analysis (PCA) to reduce the dimensionality and generate new features that capture the most significant information.

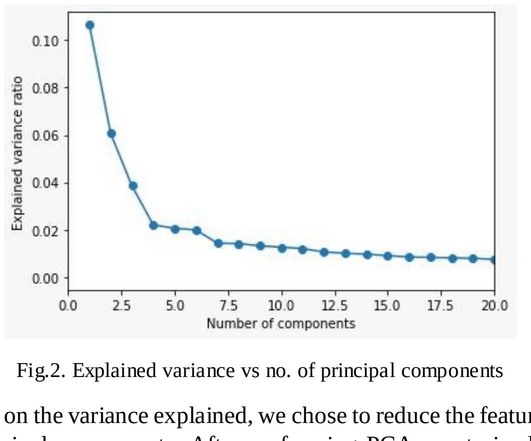

Fig.2. Explained variance vs no. of principal components

Based on the variance explained, we chose to reduce the features to 5 principal components. After performing PCA, we trained two logistic regression models. The first model was trained on the features extracted from PCA and the second model was trained on the node embedding features. A logical OR was performed on the results obtained from the two models to make the final predictions. However, this approach did not work well and did not increase the accuracy.

### 6.2 COMMUNITY DETECTION BASED FEATURES

We implemented Greedy Modularity Algorithm to perform community detection and identify nodes that are similar to each other and extract some properties of connectivity from the graph.

We generated 65 communities using the algorithm after which we created an indicator feature where 0 implies that the source node and target node belong to different clusters and 1 implies that they belong to the same cluster.