Link Prediction Challenge - Team: "N'importe quoi !"

LEONARDO BASILI[1], PAUL BÉDIER[1], AND LASSE SCHMIDT[1]

[1] *ESSEC Business School & CentraleSupélec, France*

Compiled March 27, 2023

This report summarizes our work during the Kaggle challenge within the Machine Learning in Network Science class at CentraleSupélec. The challenge was about predicting missing links in an actor co-occurence network and we reached a score of 78.17 % on the public test set.

Link to GitHub Repository: https://github.com/lassefschmidt/Network-Science_Challenge

A. Introduction

Within this project, we intend to predict randomly removed edges to accurately reconstruct the initial network using both graph-theoretical and node feature information. The targeted graph dataset is an actor co-occurence network where each node corresponds to an actor and the edge between two nodes denotes co-occurence on the same Wikipedia page. In consequence, edges between two given actors can be seen as proxies for their joint participation in the same movie, for their personal relation, or for their level of fame.

The report is organized as follows. In section B, we present our work regarding Data Cleaning, while section C is devoted to Exploratory Data Analysis and Feature Engineering. Section D covers Feature Selection and section E covers Hyperparameter tuning and Modeling. Finally, our results are stated in section F.

B. Data Cleaning

The provided training data contains 10,496 labeled node pairs, while the test data encompasses 3,498 unlabeled node pairs. The labels identify whether or not there is actually an edge between the two nodes. In addition, we are provided with an encoding of the textual information contained within each of the 3,597 nodes. The encoding consists of 932 binary features that denote the presence of a given keyword.

Overall, the data set is quite clean, however there are some issues:

a) training edgelist contains 31 node pairs where source and target node are identical,

b) test edgelist contains 17 node pairs where source and target node are identical, ACTION: we predict 1 (a link) in this case

c) node-level information contains 3 columns (out of the 932) that are completely empty (that is no node actually contains the given keyword), ACTION: we drop these 3 columns

d) node-level information contains only 2,972 unique encodings (the most represented embedding relates to 82 nodes and the top 10 embeddings relate to over 288 nodes),

e) and 22 nodes have the same embedding as well as the same set of neighbors.

C. Data Analysis and Feature Engineering

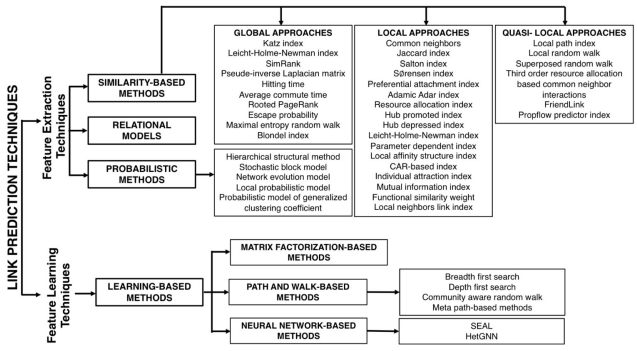

Figure 7. A taxonomy for the feature extraction techniques and feature learning methods in link prediction literature.

Fig. 1. Link Prediction Literature Review (taken from [1])

The feature engineering process was based on the link prediction framework provided in [1], summarized by the following Figure 1. Overall, we ended up implementing various feature extraction as well as feature learning techniques. More specifically, we have implemented 9 local, 3 global and 1 quasi-local approach for feature extraction, and two techniques for feature learning (Variational Graph Normalized Auto-Encoder, as presented in [2], and node2vec). The generated features have subsequently been used in supervised as well as unsupervised prediction approaches and allowed us to experiment with a wide array

of different techniques. Before going into more details, let us provide a general overview of the provided graph.

Graph Structure

Overall, the graph is quite different from a random graph, as can be seen in the following table 1. In consequence, we should be able to extract / learn meaningful features to predict the missing links to accurately restore the initial network.

Metric	Training Graph	Random Graph	Comparable?
Degree distribution	Power law (see Fig. 2)	Normal	No
Ø Path length	6.04	8.2	Yes
Clustering coefficient	2×10^{-2}	8×10^{-8}	No
Conn. Component	1	GCC exists when c>1	Yes

Table 1. Comparison of Training Graph to Random Graph

Figure 2 provides an in-depth overview over the graph. For example, you can easily see that the degree distribution of our training graph does indeed follow a power law distribution (linear relation in log-log scale). Its construction was robust 4.

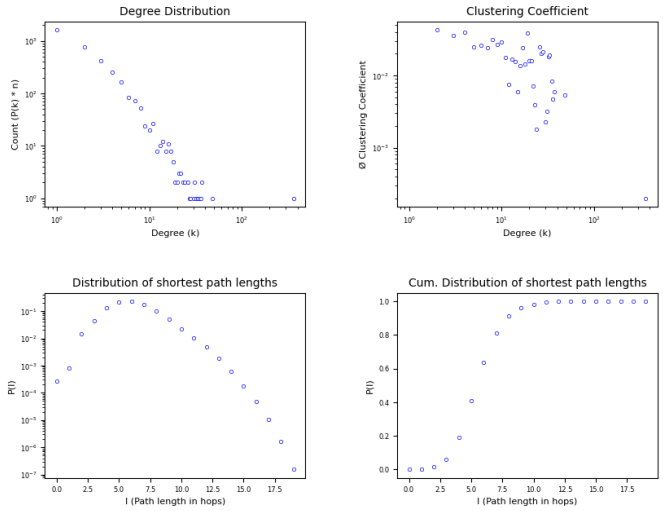

Fig. 2. Structural Analysis of Training Graph

Graph-based features

We extracted multiple graph based features which represent the structural properties of the network and can be really useful for the link prediction tasks. The implemented graph-based features are as follows:

Global approaches

1. **Katz Index** measures the proximity of nodes in a network by considering the total number of paths between them, with a damping factor applied to penalize longer paths.

2. **SimRank** measures the similarity between two nodes based on the notion that two nodes are considered similar if their neighbors are similar. It is assigned to each pair of nodes in the data frame.

3. **Rooted Pagerank** measures the importance of nodes based on the probability of reaching each node from the root node. Nodes that are highly likely to be reached from the root node will have a high Rooted PageRank score, while nodes that are less likely to be reached will have a lower score.

Local approaches

1. **Common Neighbor Centrality (CNC)** captures the importance of a node in a network based on the number of common neighbors it shares with other nodes.

2. **Jaccard Coefficient (JCC)** measures the similarity between two nodes based on the ratio of their shared neighbors to their total unique neighbors.

3. **Salton Index (SaI)** This index measures the similarity between two nodes based on the product of their degrees divided by the square root of the product of the degrees of their neighbors.

4. **Sorensen Index (SoI)** measures the similarity between two nodes based on the ratio of the number of their common neighbors to the average degree of the nodes.

5. **Adamic-Adar Index (AA)** quantifies the similarity between two nodes based on the inverse logarithm of the degree of their shared neighbors.

6. **Preferential Attachment (PA)** measure the tendency of nodes to acquire new connections based on their existing connections and structural properties. Preferential Attachment index indicates try to model the idea that new nodes in a network tend to connect to already well-connected nodes.

   Resource Allocation(RA) takes into account the structure of the network and the influence of shared neighbors on the potential connection between two nodes.

   Both these features had been enhanced using Collaborative Filetering approaches proposed in [3]

   - **Collaborative Filtering (CF)** makes the indexes capable to capture also the interactions similarity between two nodes.
   - **Self-included Collaborative Filtering (SCF)** incorporates the individual characteristics of the node which should influence the likelihood of new link.

7. **Logarithm of Preferential Attachment (PAlog)** calculates the natural logarithm of the Preferential Attachment score to normalize the values and reduce the impact of extreme values.

8. **Hub Promoted index (HProm)** privileges link formation between lower degree nodes and the hubs (nodes with higher degree). Conceive the idea that hubs are more likely to form connections with lower degree nodes rather than other hubs.

9. **Hub Depressed index (HDep)** privileges link formation between nodes similar degree level, therefore connections between hubs and lower degree nodes are penalized. Conceive the idea that hubs are more likely to form connections between them rather than with lower degree nodes (and viceversa).

Quasi-Local approaches

1. **Friendlink** finds similarities between nodes in an undirected graph constructed from the connection data, using on a similarity matrix can infer which couple of nodes is more likely to form a new connection [4].

Node-based features

1. **Degree Centrality (DCT)** measures the importance of a node based on its number of connections to other nodes.
2. **Betweenness Centrality (BCT)** quantifies the extent to which a node lies on the shortest paths between other nodes in the network.
3. **Page rank based features** measure the importance of nodes within the graph, assigning a score to each node based on random walks with teleportation probability. The intuition behind the following features is that the relative importance of the source and target nodes may influence the link between them:
    - **Average PageRank (PR1)** calculates the average PageRank score between the source and target nodes.
    - **Squared difference in PageRank (PR2)** calculates the squared difference between the PageRank scores of the source and target nodes.

Graph- and Node-based features

1. **Variational Graph Normalized Auto-Encoder (VGNAE)** consists of an encoder and a decoder: the former takes as input the node level features and edgelist of our graph and outputs a low-dimensional representation of each node while the latter takes this low-dimensional representation and outputs a probability score for each possible edge in the graph [2]. The model used for this procedure is trained using a combination of reconstruction loss and KL-divergence loss. Nodes are also enriched before training with additional features seen previously (DCT, BCT, PR, HUB, AUTH etc.).
2. **Node2Vec** provides node embeddings that capture the structural and relational information of nodes in a graph by generating biased random walks from each node [5]. These embeddings are then used to represent node pairs in a link prediction task, serving as input features for a machine learning model or with a binary operator (Average, Hadamard, WeightedL1, WeightedL2) as a measure of similarity to predict the likelihood of a link between nodes. Hyperparameter q was finetuned to be quite low (=0.25), which makes the embedding capture the homophily of the network (i.e. potential communities 6).

Node Information Features

The node information features relate to the 932-dimensional keyword embeddings of each node. As can be seen in Figure 5, the embeddings are very sparse with over 80% of nodes having less or equal to 10 keywords (and vice versa, most keywords are related to less than 10 nodes). Besides the VGNAE that takes these embeddings as input for its decoder, we tried many other methods to get insights from this feature matrix, such as

1. computing the cosine similarity between the keyword embedding of the source and target node,
2. applying dimensionality reduction techniques on the node information feature matrix such as PCA or UMAP,
3. applying clustering algorithms on the result of the previous step (such as HDBSCAN), as well as
4. using the apriori algorithm to detect potential association rules between the keywords of the respective source and target nodes for all positive edges in our graph.

However, all these approaches did not give sufficient results. We suppose that this is the case because we are dealing with a co-occurence graph where two given actors (nodes) are linked when they co-occur on the same Wikipedia page. As actors can work on a wide range of different movies or be in a relationship with each other, actors with completely different embeddings may still co-occur on some Wikipedia page).

D. Feature Selection

For both our unsupervised and supervised modeling methods, we performed feature selection. We computed correlation matrices between features, and removed one of each pair of highly correlated features. In the case of unsupervised learning, we also used grid search on possible combinations of features, ranking solutions by validation accuracy and retaining the features for which greatest accuracy was obtained.

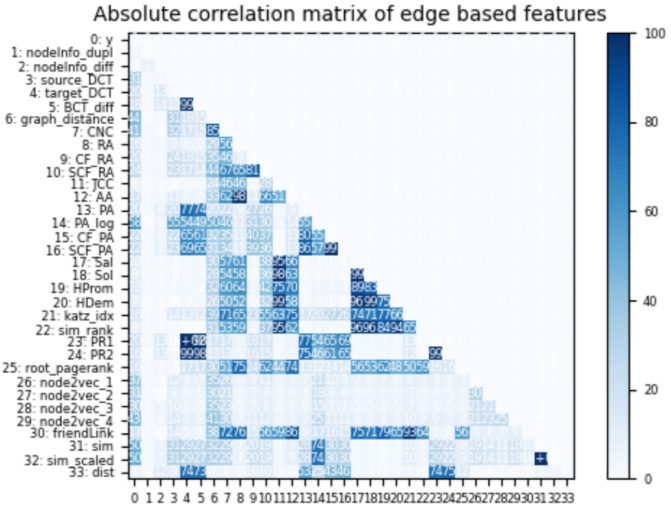

Fig. 3. Feature Correlation Matrix

E. Hyperparameter Tuning and Modeling

We performed our training validation split, preventing data leakage during training and allowing us to limit overfitting. This split was achieved by removing 20% of edges from original graph to create the training graph, while making sure it stays connected throughout the process. Our validation set is then the 20% removed edges for which we make predictions, allowing us to evaluate the model.

The unsupervised model is a very simple yet efficient feature aggregator, making predictions based on one of three different

heuristics (average, rank_average and whitened sigmoid) and using a threshold value (i.e. superior to 0.5 or median). As mentioned in the previous part, it is built upon a grid search over possible combinations of features.

For our supervised models with many hyperparameters (Random Forest & XGBoost) as well as for our feature learning models (VGNAE, Node2vec), tuning process was achieved using RayTune library for large-scale experiments and hyperparameter tuning, as well as its scikit-learn API. We used grid search over model hyperparameters for VGNAE autoencoder architecture parameters, as well as Node2vec random walk parameters and target number of embedding dimensions. For supervised models, we used randomized search, guided by a Bayesian optimization algorithm (Tree-of-Parzen, hyperopt library).

We evaluated our models using validation accuracy and AuC (area under ROC curve). The results for each best model can be found in the table below:

Table 2. Model Evaluation Results (accuracy)

Model	Train Acc.	Val. Acc.	Test Acc. (public)
Logistic Regression	0.8308	0.7726	0.7488
Random Forest	0.9042	0.7678	0.7509
XGBoost	0.9123	0.7764	0.7544
SVM	0.8387	0.7354	NA
Unsupervised	0.9067	0.78022	0.7817

F. Conclusion

For the task of link prediction on the actor co-occurence network, we applied a rigorous framework consisting of feature learning algorithms (global, local and quasi-local), as well as node/edge embeddings using neural network autoencoders and advanced random walk procedures, and proceeded to test a series of supervised and unsupervised models. About midway throughout the project, we noticed our best performance was consistently obtained through unsupervised methods as the very specific training / validation split (necessary to avoid data leakage) prevented us from applying k-fold cross validation for supervised models, leading them to overfit most of the time and fail to generalize as well. This is why we dedicated most of our work towards additional features to test in an unsupervised model. Our results rank us 4th team on the public leaderboard a few hours before submission time, indicating the soundness of our approach and methodology for this project.

REFERENCES

1. E. C. Mutlu, T. Oghaz, A. Rajabi, and I. Garibay, "Review on learning and extracting graph features for link prediction," Mach. Learn. Knowl. Extr. **2**, 672–704 (2020).
2. S. J. Ahn and M. Kim, "Variational graph normalized AutoEncoders," in *Proceedings of the 30th ACM International Conference on Information &amp Knowledge Management*, (ACM, 2021).
3. Y.-L. Lee and T. Zhou, "Collaborative filtering approach to link prediction," Phys. A: Stat. Mech. its Appl. **578**, 126107 (2021).
4. A. Papadimitriou, P. Symeonidis, and Y. Manolopoulos, "Fast and accurate link prediction in social networking systems," J. Syst. Softw. **85**, 2119–2132 (2012). Selected papers from the 2011 Joint Working IEEE/IFIP Conference on Software Architecture (WICSA 2011).
5. A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," CoRR. **abs/1607.00653** (2016).

APPENDIX

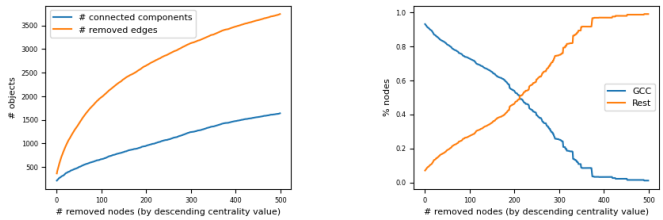

Fig. 4. Robustness Analysis of Overall Graph

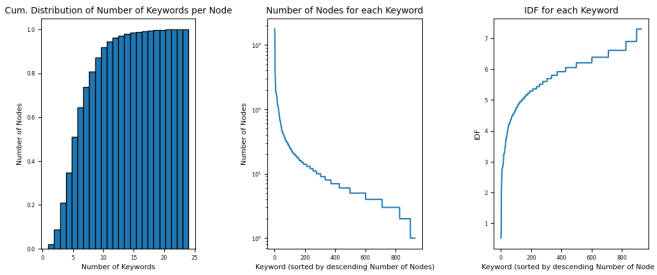

Fig. 5. Sparsity Analysis of Node Information Features

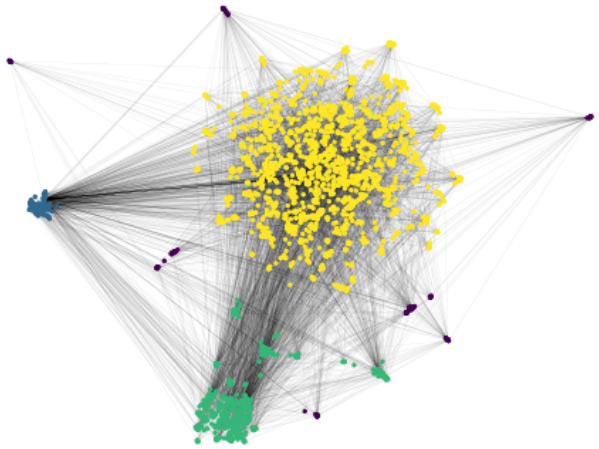

Fig. 6. Graph Clustering