NGSA Kaggle Report : Predicting missing links in a citation network

Brice Rauby, Fatma Moalla, Joel Mbouwe, and Joseph El Hachem
Team's name: **Joel&Fatma&Joseph&Brice**

CentraleSupélec, Gif sur Yvette 91190, France
{brice.rauby, joel-jores.mbouwe-nanmou, fatma.moalla, joseph.el-Hachem}@student.ecp.fr

1 Introduction

In this this Kaggle project, which is part of the NGSA course, we are given a citation network as a graph where nodes are research papers and edges represent the citation link, meaning that one of the two papers cite the other. Each node of the graph is also associated with information such as the title of the paper, publication year, author names and a short abstract. Edges were deleted at random from the citation network and our goal is to build a model that accurately reconstruct the initial network using graph-theoretical, textual, and the information on the nodes.

2 Feature Engineering

Given a pair of nodes (a, b) we create a total of 300 features describing the articles and that are significant for predicting the relation between two nodes. We used for this, the underlying citation network, the textual information and the meta-data characterizing each node.

2.1 Attributes of the nodes

We concatenate the original abstract and the title of the article and use the output as the real abstract of our article. We then use the NLTK library to stem each word of the abstract and we compute the TF-IDF of each abstract of the articles. For computing the TF-IDF we analyzed the distribution of the words of our corpus and we only consider the ones that appear at least 2 times and at most 5000 times. By using the obtained TF-idf and the other information we had, We manage to extract five features from a pair of nodes :

- The proportion of **common authors** between the articles a and b defined by $C_{authors} = \frac{|\mathcal{A}_a \cap \mathcal{A}_b|}{|\mathcal{A}_a \cup \mathcal{A}_b|}$
- The proportion of **common words** between the abstracts of the articles : $C_{words} = \frac{|\mathcal{W}_a \cap \mathcal{W}_b|}{|\mathcal{W}_a \cup \mathcal{W}_b|}$.
- The difference in **publication years** between a and b : $\mathcal{Y}_a - \mathcal{Y}_b$. One interesting point we noted about the publication year is that we had in our database pair of articles (a, b) (where a is supposed to cite b) for which the publication year of a was smaller than the one of b. Which is not quite logic. These couples represent 1.64% (2124) of the edges.
- The cosine similarity between the **TF-IDF representation** of the abstract of the pair : which is defined by the dot product of their tf-idf (since the tf-idf computed are normed by default). It measures how close the abstracts are.
- The cosine similarity between the abstract that we embedded in a **128 dimension space**. The embedding consists in training a new doc2vec [1] model that will be really specific to the contain of our abstracts. The advantage of a word embedding model over the TF-IDF representation lays in the fact that its Doc2vec can capture the context related to words.

2.2 Graph structure

We built, using NetworkX, a directed graph and we constructed about 11 features related to the structure of the graph that can characterize the similarity between two nodes.

- **Common neighbors** we compute the proportion of common neighbors between a and b as $C_{neighbors} = |\mathcal{N}_a \cap \mathcal{N}_b|$

- **Adamic Acar Index** which is defined as inverted sum of degrees of common neighbors for given two nodes : $c(a,b) = \sum_{x \in \mathcal{N}_a \cap \mathcal{N}_b} \frac{1}{\log(|\mathcal{N}_x|)}$
- **Jaccard Coefficient** defined as $J_{neighbors} = \frac{|\mathcal{N}_a \cap \mathcal{N}_b|}{|\mathcal{N}_a \cup \mathcal{N}_b|}$
- **Resource Allocation Index** $R(a,b) = \sum_{x \in \mathcal{N}_a \cap \mathcal{N}_b} \frac{1}{|\mathcal{N}_x|}$
- **Page Rank** of the target node. PageRank works by counting the number and quality of links to the target node to determine a rough estimate of how important the node is.
- **Hits coefficients**: Hyper-link induced topic search (HITS) identifies good authorities and hubs for a topic by assigning two numbers to a node : an authority and a hub weight. Authorities estimate the node value based on the incoming links. Hubs estimates the node value based on outgoing links.
- **Shortest Path** We remove the edges (a,b) (if it exists)from the graph and then compute the shortest path between the two nodes. (We then add again the edges to the graph)
- **Katz similarity** defined as $c(a,b) = \beta|\mathcal{P}_{a,b}^{<1>}| + \beta^2|\mathcal{P}_{a,b}^{<2>}|$ where $\mathcal{P}_{a,b}^{<l>}$ is the set of all paths of length l from a to b. We wanted to implement this similarity measure but due to time complexity, we finally drop the idea.

We add to these features, the **In and Out degree centrality** of the source and target along with their **preferential attachment** which is the product of the out-degree centrality of the source node and the in-degree centrality of the target. Therefore a pair of node (a,b) will be represented as the concatenation of the embedding of each node thus defining a vector of shape 256.

2.3 Node Embedding

Along with the previous features, we found interesting to evaluate the impact of learning continuous feature representations of the nodes. For this task, we implemented a **Node2Vec** [2] model which is quite similar to the Deep Walk algorithm but involves biased random walks. We embedded the nodes in a 128 dimension space by generating 25 random walks each walk being of length 50. The goal is to extract additional hidden features from the graph that represent the nodes.
Note: The features we built from the graph structure were highly correlated with the embedding we constructed using Node2Vec. We did not implement a dimensional reduction algorithm mainly because of the huge amount of data.

3 Model Tuning and Comparison

The global dataset was composed of 615512 pairs of nodes corresponding to 27700 nodes and 335130 edges. We split the dataset and applied various algorithms to the features we construct in order to adress the missing link problem.

3.1 Validation set construction

In order to have an accurate validation set, we remove from the original dataset about 15000 edges making sure, of course, that did not affect the structure of the graph, especially the preservation of the number of nodes. All the operations we did related to graph were not taking into account those 15000 edges. Ultimately, we train our model on 600512 pair of nodes and validate it on the remaining 15000 pairs that we initially removed.

3.2 The models

We implemented machine learning algorithms such as Logistic Regression with L-2 penalization, Support Vector Machine and eXtreme Gradient Boosting. Each of the parameters of the machine learning algorithms were optimized by cross-validated randomized-search over a parameter grid validation which allows us to find the optimal hyper parameter value while at the same time to prevent the model from over-fitting. Before applying a model, we made sure to standardize the data in order to stabilize the training process which improved the convergence speed. We also applied a PCA in order to adress the correlation issue between our feature issue but it didn't improve the performance of the models. Each pair of our dataset was composed of 5 features extracted from the information about the nodes, 11 features obtained with the graph structure and 256 features related to the Node2Vec model.

Logistic Regression & Support Vector Machine. In order to compare the features extracted from the graph structure and the ones from the attributes of the nodes, we train these two models with the two first group of features separately. Table 1 shows the results. The concatenation of the two groups of features improved the f1-score on the validation set.

Extreme Gradient Boosting outperforms the previous models but the training process along with finding the hyper parameters took a lot of time. Table 2 shows the future importance outputs by XGBOOST : the shortest path appears to be the most discriminant variable.

Neural Network along with the 3 machine learning algorithms, we tried a neural network which turned out to be the model with the best performance. We stacked three layers of shape respectively 512, 32 and 8, each layer was followed by a Rectified Linear Unit as the activation function along with a Dropout layer. The Dropout prevent the neural network from over-fitting. We trained the neural network only on the concatenated group of features.

Ultimately we concatenate all three groups of features and train an updated neural network. This models score **99.94 %** in terms of f1-score on our validation set, and **99.29%** on the public part of the leaderboard. Table 1 summarize the performance in terms of f1-score of all the models that we implemented so far.

Table 1. Performance of the tested models

MODEL	F1-SCORE
update NN + all features	**0.99957**
NN + node2vec features	0.999333
NN + all features	0.998197
NN + graph & node features	0.974219
xgboost + graph & node features	0.972251
xgboost +graph features	0.961866
logistic + graph & node features	0.957320
svm + node features	0.955614
logistic + graph features	0.921562
xgboost + node features	0.921406
svm + graph features	0.914656
logistic + node features	0.877750

Table 2. Future Importance

FEATURE	IMPORTANCE
shortest path	100.000000
common neighs	42.303875
resource	39.663643
adamic	12.495809
jaccard	8.823529
tf idf cosine	6.119644
common authors	3.851392
diff year	3.079254
page rank	1.465281
pref att	1.279755
authority	1.108819
hub score	1.108503
in deg	1.038362
common words	1.020769
out deg	0.863967
doc2vec cosine	0.685441

4 Conclusion and Perspectives

This project introduces us in an active field of graph analysis which is link prediction. We have created a lot of features by taking into account both the underlying graph structure and the information about the nodes. We have trained and validated several models along with the optimization of their hyper parameters and preventing them from over-fitting. We are a bit disappointed that our models weren't able to achieve an accurate performance. However the project was quite interesting. We learned a lot of things.

References

1. Quoc Le, Tomas Mikolov: Distributed Representations of Sentences and Documents https://arxiv.org/pdf/1405.4053.pdf
2. Grover, Aditya and Leskovec, Jure : Scalable Feature Learning for Networks https://cs.stanford.edu/ jure/pubs/node2vec-kdd16.pdf