Assael Jeremi
Auriau Vincent
d'Esquerre Arnaud

13/01/19
Team name: « Wewereonabreak"

Assignment 2: Report

Context

Edges have been deleted at random from a citation network. Our mission is to accurately reconstruct the initial network using graph-theoretical, textual, and other information.

In this competition, we define a citation network as a graph where nodes are research papers and there is an edge between two nodes if one of the two papers cite the other.

1. Feature engineering

1.1. Features from public baseline

- Number of overlapping words in title
- Temporal distance between the papers
- Number of common authors

1.2. Features from class and research papers [1][2][3]

1.2.1. Neighborhood-based methods

- Number of common neighbors: Papers citing the same paper are usually about the same subject
- Jaccard coefficient: The probability that both x and y have common neighbors.
- Preferential attachment: Based on the intuition that the probability that a new edge has node x as its endpoint is proportional to the number of neighbors
- Adamic Adar: Assigns large weights to common neighbors z of x and y which themselves have few neighbors (weight rare features more heavily)

1.2.2. Proximity-based methods

- Shortest path length: You will most probably read and cite papers that are cited in papers you cite. The closer the paper the higher the probability of citing it.
- PageRank, feature based on random walks in the graph.

1.3. Other graph features

These features give additional information of the structure of the graph and the trend on some of its nodes to cluster together.

- Resource allocation: $\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{|\Gamma(\mathrm{w})|}$ where $\Gamma(u)$ denotes the set of neighbors
- Node triangles for source and target nodes.
- Node degree for source and target nodes.
- Clustering index for source and target nodes.

1.4. Text features [1]

The subject of a paper is one of the most important reasons of citing it. We tried to build features that would capture it.

- TFIDF of abstract: Compute similarity in term of words in the abstract
- Number of overlapping words in abstract

- Topic modeling: Using Latent Dirichlet Allocation [4] we tried to get topics out of the titles. Here are the found topic and importance:

(0, '0.062*"quantum" + 0.026*"integr" + 0.023*"brane" + 0.021*"cosmolog"')
(1, '0.031*"effect" + 0.027*" action" + 0.026*"solut" + 0.025*"model"')
(2, '0.050*"black" + 0.047*"hole" + 0.022*"string" + 0.020*"dualiti"')
(3, '0.038*"symmetri" + 0.033*"model" + 0.022*"supersymmetri" + 0.022*"group"')
(4, '0.114*"theori" + 0.050*"n" + 0.046*"2" + 0.045*"gaug"')

1.5. Feature selection

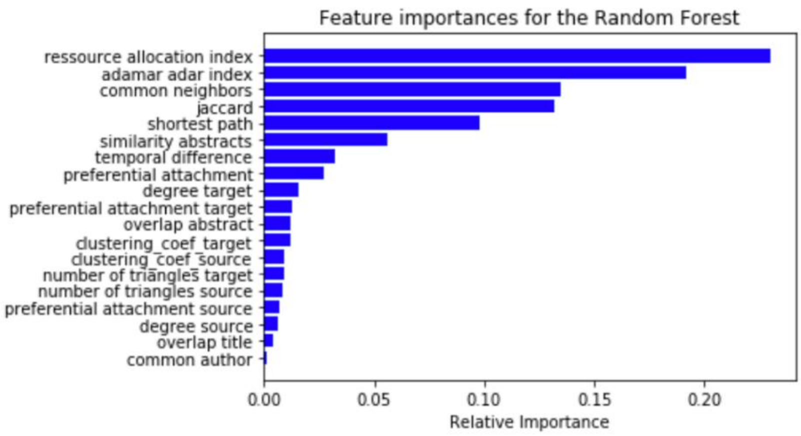

Figure 1: Feature importance for the Random Forest

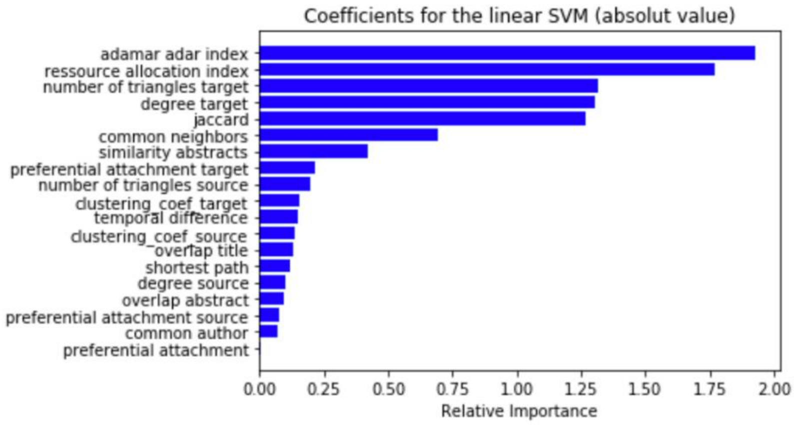

Figure 2: Coefficient for the linear SVM (absolut value)

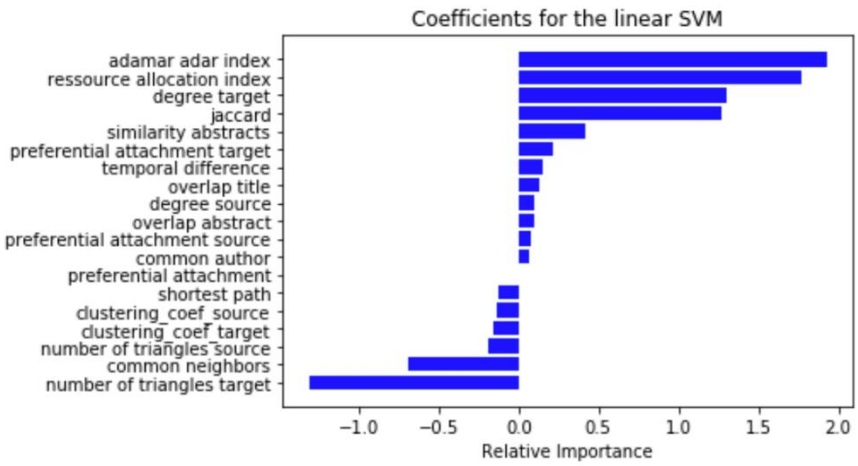

Figure 3: Coefficient for the linear SVM

Classifier	Linear svm	Linear svm	Random Forest	Random Forest
Number of features	19	5	19	5
Performance	0.9687	0.9527	0.972	0.953
Training time	3.59	3.18	21.8	7.9
Predict time	0.968	0.952	0.16	0.14

Table 1: Performance comparison of different models according to the number of used features

The five features used to get these scores are the five most important features given on the above figures.

Dividing the number of features by 4 just slightly harms prediction (2% F1 score), and can divide training time by 3 for the RF.

2. Model tuning and comparison

2.1. Classifier comparison [1]

Classifier	Linear SVM	Logistic Regression	Random Forest	Adaboost	XGboost
F1 score on validation set	0.9653	0.9664	0.9721	0.9586	0.9642

Table 2: Performance comparison of different models using F1 score

The model that provided us with the best results is the Random Forest. It is the one used for the predictions on Kaggle.

2.2. Model tuning

We performed a random search for the most performing classifiers with sklearn.RandomizedSearchCV. It helped finding the best hyperparameters and crossvalidation prevented overfitting. Here is the grid for the random forest.

Feature	Grid	Best
Bootstrap	True, False	True
Max_depth	10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None	None
Max_features	Auto, sqrt	Auto
Min_sample_leaf	1,2,4	2
Min_samples_split	2,5,10	2
N_estimators	200,400,600,…,2000	1800

Table 3: Grid of the random search for hyperparameter tuning of the Random Forest model

References

[1] Mohammad Al Hasan, Vineet Chaoji, Saeed Salem, Mohammed Zaki, *Link Prediction using Supervised Learning*, https://archive.siam.org/meetings/sdm06/workproceed/Link%20Analysis/12.pdf
[2] William Cukierski, Benjamin Hamner, Bo Yang, *Graph-based Features for Supervised Link Prediction*, https://ieeexplore.ieee.org/document/6033365
[3] Saoussen Aouay, Salma Jamoussi, Faiez Gargouri, *Feature based link prediction*, https://ieeexplore.ieee.org/document/7073243
[4] David M. Blei, Andrew Y. Ng , Michael I. Jordan, *Latent Dirichlet Allocation*,, http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf