LINK PREDICTION IN THE FRENCH WEBGRAPH

INF554 Data Challenge - Team A.P.J.

30 December 2019

Julie Dessaint: julie.dessaint@polytechnique.edu
Arnaud Sarfati: arnaud.sarfati@polytechnique.edu
Pierre Fernandez: pierre.fernandez@polytechnique.edu

1. Feature engineering

NetworkX Features

We used five different node descriptive metrics from networkx: Jaccard cefficient, preferential attachment score, resource allocation index, adamic adar index and page rank. A more detailed description of these features can be found in the appendix. Those features have been used in the classifiers where we have been able to see their importance and keep only the major ones. As an example, using only these five features in a random forest classifier gave a F1 score of approximately 0.88 on the test set and using only Jaccard coefficient and preferential attachment gave F1 scores of 0.899 with a random forest, 0.897 with gradient boosting and 0.837 for logistic regression (on a validation dataset).

Text embedding with Doc2Vec

While all the previous features concerned graph related notions, text embedding aimed to capture the information given by the text documents on each website. The solution that we used came from the library gensim that implements 2 main algorithms : Word2Vec [3] and Doc2Vec [4] that encapsulate different relations between words, like synonyms, antonyms, or analogies. The Doc2Vec model allows to train a vector for each document of the training set. The created vectors for each documents have length 20.

Before training the models, we decided to preprocess the data a little in order to have more accurate relations between documents. Especially, the preprocessing removes english and french stop words.

The text embedding allows to generate a similarity coefficient that has been used in classifiers with some improvement (cf. Figure 3). However, when the document vector is used in the neural network classifier it significantly improves the F1 score of the prediction (0.92508 instead of 0.91394).

Node embedding and neural network probabilities

In a similar way, graph's nodes could be embedded onto a lower dimension vector. To do so, we found the node2vec algorithm depicted in [5]. It creates a mapping of nodes to a low-dimensional space of features that maximises the likelihood of preserving network neighborhoods of node by simulating biased random walks. Therefore the model depends on how many walks one takes, on how long they are and if one considers the graph as a directed one. We have tried different models based on the previous statement, and we kept the one where best results were obtained.

The created vectors have length 64 (length found in the documentation) and have been used as input in the neural network. That led, alone, to a good F1 Score (0.91394). We also used this embedding in an indirect manner by using the last layer output of the neural network as a feature per se (the probability output of the neural network). Adding this feature to the different classifiers (e.g. logistic regression, random forest) lead to significant improvement (cf. Figure 3).

Features from Regression Models

We also added the prediction made by a gradient boosting regressor on the dataset containing the NetworkX features, as well as a Doc2Vec similarity coefficient. Instead of using the Gradient Boosting directly as a classifier (which we will do later), we first use it as a regressor to have somekind of weighted vote.

A heatmap representing the correlation between all the features presented in this section can be found in the appendix.

2. Model tuning and comparison

Training dataset

The first note we can make concerns the training dataset. At the beginning we were training our classifiers on all the given dataset. However we realised that it contained 62,5% of 1 and we feared that it would make our classifiers predict more 1 than 0. Thus we did the same with a restricted dataset containing as many 1 as 0 (created by randomly removing lines where there was a link in the training set), but the results were not as good as before, and the number of predicted 1 was not decreased as mush as we thought, so we dropped this lead.

Neural Network

One of the classifier we used is a simple neural network that gave as output a probability (the architecture is in the appendix). It can also predict classes (0 or 1). It takes as input the mapping of 2 nodes obtained thanks to the concatenation of the doc2vec and the node2vec outputs for a node. To avoid overfitting on that model we followed 2 procedures. The first one was to use Dropout(0.2) layers. The resulting F1 score was not as the good as previous ones on Kaggle (0.87 vs 0.92) so we decided to proceed otherwise and used a callback during the training of the model (scores were maintained).

Support Vector Machine

We used sklearn.svm.SVC to implement our model of the support vector machine algorithm. First, Support Vector Machine Algorithm's are not scale invariant, we have therefore centred and reduced all our variables using scikit learn standard scaler.

The first parameter to choose, and the most important one, was the choice of the kernel. We used cross-validation on five diferent train/test set to choose between linear, Gaussian, polynomial and sigmoid kernels. We used the F1-score and confusion matrix to select the best kernel for our dataset. We finally choose a linear kernel. Still using cross validation, we selected value 1.0 for the regularisation parameter C.

Finally, the average F1 score we got with the support vector machine model was around 0.89 which was far lower than what we obtained with Random Forest. Thus, we decided not to use this classifier in our final model.

Final Classifiers

We used three types of **classifiers** as a final tool to predict the existence of links in the webgraph: **Logistic Regression**, **Gradient Boosting** and **Random Forest** .

We have evaluated the performance of these three classifiers, by training them on an increasing number of the features presented in the first part of this report.

Parameters tuning and preventing overfitting We have focused on the optimisation of the Random Forest Classifier and the Gradient Boosting Classifier.

To do so, we used a Randomnized Search Cross Validation followed by a Grid Search Cross Validation. Consider the optimisation of the Random Forest Classifier. Among the various parameters it involves, we focused on the following:

- $n_{estimators}$: number of trees in the forest
- *max depth*: maximum depth of a tree
- *min samples split*: minimum number of samples required to split an internal node
- *min samples leaf*: minimum number of samples required to be at a leaf node
- *max features*: number of features to consider when finding out the best split
- *bootstrap*: boolean indicating whether the whole dataset should be used to build each tree

To find an approximation of the optimal parameters, we created a grid containing several values for each parameter above, trying to span the largest range they could be in. The Randomnized Search Cross Validation algorithm enabled us to go through some of the combinations of the grid (not all of them), returning the best it found.
From this combination, we created another grid exhibiting small variations in each parameter (i.e if RSCV gave 8 as max depth, we would create a grid with values 6,8,10 for max depth). We then applied the Grid Search Cross Validation algorithm which enabled to go through every possible parameters combination of the new grid, and to return the best parameterization.

Regarding overfitting, this method allowed us to prevent this effect by choosing the optimal number of trees in the random forest, a very impactful parameter to prevent overfitting for this classifier.

We used the same method to fine tune parameters of the gradient boosting classifier.

Performances comparison To compare performances, we have determined the F1 score for each classifier, while incrementing the number of features in the dataset used to train them (i.e we start with the Jaccard coefficient only, and add the other features one by one). The result is shown on Figure 3 in the Appendix.
We can observe that the Random Forest and the Gradient Boosting Classifier produce exactly the same result in terms of F1 score (about 0.98 at most). The Logistic Regression model is slightly less effective, especially when the number of features is low (i.e when we only have the Jaccard and preferential attachment coefficients). It is very interesting to observe which coefficients produce the most significant improvements in the prediction. We can see that the preferential attachment, the doc2vec similarity, and the source PageRank have some impact on the F1 Score, while the target PageRank for instant does not produce any improvement. However, the most significant impact comes from the predictions of the Gradient Boosting Regressor and the Neural Network. When displaying the relative importance of each feature (Figure 4 in the Appendix), this observation is confirmed as we see that the two predictions weight a lot in the decision of the random forest/gradient boosting.

3. Appendix

Bibliography

[1] Dong L., Li Y., Yin H., Le H., Rui M. "The Algorithm of Link Prediction on Social Network," Mathematical Problems in Engineering, vol. 2013, Article ID 125123, 7 pages, 2013. https://doi.org/10.1155/2013/125123.

[2] Gao F., Musial K., Cooper C., Tsoka S. "Link Prediction Methods and Their Accuracy for Different Social Networks and Network Metrics," Scientific Programming, vol. 2015, Article ID 172879, 13 pages, 2015. https://doi.org/10.1155/2015/172879.

[3] Mikolov T., Chen K., Corrado G., Dean J. Efficient Estimation of Word Representations inVector Space. 2013. https://arxiv.org/pdf/1301.3781.pdf

[4] Le Q., Mikolov T. Distributed Representations of Sentences and Documents, Google Inc, 2014. https://arxiv.org/pdf/1405.4053.pdf

[5] Grover A., Leskovec J. node2vec: Scalable Feature Learning for Networks. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016. https://arxiv.org/pdf/1607.00653.pdf

[6] Liben-Nowell D., Kleinberg J. The Link Prediction Problem for Social Networks, January 8, 2004. http://www.cs.cornell.edu/home/kleinber/link-pred.pdf

Networkx features

Here are some common features used for link prediction [1,2]. In all the formula below $\Gamma(x)$ denotes the set of neighbors of x.

- **Jaccard Coefficient** : $s_{xy} = \dfrac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$.

  The Jaccard coefficient[6] measures the probability that both x and y have a common neighbor for a randomly selected neighbor v that either x or y has.

- **Adamic Adar Index** : $s_{xy} = \sum_{u \in \Gamma(x) \cap \Gamma(y)} \dfrac{1}{\log |\Gamma(u)|}$.

  Adamic Adar Index [6] is strongly related to Jaccard coefficient as it is also based on the number of common neighbors between two nodes. We can see in the heatmap of the correlation between the features (Appendix, Figure 2) that Adamic Adar Index and Jaccard coefficient are highly correlated.

- **Preferential Attachment Score** : $s_{xy} = |\Gamma(x)||\Gamma(y)|$.

  Preferential attachment [6] models the growth of networks. The idea behind this score is that the probability that a new edge involves node x is proportional to the current number of neighbors of x.

- **Resource Allocation Index** : $s_{xy} = \sum_{u \in \Gamma(x) \cap \Gamma(y)} \dfrac{1}{|\Gamma(u)|}$

- **Page Rank** : The ranking of the node in the graph based on the structure of the incoming links.

Figures

Layer (type)	Output Shape	Param #
flatten_20 (Flatten)	(None, 168)	0
dense_58 (Dense)	(None, 32)	5408
dense_59 (Dense)	(None, 10)	330
dense_60 (Dense)	(None, 1)	11

Total params: 5,749
Trainable params: 5,749
Non-trainable params: 0

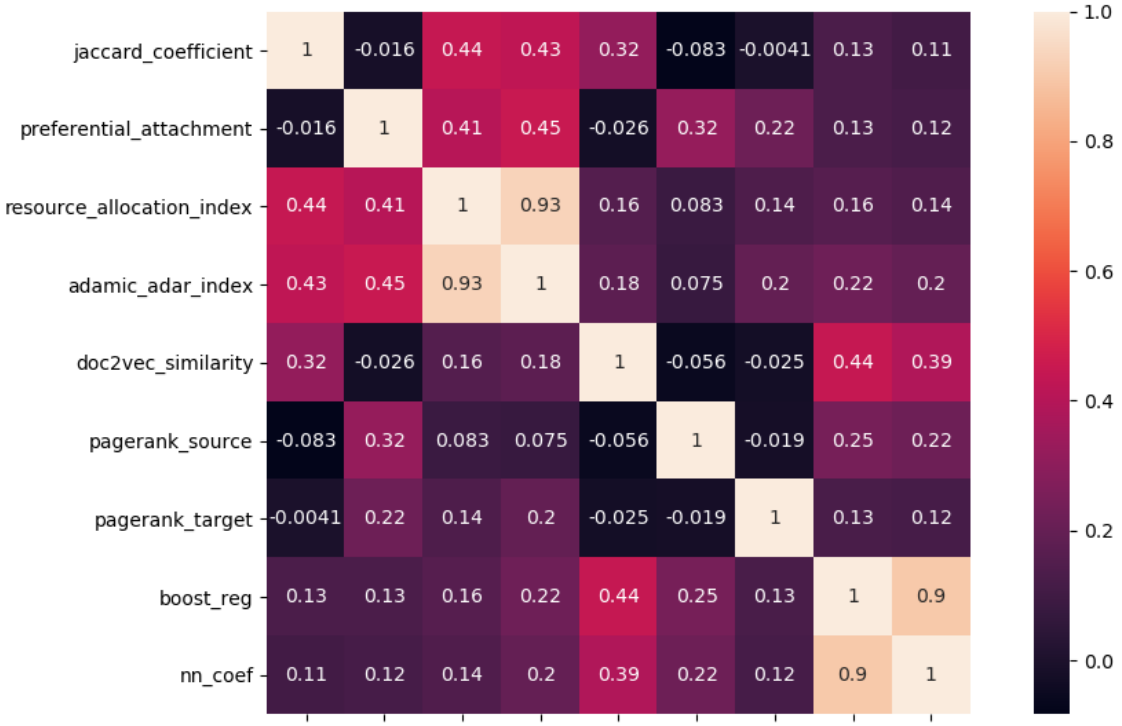

Figure 2: Correlation matrix of features

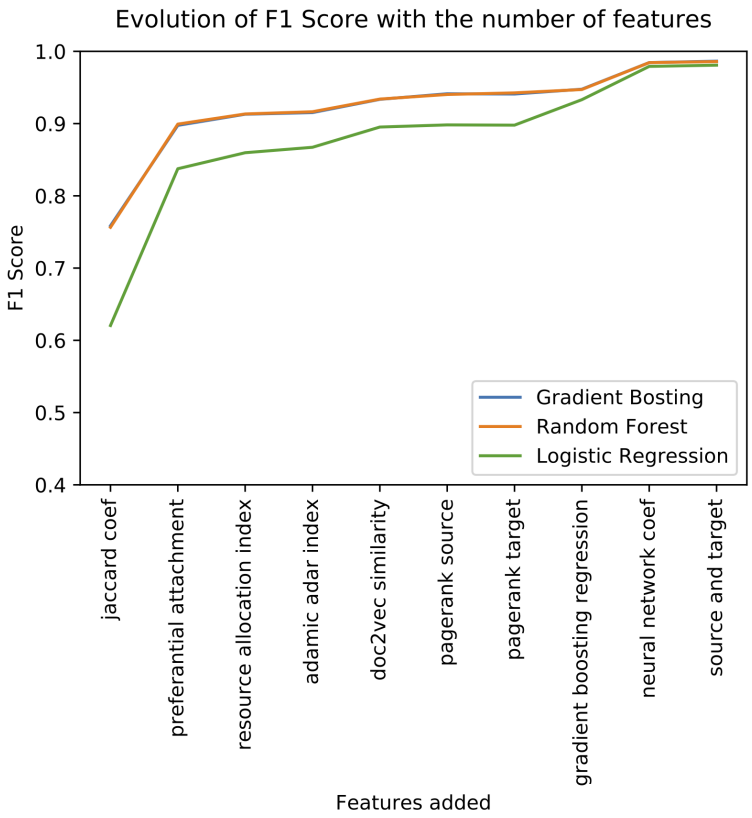

Figure 3: Evolution of F1 score when features are added one by one

```
M↓
RF.feature_importances_

array([0.00403487, 0.08820349, 0.0378141 , 0.01634349, 0.01037726,
       0.09843598, 0.00174762, 0.26814368, 0.47489952])

M↓
boost_reg.feature_importances_

array([0.01003721, 0.1002796 , 0.02805366, 0.00515838, 0.01300984,
       0.03714958, 0.00196876, 0.20760874, 0.59673423])
```

Figure 4: Arrays representing the features' importance in Random Forest (top) and Gradient Boosting Classifiers (bottom). The features are in the following order: (Jaccard Coefficient, Preferential Attachment, Resource Allocation, Adamic Adar Index, Doc2Vec Similarity, Source PageRank, Target PageRank, Gradient Boosting Regressor prediction, Neural Network prediciton)