NGSA: Network Science Analytics

Assignment 2 - Team: *Benayada_Debbagh_Hadji_Mikou*

Mohamed Aymane Benayada
Ecole Centrale Paris
mohamed-aymane.benayada@student.ecp.fr

Driss Debbagh Nour
Ecole Centrale Paris
driss.debbagh-nour@student.ecp.fr

Soufiane Hadji
Ecole Centrale Paris
soufiane.hadji@student.ecp.fr

Mehdi Mikou
Ecole Centrale Paris
mehdi.mikou@student.ecp.fr

.

DESCRIPTION

The goal of this assignment is to predict missing links in a citation network of research articles. A citation network is represented as a graph G = (V, E), where the nodes correspond to scientific articles and the existence of a directed edge between nodes u and v, indicates that paper u cites paper v. Each node (i.e., article) is also associated with information such as the title of the paper, publication year, author names and a short abstract. A number of edges have been randomly removed from the original citation network.

Your goal is to accurately reconstruct the initial network using graph-theoretical and textual features, and possibly other information. Your solution can be based on supervised or unsupervised techniques for link prediction or on a combination of both. You should aim for the maximum F1 score.

This report will follow the structure of our work: we will begin with a quick overview of the preprocessing tasks and have a look at the features created and selected for the prediction model, and finish with the prediction model choice itself.

FEATURE ENGINEERING

Preprocessing

From the csv and text input files, we built a list of IDs, nodes and edges, and create a graph. We also extract our training/test sets, and the labels in a machine learning oriented format. We first computed all features for the whole dataset, but ended up with worse prediction results than with a reduced dataset (basically 5 or 10%), which was surprising since more data should bring more generalization capacity to the model. Therefore we decided to work with 5 % of the training set.

Moving through our training set, for each node pair (source, target), we had the label (1 if they were connected, 0 otherwise). We computed a certain number of features for each one of these couples, to build a design matrix for our prediction task. Different types of features were generated: NLP features and metadata-based features, as well as graph theory based features. This is motivated by the fact our network is a citation network, therefore analyzing the paper description can bring a lot of information on the potential links, the graph features are more related to the structure of the potential connections between papers. Metadata can be a very useful complement to these as it brings specific information that can be used to predict similarity between papers. Feature description is the subject of the next section.

NLP features

- Overlapping title - $overlap\_title$ **:** Very intuitive feature, we count the number of common words in the titles of the papers in the (source, target) pair.

- Temporal difference - $temp\_diff$ **:** We compute the number of years separating both paper publications.

- Number of common authors - $comm\_auth$ **:** We count the number of common authors to both papers. Obviously, two papers with common authors are very likely to cite each other, or at least for one to be cited by the other.

- Number of common words in description - $overlap\_desc$ **:** We count the number of common words in the description of the papers. This is also a very natural feature to compute.

- Number of authors - $number\_author\_...$ **:** We count the number of author for each node of the (source, target) pair.

- Difference in number of words in title, and description - $..._diff_count$ **:** We count the number of words in the title and the description for the source and the target, then compute the difference. This type of feature are usual NLP features we found by reading articles and blog posts (see references section).

- Difference in number of characters in description $desc\_diff\_charac\_count$

- Difference in average word length in the title, and description - $..._diff_average_length$ **:** We list the length of words in the title (resp. description), and deduce the average word length.

- Difference in number of stopwords in description $desc\_diff\_stop\_words$

Graph features

Directed graph.

- **Page Rank** - *page_rank* **:** this features establishes a ranking of the nodes in the graph based on the structure of the incoming links.

- **Centrality** - *centrality* **:** Computes the degreee centrality of nodes.

- **Number of common successors and predecessors :**
comm_successors & comm_predecessors
In a directed graph, for each node, we list the nodes pointing to the node (predecessors) and the ones this node points at (successors). We then count for each pair (source, target) the number of commons predecessors, and the number of successors. The intuitive idea behind this feature is that two papers are similar if they cite the same papers or are cited by the same papers.

- **In-degree and out-degree :** In the same spirit, we get the in-degree and the out-degree of both the source and the target in the (source, target) pair. The in-degree represents the number of edges incoming to a vertex (number of papers citing the paper). The out-degree represents the number of edges outgoing from a vertex (number of papers cited by the paper). This gives an idea of the importance of a paper in the network. For example, a high out-degree node and a high in-degree node are likely to be associated, while a low out-degree node and low in-degree node are much less likely to be.

Undirected graph.

- **Jaccard's coefficient:** it is a similarity metric that is commonly used in information retrieval. It measures the probability that both x and y have a feature f, for a randomly selected feature f that either x or y has. If we take "features" here to be articles cited, then this measure captures the intuitively appealing notion that the proportion of the papers cited by x who are also cited by y (and vice versa) is a good measure of the similarity of x and y.

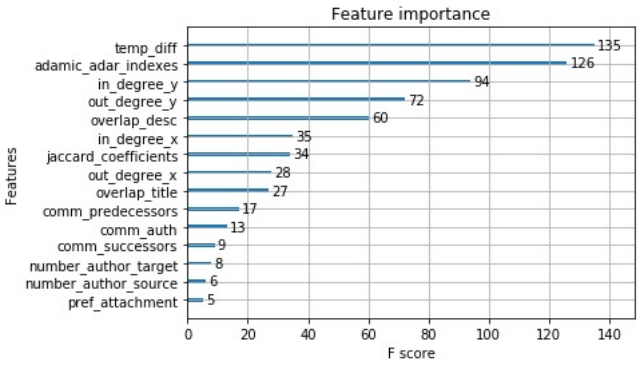
- **Adamic/Adar index:** this measure refines the simple counting of common features by weighting rarer features more heavily. The Adamic/Adar predictor formalizes the intuitive notion that rare features are more telling; papers that share the phrase "for example" are probably less similar than documents that share the phrase "clustering coefficient."

- **Preferential attachment:** it is based on the concept that papers with many citations tend to create more connections in the future. This is due to the fact that the rich get richer. We estimate how "rich" our two vertices are by calculating the multiplication between the number of citations each vertex has.

MODEL TUNING AND COMPARISON

We generate these features for the test set as well, and scale them.

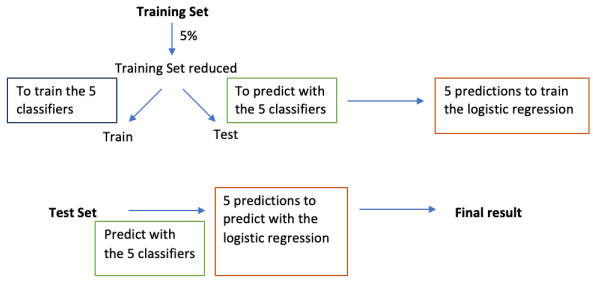
Feature selection : we want to have an idea of the importance of these new features for the prediction (and therefore their contribution to the F1 score). We fit a XGBoost classifier using all the features we have with our training set and the labels, and plot the feature importance histogram for the 15 most important features only :



We decided to keep the 11 most important features.

Methodology : for the prediction, we decided to use five classifiers which input the features created and outputs the predicted labels, and then a logistic regression that inputs the five prediction vectors, and outputs one final prediction vector.

Therefore, we need to train/test our five classifiers, but also our logistic regressor. To do so, we separate our training set into a training subset and a test subset. Using the training subset from the initial training set, we train our five classifiers. Using the test subset from the initial training set, we predict labels with our five classifiers. Thus we generate the training data of our logistic regressor. Once the logistic regressor trained, we can do the final prediction: we feed the five classifiers with the initial test set, predict five label vectors which we use to predict with our logistic regressor the final label vector.



The 5 prediction models used are the following:

- Model 1: XGBClassifier on all features
- Model 2: XGBClassifier on selected features
- Model 3: XGBoost with chosen parameters
- Model 4: Random Forest Classifier
- Model 5: Random Forest Classifier Selected Features

Model selection : we decided to use XGBoost and Random Forest classifiers they are known to be the most efficient for classification Kaggle tasks. This is a very empirical choice based on experience and the good results (on the partial test set) justify such a choice.

Parameter tuning and overfitting: For the parameter tuning, we tried several values and quickly found the chosen parameters. We decided to tackle overfitting by using the technique described earlier in the Methodology subsection, and learned in an online course (reference [5]). Our final prediction is the result of a logistic regression on 5 predictions from 5 different models with different parameters. This allows our prediction to be more general.

Results : we evaluate the model with the F1 score, and end up with a best score of 0.96714 on the public test set.

REFERENCES

[1] https://networkx.github.io/
[2] http://be.amazd.com/link-prediction/
[3] Shubham Jain, *Ultimate guide to deal with Text Data (using Python) - for Data Scientists Engineers*, Feb. 2018
[4] Hicham El Boukkouri, *Text Classification: The First Step Toward NLP Mastery*, Jun. 2018
[5] Stephane Mallat, Pierre Courtiol, *S'attaquer a une competition de machine learning : methodologie et exemples pratiques*, College de France
(https://www.college-de-france.fr/site/stephane-mallat/seminar-2018-02-21-11h15.htm)