INF554 Report: Link Prediction of a Citation Network

Stormtroopers
Guillaume Dufau (guillaume.dufau@polytechnique.edu)
Christopher Murray (christophe.murray@polytechnique.edu)
Léon Zheng (leon.zheng@polytechnique.edu)

January 2018

1 Introduction

Reconstruction of citation networks is a common example of link prediction which can be approached with various types of machine classification models. This challenge is a **supervised classification problem** since there is already a large corpus of citations of this network. The objective is then to predict if one paper from the network is cited by another one. To solve this classification problem, we firstly constructed features derived from **contextual information** of the papers as well as **topological information** from the citation graph, which can all meaningfully describe each pair of nodes. Then, we used them to trained various classification models to predict whether or not a given pair of nodes are connected or not.

2 Feature engineering

The feature engineering for our model focused primarily on the *contextual features* of the papers (extracting and comparing the meaning of abstracts, difference in publishing year, etc.), and the *topological features* of the citation network (in-degree, out-degree, Jaccard coefficient, etc.).

2.1 Contextual features

Since the data contained relational information of the graph (node connectivity), as well as a lot of node information (authors, title, year published, abstract, and journal), the first features defined for our classification models were built upon the provided node information.

Our first task was to understand text data. After removing stop-words, the simplest features that can be built are the number of co-occurring words in the abstracts, titles, whether or not the papers were published in the same journal, the number of shared authors, and the difference in publishing years. Therefore, the contextual features used for the project are the following.

Same-authors Number of authors who worked on both papers.

Co-occurrence-abstract Number of words appearing in both abstracts (stop words removed).

Co-occurrence-title Number of words appearing in both titles (stop words removed).

Same-journal Binary value, being true if both papers were published in the same journal.

Years-difference Difference between both publication years (can be negative).

TF-IDF Cosine Similarity The cosine similarity between the papers' TF-IDF values.

Since the co-occurrence information is based on absolute frequencies there might be some terms diminishing the relevance of other terms because they occur more. The TF-IDF model (Term Frequency-Inverse Document Frequency) uses scaling and normalizing in its computation to solve this issue.

Let tf_t be the term frequency, number of times term t appeared in the document divided by N the number of terms in the document, and df_t, the number of document with term t in it. Then $tfidf_t$ is defined as:

$$tfidf_t = tf_t * idf_t \text{ , where } idf_t = log\frac{N}{df_t} \tag{1}$$

2.2 Topological features

We can also characterize each pair of papers by their position in the citation network, which is modeled by a directional graph with the vertices being research papers, and the oriented edges demonstrate which papers cite another. Intuitively, the citation network might present several clusters, where each cluster has strongly connected components; each cluster would then correspond to a certain research domain, or topic. Then, if two vertices in the graph are close in the way that they share a similar neighborhood, they might represent papers that are in the same domain, and thus have a higher chance to be cited by the other.

Therefore, we used several popular heuristics to construct those graph features, mainly from [1]. We note (u, v) a pair of nodes, and define the following topological features.

Degree features In-degrees and out-degrees of each node: $d_{in}(u)$, $d_{out}(u)$, $d_{in}(v)$, $d_{out}(v)$.

Triads features Number of partial triads that u and v form with common neighbors w. There are four types of triads, just as we can see in table 1. The features are then the number of the different types of partial triads for the pair (u, v), t_1, t_2, t_3, t_4, and their frequencies, $\frac{t_1}{C(u,v)}$, $\frac{t_2}{C(u,v)}$, $\frac{t_3}{C(u,v)}$, $\frac{t_4}{C(u,v)}$, where $C(u, v)$ is the number of common neighbors of u and v.

$u \to w \to v$	$u \to w \leftarrow v$	$u \to w \leftarrow v$	$u \leftarrow w \leftarrow v$
t_1	t_2	t_3	t_4

Table 1: The four triad types that u and v can form with a common neighbor w

Common neighbours Number of common neighbors of u and v, $|\Gamma(u) \cap \Gamma(v)|$ where $\Gamma(\cdot)$ is the set of neighbors connected to a vertex.

Adamic-Adar coefficient Frequency-weighted common neighbors: $\sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{log|\Gamma(w)|}$

Jaccard coefficient Proportion of common "friends" over total "friends": $\frac{|\Gamma(u) \cap \Gamma(v)|}{|\Gamma(u) \cup \Gamma(v)|}$

Preferential attachment coefficient Measure of how "rich" is the neighborhood of u and v: $(|\Gamma(u)| \cdot |\Gamma(v)|)$

Katz score Measure of the similarity of two nodes by counting the number of short and long path that connect the two nodes. The score is high when there are a lot of short path connecting the two vertices: $\sum_{l=1}^{\infty} \beta^l |path_{u,v}^l|$ where $path_{u,v}^l$ is the set of paths that connect u and v and has a length l, and β a positive constant.

HITS Algorithm defining Hub and Authority value A paper considered as hub refers to many authority nodes in the graph $hub(i) = \sum_{i=1}^{n} auth(i)$. An authority paper is pointed by many hub nodes $auth(i) = \sum_{i=1}^{n} hub(i)$.

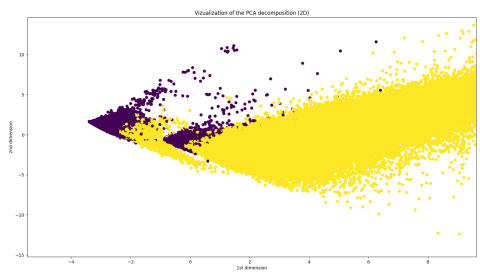
Page Rank Counting the number and quality of links to a paper (the sink), giving an idea of the importance of a paper based on the structure of the incoming links.

2.3 Data visualization

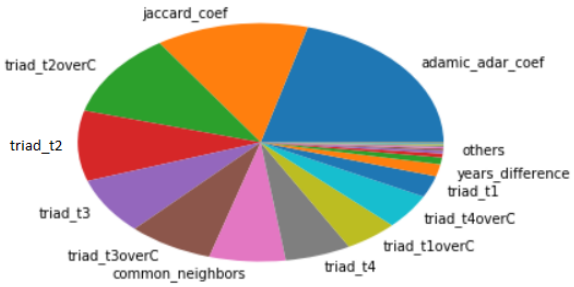
After constructing all the features for all pairs of nodes, we visualize the data set to check if those features are relevant enough to determine the existence of a citation link between the two papers. Since there are more than 20 features, we applied a PCA decomposition on the data set to get two main features that would allow us the visualize the data set on a 2D plot. The result is shown in figure 1a. We observe that data points are quite well separated according to their label, which means that the features we have chosen are able to discriminate, with limitations, which pair of papers have a link.

2.4 Features selection

For features selection, we used the *BestKFeatures* method from the *sklearn* toolbox with the *f_classif* function. The *f_classif* function uses the ANOVA F-scores to select which features to discard. An F-score is the ratio of the variance of the means of different classes over the variance of samples of a given class, and is used to test the validity of different classes. Using this function we can select the number of features we want to keep, and we observe for example that features like *same_journal* and *preferential_attachment_coefficient* are the first left apart. Despite this they still make a difference for the predictions so we keep them. An example of removed variable is *shortest_path* that was irrelevant for better generalization and was the first feature taken out thanks to this approach.



(a) Data visualization after PCA decomposition (label "0" in purple, label "1" in yellow).



(b) Feature importance according to RandomForestClassifier (max_depth=9)

Figure 1: Data visualization after PCA decomposition and features selection.

We also removed the Katz score because of its high computational cost and the fact that it didn't help the model to have better predictions.

According to the $feature\_selection$ option of the $RandomForestClassifier$ function our topological features tend to give better information. A future work might be to do more advanced NLP to grasp better knowledge from contextual information of the papers. Even if we did try to apply auto-encoders for example there's still many possibilities to look at. Features selection results are shown in Figure 1b. We observe that topological features are very important for this classification problem.

3 Model selection

3.1 Cross validation

In order to select the best possible model to predict the links, we used *cross-validation*. To grasp the effectiveness of the graph-based features we built the graph only on information we had from the training set. That is why we could only use hold-out cross validation, splitting 90% of the entire *training_set.txt* file as our training set, and the 10% left as our test set. It allows f1-score computation, our predicate error. It also helped us selecting the best hyper-parameters in the selected model. The table 2 shows the performance of different learning models from the *scikit-learn* library.

Models (best parameters if tested)	f1 score with the split
Linear SVM	0.922351
Logistic Regression	0.932125
K nearest neighbors (k=3)	0.889214
Naive Bayes	0.892664
Linear Discriminant Analysis	0.951406
Adaboost (Decision Tree based)	**0.961157**
XGBoost (See bellow for details)	**0.958008**
Multi-layer-perceptron (tolerance rate: 1e-5)	**0.971211**

Table 2: Comparison of the prediction score of different models (with the remaining 10% of the provided *training_set.txt*). The best model is the Multi-layer-perceptron.

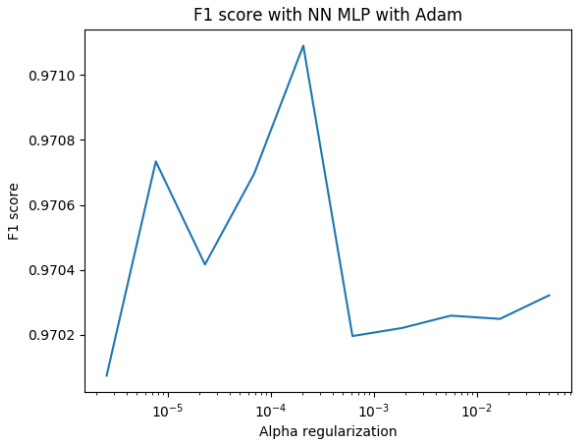
In the case of the XGboost, after much hyperparameters tuning, we ended with these parameters: $max\_depth = 9$; minimum positive effect on the loss function required to split the variable: $gamma = 3000$; $learning\_rate = 0.0001$; and some parameters for the XGboost library: $min\_child\_weight = 2$, $colsample\_bytree = 0.35$, $subsample = 0.8$, $reg\_alpha = 3$, $learning\_rate = 0.0001$.

3.2 Hyperparameters tuning for Multi-Layer Perceptron

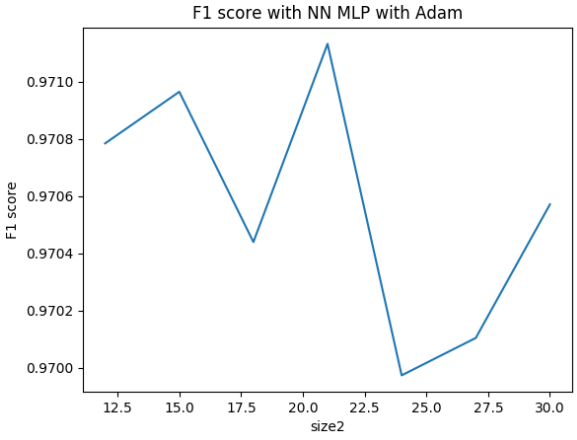
Then, we tried to find the best generalization possible, which means the best f1-score on our test set, by testing different important parameters in the *MLP scikit-learn* function. After many trials, we ended with a 2 hidden layers perceptron. Ending with a neural network of neurons sizes (25,65,21,1).

A ReLu activation function tend to give better scores than sigmoid, and since we're using many training example the Adam optimizer is perfectly fitted to this case. After several tries, our best tolerance rate is $5 \cdot 10^{-5}$.

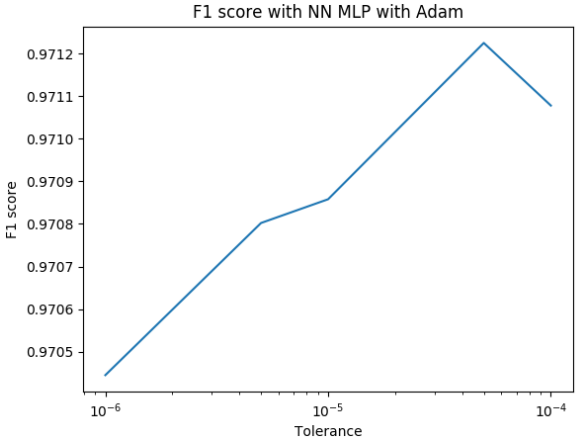
Similarly, the best regularization term alpha is pretty low, $\alpha = 1.74 \cdot 10^{-4}$. Examples of hyperparameters tuning graphs we used are shown in Figure 2.



(a) Tuning the α regularization term



(b) Tuning the second hidden layer size



(c) Tuning the tolerance rate

Figure 2: F1-score of Multi-layer perceptron classifiers with different hyperparameters. The best α regularization term to choose is $1.74 \cdot 10^{-4}$. With a fixed first hidden layer size of 65, the best size for the second hidden layer is 21. The best tolerance rate is $5 \cdot 10^{-5}$.

4 Conclusion

The final predictions submission on Kaggle have been obtained by constructing contextual and topological features, and using a **Multi-layer Perceptron** learning model with fine tuned hyperparameters. The final F1-score obtained for the testing set is **0.97427**.

To improve the model, one can find better features. For example, with autoencoders in [3], it is possible to construct more complex graph features so that models can take into account more precise information about the topological structure of the citation network. Autoencoders could also be used to do the greedy initialization of the MLP layers. It is also possible to use the recent SEAL framework [2] to build neural network that can learn by itself relevant graph features, while using explicit features like the contextual ones to consider the maximum information for the predictions.

References

[1] Wayne Lu Kyle Julian. Application of machine learning to link prediction. *Stanford University*, 2016.

[2] Yixin Chen Muhan Zhang. Link prediction based on graph neural networks. *Washington University in St. Louis*, 2018.

[3] Phi Vu Tran. Learning to make predictions on graphs with autoencoders. *Strategic Innovation Group, Booz-Allen-Hamilton, San Diego, CA USA*, 2018.