INF582 Data Challenge - Link Prediction in Citation Networks

Jacob Jonsson, Diego de Rochebouët, Dinara Veshchezerova

March 2018

1 Introduction

For this project, a citation network with removed links is presented, and the problem of identifying which node pairs had a link from a set of possible links needs to be solved.

To predict whether a suggested link was true or not, we had to design features based on textual information from the papers' titles and abstracts, graph-theoretical properties, and meta-data such as the year of publication. Further, we had to evaluate different models using the features we engineered.

The challenge was evaluated using the mean F1-score.

2 Feature engineering

2.1 Text Mining

2.1.1 Tf-idf

Abstracts tend to use a concise language and often try to use a language which makes the reader understand the scope of the content. By doing this, domain-specific terms are frequently used. For that reason, we assume that papers citing each other should share this specifics terms and that papers that do not share any domain-specific terms will probably not cite each other. This is exactly what the Tf-idf score between two abstract reflect, which is why it was computed and used.

2.1.2 Word Mover's Distance

The Word Mover's Distance (WMD) is a measure of the distance between two text documents defined by the minimum cumulative distance that all words from one document have to travel to exactly match the other, when mapped to a Word2Vec vector space.[2]

We used the WMD between abstract texts as one of the textual features. Our reasoning was that since abstracts are intended to capture the essence of papers, documents referring each other could be assumed to have a correlation between their semantic content, as opposed to only the exact terms used, as is the case for the tf-idf feature. The WMD has been proven effective for this purpose.

2.2 Graph-theoretical

2.2.1 Author citation graph

According to this article [1], it is useful to take into account some features that we can extract from a graph named author citation graph: "*... if an author X's papers have been cited by another author Y for k times in total, there is a directed edge from Y to X with weight k ...*".

This can be explained by the fact that people tend to cite papers written by authors that they have cited before. To add this feature, first we calculated the dictionary which contains the information about the number of citations from author_1 to author_2 for each pair of posible authors. And then, for each pair of papers, the average number of author citations between the authors of each paper was computed.

2.2.2 Degree of nodes

Another idea that we had was the fact that some papers should be more popular than others. If a paper is revolutionary in its field, it is probable to be cited a lot. This gave us the idea to compute the degree of every paper in the graph and use as a feature. Since we always have a pair of papers to compare, the degree of both papers was computed and used as two different features.

2.2.3 Number of shared neighbors

A third and last graph feature was the number of shared neighbors between the two papers. The reasoning is that if one paper is going to cite another, they are likely to cover related topics. If there then is a third paper on the same topic, then all three would be probable to be connected in the graph.

2.3 Feature selection

All the features were computed and as each feature was added, the score improved. To see which features were better, we computed using cross-validation and the LightGBM classifier the score we would get with each and every combination of feartures, as seen in the Figure 1. We saw here surprisingly that the most promising features, the author citation graph and the WMD, almost didn't add to the final score, with the simple features being the most important. We have to give particular remarks to the **number of shared neighbors**, by adding only this feature, the score can go up **from 0.814 to 0.966**. But none of the features reduced the score so the simple solution of using all the features was the one that prevailed.

3 Model tuning and comparison

3.1 Baseline: Linear SVM

A linear-kernel SVM was given, and was the classifier we used in the initial stage of the challenge, which was feature engineering. Using our two first graph-based features, the WMD feature and the author citations feature, we started to reach acceptable results with F1 scores at 0.9444, training on 5% of the data. However, when we tried using the same features with an LightGBM (LGBM) classifier, the result was immediately more than 0.02 better. For that reason, we didn't tune the linear kernel.

3.2 RBF-kernel SVM

Using an RBF-kernel, our best score was 0.9624 using the two first graph-based features and the author citations feature, and utilizing cross validation in the training, training on 5% of the data.

The model was tuned using grid search over six cross validation folds over the following parameters:

`C: [2**(-3), 2**(-1), 2**0, 2**2, 2**4, 2**6],`

γ: `[2**(-7), 2**(-5), 2**(-3), 2**(-1), 2**1]`

The optimal parameters, used for the reported score, were C = 16, $\gamma = 0.125$.

3.3 Random forests

The best score received using a random forest was 0.9709. All features and all data was used for the training, along with the following hyper-parameters: `'max_features': 'auto', 'n_estimators': 45, 'max_depth': 25, 'min_samples_leaf': 2`.

Using a ten-fold cross validation to reduce over-fitting, the grid of values the model was tuned on were:

```
'n_estimators': [9, 18, 27, 36, 45],
'max_features': ['auto', 'sqrt'],
'max_depth': [5, 10, 15, 20, 25],
'min_samples_leaf': [1, 2, 4]
```

3.4 LightGBM

The LGBM model was the most efficient stand-alone classifier, with a score of 0.9712, trained on the full data set with all features. The hyper-parameters used were: `'num_leaves': 127, 'reg_alpha': 0.5, 'max_depth': 8, 'min_data_in_leaf': 16`.

The model was tuned using a six-fold cross validation grid search over the following values:

```
'num_leaves': [10, 31, 127],
'min_data_in_leaf': [4,8,16,32,64],
'max_depth': [4,8,16,32,64],
'reg_alpha': [0, 0.1, 0.5]
```

3.5 Multi-layer perceptron

We evaluated the use of a neural network classifier by training a simple fully connected multi-layer perceptron. We decided to not do extensive tuning due to the number of alternatives, but instead tested locally with architectures using one or two hidden layers. The number of neurons in each layer was hand-tuned between 5 and 20, and the best local results were reached using two hidden layers, 12 neurons in each layer and optimized with an Adam optimizer. This model scored 0.9071.

3.6 Ensembles

We tested several ensemble methods. Using 3-6 identical RF or LGBM models with different random seeds, the result was slightly worse in local tests, and therefore we discarded that idea.

The best results overall, however, was reached using a combination of LGBM and RF. When we averaged the results from one tuned classifier of each sort, and trained using all features and all data, along with ten-fold cross validation to reduce overfitting, the score was 0.9713.

Notable is that we only averaged the probabilities from the classifiers in the ensembles. Vote-based approaches should also be explored, but we did not evaluate this.

3.7 Tweaks

A problem we encountered was the fact that the feature computation was really slow. Because of this, we first started by just using 5% of the data. However, this was later solved by saving the features computed with all the data to a file that was then loaded. This allowed us to test the models rapidly and saved us a lot of time.

To test our models locally, cross validation was used, to allow us to have a more general score and not specific to the subset of data we chose to train/test, which allowed us to trust those scores.

4 Conclusion

This project was really interesting and it made us learn a lot of text mining but also of machine learning in general. We now understand that feature engineering is the most important part of the challenge. A better model can make the score go up and tuning it can get the score slightly up, but most of the improvements and the real challenge come from choosing the correct features and testing them. Computing these features can mean heavy computations or a slow process in general, which is why it's a good idea to save them if they do not have to be recomputed.

For this project in particular, we saw that all the possible features need to be considered. Here, the graph features proved to be as important or even more important than the text features. this doesn't mean that this features where not important, on the contrary, it was thanks to the tf-idf that the score could get to the 0.97 mark, and it ended up being one of the most important features.

It would have been really interesting to try other features, in particular the ones that we saw in class, like using the graph of words, and we expect them to have improved our score had we had the time to do them. However, we will have to leave them for the next time we come across a text mining problem. And we know we will come across another text mining problem *because text is everywhere* and the sheer amount of text data available just keeps growing and growing with the need of processing it and understanding it.

5 Appendix

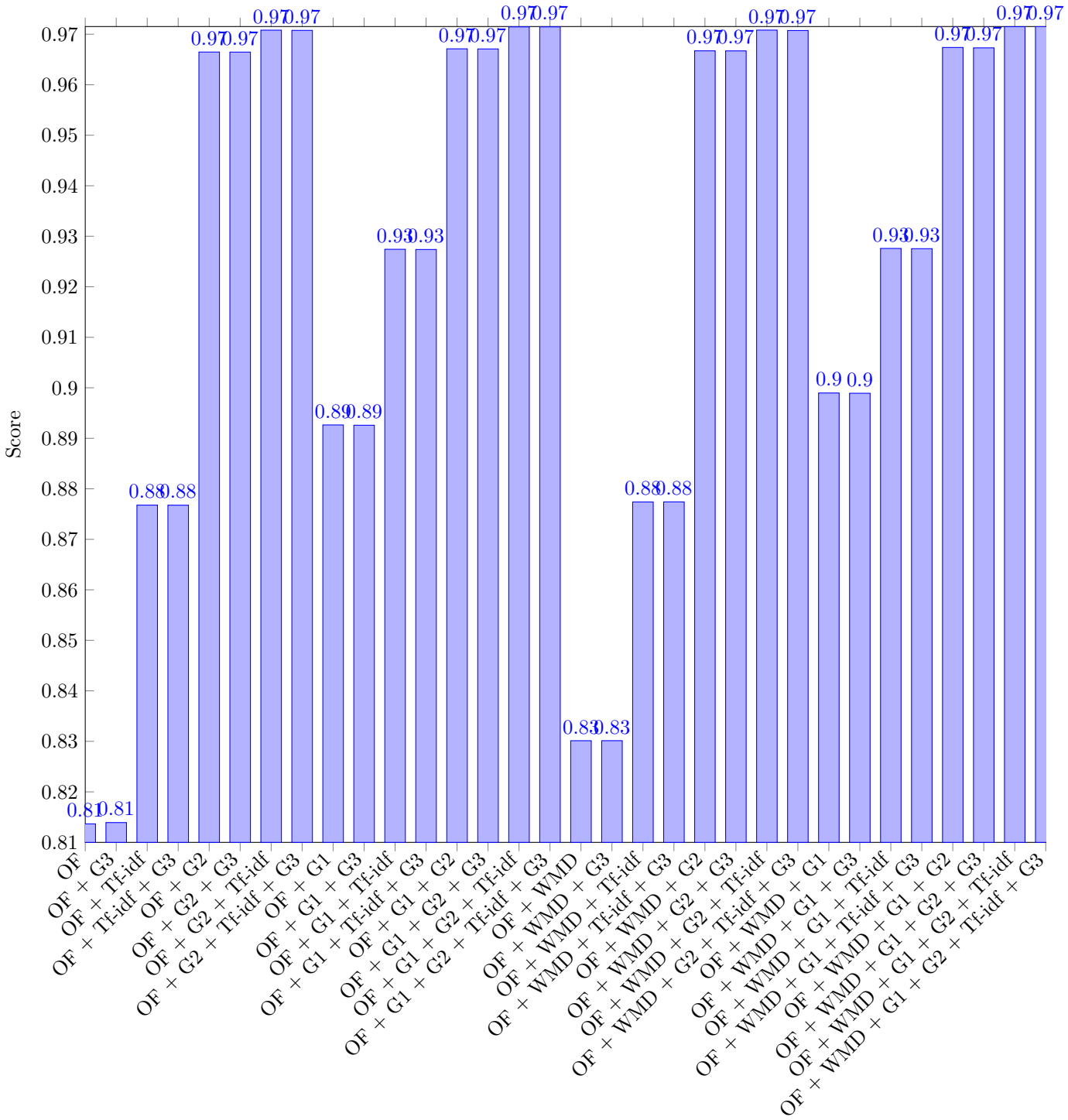

Figure 1: Cross-Validation Score for the LGBM with all the data for different sets of features.

Features: OF: Original Features (overlap_title, temp_diff and comm_auth) ; G1: Degree of nodes ; G2: Number of shared neighbors ; G3: Author citation graph ; WMD: Word Mover's Distanc ; Tf-idf: Tf-idf score

Classifier	Best score
Linear SVM	0.9444
RBF-kernel SVM	0.9624
Random forests	0.9709
LightGBM	0.9712
Multi-layer perceptron	0.9706
LightGBM + Random forest	**0.9713**
LightGBM + Multi-layer perceptron	0.9708

Table 1: Best score on private set for different classifiers. Note that different features, tuning and train sets were used, as specified in the corresponding subsections.

References

[1] Jinjian Zhai Jingyu Cui, Fan Wang. Citation networks as a multi-layer graph: Link prediction and importance ranking. *CS24W Project Report*, 2010.

[2] N. I. Kolkin M. J. Kusner, Y. Sun and K. Q. Weinberger. From word embeddings to document distances. *Proceedings of ICML*, 2015.