# NLP Data challenge report

Team name : QDTTS/ Axel de Lavergne, Mohamed Salah Zaiem

March 16, 2018

## 1 Introduction

The challenge was very interesting because we started, for once, without a features table. We had to design entirely by hand all the features needed. Graphs offer a large number of possibilities and this was where difference could be made. In this challenge, we arrived **2nd out of 28 teams**, with a score of **0.97991** out of 75 submissions, and we will present our work in the following.

## 2 Features engineering

During the feature engineering, we identified two different types of features : semantic (using the text), and topological (using the graph representation of the links). Our feature engineering is mainly based on these different publications : [1], [2], [3], [4], [5], [6].

### 2.1 Textual and semantic features

We tried to focus first on text related features. These features would therefore only concern the title of the paper, the text of the abstract and the names of the authors.
From these three texts, we created 8 features :

- overlap title : number of overlapping words in title
- overlap abstract : number of overlapping words in abstract
- temp diff : temporal distance between the papers
- comm auth : number of common authors
- comm journal : 1 if they are in the same journal
- cosine similarity : Cosine similarity between abstracts
- lsa distance euc : lsa distance between abstracts

As we can see in Figure 1, LSA distance and cosine are very valuable features. Abstract overlap brings valuable information too.

### 2.2 Topological features

Using Igraph library first then Networkx, we tried to take advantage of the graph given since it offers a very large number of features possibilities.
We were not aware of all the features a graph offers before reading some papers on link prediction. We discovered some crucial indexes for our classifier. We first implemented Jaccard Coefficient and it highly boosted our score, around 0.95. We decided to search for more graph indexes and it lead us to a total of 14 topological features.
Using Igraph we added :

- pagerank : very known index for his role in the creation of Google.
- is same cluster : we created clusters using Newman's eigenvector community structure detection.

```
Plot feature importances...
```

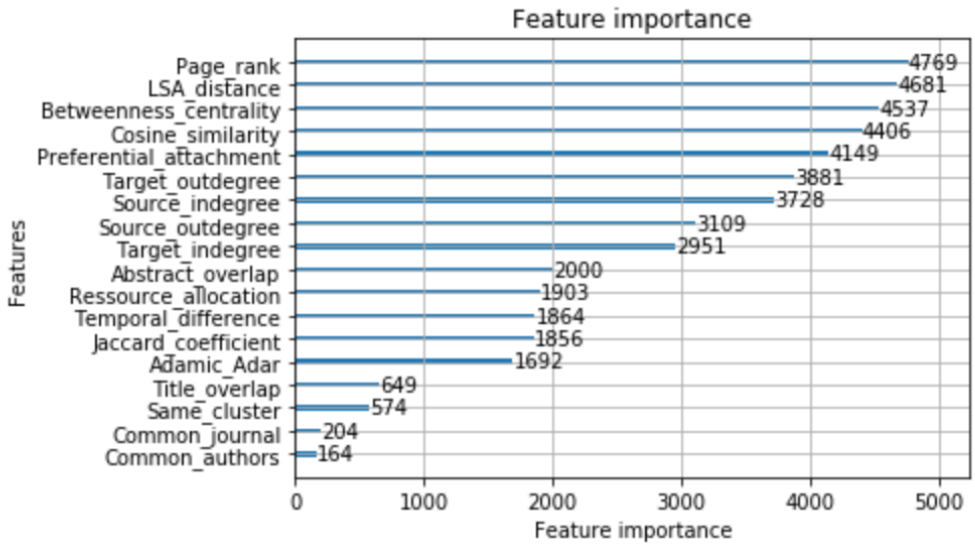


Figure 1: Features importance for the LGBM Classifier

- Betweenness centrality : a measure of centrality based on the number of shortest paths between two vertices.

These features are of the highest importance. For the LightGBM classifier, Pagerank and Betweenness centrality are respectively first and third most important features.
Using Networkx we created a directed graph that gave us access to other basic topological features such as indegree and outdegree for target and source.It also let us add three specific features :

- Jaccard coefficient : simple index based on the number of common neighbors.
- Adamic adar : (Frequency-Weighted Common Neighbors) This measure refines the simple counting of common features by weighting rarer features more heavily.
- Ressource allocation : index based on the number of neighbors of the common neighbors of target and source
- Preferential attachment : the product of the number of neighbors of the target with the number of neighbors of the source

# 3 Model fitting and comparison

If you need more information about this section, you can find all the parameters in Prediction-with-different-classifiers.ipynb file.
We are talking about classifiers only here but we were adding some features all along the way.
We mainly tried four classifiers on this challenge. We started with a SVM Classifier as it was the one given in the example. We reached a score of 0.95668 with SVM. Since the number of features was not too high regarding the number of points, we thought SVM was not the most adapted classifier to our problem and we moved on to Random Forests.
We had better results with Random Forests as we reached 0.96398. But as the competition were going and we were losing ranking, we thought we should try more complex classifiers.

We then used Neural Networks classifiers using keras library. After several tries, we opted for a multi-layered architecture with many dropouts to avoid overfitting. You will find the complete architecture in Figure 2. This made us reach a score of 0.96530 which was our best so far.//
We finally tried Gradient Boosting with the LightGBM library, as it is way more faster than Xgboost. The problem with Gradient Boosting is always about tuning the parameters. As you will

```
from sklearn import cross_validation
X_train, X_valid, y_train, y_valid = cross_validation.train_test_split(training_features,labels_array,
                                                                       test_size=0.4, random_state=0)
Y_train = np.array([y_train,(1-y_train)]).T
Y_valid = np.array([y_valid,(1-y_valid)]).T
```

```
# neural network building...
model = Sequential()
model.add(Dense(16, input_shape=(21,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(32))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(2))
model.add(Activation('softmax'))
```

```
# choose a optimizer for our neural network
rms = RMSprop()
model.compile(loss='categorical_crossentropy', optimizer=rms)
```

```
from keras import callbacks
early_stopping = callbacks.EarlyStopping(monitor='val_loss',
                                patience=2,
                                verbose=0, mode='auto')
```

Figure 2: Neural network architecture

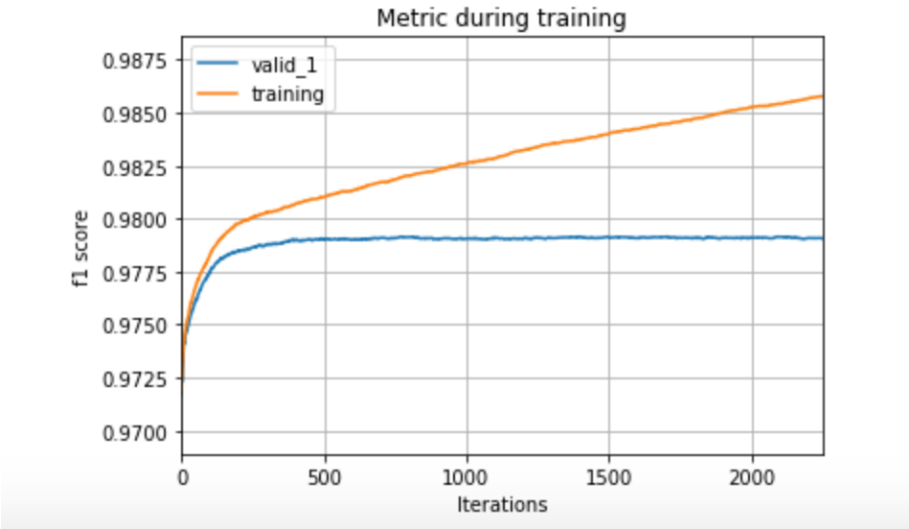


Figure 3: F1 Score evolution

find in Model-tuning-LGBM-classifier.ipynb, we tuned numerous parameters using cross validation techniques. This tuning allowed us to notably avoid overfitting and achieve a very good result. We picked the learning rate by hand after several tries on the validation set. This made us reach our best score which 0.97991. You will find in figure 3 and 4 the evolution of the binary log loss and the F1 score on training and on validation set.

# 4 Conclusion

First of all, this project allowed us to discover a problem of supervised learning on NLP tasks where there aren't any features in the beginning. It was really challenging, as we had to study the literature and think of relevant features to make the predictions. It also made us test our ability on different classifiers and compare their performances, in order to find new solutions to improve our score.

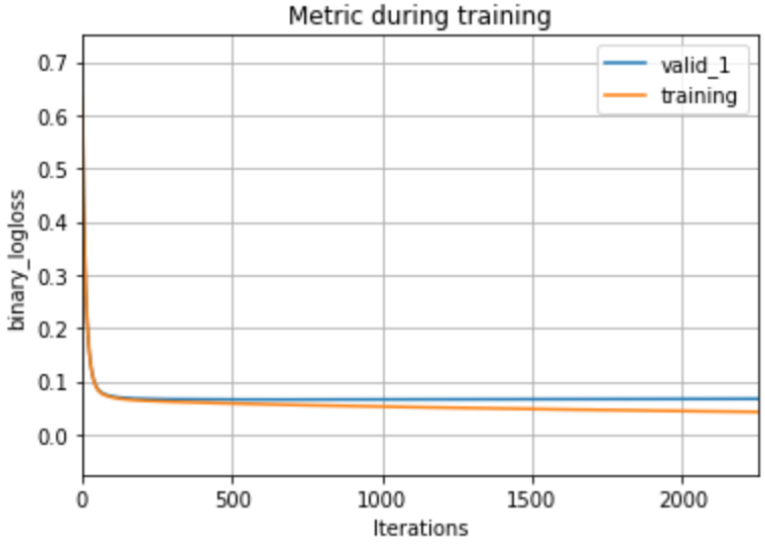


Figure 4: Binary logloss evolution

| Classifier | Best Score |
|---|---|
| SVM | 0.95668 |
| Random Forest | 0.96398 |
| Neural Network | 0.96530 |
| Light GBM | 0.97991 |

Table 1: Classifier's comparison

# References

[1] Feature Engineering for Supervised Link Prediction on Dynamic Social Networks, Jeyanthi Narasimhan, and Lawrence Holder ,School of Electrical Engineering and Computer Science, Washington State University - 2014.

[2] Graph-based Features for Supervised Link Prediction, William Cukierski, Benjamin Hamner, Bo Yang, Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011.

[3] Link Prediction in Social Networks using Computationally Efficient Topological Features, Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach and Yuval Elovici, Deutsche Telekom Laboratories at Ben-Gurion University of the Negev.

[4] Link prediction using supervised learning, SDM Workshop of Link Analysis, Counterterrorism and Security, M. A. Hasan, V. Chaoji, S. Salem, and M. Zaki, 2006.

[5] An efficient method for link prediction in weighted multiplex networks, Sharma and Singh, 2016.

[6] Link prediction in citation networks, Naoki Shibata, Yuya Kajikawa, Ichiro Sakata, 2011.