NGSA - Kaggle
Predict missing citations
Team name: TeamPuget

Maxime Nannan
Camille L'Huillier
Arielle Kuperminc
Eliott Rabin

January 2019

1 Features engineering

We mainly used **python** and the libraries **pandas**, **igraph** and **scikit-learn**.

1.1 Data processing

We cleaned titles, parsed list of authors with regular expressions and replaced NaN and infinite values in our extracted features by respectively 0 and a large value.

1.2 General features

We created dummy features on **title_overlap**, **publication_date_diff** (source date - target date), **same_journal** and number of common authors **common_authors**.

1.3 Graph features

For graph features we built two non directed and non weighted graphs:

1. a **collaboration graph** where nodes stood for authors and edges indicated that they had collaborated at least once

2. a **citation graph** where nodes stood for papers and were connected if one of them had cited the other

1.3.1 Authors graph

We extracted 3 kinds of features from this graph.

First there were features linked to the distance between source authors and target authors so we computed the distance between all the pairs of authors in source and target and got the number of authors that had collaborated **authors_collaboration**, the minimal distance **collaboration_min_distance**, the mean distance **collaboration_mean_distance**.
It should quantify how close are the authors.

Then we extracted the same kind of features than the previous ones but between the source paper first author (the writer) and target authors because we though that it was more important to know how close is the papers' writer

from cited authors than the others source authors. We obtained the following features **writer_collaborators**, **writer_collaboration_min_distance**, **writer_collaboration_mean_distance**
We also had a Boolean feature **writer_in_target_paper** that told whether the writer was in the cited article or not.

Finally, we extracted neighborhood based metrics **Adamic adar**, **Common neighbors**, **Jaccard coefficient**, **Preferential attachment** between authors from source and target and got the maximal value for each metric **max_authors_adamic_adar**, **max_authors_common_neighbors**, **max_authors_jaccard_coefficient**, **max_authors_preferential_attachment**

1.3.2 Papers graph

From the papers graph extracted the number of time target and source had been cited **(source—target)_paper_citations** with the idea that papers that had been cited a lot will probably be cited by others papers. We extracted too the number of papers cited by source and target because papers that had cited a lot of papers tend to cite less other papers **(source—target)_number_of_papers_cited**

Then we computed neighborhood based metrics between source paper and target paper **adamic_adar**, **common_neighbors**, **jaccard_coefficient**, **preferential_attachment** because those features allowed us to know if source and target had the same neighborhood (idea behind was that friends of friends tend to be my friend)

We also computed pagerank of the target and the source papers. **pagerank_(source—target)**.

We wanted to compute path based metrics but it was not easy to compute because we would have to create one graph for each pair source target to do so.

1.4 NLP features

In order to compute a text based similarities between source and target we used the tf-idf embedding and the cosine similarity between embeddings of:
- source and target titles: **title_cosine_similarity**
- source and target abstracts: **abstract_cosine_similarity**
- source and target journals: **journal_cosine_similarity**

- title of one paper and abstract of the other: **source_title_target_abstract_cosimilarity** , **source_abstract_target_title_cosimilarity**

2 Models

We decided to compare Random Forest and Light Gradient Boosting Machine and picked the best one.

2.1 Model tuning

We used a grid search to find the best parameters for both models and an early stopping for the LGBM to control overfitting.

2.2 Selection

By running a cross-validation and comparing the f1 scores we were able to compare models.

2.3 Features selection

As we only used LGBM and Random Forest we did not have to do features selection because those algorithms automatically used features that were the most discriminating.

2.4 Results

See below the f1 score of the different models on the leader board and obtained by cross validation.

	Cross-Validation	Leader board
Random Forest	0.9777	0.97023
LGBM	0.9787	0.96659

So we finally decided to use the Random Forest model because it was more robust to overfitting and got a 0.97023 f1-score on the leader board.

You can find bellow the features importance after training a Random Forest.

Figure 1: Features importance