Kaggle Competition
Team name: GALEX

Gaël de Léséleuc de Kérouara
gael.de-leseleuc@student-cs.fr

Alexandre Duval
alexandre.duval@student-cs.fr

1 APPROACHES

Let's remind that we have three data sources. The first one, *node info*, contains information about the paper itself, including id, publication date, title, authors, abstract and journal. The second provides information about citations. It shows pairs of paper ids, let's say A and B, as well as a label (1 if paper A cites paper B and 0 if not). So we have basically the true edges of the graph as well as some random pairs of edges that don't correspond to an edge, meaning where there is no citation between those papers. The test set also has a list of pairs of papers id but without labelling. Our goal is to label those, so in different terms, to predict which paper will be cited in the future and by who.
We in fact looked at two very different approaches to the problem, both giving great results.

1.1 Edge prediction approach

The first approach we chose is the following. We split the training data (citations) into two subsets: the training and target data. The former contains 80% of the data and is used to create a graph approximating the full citation paper network. Note that only true edges are included in this graph, meaning edges whose label equal 1 in the training file. Indeed, we do not want to create an erroneous graph containing fake edges (label = 0). In this graph, nodes refer to the papers and directed edges refer to citations. For instance, a direct edge from u to v describes a citation of paper v in the paper u.
We associate to each node information relative to the corresponding paper and extract graph features relative to nodes (betweeness similarity, degree, etc.) or edges (resource allocation index, adamic-adar, etc.). More details concerning those features are provided in the next section.
Once we have extracted a feature vector for each edge in and outside the graph, we try to predict the label of all edges that are included in the target subset. Note that these edges are not included in the graph and are therefore completely new. In fact, it is like we excluded some edges of the graph and added some unreal edges (random pairs of nodes not in the edge set) to create a dataset where we could train our classifier to recognise (using true graph properties and info on papers) if the edge was originally in the graph or not. Note that we dealt with the unbalanced dataset issue and made sure to include as many true edges as fake edges in the target set

We thus decide to train the model to predict the label of these unseen edges (in the graph we work on) because it is exactly what we will try to do for the test set. In other words, we use this graph's properties and information on each paper to predict the existence or not of a new set of edges, non-existent in the current graph. The drawback of this method is that we do not train the model on the full graph and consequently loose some information. To tackle this limit, we also proceeded to an ensemble of models, all following this approach but trained on different train/target splits of the data.

1.2 Label prediction approach

In the second approach, we train on the full dataset (graph), meaning we train the model to predict all edges' label contained in the training set. This includes label 0 and 1 edges, as well as test data edges. You thus learn to predict correctly if an edge has the label 0 or 1 (for the whole training dataset) and then apply your classifier on the test data edges, that are already in your graph. The advantage of this approach is that we can use the full graph to train the model. The drawback and we will even say, inconsitency, is that you use a fake graph to extract feature vectors. Indeed the graph being composed of both label 0 and 1 edges, is actually unreal. This is a strong factor that led us to favour the other approach. Nevertheless, this method still worked very well and resulted a 0.98 f1 score. Finally, note that we compute the features for edges at once in this case, unlike in the first scenario where we computed it in two parts (train then test).

2 FEATURES EXTRACTION

Once we have defined our approach and built the framework, the main task involves creating the feature vector for each edge since this vector is later used to classify the edges by our xgboost classifier.

There are two main types of features: graph based features and features that are intrinsic to the papers.

2.1 Intrinsic features

We use text based features to capture how likely a paper is to cite another based on some logical reasons. Indeed, a paper written before another can hardly cite something that has not been written yet. Similarly, papers often cite papers treating of a similar topic, as you only quote papers that are relevant to your work. An author often quotes work that he is familiar with, so probably papers written by his friends, his colleagues or published in the same journal as his. Therefore, to capture these correlations, we have included a few features:

(1) paper diff: difference in publication years between two papers
(2) abstract: cosine similarity between pre-processed (no stopwords, lowercase, lemmatize) and vectorized (using TF-IDF) abstract of two papers
(3) title: cosine similarity between two paper's title (also pre processed and vectorized)
(4) journal: 1 if they were published in the same journal, 0 otherwise.
(5) author: number of authors in common

2.2 Graph features

On the other hand, there are graph based features, most of which we have seen in lectures. We tried to included features as diverse as possible to capture as many aspects of the problem as possible. Note that features are extracted either for edges directly or for the two nodes forming an edge. We are providing below some basic intuition for their inclusion.

First of all, some basic properties such as the *source/target node in/out degree*, the *distance* (shortest path) and the *difference of in/out links* between the two nodes are included. This provides information about the connectedness of each node and supports the preferential attachment intuition. The probability of a new edge involving node u is proportional to the number of links of u. With this model, a node with many citations will receive even more citations.

On a similar node, we have integrated measures of node centrality, such as *degree centrality, k-core or a difference in betweeness centrality* (the others are a bit too computationally expensive). The latter is particularly interesting. The intuition is that a paper with a large betweenness centrality bridges unconnected papers and is therefore anticipated as a previously unexplored seed of innovation. It sort of measures the influence a paper has on the spread of information through the network.

In many networks, nodes are highly clustered locally, meaning that two papers with many *common neighbours* tend to be highly connected together and show a bigger proportion to cite each other. We have therefore included the number of common neighbours (undirected edges) from the network this to capture this phenomenon.

Since this task is about link prediction, we added some similarity metrics, such as the *Jaccard coefficient, the resource allocation index and the adamic-adar metric*. All of them are of great help to predict the existence of a link despite using different techniques. We wished to include more complex metrics, especially Katz or Pagerank link prediction, but they were quite computationally expensive and we already obtained great results without.

2.3 Creation of new features with the generation of a author-based-graph

We have dedicated a lot of our time to carefully clean the author name features and make the most of its potential because we believe some co-authorship proprieties (described below) will convey meaningful information for citation prediction.

Author name cleaning. We observed that author name writing form greatly varied from one paper to another.

Here are some examples :

```
1. M. Cvetic, H. Lu, C.N. Pope
2. L. Cornalba (I.H.E.S.)
3. Y.S. Myung, Gungwon Kang
4. V. Fateev (Montpellier), A. Zamolodchikov (Rutgers),
Al. Zamolodchikov'
5. Satoshi Iso, Hikaru Kawai, Yoshihisa Kitazawa
```

.

Some author are designated using only their initial for the first name (see example 1), while other are written with their full first name (see example 5). For some authors, their university or their location are also specified (see example 2 and 4). So we had to process the authors names so that a particular author is characterised by a unique representation of his name. We proceed this way :

(1)

: some are written with their first name, some with university where the authors works, etc. Using regular expression, we succeed to convert every name to a unique format.

Generation of a new graph. For now, we only use author name to compute the number of common authors between two papers. However, let's imagine that in paper A there is an author X, and in paper B there is an author Y. Now let suppose that in author X and author Y worked together on a paper C. This a very valuable information when the problem is to evaluate the probability of citation existence between paper A and paper B. To access this data, we had the idea to generate a new graph where the nodes still represented a paper but now the edges represents the existence of a common author between two papers. If we take back our previous example, in that case, paper A and B will have a distance of only two in the new generated graph. So to exploit this information, we simply decided to add graph features from this new author based graph.

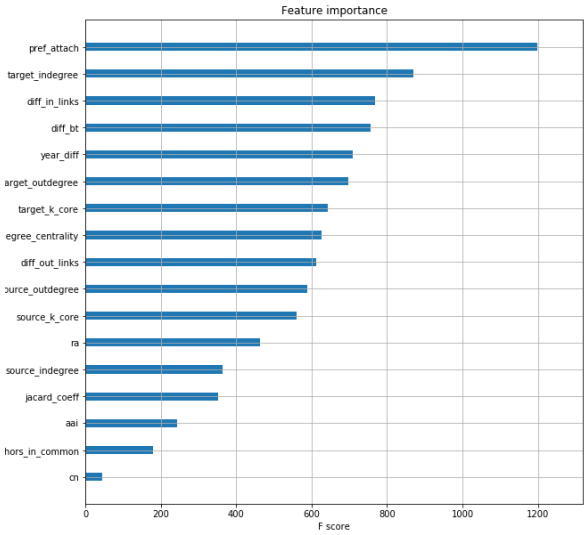

Figure 1: Weight feature importance (top)

2.4 Feature importance

At each iteration, we studied the performance and the relevance of our new features using some feature importance metrics, easily accessible when using xgboost.

3 MODEL AND PERFORMANCE

3.1 Results

In this section, we bring some last precisions concerning the models that we have used.

We started with the SVM model but quickly changed it for a tuned XGBoost model, which largely improved our results. We will not delve into its functioning as it was deeply used and understood in the machine learning module of Term 1. The ultimate version had 200 trees with a depth of 7.
We evaluated our model performance both using cross validation and submitting on the Kaggle, as the number of submission was not limited, it allowed us to gain some time.
Note that our best result on Kaggle is 0.9887, and is the result of our ensemble of three models with the first approach. We indeed built three similar models, using a different target set and of different sizes, to make our model more robust. We chose the final output for each edge prediction sample using a majority rule. Please refer to the Approach section above for more details).

3.2 Limitations and improvements

The main limitation is computational. Using networkx to calculate most metrics, as it has a bigger library, we cannot compute the Katz similarity for instance, as it takes too long. However, although we are not first, we end up with a great accuracy so it is not essential either.

If we had more time and less work, we would have liked to try an unsupervised approach for link prediction. In particular, we would have liked to use node2vec in order to extract features for each node (and then edge edge) in an unsupervised way. We could even combine these unsupervised features with the supervised features that we created and run a more complex model using them all.