Link Prediction in Citation Networks

Kaggle Competition Report

Raghuwansh Raj
raghuwansh.raj@student-cs.fr
CentraleSupélec
Gif-sur-Yvette, France

Greshma Babu
greshma.babu@student-cs.fr
CentraleSupélec
Gif-sur-Yvette, France

Wanci He
wanci.he@student-cs.fr
CentraleSupélec
Gif-sur-Yvette, France

Vivian Koutroumani
vivian.koutroumani@student-cs.fr
CentraleSupélec
Gif-sur-Yvette, France

CODE

Link to the git repository containing all code and data used for this paper: https://github.com/rajraghuwansh/kaggle_MLNS2022_-Link_Prediction

1 INTRODUCTION

For this competition, we are given datasets containing information about research papers, and the citations between them. The training set consists of 615,512 source-target paper pairs, labeled with 1 if there is a citation from sourve to target, or 0 otherwise. We are also given a file containing useful information about the papers, like ID, publication year, title, authors, name of journal and abstract. Based on this information, our goal is to predict the links for a test set that consists of 32,648 source-target pairs.
We reached the leaderboard score of 0.98195 making us finish **3/44** in the competition. For the purposes of the competition we considered papers as nodes and citations between them as edges, so that we can treat them as a graph in which we need to predict the links[2].

2 FEATURE ENGINEERING

For our model to perform the maximum, we had to create features that would improve the predictions. We considered two types of features: The **semantic** ones, and the **topological** ones.

2.1 Semantic Features

Semantic features are the features that are related to the word's meaning. We define our different semantic features based on the property that the more related two papers are, the more probable it would be to have common words in the titles and abstracts and to be published in similar journals or the same journal or written by the same people[4].
Before we compare the different words, however, we remove words that are so frequent in the language but do not add information to the text, called Stop words as these might tend to give unimportant information to our models.
The different features we define are as follows:

- Overlap_title: This feature measures how many words are common in the titles of the two papers.
- Overlap_abstract: This feature measures how many words are common in the abstracts of the two papers.
- Overlap_journal: This feature measures how many words are common in the name of the journals that the two papers are published in.
- Comm_auth: This feature measures how many authors are common among the two papers.

We also compare papers by converting them to vector representations and finding the similarity between them. The features generated so are:

- Cosine_similarity: There are some words that are more representative of a document whose information is more definitive of the paper and related papers. To find these words, we make use of the methodology called TF-IDF vectorization. What the TF-IDF does is it calculates the frequency of terms in a document (Term frequency, TF) and compares it with how rare or common the word is in the set of documents (Inverse Document Frequency, IDF). This is done by multiplying the frequency of the term with the inverse logarithm of the ratio of the total number of documents to the number of documents the term appears in.

$$\text{Term frequency (TF)} = \frac{f_{t,d}}{\Sigma f_{t,d}} \quad (1)$$

$$\text{IDF} = \log \frac{N}{|\{\text{d} \in \text{D: t} \in \text{d}\}|} \quad (2)$$

$$\text{Cosine Similarity (d1, d2)} = \frac{\text{Dot product(d1, d2)}}{\|\text{d1}\| * \|\text{d2}\|} \quad (3)$$

where t is the term we are considering, d is the document, D is the corpus of documents, $\text{f}_{t,d}$ is the count of term t in document d and N is the total number of documents. For cosine similarity, d1 and d2 are two vectors.

We use the inbuilt TfidfVectorizer from the sklearn library's feature_extraction.text module to convert the documents into a vector of tf-idf features. To compare more effectively, we use both unigrams and bigrams. Once we have the vector representation, we can approximate the similarity of the two texts using the cosine angle between them.
- LSA(Latent Semantic Analysis) is a technique of analysing relationships between a set of documents. A matrix containing word counts per is constructed from a large piece of text and a mathematical technique (SVD) is used to reduce the number of rows while preserving the similarity structure

among columns. This is done computationally by making use of the TruncatedSVD from the sklearn.decomposition module. Documents are then compared by taking the cosine of the angle between the two vectors.

Lastly, another feature we study is the:

- Date_Difference- This feature measures the temporal distance between the years the papers were published in.

2.2 Topological Features

Topology is a mathematical approach that allows us to structure data based on the principles of feature adjacency and feature connectivity[1]. We created topological features in order to manage spatial relationships between the research papers. First of all, we created our graph using networkX python package. Given our graph we created and used the following features:

- PageRank: We used the inbuilt PageRank method that computes a ranking of the nodes in the graph G based on the number of the incoming links. We set the damping factor to 0.5, which is the probability of someone to stop clicking on citations, at any step.
- Betweenness centrality : Using the package betweeness, each node of the graph receives a score, based on the number of shortest paths that pass through the node (nodes that more frequently lie on shortest paths between other nodes will have higher betweenness centrality scores). Then to calculate the betweness centrality we abstracted the target node's score from the source's score.
- Is samecluster : Using community leading eigenvector, we firstly computed the clustering coefficient for nodes. Then we compared the cluster information for source node and target node to check if they are in the same cluster or not.
- Jaccard coefficient : This index is based on the number of common neighbors.
- Adamic adar : Index based on the amount of shared links between two nodes. Mathematically, it is the sum of the inverse logarithmic degree centrality of the neighbours shared by two nodes. This measure refines the simple counting of common features by weighting rarer features more heavily.
- Resource allocation : Index based on the number of neighbors of the common neighbors of target and source nodes.
- Preferential attachment : The product of the number of neighbors of the target with the number of neighbors of the source.

The last 4 features, are automatically calculated by networkX package and are used for undirected graphs, but we used them cause they seemed to improve our model's performance.

2.3 Vertex Features

Below we explain the **vertex features**[3] for directed graphs we created:

- Target in degree, Source indegree :Number of edges directed inwards divided by node degree.
- Target out degree, Source out degree : Number of edges directed outwards divided by node degree.
- Target scc, Source scc: Degree of a node divided by the number of strongly connected components (where every vertex is reachable from every other vertex) in the neighborhood subgraph.
- Target wcc, Source wcc : Same as previously, using the number of weakly connected components (where all vertices are connected to each other by some path, ignoring the direction of edges) in the neighborhood subgraph.
- Len path st : Shortest path length from the source to the target.
- Len path ts : Shortest path length from the target to the source

2.4 Edge Features

What is more, we added some features related to **edges, clusters, K-core**:

- Target nh subgraph edges, Source nh subgraph edges: Number of edges in the neighborhood subgraph generated by a considered node.
- Common friends : For a given pair of nodes (target,source), we want to know the number of vertices that are at a 1 edge-distance of for target and source, meaning how many straight citations are there.
- Total friends : For a given pair of nodes (target, source), total number of nodes in the union of the two neighborhood subgraph.
- Friends measure : For a given pair of nodes, this is the number of vertices that are at a 3 edge-distance of for target and source.
- Sub nh edges : Number of edges in the neighborhood subgraph.
- Len path : Shortest path length between the source and the target

2.5 Additional Features

- Target clustering, Source clustering : Compute the cluster coefficient of a node. clustering(v) = 2T(v) d(v)(d(v) 1) T(v) the number of triangles using v as one the vertices.
- Target core, Source core : K-core is a subgraph containing only nodes of degree k or more. This core number of a node is the highest value of k of a k-core containing that node.
- Target pagerank, Source pagerank : Use the PageRank algorithm to determine a value for a node. Imagine an agent randomly choosing edges to go to and outputting the number of times it would go through that node.

Degree is the number of neighbors of a node.

3 MODELLING AND PREDICTIONS

3.1 Models Trained

We mainly tried five classifiers on this challenge with continuous addition of more graph features and hyperparamters tuning.

3.1.1 SVM. We started with a SVM Classifier as it was the one given in the example. We reached a score of 0.9644 with SVM.

3.1.2 Random Forest. Since the number of features was not too high in the begining 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 0.9745.

3.1.3 Keras Neural Network. 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. This made us reach a score of 0.9714 and the loss and accuracy are as shown in figure 1 and figure 2 But as the competition were going and we were losing ranking, we thought we should try more complex classifiers.

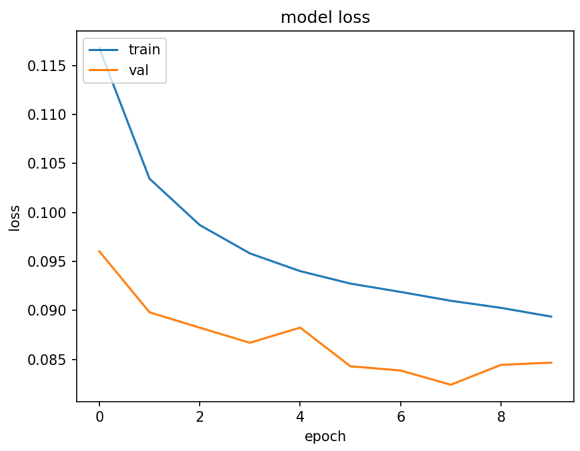

Figure 1: Keras Neural Network Loss

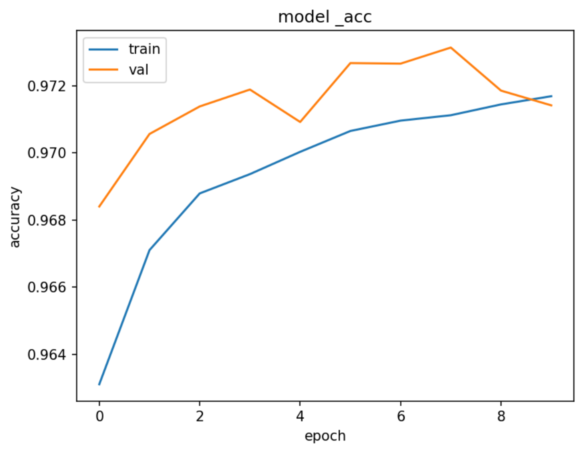

Figure 2: Keras Neural Network Accuracy

3.1.4 Boosting. In order to predict the existence of an edges given our input feature vector, we use a XGBClassifier available in the XGBoost library which provides scalable and flexible Gradient Boosting method. We use a LGBMClassifier from LightGBM and it scored higher than the previous one. To monitor the training and avoid overfitting we use K-folds method for cross validation. It allows us also to have an average prediction for the testing-set which improves the results since we smooth the result by keeping the interesting trends captured by each training. We reached a accuracy of 0.9785 on the validation set but the model was slighlty overfitting on the leaderboard, hence we decided to tuned the parameters that control the overfitting.

Table 1: LightGBM Parameters

Parameters	Values	Parameters	Values
learning_rate	0.007	min_data_in_leaf	100
max_depth	5	lambda_l1[1]	0.8
feature_fraction	0.8	lambda_l2[2]	0.9
num_leaves	30	n_estimators	1250

[1] L1 regularization term on weights.
[2] L2 regularization term on weights.

3.2 Hyperparameters tuning

The problem with Gradient Boosting is always about tuning the parameters. 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 0.9734.on kaggle. Table 1 contains hyper parameter trained for LightGBM model and figure 3 ranks the importance of the features.

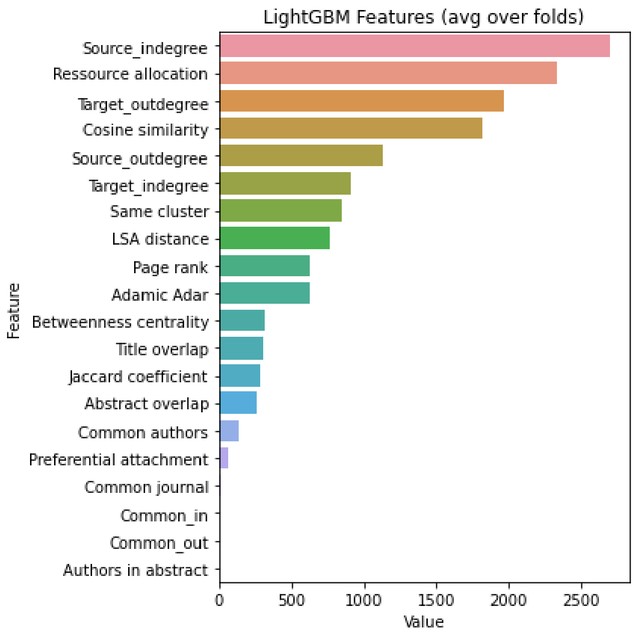

Figure 3: Importance Order of Features

At this point we decided to add more advanced graphical features to the feature body that was generated by exploiting nodes and edges of the graph, we were able to thrown in around 30 more graph features using this. These features can be found in Advanced

Table 2: Multilayer Perception Model Parameters

Parameters	Values	Parameters	Values
solver	adam	alpha	0.0094
activation	relu	hidden_layer_sizes	(72,36)
tol	5e-5	max_iter	250
verbose	1		

Table 3: Performance Comparison

Models	Training Score	Leader Board Score
LGBM	0.9815	0.9734
MCP	0.9861	**0.98195**

Feature Engineering notebook that is shared. We reiterate over the tuned LGBM but the performance seemed plateued, Then we turned into multilayer perceptron model given by scikit learn and start playing with thier parameter after some feature tuning we were able to clock an accuracy of 0.98195 on the leader board,the model loss was also all time low at 0,0481.We decided to use this model as our final model for the challenge.
Table 2 specifies the parameters we used for the multilayer perception model and figure 4 shows the loss while training.

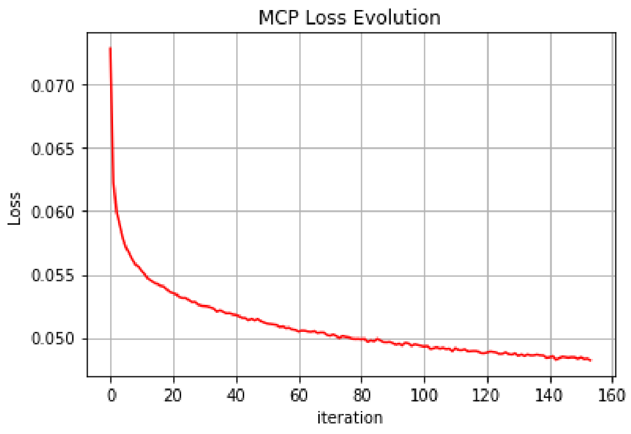

Figure 4: MCP Loss

4 RESULTS

Performance comparison of lGBM and MCP on validation data and leaderboard are shown in the table 3.

5 CONCLUSIONS

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. It also helped us to explore the fascinating possibility of graph features in machine learning.

REFERENCES

[1] Michael Fire, Lena Tenenboim, Ofrit Lesser, Rami Puzis, Lior Rokach, and Yuval Elovici. 2011. Link Prediction in Social Networks Using Computationally Efficient Topological Features. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*. 73–80. https://doi.org/10.1109/PASSAT/SocialCom.2011.20

[2] Mohammad Al Hasan and Mohammed J Zaki. 2011. A survey of link prediction in social networks. In *Social network data analytics*. Springer, 243–275.

[3] Xi Wang and Gita Sukthankar. 2014. Link prediction in heterogeneous collaboration networks. In *Social network analysis-community detection and evolution*. Springer, 165–192.

[4] Till Wohlfarth and Ryutaro Ichise. 2008. Semantic and Event-Based Approach for Link Prediction, Vol. 5345. 50–61. https://doi.org/10.1007/978-3-540-89447-6_7