Predict Missing Links in a Citation Network
ESSEC Business School | CentraleSupelec
Network Science Analytics – Team PVPS

Shahmir Kazi Veerendarnath Naladala Priyanka Pippiri Pravalika Mavilla
ESSEC Business School | CentraleSupelec
B00771255@essec.edu B00760534@essec.edu B00746159@essec.edu B00767354@essec.edu

Abstract

The team explores and proposes a solution for "Missing Link Prediction" in a citation network. Using Network Theory [NetworkX] and using a directed graph theory approach [DiGraph] and multiple classifiers based on ensemble learning and boosting techniques, the team arrives at a ***98.3% accuracy*** *using Ada Boost Classifier [1]. Compared to the choice of technique, the "Feature Engineering" was the most significant contributor to the successful result.* ***25 features*** *were created and all were used to train, validate and subsequently test on an unseen data set.*

1. Motivation

The problem of link prediction is of significant use in many industries such as security, e-commerce and recommender systems. [3] The team wishes to learn these techniques through the given problematic, to be able to use them in real world situations.

2. Problem Definition

From a citation network of research articles, edges have been removed from the nodes. The problematic consists of a directed network and the team is working on the following:

Given An unseen and varied directed citation network,

Use Feature engineering, Network science/ Graph theory and Machine Learning classification techniques

To Predict whether a citation link is present between two articles or not.

Hence, it is a binary classification problem requiring use of multiple areas from the field of data science.

3. Available Data

- **Training_set.txt**

This file contains three columns: the first node is the source (quoting article) and the second node is the target (quoted article). The third column shows if there is a link or not (0,1)

- **Node_information.csv**

This file contains information for each of the nodes**.**

We name the columns as the following:

id', 'pub_year', 'title', 'authors', 'journal_name', 'abstract']

- **id** – unique id for each article
- **pub_year** – publishing year
- **authors**
- **journal_name** - article is publication
- **abstract** – a summary of the article

4. Methodology followed

The below is followed in chronological order and also detailed in the Python Jupyter notebook.

1. Visualizing the network as a graph using DiGraph and NetworkX. The below output is obtained.

```
Number of nodes : 27770
Number of edges : 335130
```

2. Create a 5% subset of the data for computational purposes (we tried increasing it to 15% and accuracy was not affected even under cross validation)
3. Feature Extraction – for underlying structural, graphical and textual similarities between source and target.
4. Running multiple ensemble classifiers under a 3 fold Cross Validation to assess optimal classifer and best hyper parameters.
5. Run the optimal classifier (Random Forest) on an unseen dataset.

5. Feature Extraction

25 features are created for deriving information:

Graph Related Features

- ***source_out_centrality:*** the percentage of papers the article is quoting [1]
- ***target_in_centrality:*** The percentage of article quoting out target article [1]
- ***source_centrality***: the relative importance in terms of out links [1]
- ***target_centrality***: the relative importance in terms of in links [1]
- ***source_evc:*** Eigenvector Centrality, is computed using adjacency matrix. It represents the source's influence in the overall network. Hence, it is also referred to as Prestige Score. It further represents whether the source node network structure is structural similar to the target node [2]
- ***target_evc***: Eigenvector Centrality computed for target node [1]
- ***shortest_path***: number of nodes between source and target [3]
- ***target_pagerank***: the ranking of the node based on the structure of incoming links [3]
- ***preferential_attachment***: the likelihood of a connection between two nodes, based on their existing connectivity [3]
- ***source_hub_score***: a hub will will cite many other nodes (out links from souce) [1]
- ***target_authority_score***: an authority will be cited by other nodes (in links to target) [1]
- ***jacard***: useful as it counts common neighbors but normalizes by the number of connections made by the two nodes [4]
- ***common_neighbors:*** closeness between two articles [4]
- ***out_neighbors***: for computing likelihood of 1 citing 2. If 1 cites none, it will not cite 2. [4]
- ***in_neighbors***: The likelihood of an article being cited can be gauged from this [4]
- ***popularity***: The number of neighbors of the nodes pointing towards the target [2]

Meta Data Features

- ***pub_year_difference***: recent articles are likely to cite each other [1]
- ***common_authors***: are likely to collaborate [5]
- ***common_journals***: likely to have common topics and authors at a combined space [5]

Textual Features

- ***title_similarity***: Sentence2Vec word embedding similarity used from Spacy [2]
- ***abstract_similarity***: similar method as titles [2]
- ***testing_dist_absract:*** Cosine similarity between abstracts. A simple BoW approach would even take into account common words but through TF- IDF "inverse" representation, we give weightage to less frequent words and compute cosine similarity. [5]
- ***training_dist_title:*** Cosine similarity between titles, same as above. Used as a measure of closeness. [5]
- ***common_successors:*** the articles to which both source and target point towards [2]
- ***common_predecesors***: the articles which pointed to both source and target [2]
- ***overlap_title***: Common words within titles. This is useful as Stemming has been performed, making this method more insightful [2]
- ***overlap_abstract***: Common words within abstracts. [6]

For feature selection and overfit testing, multiple trial comparisons using validation data and Kaggle test submission were performed.

6. Choice of classifier and Cross Validation for hyper parameter tuning based on 25 features

Initial runs with complete feature engineering were **run for all models** using GridSearchCV. It showed Random Forest to have the highest accuracy amongst classifiers.
Hence, we show the code for 3 Fold GridSearchCV for RF. The optimal parameters **[clf_best]** obtained through train/validation data are used for final testing on test data.

```
#hyperparameter grid search for randomforest
n_estimators = [200, 400, 600, 800, 1000]
max_depth = [5, 10, 15]
min_samples_split = [2,4,6]
min_samples_leaf = [1, 3, 6]

from sklearn.model_selection import GridSearchCV

hyperF = dict(n_estimators = n_estimators,
              min_samples_split = min_samples_split,
             min_samples_leaf = min_samples_leaf)
clf = GridSearchCV(RandomForestClassifier(), hyperF, cv = 3, verbose = 1,
                   n_jobs = -1, scoring='accuracy')

clf.fit(X_test, y_test)
clf_best = clf.best_estimator_
```

9 other classifiers were trained using GridSearchCV, on the complete training data with 25 features, with the below scores.

Validation Scores	
Random Forest	**98.004**
XG Boost	97.470
Logistic Regression	94.175
Stacking	97.749
Ada Boost	**97.204**
Decision Tree	95.440
Extra Trees	97.470
Gradient Boosting	97.540
Linear SVC	95.544
SVC	94.987

For preventing overfitting, feature wise comparison explained earlier was coupled with model wise comparison to make sure none of the models overfit the data.

Additional features were added based on validation and test accuracy improvement. Initially CV was run on 18 features to yield an accuracy of 91.6%. Subsequently, additional features were added to achieve the above accuracy also shown in the submitted Python code.

7. Running on test data (Team Name = PVPS)

Using the Random Forest Classifier with optimal hyperparameters, we submit the results on Kaggle to obtain an accuracy of 98.2%.

Due to highest cross validation score, it was chosen as the optimal model. However, running Ada Boost on test data shows AdaBoost to have the highest accuracy. Therefore, for test data, we may finalize AdaBoost with an **accuracy of 98.3%**.

Choice of Classifier	
Random Forest	98.205
XG Boost	98.074
Logistic Regression	97.026
Stacking	98.170
Ada Boost	**98.307**
Decision Tree	96.698
Extra Trees	97.698
Gradient Boosting	98.075
Linear SVC	97.861
SVC	97.700

8. Related work and References

[1] M. Al Hasan, V. Chaoji, S. Salem, M. Zaki ”Link Prediction using Supervised Learning” Rensselaer Polytechnic Institute, Troy, New York 12180

[2] What will Facebook Friendships look like tomorrow? http://be.amazd.com/link-prediction/

[3] M. Al Hasan, M. J. Zaki ”Link Prediction in Social Networks” eBay Reseach Center, San Jose CA and Rensselaer Polytechnic Institute, Troy, New York 12180

[4] Kdeng ”Feature Selection” Carnegie Mellon University of Computer Science https://www.cs.cmu.edu/ kdeng/thesis/feature.pdf

[5] M. Vazirgiannis INF 582: Text Mining and Natural Language Processing Ecole Polytechnique

[6] Raphael Montaud & Gabriel Misrachi. Link Prediction in a Citation Graph. https://github.com/raph-m/link-prediction/blob/master/link-prediction-report.pdf

[7] Amir Mahdi Abdolhosseini-Qomi. Link Prediction in Real-World Multiplex Networks via Layer Reconstruction Method. https://arxiv.org/pdf/1906.09422.pdf