Link Prediction in Citation Networks

Maria Kouvela, Napoleon-Christos Oikonomou
Data & Web Science Post-graduate program
School of Informatics, Aristotle University of Thessaloniki
{mvkouvela, onapoleon}@csd.auth.gr

ABSTRACT

In this work, which was part of the Technologies for Big Data Management and Analytics course, we focus on two tasks. Given a set of edges that correspond to citations (or lack thereof) between two scientific papers we are asked to 1) Train a classifier that, when given two scientific papers, is able to predict if there will be a link (citation) between them and 2) Given only a set of papers, create a model that identifies possible citations between them.

PART 1

I. INTRODUCTION

In this part, a citation network is given as a graph where nodes are research papers and an edge between two nodes signifies that one of the two papers cites the other. From this citation graph, edges have been removed at random. The goal is to create a model that when given two of the network's nodes, predicts whether these nodes share a link or not. Given that for every node there is plenty of information available: title, abstract, authors and so on, this problem combines natural language processing and network link prediction. Thus, we used tools and techniques from both fields to distil information about the data and extract features.

II. DATASET

There are four datasets provided. A short description of each one is presented below:

- `training_set.txt`: 615,512 labelled node pairs (1 if there is an edge between the two nodes, 0 else). One pair and label per row, as: source node ID, target node ID, and 1 or 0. The IDs match the papers in the `node_information.csv` file.
- `testing_set.txt`: 32,648 node pairs. The file contains one node pair per row, as: source node ID, target node ID. Evidently, the label is not available, and we need to predict it.
- `node_information.csv`: for each paper out of 27,770, contains the following information (1) unique ID, (2) publication year (between 1993 and 2003), (3) title, (4) authors, (5) name of journal (not available for all papers), and (6) abstract. Abstracts are already in lowercase, common English stopwords have been removed, and punctuation marks have been removed except for intra-word dashes.
- `Cit-HepTh.txt`: this is the complete ground truth network. This file was used only to evaluate our solutions with respect to the accuracy.

III. METHODOLOGY

Our dataset consists of information about 27,770 research papers. We have information for 615,512 pairs, which account for 0.16% of the possible pairs, given the fact that we treated the graph as undirected. Although we do not have direct access on how or why these particular pairs where selected to be the training set, we do know that for 54% of these do correspond to citations, while the other 46% do not. The testing set on which we are called to predict citation links consists of 32,648 pairs or 5% of the training set's size. Hence, taking also into account how balanced the training set is, one could argue that the training set network is a sufficient amount of information to find the unknown labels of the testing set. Even so, we used both content-based features (from `node_information.csv`) and network-based featured from the citation graph, to build our classifiers. The enhanced dataset we created contained the features that are described below.

A. Content-based features

1) Title

For each node in a pair, its title was lemmatised and then tokenized to word lists, after removal of stopwords. Then we considered the number of same tokens as a feature.

2) Abstract

We used the abstract of each node to extract two features. Firstly, the number of same words between the pair's abstracts, after lemmatisation and tokenization to word lists. Secondly, we converted the abstract's text to a vector of token counts and calculated the cosine similarity between these two vectors, as another feature.

3) Authors

We added a binary feature that signifies if two nodes share one or more authors, or not.

4) Year

We used the pair's temporal difference, that is, the year gap between the two publications.

B. Network-based features

1) PageRank

PageRank works by counting the number and quality of links to a node in order to determine a rough estimate of how important the node is. The underlying assumption is that more important papers are likely to receive more citations, and that important papers cite other important papers as well. Consequentially, we added two more features; the PageRank index of each end of the pair.

2) Indegree & Outdegree

The purpose of this feature is to measure the importance of a node in the citation network through in degree centrality criteria. As stated above, we focused on a simplified version of the problem, not caring if *A* cites *B*, or *B* cites *A*, so we used both in-degree and out-degree of each node, adding four more features.

3) Neighbours

Last but not least, we calculated the number of common neighbours for every pair's nodes. The idea behind this, is that nodes that share a lot of common connections have a higher probability of being connected with one another.

C. Classification

Following the analysis above, we enhanced the original dataset by adding: `cosine_similarity`, `num_of_same_words_in_title`, `num_of_same_words_in_abstract`, `source_indegree`, `target_indegree`, `source_outdegree`, `target_outdegree`, `year_gap`, `neighbour_similarity`, `have_same_authors`, `source_pagerank` and `target_pagerank`. We transformed each instance of features into a vector, which we normalized for mean and variance. Then, we fed the above feature set into multiple classifiers; Logistic regression, Decision tree, Random forest, Gradient-boosted trees, Multilayer perceptron and Linear Support Vector Machine. After hyper-parameter tuning, the resulting classifiers alongside the target metric of our task (mean F1 Score), are presented in Table 1 below. As a reference, Table 2 shows the time it took for each classifier to train, when using a 2012 MacBook Pro equipped with an i7-3615QM @ 2.3GHz with 4 physical cores and 8 GB of RAM. Note that the dataset where already stored in memory, but their creation took ~6 hours.

TABLE I. OPTIMAL CLASSIFICATION PARAMETERS & RESULTS

ALGORITHM	F1 SCORE	PARAMETERS
LSVC	87.4%	maxIter: 2000 regParam: 0.001 tol: 1.0E-8 aggregationDepth: 10
Logistic Regression	87.9%	maxIter: 2000 tol: 1.0E-8
Multilayer Perceptron	73.67%	layers: [14, 32, 18, 2]
Random Forest	93.59%	maxDepth: 15
GBoost Trees	95.92%	maxDepth: 15

TABLE II. TIME FOR TRAINING & TESTING

ALGORITHM	TRAIN	TEST
LSVC	$102sec$	$6sec$
Logistic Regression	$16sec$	$3sec$
Multilayer Perceptron	$201sec$	$4sec$
Random Forest	$76sec$	$17sec$
GBoost Trees	$681sec$	$53sec$

As it is evident, the method that performed the best was GBoost tree ensembles. Random Forest, another kind of ensemble, performed acceptably close and took far less time to train. This is because while GBoost adds classifiers serially, Random Forest does it in parallel.

IV. CONCLUSIONS & FUTURE WORK

In this part of our work we have tried to create a model that predicts links in a citation network. Using features both topological and content-based we created a dataset consisting of 615K citations with 12 features. We then experimented with different classifiers and found that GBoost trees produce a F1 score ~96%. This validates our original hypothesis that the nature of the original dataset is sufficient enough, to predict the unknown links in the network.

Future work lies in many directions. One interesting approach is to create a second network with authors as nodes and try to distil more information about each candidate citation.

PART 2

I. INTRODUCTION

In this part of the work, we are given only the nodes of the network and are tasked to create a model that recreates the graph. In other words, without having any knowledge if a paper cites another or not, find every citation that exists. As one can imagine, this is a much more difficult problem, because not only is the number of possible edges quadratic to the number of papers, but also, even for an expert of a field, it is very difficult to distinguish papers that cite one another.

II. METHODOLOGY

Given the complexity of the task and our limited computational power resources, we decided to focus on a small sample of the original dataset. We used a network consisting only of 700 nodes, or ~2.5% of the original dataset. In our dataset, out of $\frac{700*(700-1)}{2} = 244{,}650$ possible edges, we measured 137 existing ones, a graph density of ~0.06%. This is on par with original graph's density of 0.06%. The main focus of our work is based on the assumption that "real" citations have different properties from "fake" ones and so one can separate them into two groups. Hence, we used unsupervised learning techniques, to try and separate our edges into these two groups and selecting the smallest of them as the cluster of "real" citations.

A. *Features*

Our dataset is unlabelled and as such we used only Content-based features, as described in Part 1, III.A.

B. *Clustering*

Following the analysis above, we enhanced the original dataset by adding: `cosine_similarity`, `num_of_same_words_in_title`, `num_of_same_words_in_abstract`, `year_gap` and `have_same_authors`. We transformed each instance of features into a vector, which we normalized for mean and variance. Then, we used different clustering algorithms to separate our instances into "real" and "fake" citation; k-Means Clustering, Hierarchical Clustering and clustering using Gaussian mixtures. After hyper-parameter tuning, the resulting classifiers alongside the target metric of our task (mean F1 Score), are presented in Table 3 below. As a reference, Table 4 shows the time it took for each classifier to train, when using a 2012 MacBook Pro equipped with an i7-3615QM @ 2.3GHz with 4 physical cores

and 8 GB of RAM. Note that the dataset where already stored in memory, but their creation took ~2 hours.

TABLE III. OPTIMAL CLUSTERING PARAMETERS & RESULTS

ALGORITHM	F1 SCORE	PARAMETERS
k-Means	76.17%	maxIter: 1000 tol: 1.0E-9
Hierarchical Clustering	33.59%	maxIter: 50
Gaussian mixtures	2.95%	maxIter: 1000 tol: 1.0E-9

TABLE IV. TIME FOR TRAINING & TESTING

ALGORITHM	TRAIN	TEST
k-Means	$9sec$	$1sec$
Hierarchical Clustering	$26sec$	$1sec$
Gaussian mixtures	$49sec$	$1sec$

As it is evident, the method that performed the best was k-Means Clustering. Hierarchical Clustering and Gaussian mixture models don't seem able to create acceptable clusters. This is because these techniques cannot take into account the enormous class imbalance.

C. Outlier Detection

In an effort to try and improve our results, we thought of the problem as an anomaly detection one. Most real graphs, including our sample, are very sparse and finding an edge between two nodes seems rare. To try and find these outliers we used the following logic:

- For each feature, calculate its mean and its variance across all data instances.
- Given a new example x, compute the probability of each feature using the Gaussian distribution function:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Take the product p of all probabilities. If $p < \varepsilon \rightarrow anomally$.
- Repeat process 5 times using different portions of the dataset. Consider an edge anomalous if it is classified as such all 5 times.

Although the method described above considers input dataset to contain no anomalies and as such is riskier to use that clustering techniques, is outputted better results that k-Means Clustering. It had a mean F1 Score of 86.73%. Note that in our case $\varepsilon = 0.002$

III. CONCLUSIONS & FUTURE WORK

In this part of our assignment we have tried to create a model that recreates-from-zero a citation network. Using content-based features we created a dataset consisting of 245K citations with 5 features. We then experimented with different clustering techniques and found that k-Means Clustering produces an F1 score ~76%. Then we considered "real" edges to be outliers and created an anomaly detection algorithm that improved F1 Score to ~87%.

Future work lies again in many directions. A much necessary task is to expand our sample and test if we get similar results for bigger networks. One other interesting approach would be to consider papers and not citations as data points.