Citation Prediction Report

Lucas Elbert and Jacob Pfau

Kaggle Team: Cappucinos

1 Introduction

Our task is to predict citations between physics papers given paper titles, authors, publication years, name of journal and abstract. We are given a training set with 615,512 pairs of papers for which we know whether or not one cites the other. Our goal is then to predict given a set of 32,648 source/target pairs whether the source predicts the target.

The first step in our pipeline was to construct a graph representing the citation relation between papers. Then we created features quantifying the relationship between each source/target pair. These features involved comparisons of abstract text and other paper descriptors and some used combinations of both text comparison and citation graph properties.

We experimented with various machine learning models including logistic regression, support-vector machines (SVM) and ensembled decision trees. Following hyper-parameter tuning and feature selection, we determined that the library Light GBM — using a variant of gradient boosted trees — provides optimal citation prediction with yielding a ~0.980 f-score.

2 Dataset

Our dataset consists of information on 27,770 physics research papers. Out of the 771,172,900 possible pairs of papers we have citation information on 0.16% or 615,512 of these pairs. Although we do not have access to the distribution information on how these pairs were selected, we do know that for 54% of this 615,512 the source paper cites the target and for the other 46% the source does not cite the target.

The testing set on which we must predict citation links consists of 32,000 pairs and as such is a small fraction of the total training+test dataset. Hence we expect that using the graph generated from the citations given in our training set is a close approximation of the full graph defined by the citation links in both the training set and the unknown testing set.

3 Methodology and Pipeline

To predict the presence of links in a graph there are multiple possible approaches ranging from entirely automated (i.e. devoid of feature creation) to entirely feature dependent. For example one may use a graph-based recurrent neural network to predict whether or not a given source paper cites a target. This would be a featureless approach. We opted to construct features describing both individual papers and source/target paper pairs.

Once we computed our 40-dimensional feature vector for each source/target pair we trained a classifier to evaluate this feature vector and predict whether that pair shares a citation link. After experimentation with feature selection and hyper-parameter tuning we concluded that the Light GBM gradient tree boosting library provided both the fastest and most accurate solution to our citation prediction problem.

In implementing the above prediction methodology we used the following pipeline:

1. Construct the training graph using papers as nodes and gold-standard training set citations as edges
2. Precompute data structures to facilitate feature computation: LSA, k-d tree, shortest path length
3. For each source/target pair: remove the edge (source, target) from training graph build the feature vector then re-add the removed edge
4. Train a classifier on the training set and predict citation links for all pairs in the testing set
5. Construct iterated-training graph by adding citation links predicted in step 4 to the training graph
6. Repeat step 3 using the iterated-training graph instead of the training graph
7. Repeat step 4 i.e. generate final predictions for the testing set

4 Features

In constructing features we attempted to quantify similarities in the given attributes (title, author, abstract, etc.) into variables which would correlate with citation likelihood. We will first enumerate our features by name and then elaborate on those which are not self-evident:

1. Number of common authors
2. Temporal Fit
3. Abstract LSA Distance
4. Title TFIDF
5. 2-3 gram Abstract LSA Distance
6. Node Degree
7. Path Length
8. Citation Check
9. Successors of Predecessors
10. Max Similarity
11. Peer Popularity

Fig 1, List of Features

(3-5) To quantify similarity in titles and abstract we vectorized both by term-frequency/inverse-document frequency (TFIDF) That is to say we created vectors for each document (title/abstract respectively) in which all occurring vocabulary terms are assigned a coordinate and the value of that coordinate is determined by the number of occurrences of that term and by the global number of occurrences across all documents. The Latent Semantic Analysis (LSA) matrix is then constructed by dimensionality reducing the matrix of all TFIDF vectors by singular-value decomposition. Distance between documents is then determined by taking the cosine distance between vectors.

$$\text{CC} = \Sigma_i [E(i, T) * \ log_2(1000 - i + 1)] \ / \ 8500$$

(8) The citation check feature computes the percentage of the 1000 nearest neighbors (by LSA distance) to the source which cite the target. This percentage is

weighted by LSA distance ranking as in the above equation — i ranges over the ordered 1000 nearest neighbors and the E function is the indicator function for whether paper i cites T the target. Similarly we compute the weighted percentage of the 500 nearest neighbors of the target which are cited by the source.

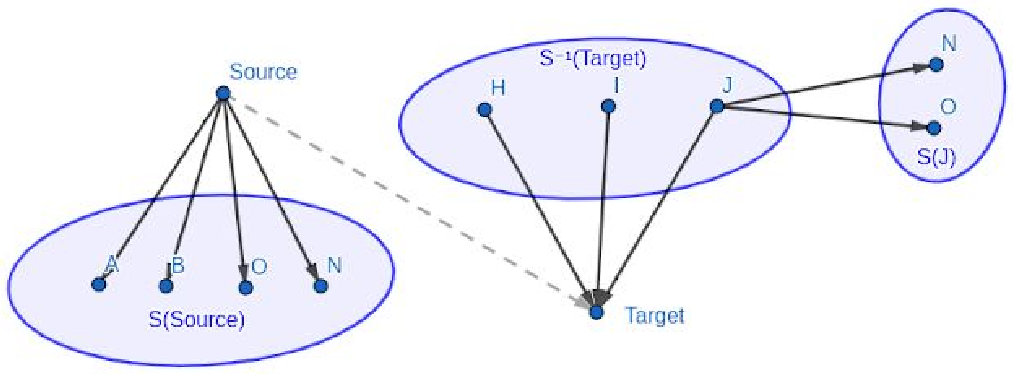

Fig 2, Successors of Predecessors Feature

(9) The successors of predecessors feature compares the set of papers cited by the source — *S(Source)* — to the set of papers cited by papers known to cite the target — $S(x)$ *for* x *in* $S^{-1}(Target)$ (see figure 2). We quantified this comparison in a number of ways: e.g. by taking the normalized maximum cardinality of the intersection $|S(Source) \cap S(x)| / |S(Source) \cup S(x)|$. The feature value would be 0.5 in the above figure.

(10) Max similarity follows the same principle as feature (9), but this time comparing the source to elements of $S^{-1}(Target)$ in terms of their LSA distance instead of their successors' overlap. We then take the mean of the LSA distances from source to the three closest elements of $S^{-1}(Target)$ and as another coordinate the mean of the three furthest.

(11) Peer popularity computes the percentage of nodes in the neighborhood (i.e. successors and predecessors) of Source which cite target.

5 Classification Algorithms

We input subsets of the above feature set into a Logistic Regression, K-Nearest Neighbors, Perceptron, Random Forest, and Light GBM. We normalized the feature matrix for mean and variance and used this normalized data for a SVM classifier. After hyper-parameter tuning and feature selection by cross-validation we determined that Light GBM using gradient boosted decision trees performed best.

We further experimented by ensembling logistic regression, KNN, SVM and a decision tree into a meta-classifier decision tree. This still under-performed relative to Light GBM results.

6 Feature Selection and Parameter Tuning

In order to select optimal subsets of the full feature set we used two approaches: The first is Scikit-learn's Recursive Feature Selection using Cross-Validation, and the second is randomized testing of subsets. We also experimented with polynomial feature creation followed by feature selection, but no combination of polynomial features outperformed the raw feature set.

For parameter tuning we used Scikit-learn's RandomizedSearchCV. From the RandomizedSearchCV results we determined that using the mode of each hyper-parameter (over the top-20 hyper-parameter combinations) showed optimal performance. In our final model, Light GBM we found that the default parameters worked best.

7 Results

Classifier	**Hyper-Parameters**	**Feature Set**[1]	**Result (f-score)**
Light GBM	Default	- 4,9,12,39	0.981
Random Forest	Min_samples_split: 5 min_samples_leaf: 4 max_depth: 6	- 0,1,4,8,9,12,13	0.980
SVM	Kernel: linear Loss: hinge	Normalized, All	0.976
Logistic Regression	Penalty: l1	- 5,26, 27, 29, 30	0.974
KNN[2]	N_neighbors: 9 Weights: l2 distance	Normalized, All	0.971

Fig 3, Optimal Parameters and Results

8 Further Directions

Time limitation prevented us from exploring all possibilities. Notable alternative features we considered but did not implement include using a co-authorship graph, PCA on graph adjacency matrix, deep word embeddings, including unlabeled pairs in training data (by labelling them '0'). We also omitted a number of poorly performing features and variations including inter-journal citation statistics, max sim and citation check using LSA n-grams, using raw TFIDF vectors instead of LSA.

[1] '-' is used to signify that this classifier used all features excluding the given numbers.
[2] Only trained on 150,000 training samples, because of prohibitive time complexity