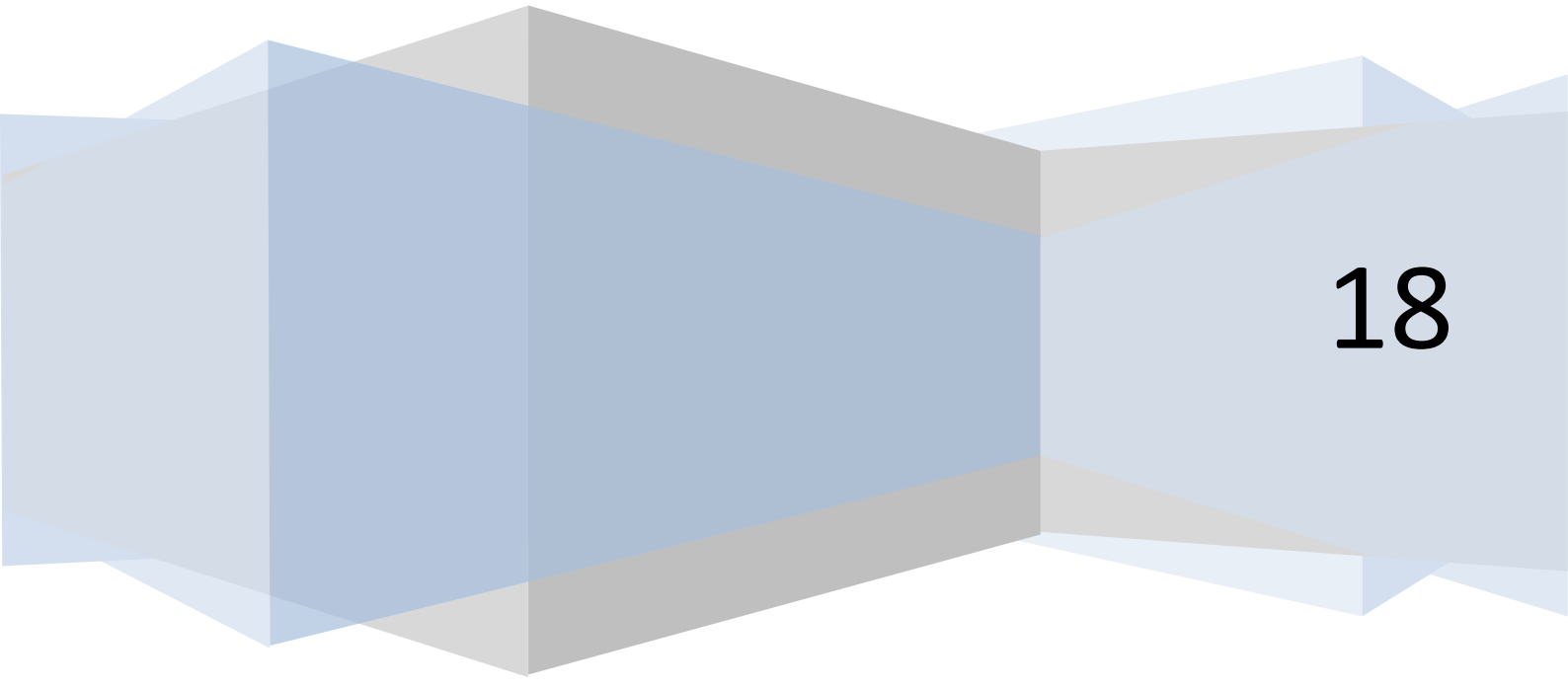
BIG DATA AND CLOUD COMPUTING

Missing links prediction in citation network

Team name in Kaggle:
Smells_like_chicken_nuggets

Mantzoufa Ioanna
Papoutsoglou Iordanis
Stavrothanasopoulos Klearchos



18

Abstract

This document presents our solution for the Kaggle data challenge on the prediction of missing links in a citation network. Our final score is 0.97072 and we are ranked 8th out of 12.

Introduction

A citation network is defined as a graph in which the nodes represent research papers and the edges the connections between these papers. More precisely, there is an edge between two nodes if one of the paper quotes the other. In this challenge, the aim is to accurately reconstruct a citation network in which the edges were randomly deleted.

I. Performance Evaluation

There are narticles = 27,268 articles and nauthors ≈ 15,017 authors. The training and testing sets respectively contain 615,512 and 32,648 data which are tuples (source article, target article, label). The label is 1 if there is a connection between the source article and the target article, 0 otherwise. 54.4% of the training data are labeled 1. Given two articles, we wish to predict whether there is an edge or not for each node pair in the testing set.

The classification performance is evaluated by the mean F1 score. It measures both precision p and recall r defined by

$p = tp/(tp + f p)$ and $r = tp/(tp + fn)$

where tp, f p, and fn stand for true positive, false positive, and false negative respectively. The F1 score is given by

$F1 = 2 \times[(p \times r) / (p + r)]$

In our model, we randomly chose a subset containing 30% of the original training set to learn the parameters (learning set) and 3% of the original training set to compute the F1 score (validation set) in a 10-fold validation.

II. Classifier Selection

Families of classifiers

We considered five families of classifiers:

• **Support vector machines (SVMs)** are a set of supervised learning methods used for classification, regression and outliers detection.

They have many advantages like that they are really effective in high dimensional spaces and they are still effective in cases where the number of samples is smaller than the number of dimensions. Moreover, SVM does not use all the training points in the decision function, but a subset of them in order to be memory efficient. Also, they are really flexible, as different Kernel functions can be specified for the decision function.

However, SVMs have also some disadvantages. One of them is that when the number of samples is smaller than the number of features, over-fitting is possible. In addition, the probability estimates are calculated through an expensive process of five-fold cross-validation, as the SVMs do not provide them immediately.

Even though the support vector machines support both dense and sparse vectors as input, to make predictions for sparse data, it should first have been fit on this kind of data.

• **Random forest (RF)** fits a number of decision tree classifiers on many sub-samples of the dataset and uses averaging to improve the accuracy of the predictions and avoid over-fitting

This classifier has many advantages, with the most important being that it can handle the missing values with great accuracy. Also, it does not over-fit the model and it easily handles large data sets with high dimensionality.

As with all the other classifiers, Random Forest has also some disadvantages. It has a questionable performance on classification compared to regression. Another negative characteristic is that we have no much control over what the model does.

• Logistic regression is a technique based on the logistic function and used for binary classification problems. An advantage of this method is that in terms of robustness is better than the most classifiers, as the independent variables do not have to be normally distributed, or have equal variance in each group. This classifier may handle even nonlinear effects, even though it is not the most appropriate one. Moreover, we can add explicit interaction and power terms, in order to make a more efficient prediction. Another important benefit of logistic regression in comparison to the rest classifiers is that it performs better than them when the signal to noise ratio is low.

• Gradient Boosting involves three elements. First of all, there is a loss function that needs to be optimized and it depends on the type of problem we solve. We can define our own loss function, as there are many supported standard loss functions. The second element is a weak learner, in order to make predictions. In gradient boosting, decision trees are used for this purpose. Additive model includes trees that added one at a time, where existing trees in the model are not changed, is the third element involved in the gradient boosting. This classifier is used to minimize the loss when adding trees.

• Voting

It is an ensemble method where a new classifier is constructed by combining various classifiers. The individual classifiers cast their vote to classify an unseen case. A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse as Hansen et al (5) noted. An accurate classifier is one that has an error rate of better than random guessing on new x values. Two classifiers are diverse if they make different errors on new data points.

In our effort, we combined three classifier giving a single vote to each one of them. The three classifier were the random forest, the logistic regression and Gradient Boosting classifier.

Grid search for parameter selection

The parameters of each classifier were chosen with the exhaustive grid search method. The idea is to specify a set of ranges we want to try, perform a k-cross-validation (with k = 10) and keep the combination that produced the best result.

III. Feature Selection

Data pre-processing

In order to enhance the algorithm's speed, it was deemed as necessary to perform a data pre-processing before extracting the features in a for loop. A single paper could be found multiple times in the training and test set and as a result the for loop would take longer to mine the features. The pre-process consisted of three different actions: 1) splitting the sentence into words, 2) change the words in lower case letters, 3) removing stopwords and performing stemming. The result of this process is to transform a text into a list of words.

Feature Engineering

Feature engineering might be important, but creating many features can be problematic for the classification problem. There is the probability that redundant features with little contribution to the classifier, where created during the engineering phase. The feature selection step is aiming to tackle overfitting, caused by the high dimensionality, by removing these redundant features and lead to a simpler and better performing classifier.

After the end of the feature engineering, a correlation matrix was created in order to detect highly correlated features. Highly correlated features have different effects depending on the used model. Linear models will be affected by the multicollinearity and yield solutions that are wildly varying. Random forests may be good at detecting intersections between the different features, the high correlation can hide these intersections. In other words, dropping highly correlated features can build a more accurate model.

The first feature selection method which was used was Recursive Feature Elimination (RFE). As the name of the method is implying, the feature selection springs from the recursive removal of a feature. The removal process is based upon a selected scoring method. The one which was implemented in our process was the f1 score.

A second method, which was applied, concerning the selection of the best features by the importance to the classifier. A threshold was necessary to be set as the parameter to decide the best features. Features with lower importance than the set threshold were discarded.

Text similarities measurements

The given node information, which was provided, were extensive in text since titles and abstracts were provided. The goal was to enumerate and compare the texts in order to find similarities between them. Different measurements were employed to achieve this goal and the used measurements were the overlapping words shared by a pair of texts, the Jaccard text similarity, and the cosine text similarity.

Before getting to the creation of any of these measurements, a text pre-process step was deemed as necessary. In this pre-processing step, the text was covered to lower case letters. Moreover, stopwords

were removed to have a smaller collection of words. Finally, two stemmers were used to extract the root of words and improve the similarity score. These two stemmers were Porter and Lancaster stemmer as provided by the ntlk library. The comparison between the used stemmers gave us the notion that the model's accuracy did not significantly change.

The first similarity measurement that was engineered was the overlapping words on the titles and abstracts between a pair of papers. It is the most based and simplistic measurement that provides the number of same unique words shared between two texts. The idea behind the creation of the feature was that the greater the number of common words are, the greater the chance that the context of the texts is the same. The pre-processing of the text was critical to remove noise such as stopwords. The two texts were handled as a set of words and the intersection between the two texts was the extracted feature.

The second text similarity measurement was the Jaccard text similarity. The Jaccard similarity is an extension of the overlapping words since the intersection between the sets of words of texts is still used. The difference is that the Jaccard similarity normalizes the similarity by dividing the intersection of sets of words, Jaccard similarity=$\Gamma(x) \cap \Gamma(y) / \Gamma(x) \cup \Gamma(y)$. In other words, Jaccard similarity takes into account the length of the text and the fact that extensive texts will most likely share terms. A flaw in the Jaccard similarity has to do with the absence of the word frequency. Rare terms carry greater importance than frequent ones since they are more informative for the context of the texts.

The last similarity measurement is the vector space text similarity of cosine. The terms were picked to be the different words in the texts and every word in the vocabulary is an independent dimension in the vector space model. The different words were defined by their respective TF-IDF score instead of the base measurement of the word frequency. With the numeric score of a document defined, the similarity of texts was computed as the cosine of the angle between the two vectors.

Based Topological Similarity

Local similarity-based approaches use node neighborhood-related structural information to compute the similarity of each node with other nodes in the network. The neighborhood of a vertex v is the subgraph of G included by all vertices that are connected to v by an edge. The advantages of the local similarity methods are that faster than the nonlocal ones and highly parallelizable. On the other hand, the local methods are restricted to set of adjacent vertices and are not expanded to distance-two vertices.

The common-neighbors predictor captures the notion that two nodes who have a common linked node may form a link between them. This link creation has the effect of "closing a possible triangle" in the graph and feels like a common mechanism in real life. The common-neighbors are defined by the intersection of the sets of neighbors of two nodes, score(x, y)=$\Gamma(x) \cap \Gamma(y)$. Newman has computed this quantity in the context of collaboration networks, verifying a positive correlation between the number of common neighbors of x and y at time t, and the probability that x and y will collaborate at some time after t.

The Jaccard coefficient is a similarity metric that is commonly used in information retrieval. It measures the probability that both x and y have a feature f, for a randomly selected feature f that either x or y has. If we take "features" here to be neighbors, then this measure captures the intuitively appealing notion that the proportion of the neighbors of x who have also linked with y (and vice versa) is a good measure of the similarity of x and y.

The Adamic/Adar predictor measure refines the simple counting of common features by weighting rarer features more heavily. The Adamic/Adar predictor formalizes the intuitive notion that rare features are more telling; documents that share the phrase "for example" are probably less similar than documents that share the phrase "clustering coefficient."

Global similarity-based indices

Global similarity-based indices use the whole network topological information to score each link. These methods are not limited to measuring the similarity between distance-two nodes. However, their computational complexity can make them unfeasible for large networks.

The set of nodes connected through an edge to a node $x \in V$ is called the neighborhood of x and is denoted as Γx. In undirected graphs, the degree of a node x is defined as the number of edges connected to the node and will be denoted as $|\Gamma x|$. In directed graphs, the degree of a node is the sum of the out-degree and the in-degree, which are the count of outgoing arcs and incoming arcs, respectively. The average degree of a

network is denoted as <Γx> and is equal to the average degree of all its nodes.

The negated shortest path is a basic graph similarity measure that requires one to compute the shortest path between two nodes. The shortest path can be computed by Dijkstra's algorithm and then given a negative value, s(x, y) = −|shortest path x,y|. The shortest path prediction accuracy is rather poor and other global similarity methods could be used The advantage of the shortest path is that it possesses a faster converge than the rest of global methods.

TF-IDF

Term frequency-inverse document frequency (TF-IDF) is a common numerical statistic used as a weighting factor for text mining applications. It intends to reflect the relative importance of a word in a document and increases as the number of occurrence of the word increases. The term frequency tf(t, d) of the term t in document d is defined as the number of times t occurs in d. The inverse document frequency idf(t, D) measures the rarity of t across a corpus of documents D:

$$idf(t, D) = \log |D| / |\{d \in D : t \in d\}|.$$

TF-IDF is then calculated as:

$$tf\text{-}idf(t, d, D) = tf(t, d) \cdot idf(t, D) \neq 0.$$

Data scaling

The range of values for the independent features can widely vary and there are various implications on the predictive models. There are classifiers which generally use distances for the classification. When a feature has a wide range of values, the distance will be governed by this particular feature. Another reason has to do with a faster convergence to the gradient descent. In other words, data scaling is a pre-processing step to diminish the range of values of features in order to produce a reliable model.

The scale method adjusts the values around the mean and component-wise scale to unit variance.

The min-max method employs two statistics for the feature normalization as the name implies. The new normalized value is computed as y=(x-min)/(max-min), where x is the value of the feature in every instance. The y are fluctuating between zero and one since x can take values between its minimum and its maximum.

Conclusion

We used a total of six features in our model: two features based on cosine distance between TD-IDF abstracts and titles, the temporal difference between the articles, the number of common neighbors between articles, the adar similarity between the articles and the degree difference between the articles. The best score we obtained was 0.97072 with a gradient boosting classifier with its hyper-parameters tuned by the grid search method. From all the families of classifiers we tested, we observed that the best classifiers were classifiers based on the aggregation of trees (Random Forest, Gradient Boosting).

References

http://scikit-learn.org/stable/modules/svm.html

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

https://machinelearningmastery.com/logistic-regression-for-machine-learning/

(5) Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence, 12(10), 993-1001.