Paper Author Identification Problem
REPORT
Team - 37

Mahtab Sandhu
Vanjape Rajas Mangesh
Venkat Parthasarathy

Abstract:-
This project explores the problem of finding the correct authors given a publication. Several machine learning algorithms have been implemented and their results are discussed in this report.

Introduction:-
The dataset provided is a noisy one as the authors provided for a given publication is a superset of the actual/ground-truth set of authors. This problem arises because search engines if asked to retrieve the authors of a publication might retrieve authors with similar names or the same author who had written papers under different names etc. A simple filtering option will not work because sometimes the results returned may be correct as well. (For example, “Michael Wu”, “Mike Wu” and “M.Wu” may be the same person or different people and a generalization cannot be done.)
To break this problem down, first we need to extract some form of training data from the available data so that supervised algorithms can be applied. So, conditions for unambiguous authors are introduced to use as training data and on top of that supervised learning algorithms such as Random Forest and SVM are implemented.

Feature Extraction:-
Condition for unambiguous author:- If none of the coauthors share the same last name as the author or the author is the sole candidate for that publication.
1) 16 highest tf-idf of matching keywords of p and A.
2) 8 highest tf-idf scores of matching keywords of journal of p and A.
3) 4 highest number of overlapping “non-ambiguous” publications of A and those of other possible authors of p.
4) Number of keywords related to A.
5) Number of keywords occurring in p.
6) The maximum Levenshtein distance between the affiliation of the target author and affiliations of coauthors in the paper.
7) The minimum Levenshtein distance between the affiliation of the target author

and affiliations of coauthors in the paper

Algorithms:-

Naive Bayes:-

Mean and standard deviation of each feature (and the inverse of that feature) are computed using MLE. Assuming each feature is distributed according to a Normal Distribution, we use the mean and standard deviation computed in the previous step and the Normal Distribution Formula to compute the probabilities of each feature. The product of these features and the prior produces the probability that the paper is written by the author. So, for each paper, author pair this is computed and the output is sorted in decreasing order according to these probabilities.

K-Means:-

Given author and a list of ambiguous publications, we find the features of each author, publication pair and run the k-means algorithm with value of k=2. There exists two clusters because one cluster contain the publications that was indeed written by the author and the other which was not written by the author. Then, each cluster is sorted according to a matching score (Levenshtein Distance) and the result is outputted as first the sorted list of the more probable cluster and then the sorted list of the other cluster.

Support Vector Machine:-

The training data of the SVM is the features corresponding to the confirmed/unambiguous publications of the author (condition for non-ambiguity explained in previous section) and a random sample of size twice of the size of the confirmed publications that was not written by the author. The SVM is then trained using two kernels, RBF and Linear, to predict first whether the publication was written by that author and then within each class (0 or 1) sorted according to a matching score. (Levenshtein Distance)

Random Forest:-

The process of obtaining training data is similar to Support Vector Machine. Inbuilt python libraries with default parameters were used.

Decision Tree:-

The process of obtaining training data is similar to Support Vector Machine. Inbuilt python libraries with default parameters were used.

Results:-

For Naive Bayes and K-Means, we were able to submit to Kaggle to verify the score. (0.66 and 0.71 respectively) But, unfortunately as we added more features and now our keywords of a paper did include the title of the paper, the memory increased by two-folds. So, we weren't able to compute results of all 2245 test samples as it was consuming too much time. However, we choose a random sample of size 100 from the test data and verified it with the output of the paper. (which was available in the github repository) The results are presented below:

Classifier	**Mean Average Precision**
Naive Bayes	0.78
K-Means	0.85
Support Vector Machine	0.82(Linear), 0.84(RBF)
Random Forest	0.86
Decision Tree	0.83