Team /'ku:l/

CS60050: Machine Learning Term Project

IIT Kharagpur - Fall 2016

By

12EX20006 - ANKUR AGARWAL

12HS20046 - VINIT HASE

1. Project Title

Identification of Publication House given Editorial Article

2. Problem Definition

We would like to build an engine that would identify certain common trends in editorial pieces of leading newspapers and given a brand new piece can identify the newspaper it was published on.

2.2 Problem Overview

Author identification of a text is a real problem that is faced across many different fields. Academics are constantly on the lookout for plagiarism, while historians and archaeologists often recover unattributed texts whose authors need to be identified.

It is this particular reason that makes this field in Machine Learning particularly attractive and active. It is an open-ended task with many researchers working on it. This problem is, unsurprisingly, challenging and can be solved using multiple different approaches with different success rates.

We in our project aim to contribute our 2 cents to this area of text analytics. Building on the existing body of work we train learning algorithms to identify trends in the editorial pieces published by a news agency everyday and then attribute any given editorial to its publishing house. We think this will be possible because unlike normal reporting, the editorial article published by any news agency is written by a small editorial team and is always in line with the editorial standards of the newspaper. Furthermore, we also believe that every newspaper targets a particular audience and thus the richness of the writing would vary from newspaper to newspaper according to its target audience. Therefore, our underlying assumption here is this:

Every newspaper's editorial team has a "true," statistically-representable style that their works are slight variations on.

3. Methodology

With this underlying assumption, we researched existing literature and similar projects for leads on potential methodologies. We realised the most common methodology used to solve popular text analytics problems such as sentiment analysis and document classification is the semantically based deep text analysis approach. But for our purpose semantics wouldn't be identifiably different from one newspaper from another.

Let's take for example a sentiment analysis problem where we are classifying tweets into positive and negative categories. In this case we know that a positive tweet will have words like 'good,' 'impressed,' 'excellent,' etc., whereas a negative tweet will have words like 'disappointed,' 'bad,' 'enraged,' etc. But for different newspaper articles on the same topic the keywords would all be similar. They would differ from each other more at a level of writing.

It is with this logic that we chose to pursue the shallow text analysis approach as it is well-aligned with our assumption of a statistically based author style and, more importantly, is significantly less expensive to compute.

Further research into shallow text analysis suggested three main approaches:

Token Based: Word-level features such as word length and n-grams.
Vocabulary Based: Vocabulary richness indicators, such as the number of unique words.
Syntactic Features: Sentence-based features, such as parts of speech and punctuation.

Each of these feature approaches have their benefits and drawbacks; token level features are easy but, especially in the case of n-grams (contiguous sequences of n words), slow to compute. Vocabulary features are both fast and easy to compute, but can have a high variance. Syntactic features share many of their qualities with vocabulary features, but can be quite complex to compute.

Our input files are JSON files which contain the editorial articles and some metadata scrapped off the website of 6 different news agencies using scrappy. For this project we have used data from The Guardian, Times of India, Economic Times, Financial Express, Indian Express, and Deccan Chronicle. We were able to scrape off 29,000+ news articles in total.

3.2 Language Model

Given the huge news corpora we had to analyse, we opted to forgo computationally expensive features (such as n-grams) altogether and instead focused on easily computable features, particularly those pertaining to vocabulary and syntax. Just as the long lengths of our texts make the expensive features unappealing, they also make the cheaper ones powerful.

A table is given below which lists the 108 features we will be using and their descriptions. These are very similar to those used by a group of students at Stanford for their Machine Learning project for the course CS229 that is offered there[1]. The vocabulary indicators we have used are standard and widely used. We also decided to look at different parts of speech such as pronouns, conjunctions and prepositions as they both have small word

pools, but are still powerful in determining sentence structure, representing subject and clause shifts respectively. Our final feature vector design treats each editorial as an individual training example and consists of 108 features split across the seven categories listed in Table 1. Note that, aside from a check against a small pool of conjunctions and pronouns, we do not consider the semantic meaning of any individual word; we are only concerned with a word's length and uniqueness.

Table 1. Summary of features used in the language model.

Feature Description	No. of data points
Hapax legomena normalized by number of unique words (not counting stop words) in the text	1
Dis legomena normalized by number of unique words (not counting stop words) in the text	1
The number of unique words normalized by the total number of words (not counting stop words) in the text (as a richness score)	1
Number of stop words normalised by the total number of words	1
The distribution of sentence length normalized by the number of total sentences in the book. The first 25 bins are for sentences of length 1 - 25 words and the 26th bin is for sentences more than 26 words in length	26
The distribution of word length normalized by the number of total words (including stop words) in the book. The first 25 bins are for words of length 1 - 25 characters and the 26th bin is for words more than 26 characters in length	26
The distribution of coordinating and subordinating conjunctions normalized by the number of total sentences in the book. The first 25 bins are for sentences of that have 1 - 25 conjunctions and the 26th bin is for sentences that have more than 26 conjunctions	26
The distribution of nominative pronouns normalized by the number of total sentences in the book. The first 25 bins are for sentences of that have 1 - 25 pronouns and the 26th bin is for pronouns that have more than 26 conjunctions	26

The NLTK library was extremely helpful in feature extraction. Most of our actions like removal of stop words, stemming, parts of speech tagging, etc., were greatly simplified

because of the various NLTK libraries. The code used for feature extraction is available on our github repository for the course.

3.3 Classification Methodology

We used multiple popular classifiers to train and predict, namely - Ridge Classifier, Perceptron, Passive-Aggressive Classifier, Random Forests Classifier, AdaBoost Classifier, Bernoulli Naive Bayes Classifier and Linear Discriminant Classifier. It was made extremely use to implement all of these classifiers using the scikit-learn library.

We carried out a stratified shuffle split with 10 folds, reserving 30% of the data for evaluation. The stratified shuffle split helps preserve the class distribution of our original data in the train-test split. We also carried out some feature selection using a k-Best features metric with k = 2 * num_features / 3 in order to remove redundant features (there were some of these because the text isn't varied enough to provide non-zero values for every data point of every distribution listed in the feature table). We also experimented with PCA but dropped it from the final implementation as there was no improvement in results.

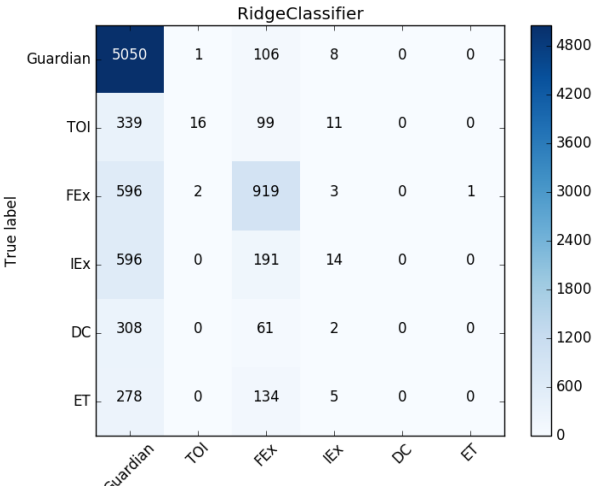
4. Results and Analysis

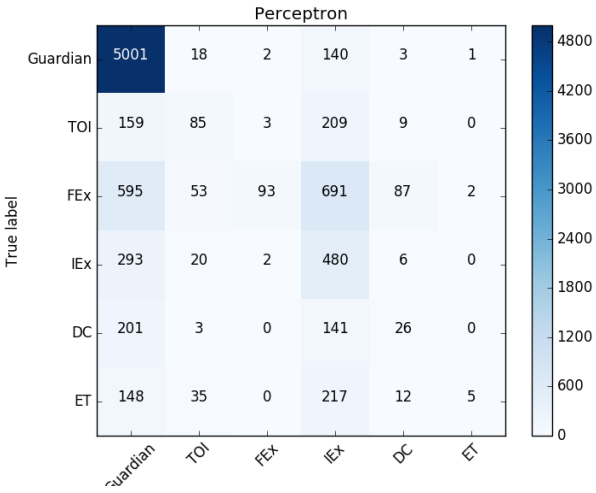
The table below summarises the run time and accuracy of the various classifiers. The highest accuracy that we were able to achieve was close to 77% using the Random Forest Classifier.

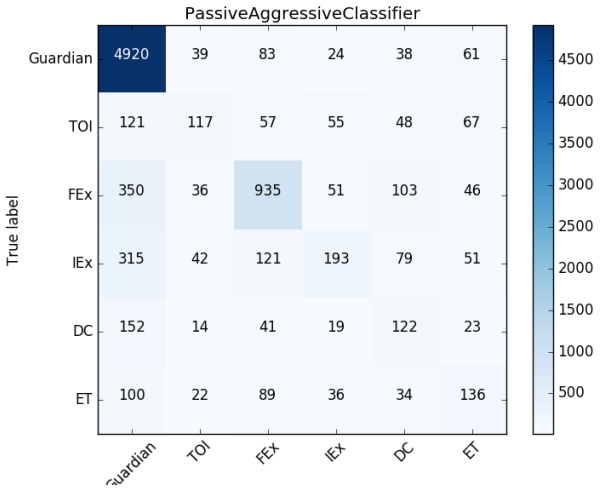
The results are consistent and slightly better than that achieved by the project trying to predict the author of a novel done at Stanford.

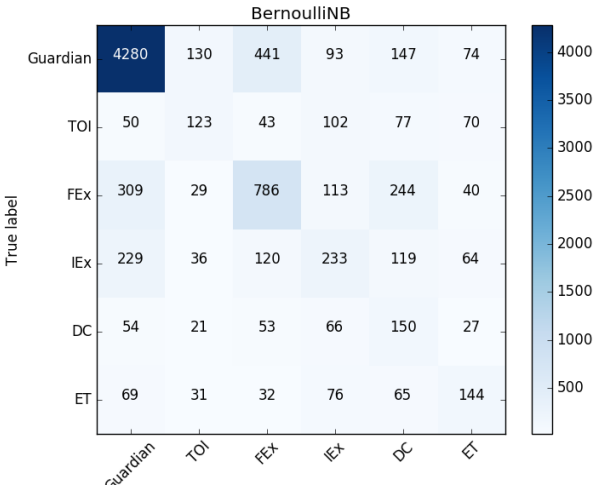
Classifier	Train Time	Test Time	Accuracy
RidgeClassifer	0.179	0.003	68.6
Perceptron	1.512	0.002	65.1
PassiveAggressiveClassifier	1.938	0.002	73.5
RandomForestClassifier	**8.598**	**0.322**	**77.1**
AdaBoostClassifier	**9.688**	**0.186**	**74**
BernoulliNB	0.127	0.015	65.4
LinearDiscriminantAnalysis	0.193	0.002	73.4

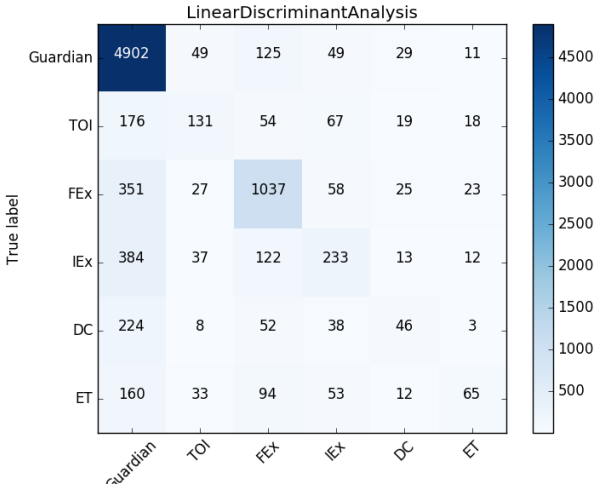
We also looked at the confusion matrices for the different classifiers. They are as furnished below.

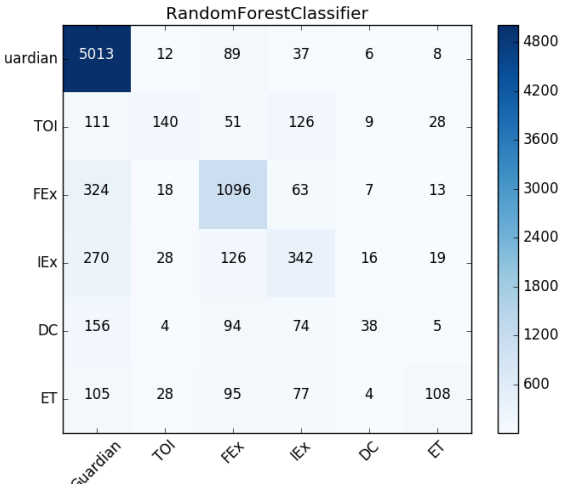


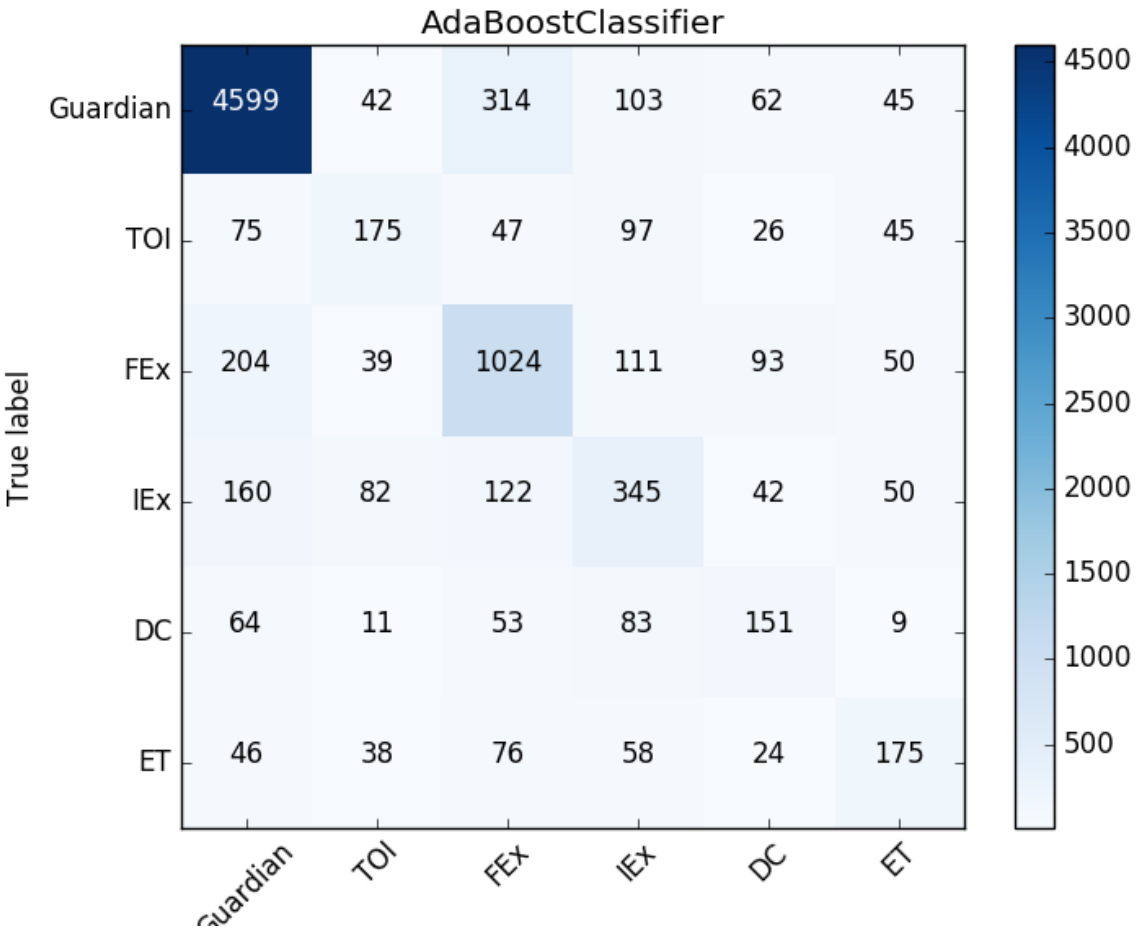












What we were able to observe looking at the confusion matrix is that RidgeClassifiers and Perceptron misclassify a lot of instances, the classifiers which produce better results in our case like the AdaBoostClassifier have a lower rate of misclassification. Also, the rate of misclassification decreases with the increase in the number of training examples. We had the maximum number of articles from The Guardian and Financial Express and the percentage of correctly classified articles is maximum for both of them.

We also tried to break the dataset into smaller binary classification problems taking editorials from only two newspapers at a time. In that case the accuracy improves to somewhere from 82% to greater than 90% for some cases. Accuracy being highest and rates of misclassification lowest when considering the two newspapers with highest data, namely The Guardian and Financial Express.

5. Future Scope of Work

We believe a lot of further work can be done in this field to greatly increase the accuracy of the model and also the type of problems. More than the classification algorithm we think work has to be put in to extract better and more revealing features about the text.

In this project we used the most intuitive measures we could come up with. A more in-depth research into the features of text might reveal better features. Also, more data will also improve the accuracy if the model. These are things we wish to continue working on in the next semester.

We also think more work could be put into identifying the topic on which the article is written. We were able to extract the tags associated with each article and they should help us in topic classification. Furthermore, though ambitious we can also try and identify the perspective from which certain political pieces are written for example whether an article argues for a liberal perspective or a conservative perspective. We believe these will all be interesting and worthwhile problems to work on in this day and age of so many opinions and news sources where we want to choose the news content we consume more carefully and have a healthy balance of perspectives.

6. Acknowledgement

We would like to thank Professor Pabitra Mitra and our course TA Mr. Anirban Santara for their teaching and constant support during the project. We would also like to thank the developers of Scrappy, NLTK and scikit-learn libraries in python which greatly simplified our tasks of scrapping, feature extraction and classification respectively.

It was a great learning exercise for us and we sincerely thank you for giving us this opportunity.

7. References

[1] Stanko, S., Lu, D., Hsu, I. Whose Book is it Anyway? Using Machine Learning to Identify the Author of Unknown Texts

[2] Luyckx, K., Daelemans, W. Shallow Text Analysis and Machine Learning for Authorship Attribution

I. Index