Authorship Identification for Documents

Devansh Dalal(2012CS10224) and Abhishek(2012CS10208)

Abstract

Authorship identification has been a very important and practical problem in Natural Language Processing. The problem is to identify the author of a document from a given list of possible authors. A large amount of work exists on this problem in literature. We develop ideas based on this work in order to build our own model for authorship identification. We also take a model from this work as a baseline for comparing the results. Our model for the task is a text classifier based on logistic regression which includes n-grams, style markers and document finger-printing as features. *Reuter_50_50* is the dataset used. The baseline achieves an accuracy of 73.08% on this dataset. We were able to achieve a close accuracy of 72.11%.

Introduction

Authorship identification has been a very important and practical problem in Natural Language Processing. It has become even more important with the widespread use of various forums where anonymity is allowed. A good system for authorship identification can handle cases of misuse of anonymity. Such a system can also be used for plagiarism detection. Analysis of authorship of historical texts has already been done many times using techniques for authorship attribution. Various spoofing e-mails can also be detected by using these techniques.

Previous Work

The problem of authorship identification has been studied extensively. Many hand-designed features have been developed in order to tackle this problem. For solving this problem, conventional text classifiers like naive bayes, logistic regression and SVM have been used, and various deep learning models like LSTM have also been tried. Different models perform differently depending upon the type of training and testing data available.

We take the results of [1] as baseline for comparison. We use the same data so that comparison makes sense. The work on authorship identification before [1] used fixed length n-grams only. This work was different in the way that it used variable length n-grams as features. It mainly used 3-grams, 4-grams and 5-grams as it was well-established that subsequent n-grams don't impact the model much. After introducing the various n-gram features, it extracts the most frequently occuring n-grams using a feature selection method. The final feature values are fed into a linear SVM for classification. The accuracy obtained by this system for *Reuter_50_50* dataset was 73.08%. We have tried to replicate these results. However, the most recent work on this dataset is [2] where the authors have obtained an accuracy of upto 75.48%. This model uses very complicated mathematical techniques which are completely different from the ones we have discussed in the class. Therefore, we did not choose it as our baseline.

Dataset

We have used the *Reuter_50_50* dataset which was easily available at the website [3]. It contains 50 training documents and 50 test documents for each of the 50 authors. All of these documents are on the same topic of news. Every document has been written by one and only one author.

Our Model

It is a supervised learning task as a label is present for each of the training examples. Since there are multiple authors and we have to assign the document just one of them, it is also a multi-class single label text categorization problem.

Our model for solving this problem is basically made up of these 3 components: logistic regression classifier, stylometric features or style markers, document finger-printing. Similar ideas have been tried out in the past but not necessarily in this particular combination. We have used Python programming language for implementing our model. We have used logistic regression classifier from the Python SciKit Learn library.

Stylometric Features

Stylometric features or style markers try to capture the writing style of the author. We have used a large number of stylometric features in our model. The motivation behind these features comes from the survey paper on authorship attribution [4]. The features pertaining to types and token are:

number of types (unique words), number of tokens, type-token ratio. Some authors have a tendency to write long sentences whereas some authors prefer shorter sentences. In order to capture this, we have included various features in our model such as average length of a sentence (both in terms of words and characters), standard deviation of sentence lengths. Some authors pose more questions as compared to others. For assimilating such ideas in our model we have introduced features like relative number of declarative, interrogative, imperative and exclamatory sentences. Other style markers include relative number of punctuations, relative number of digits in the document etc. Natural Language Tool Kit (NLTK) has been used to extract these style markers from the text.

Document Finger-printing

Fingerprint identification is a well-known technique in forensic sciences. The basic idea of identifying a subject based on a set of features left by the subject actions or behavior can also be applied to other domains. Identifying text authorship based on an author fingerprint is one such application. Finger-printing has mainly been used in the past for plagiarism detection. Basically, the idea is to generate a compact signature for each document and then those signatures can be matched for similarity.

We used 2 types of document finger-printing, character level finger-print generation and word n-gram level finger-printing. In character level finger-printing, we first create a set of all $k(k >= 5)$ length character n-grams. Then we select a small subset of these character n-grams based on their hash values using the Winnow's algorithm[5] implemented in this python fingerprint library[6]. The hash value of the sequence of chosen character n-grams is returned as the signature of the document. To make the process efficient we used fingerprint library. Similarly word level features can be used and similarity between 2 documents can be computed as the number of matches found. This word-level analysis is very slow if implemented normally. So, we used DARMC Intertextuality Analysis Library[7] for effective computation of the similarity function.

EXPERIMENTATION AND RESULTS

The implementation was done in Python using the commonly used ML and NLP libraries like scikit learn, Theano , NLTK etc. The code is available in this github repository.

We started out with the simple naive bayes classifier which gave a decent accuracy. Then we tried various other classifier models of which logistic regression performed the best for us. So we decided to go ahead with it. Then we added various stylometric features to our model. This led to a considerable increase in accuracy. After that we went further to add the document finger-printing features which again led to a significant increase in accuracy.

The accuracies obtained on the various models mentioned above have been tabulated in Table 1.

model	accuracy
NB	67.15%
SVM	69.1%
Logistic Regression (LR)	70.76 %
LR + stylometric features	71.36 %
LR + stylometric features + document finger-printing	72.11 %

Table 1: accuracies of models with progressive features

OTHER METHODS EXPLORED

Long short-term memory (LSTM) We also tried LSTMs as they have been proved to be very effective for this problem in the literature. Implementation of LSTM was done in theano, by modifying the tutorial code given for LSTM on theano official website [8]. The initial accuracies were not good. One problem that we suffered from was the lack of data. We had only 16 MBs of text, used for training and testing of 50 classes. Since LSTMs require a large amount of data for getting appreciable results, they did not help in effectively solving our problem. Therefore, we did not further consider the approach of using LSTMs for solving our problem. Table 2 lists some results for LSTM.

classes	accuracy
first vs all	92.5%
first vs second	85%
5 authors	67.2 %

Table 2: results for basic LSTM architecture

Knowledge of most confused authors Due to data insufficiency, we restricted ourselves to simpler models that use more number of features. For performing error analysis on our model, we constructed confusion matrix for the authors. From there we tried to use the knowledge of most confused authors for improving the performance of our model. One key observation was that the confusion matrix was a sparse matrix. This means that an author is confused with only a small number of other authors and not with almost every other author. In particular, there are certain pairs of authors who are confused with each other. For disambiguating between such authors we trained smaller models made exclusively for those authors. First a confusion matrix is created for the predictions on the training set. Then from the confusion matrix we pick top-k most confused pairs of authors (depending on the total number of mis-predictions among them). We train smaller models on each of these selected pairs. Now for prediction on a test example, if the predicted author is found to be one of the confused authors, we use the majority of the outputs from the smaller models. This method increased accuracy by a small amount. However, this method is computationally expensive and therefore we could not use it effectively.

Lemmatization and POS tagging We considered replacing the words with their root forms using Porter Stemmer lemmatization [9]. But it did not result in an increase in accuracy. We also tried the syntactic feature of appending the words with their Part-Of-Speech (POS) tags. Again this idea

did not work out in our favour. These features try to hide the actual word information. Perhaps due to this they did not improve the accuracy.

CONCLUSION

The accuracy obtained by our model is close to the accuracy of the baseline model. This shows that stylometric features and finger-printing features are crucial to the task of authorship identification. Also for the used dataset, traditional classifiers like naive Bayes, SVM and logistic regression are performing better than neural network models like LSTM.

FUTURE WORK

One promising approach is to use LSTMs with more amount of data and to build on the previous work[10][11] added with all the stylometric features and document finger-printing ideas. Also the idea of using the confusion knowledge from training data for disambiguating between certain pairs of confusing authors seems to improve the results. The document similarity function is still very slow and therefore we are using only a part of the text written by an author as the representative text for that author. With more efficient methods , it might be possible to include more text for each author. Since we need a lot of data for more complex models, work can be focused on creating a large authorship dataset, preferably using a web crawler. One potential source of authorship data is Quora.

The classifiers that we used were linear in nature. Therefore, it seems that the use of non-linear models can increase accuracy as non-linear models will also be able to capture non-linear properties of the documents. Adding semantic features like synonyms and antonyms can also be tried.

References

[1] John Houvardas and Efstathios Stamatatos. N-gram feature selection for authorship identification. *AIMSA*, pages 77–86, 2006.

[2] Zongkai Yang Zhi Liu and Sanya Liu. A novel random subspace method for online writeprint identification. *JOURNAL OF COMPUTERS*, 7(12), 2012.

[3] Zhi Liu. `http://archive.ics.uci.edu/ml/datasets/Reuter_50_50`.

[4] Efstathios Stamatatos. A survey of modern authorship attribution methods.

[5] Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85. ACM, 2003.

[6] kailashbuki. python fingerprint library. `https://github.com/kailashbuki/fingerprint`, 2012.

[7] Brendan Maione-Downing. Darmc intertextuality analysis library. `https://github.com/bmd/intertextuality-tool`, 2013.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Vivake Gupta (v@nano.com). porter stemmer. `https://github.com/vbuterin/spread/blob/master/spread/porter.py`, 2012.

[10] Anand Pandey and Ankit Pensia. Cs671: Natural language processing.

[11] Pranav Jindal and Ashwin Paranjape. Deanonymizing quora answers.