Authorship Attribution - Blog Data

Amogh Akshintala 108645000
Prankur Gupta 108492684
Sarang Joshi 108493027
{aakshintala,prgupta,sajoshi}@cs.stonybrook.edu

December 17,2012

Abstract

There are times when authorship of a document is disputed. Linguistic analysis of a text to determine authorship is a well established field. Computer Scientists, however, have only recently begun statistically analysing the same. However, most work has focused on significantly large data such as novels. We attempt to extend the same to much smaller writings such as blogs which typically tend to be only 3-4 paragraphs long. We compare a baseline of N-gram word based classification with various other features and present our findings.

1 Introduction

The goal of our project is to build a classifier that, given a small number of writings by a particular author and a mix of blogs by other authors who write on the same domain, should correctly identify the author when presented with an unseen blog by one of the authors it was trained on. We are trying to test the hypothesis that this is achievable given the limited size of data that is generally available in real world scenarios. In our case, we classify the blogs into two categories - A) written by Paul Graham, B) Written by other authors (here others authors can be anybody).

One of the challenges in this project is that the amount of positive data is minute in comparison to the amount of negative data available, for which we can pick from a wide pool of authors. Another challenge is that authors tend to write domain specific and mixing in data from authors who write on other fields is going to be counterproductive as this degenerates into a genre classification task if too much data is included but the lack of such data is a problem too. In our case Paul Graham writes blogs about Startups and Entrepreneurship as he runs a seed venture capital company Y-Combinator. We studied the possibility of building a classifier that correctly identifies the author of a document given a small training set and present the results which we have obtained in the paper.

2 DataSet

The dataset used, provided with this document, had to be collected by hand as we didnt find any data set that satisfied out needs. It consists of 6 authors from the domain of Entrepreneurship and Venture Capitalists,

(a) Paul Graham - 115 Blogs
(b) Ben Horowitz - 87 Blogs
(c) Fred Wilson - 90 Blogs
(d) Josh Kopelman - 102 Blogs
(e) Bill Gurley - 25 Blogs
(f) Mark Suster - 12 Blogs

We have also included for the sake of variety so as to increase the entropy of the data 117 Blogs by Matt Welsh, who blogs on Computer Science in general. We will consider his blogs to check our results for domain independent analysis. We are also on the active lookout for more quality bloggers whose work we can include to further strengthen our results.

3 Classifier Used and parameters

We used the SVM classifier from LibSVM, SMO and Naive Bayes Classifiers provided by WEKA. We also looked into the "Random Forest" classifier, which is based on a collection of decision trees. During training on the SMO classifier, we played around various parameters, some of which we list below:

(a) c - complexity constant (default 1)
(b) k - kernel to be used
(c) x - for cross validation
(d) N - Normalization Factor
(e) W - random seed
(f) T - Random Seed

We have encoded the feature vector based on tf_idf values.

4 Features Used

Apart from using basic features like, Unigram, BiGram and Trigram of Language Model, POS Sequencing and PCFG, we have also attempted some of our own stylometric features after looking at the dataset. As our blogs are written in good English and the topic of the blogs is also intellectually sound i.e. Venture Capital and Entrepreneurship, the blog data doesn't contains any informal data, which we may encode as features. We looked at the dataset and choose stop words carefully and contructed a feature from that. Apart from this, we also noticed the target author gave examples/proof of each incident/conclusion he reported in his blogs. So, we made a feature based on how many examples/points he uses in his blogs. We devised yet another Parse tree based feature namely "Parse Tree Complexity", which will be explained in its sub-section. We will also talk about a novel feature based on frequency common words, (CFF), which gives a pretty good accuracy for binary classification between two authors.

4.1 Language Models

In this section, we will discuss about our approach for encoding features based on language models.

Initially, when we did n-gram analysis namely unigram, bigram and trigram, the accuracy for each of these n-gram analysis was 100%. This was expected since most of these documents contain the names of their individual companies. This is in some sense cheating on the classification. We took a small dataset, of about 50 blogs in total, for this experiment and divided the dataset into training and test data by 80-20% rule, performed 5-fold cross validation. Even in this case, as expected, there was 100% accuracy for binary classification.

At this point we are very sure that this is due to the fact that every author is biased towards his own startups, so they tend to compare other startups with theirs. They also tend to use the same set of companies/names of people associated with these companies very often especially as examples. In order to deal with this, we replaced every proper noun (NNP) with a unique tag. In doing so, we believe we have effectively removed all bias without throwing away any other information. One could argue about how we could be throwing away useful information which would help in properly identifying the authorship of the blog. Our counter argument to that is that, what about when the authors happen to compare their companies with another authors companies? Another case where this would be bad is if one author, who used to write about company X, moved to company Y and now if the test data contains blogs by him on company Y. Thus it seems to us, a good idea to replace all proper nouns with a unique tag in our particular case.

We repeated the experiment with the full dataset after having replaced all proper nouns. We ran POS tagger on the training and test data and replaced all the nouns (i.e. words in the untagged document that were tagged NNP in the tagged corpus) with NNP keyword. The data was then fed to the SMO, NaiveBayes and LibSVM classifiers. For training and test data, we again applied 80-20% rule. For all n-gram analysis, we got more than 95% accuracy with highest being for unigram i.e. 98%. The tuning parameter C for SMO and LibSVM was taken as 0.01. The accuracy for NaiveBayes and LibSVM was better than accuracy for SMO classifier. The results are populated in Table 1 below. The data used for this experiment is
Training: Paul Graham - 90 documents, Other Authors - 233 documents
Test: Paul Graham - 25 documents, Other Authors - 88 documents

Table 1: Language Model

Approach	Features	Accuracy	Paul Graham			Other Authors		
			P	R	F	P	R	F
	$Unigram_{SMO}$	98.23	92.6	100	96.2	100	97.7	98.9
	$Unigram_{LibSVM}$	98.23	96	96	96	98.9	98.8	98.9
	$Unigram_{NB}$	98.23	96	96	96	98.9	98.8	98.9
Text Categorization	$Bigram_{SMO}$	88.50	83	83	78.4	87.1	89.2	93.1
	$Bigram_{LibSVM}$	95.57	91.7	88	89.8	96.6	97.7	97.2
	$Bigram_{NB}$	95.57	91.7	88	89.8	96.6	97.7	97.2
	$Trigram_{LibSVM}$	97.34	95.8	92	93.9	97.8	98.9	98.3
	$Trigram_{NB}$	97.34	95.8	92	93.9	97.8	98.9	98.3

4.2 POS Sequencing

In this section, we performed experiments on two different types of training and test data set. We did the usual binary classification, Positive being blogs written by Paul Graham, and Negative being blogs written written by various different authors of the same domain. Table 2 contains results obtained when the training and test data set contains equal number of positive and negative data, i.e.

Training - 90 blogs by Paul Graham, and 90 by others.
Test - 25 blogs by Paul Graham and 25 by others.

Table 2: POS Sequencing on Equal Dataset

			Paul Graham			Other Authors		
Approach	Features	Accuracy	P	R	F	P	R	F
POS Sequencing	POS_{SMO}	72	65.8	96.2	78.1	91.7	45.8	61.1
	POS_{NB}	52	58.3	26.9	36.8	50.0	79.2	61.3
	$POS(2)_{SMO}$	96	96.2	96.2	96.2	95.8	95.8	95.8
	$POS(2)_{NB}$	90	95.7	84.6	89.8	85.2	95.8	90.2
	$POS(3)_{SMO}$	98	100	96.2	98.0	96.0	100.0	98.0
	$POS(3)_{NB}$	90	83.9	100.0	91.2	100	79.2	88.4

In the table 2, the results are based on two classifiers SMO, and Naive Bayes. POS is the simple POS tagging based feature vector in which, we first convert our training and test dataset to POS tagged dataset, and then encode the features as a unigram language model of POS. The result clearly shows that encoding simple POS feature vector is a bad idea, because as we have already explained, we have a smaller dataset i.e. less number of words in each document and also less number of documents. So it is not a surprise that we achieved a lower accuracy, as different authors do end up using around the same number of different POS in their blogs. As POS- Unigram is just the count of the number of parts of speech used by a particular author in a document, we feel the need for a better feature encoding in order to achieve better accuracy, as each author tends to have different writing styles, which is not very well represented by POS tags as unigrams.

Thus we decided to do POS-sequencing. POS(2) - means we did Bigram count of the POS tags, i.e. 2 POS tags taken together. We considered this to be a feature vector. And as we expected, it showed much better accuracy. As each author has a different sentences structure, writing style definitely impacts each and every sentence and its formation, which is basically POS sequences. Our accuracy went up particularly greatly. As these POS tags combined together gives us important information regarding the structure of sentence used by the author, using POS sequencing in our case, makes a lot of sense. POS(3) - means combining 3 consecutive POS tags as one feature vector. The result again showed a very good improvement over POS(2).

We can go for larger POS sequences, but it will just increase the computation cost as well as data sparsity. The better way to deal with much higher POS sequencing is the syntactic analysis of each sentence, as even if we go on increasing the POS sequence, it might boil down to the problem of finding the length of each sentence in the blog. Because using a POS sequence which go beyond the average length of each sentence in a blog will just increase the problem of data sparsity and would not be helpful to our cause anyhow. In order to extend this POS sequencing to a safe level, we aim at deriving parse trees, and then extracting the first level of the parse trees.

Table 3 contains results obtained when the training and test data set contains unequal number of positive and negative data, i.e.
Training - 90 blogs by Paul Graham, and 252 by others.
Test - 25 blogs by Paul Graham and 66 by others.

Table 3: POS Sequencing on Unequal Dataset

Approach	Features	Accuracy	Paul Graham			Other Authors		
			P	R	F	P	R	F
POS Sequencing	POS_{SMO}	70.32	33.3	3.8	6.9	71.6	96.9	82.4
	$POS(2)_{SMO}$	91.2	78.1	96.2	86.2	98.3	89.2	93.5
	$POS(2)_{NB}$	93.4	95.5	80.8	87.5	92.8	98.5	95.5
	$POS(3)_{SMO}$	97.6	97.8	90.5	94	97.5	98.5.0	97.9
	$POS(3)_{NB}$	83.52	69	76.9	72.7	90.3	86.2	88.2

When the number of blogs written by other authors dominates (refer Table 3), then POS tagged feature vector gives us almost 0 accuracy w.r.t to the blogs written by Paul Graham. It classifies all the blogs to the others while testing, the reason to which is the POS tagged blogs of other authors dominates, as it just tells about the number of POS used in a blog. For a particular author itself, it can vary. So as we have trained our classifier on a large amount of Negative data, which have a variety of number of POS tags used in a blog, so the accuracy has dropped very low.

But when we consider the case of POS sequence tagging e.g POS(2) and POS(3), it shows us proper results with good accuracy, as POS sequence tends to utilize the style of the author. Individual author tends to use the same POS sequence throughout the blog, POS sequence feature vector uses that base and gives very good results. In order to improve the results we can also do nested cross validation, or go on higher POS(4).

4.3 Syntactic Analysis

In this section, we will talk about the deep syntactic feature which we dealt with. As we had hypothesised earlier, the sentence structure of an author is generally a good indicator of his style of writing. We generated the parse trees of all the sentences in all the documents obtained using the Stanford Parser and extracted what are known as level 1 fetures from the parse trees. Level 1 features are essentially the RHS of the grammar that was used to derive the tree at the first level. We then used these as unigram features and fed the tf-idf values for each feature to SMO and Naive Bayes classifiers and the results are tabulated in the table below. For this experiment we used unequal amounts of data as describe below:
Training - 90 blogs by Paul Graham, and 252 by others.
Test - 25 blogs by Paul Graham and 66 by others.

Table 4: Parse Tree Level1 Feature

Approach	Features	Accuracy	Paul Graham			Other Authors		
			P	R	F	P	R	F
Parse	$Unequal_{SMO}$	90.38	89.3	92.6	90.9	91.7	88	89.8
Tree	$Unequal_{NB}$	94.23	100	88.9	94.1	89.3	100	94.3

PT(1) represents the unigram of the Parse tree POS tags at Level 1.

The sentence structure is used here to classify authors based on the kind of writing styles they follow. Although the results are quite good (90%+ accuracy) it doesnt perform as well as the other two styles of modelling. Our analysis is that it could be that these authors, since they all write in the same domain, and probably know each other, read the same material and possibly even each others blogs. Writing style is heavily influenced by what you read so we hypothesise that doing a bi-gram modelling of the level1 data may or may not result in a significant improvement in the results because of the reasons mentioned above. Even going below level 1 will also not give us significant improvement in the results.

4.4 Stylometric Analysis

In this section we will discuss about the features we encoded based on the particular writing style of our author.

4.4.1 Stop Words

After going through the data set, we made note of stop words which the target author tended to use dominantly and compared it with the other authors. Some stop words frequency matched but still there was a good number of stopwords which were not used a lot by other authors compared to the target author.

We encoded the frequency of these stop words as a feature vector. The result is shown in the table below. The result are not too great but still they makes sense, as were are doing a domain specific classification here. So, the count of stop words used by different authors may vary but not by a great difference.

Table 5: Stop Words Feature

Approach	Features	Accuracy	Paul Graham			Other Authors		
			P	R	F	P	R	F
Stop Words	$Equal_{SMO}$	86	82.8	92.3	87.3	90.5	79.2	84.4
	$Equal_{NB}$	84	78.1	96.2	86.2	94.4	70.8	81
	$Unequal_{SMO}$	87.9	74.2	88.5	80.7	95	87.7	91.2
	$Unequal_{NB}$	85.71	90	77	66.7	83.3	91.2	90.9

Here, $Equal_{SMO}$ tells that the classifier SMO was fed with equal amount of positive as well as negative data. While $Unequal_{SMO}$ tells that the classifier SMO was fed with more negative data compared to positive.

4.4.2 Points, Examples and Sources (PES)

Another important feature which we came across was, many authors in this domain, while writing about a topic lay a lot of emphasis on the reason, and sources of their conclusion about a topic. They tend to write points and examples specifying how they came to that conclusion. We thought it might be a good feature to encode. This feature was encoded using the points brackets e.g. {,), (etc , words like source, examples, following, point etc, and doing a unigram analysis of these features. And the results are shared in table 6. The good thing about that approach is it makes sense with our data set. As the authors tends to write blogs on various facts from relied sources and tries to base their results with proofs.

Table 6: PES Feature

			Paul Graham			Other Authors		
Approach	Features	Accuracy	P	R	F	P	R	F
	$Equal_{SMO}$	88	85.7	92.3	88.9	90.9	83.3	87
PES	$Equal_{NB}$	84	78.1	96.2	86.2	94.4	70.8	81
Feature	$Unequal_{SMO}$	94.5	100	80.8	89.4	92.9	100	96.3
	$Unequal_{NB}$	89.01	75	92.3	82.8	96.6	87.7	91.9

4.4.3 Parse Tree Complexity

We came up with a new feature that we call Parse Tree Complexity. The idea behind this is that sentences written by different authors may be of the same length but may resolve to parse trees of varying complexity as per the inherent complexity in the sentences. Thus we set out to measure this and the simplest way that this can be done without losing much data is to count the number of nodes in the parse tree. This is better than counting the number of leaf nodes as the number of leaf nodes is generally the number of words in the sentence. Some sentences may resolve to simpler parse trees without too many subtrees in them or some may be entirely winded and complex and may result in very large parse trees. We opine that this is correctly captured by our heuristic of counting the number of leaf nodes in the generated tree and using this as a feature to classify based upon.

This however does result in pretty poor performance in case of unequal dataset but extremely good performance in case of equal dataset as can be seen in Table 7 (around 70% accuracy) as compared to the baseline. However, this feature provides a pretty good measure if we want to do a binary classification between two particular authors. Still it may show similarity between writing styles of different authors due to the fact that this complexity doesn't deal with the way individual authors mix their sentences, just bases itself off of the size of the tree generated. So, if it is used with some other feature, it would be more accurate.

As you can also observe, that recall for positive data is 100%. So, we are able to correctly classify the positive data, but we do not do

Table 7: Parse Tree Complexity Feature

			Paul Graham			Other Authors		
Approach	Features	Accuracy	P	R	F	P	R	F
Parse Tree	$Equal_{SMO}$	95.9	100	92.6	96.2	90.5	100	95
Complexity	$Unequal_{SMO}$	69.9	44.3	100	61.4	100	60.5	75.4

well with negative data. Because, here in the negative data, we have included blogs from 6 other authors, and there is good probability that some author writing style can actually match with our author of interest. So, if could just use this classifier with some another classifier, we might achieve better results.

4.4.4 Common Frequency Feature

In this section, we will discuss about another novel approach. In this approach, we consider two authors at a time. We count the occurance of each character n-gram for both the authors. We then compare both these list and discard any sequence that were not common. This was done to ensure that very infrequent n-grams, such as those that appear only once in the entire training set, would not be included, nor would n-grams that appear very often in one of the author's texts but not at all in others.

Table 8: Common Frequency Feature 1

			Josh Kopelman			Matt Welsh		
Approach	Features	Accuracy	P	R	F	P	R	F
Common Frequency Feature	CFF_{SMO}	97.95	100	95.2	97.6	96.6	100	98.2

Table 9: Common Frequency Feature 2

			Josh Kopelman			Ben Horowitz		
Approach	Features	Accuracy	P	R	F	P	R	F
Common Frequency Feature	CFF_{SMO}	97.43	95.5	100	97.7	100	94.4	97.1

Table 10: Common Frequency Feature 3

Approach	Features	Accuracy	Paul Graham P	Paul Graham R	Paul Graham F	Matt Welsh P	Matt Welsh R	Matt Welsh F
Common Frequency Feature	CFF_{SMO}	92.45	86.2	100	92.6	100	85.7	92.3

In tables (8-10), we discuss about the binary classification between two particular authors, and show the accuracy achieved by our feature. Against some other authors we were also able to achieve 100% accuracy, which is quite justifailble, as this feature is based on the frequency count of common words and their n-grams. So, what if the distribution of these words are totally bizzare which we can use in our favor. Like in case of Ben Horowitz and Paul Graham, if you look into the frequency of words that are commonly used by both of them, it is evident. Words used by Paul Graham very frequently are used seldomly by Ben Horowitz.

Table 11: Common Frequency Feature 4

Approach	Features	Accuracy	Paul Graham P	Paul Graham R	Paul Graham F	Ben Horowitz P	Ben Horowitz R	Ben Horowitz F
Common Frequency Feature	CFF_{SMO}	100	100	100	100	100	100	100

Our classifier trained on common frequent word n-grams can successfully discriminate between authors with fairly similar styles. More generally, common frequent n-grams seem to offer an adequate representation of an author's stylistic *Signature*.

5 Multi Classification

In this section, we will talk about the approach we followed for multiclassification. We did Unigram of Language Model with respect to rest of the authors, keeping others authors as our main authors, one at a time. Then given a blog, we ran it through different models of the classifier SMO. We used the blog of Paul Graham and ran it with other author's model. That blog was accepted for 2 authors, including Paul Graham itself. So, if we will now calculate the accuracy of our multilevel classification, it will be

$$\text{Accuracy} = (1 - \tfrac{N}{T}) \times 100$$

where, N - Number of wrong accepted models
T - Total number of wrong models tested against.

In this case, Accuracy = 83.33%.

6 Issues Faced

As, our dataset/blogs are written by experts in their respective fields. So, trying to find features based on misspelt words, informal words etc, are not applicable. And also, some of our novel features like Parse Tree complexity, if used indiviadually might not be that effective, but if used with any other feature, could really make a difference.

Another major problem we faced was while doing multiclassification. As there were authors in our dataset, who wrote very less blogs on the domain. So, there was a very few training data of positive to be trained on.

7 Conclusion

We manually collected and segregated (i.e. human annotated) a blog dataset for various different authors of the same domain. Some of them were quite active and wrote hundreds of blogs, and some were not that active, hence, we ended up with a small dataset. We incorporated some basic features like Language model, POS and PCFG, along with some custom features like PES and PTC, CFF, which show promise but can't be used alone for a general binary classification where we compare one author with a large number of other authors. If used with some other feature, they may give us more accurate results.

We conclude that, using some of the novel features suggested by us, we can do binary classification between two particular authors very accurately, and if we want to do binary classification between a certain target author and rest of the authors, we need to use some another feature along with our novel features, in order to achieve high accuracy.

8 References

(a) Syntactic Stylometry for Deception Detection - Song Feng,Ritwik Banerjee,Yejin Choi

(b) Domain Independent Authorship Attribution without Domain Adaptation - Rohith K Menon, Yejin Choi

(c) N-Gram-Based Text Attribution - Patrick Leahy