Who Got Style?

Learning to recognize literary styles

CS 229 Final Project
Fall Quarter 2012

Konstantinos Balafas
Simon Calvet

Introduction

Most well known authors have a distinct writing style that renders their work unique. For the erudite reader, that style is quite obvious and it is not hard to tell between Jane Austen and Charles Dickens or Leo Tolstoy, to name a few. That distinction, however, requires skill and knowledge that can take a lifetime to attain. For that reason, the application of machine learning algorithms in author identification is a problem that has garnered a lot of research interest. Our objective for this project is to apply some of the algorithms that we learned in class on the problem of author identification. More specifically, we will pose the following two problems:

- Select a book and choose between the actual author and another.
- Select a book and identify the author from a larger pool (containing the actual author)

Dataset

Our dataset consists of 112 books from 16 different authors. The books were downloaded in .txt format from the Project Gutenberg website (www.gutenberg.org). While the books and the authors, as well as their numbers, were selected arbitrarily, we tried to keep most authors between 5 and 10 books. Once the books were selected and downloaded, they were compiled into forms that could be used in our algorithms. Each book was compiled firstly into a vector of all the words appearing in the book and their occurrences. Secondly, books were compiled into vectors of the frequency of each word and sentence size.

Word Occurrence Approach

A fairly obvious measure of an author's style is the words that they use. As such, our first idea was to train an algorithm on the occurrences of words in a book. As expected, this led to input vectors of very large size, since every word that appeared at least once in any of the books in our dataset had to be accounted for. Naturally, this meant that the algorithms that we would have to run would be very computationally expensive. In order to remedy that, we tried using a stemming algorithm that would reduce the size of our input vectors. This makes sense from a linguistic point of view as well; our first approach, with a "full" dictionary differentiates between the occurrences of different forms of the same word. In other words, the "full" dictionary will consider "do" and "does" as completely different words, assuming that each one influences the style of the author to a different extent, something that is obviously not true. A stemming algorithm would lump these words together, providing more insight on the style of an author. After a brief online search, the Porter stemming algorithm (http://tartarus.org/martin/PorterStemmer/) was used. Furthermore, clearly being in a supervised learning setting, we tried Naïve Bayes and Logistic Regression as a first approach.

Naïve Bayes

A Naïve Bayes was implemented and tested for two different problems.

- The first will be called **binary classification**. Selecting two authors and a book that was written by one of the two, we want to predict the actual writer. It was tested for a few author couple, using Leave One Out Crossed Classification (LOOCV), and each time had 0% error.
- The second will be called **One Versus All**, and is a way to apply Naïve Bayes to multivariate prediction. When a prediction has to be done, the algorithm is first trained on all authors, labeling 1 if the author considered wrote the book and 0 if not. When testing on a given book, the author selected is the one that gives the highest probability of "being written by the considered author". This approach surprisingly gave good results: using LOOCV, the error obtained was 18%.

Logistic Regression

As one of the simpler supervised learning algorithms, we tried using logistic regression for our problem. At first, we used logistic regression for **binary classification**. Using LOOCV and all possible permutations of author pairs, the final test error was 29%. Furthermore, the error was not significantly affected when the reduced dictionary that was obtained from the stemming algorithm was used. Using the reduced dictionary did, however, greatly reduce the time that the algorithm needed to run.

In trying to select the author of a book from the entire set of authors, the following procedure was followed:

A logistic regression algorithm was trained for each author on the entire dataset with the exception of one book. The label 0 was the case when the book did not belong to the author and the label 1 was the case when the book belonged to the author. The quantity $h_\theta(x)$ was calculated for each author and the predicted author was selected as the one with the largest $h_\theta(x)$. Firstly, the input vector contained the occurrences of words. Due to the fact that there were some words (such as "a" or "the") had a much larger number of occurrences than others and that the zero-label training examples were a lot more, θ'x was a very large number and the exponential of it tended to infinity and the algorithm failed. Then, the occurrences were normalized by the number of words in the book (essentially representing frequencies). Even in that case, this algorithm had 100% test error.

Word/Sentence Length Approach

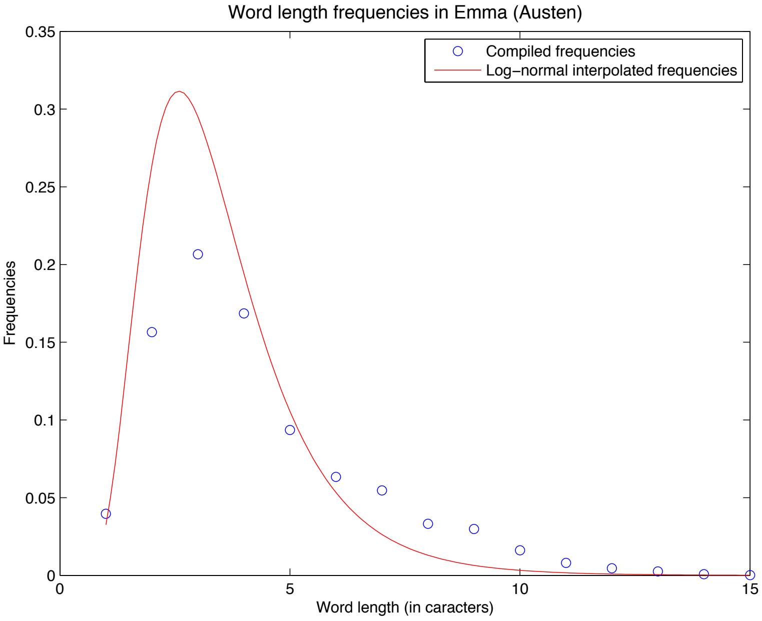

Figure 1

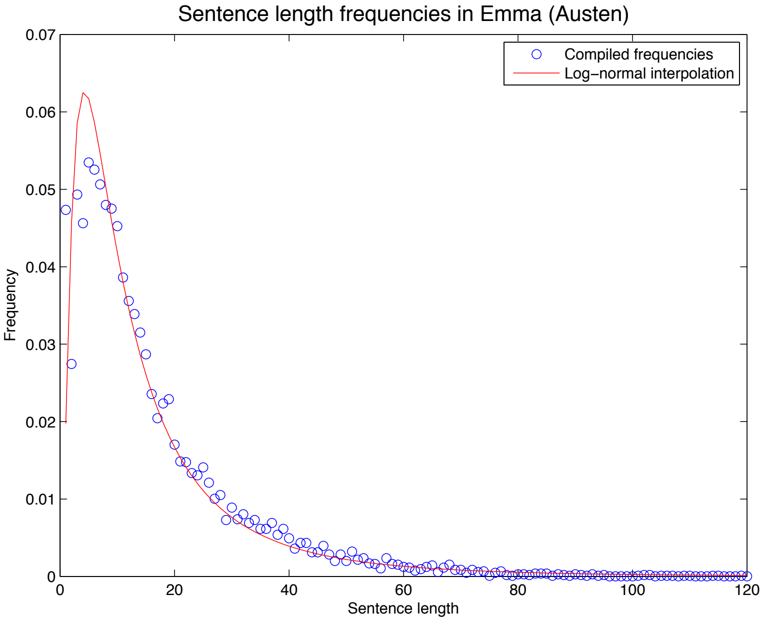

Figure 2

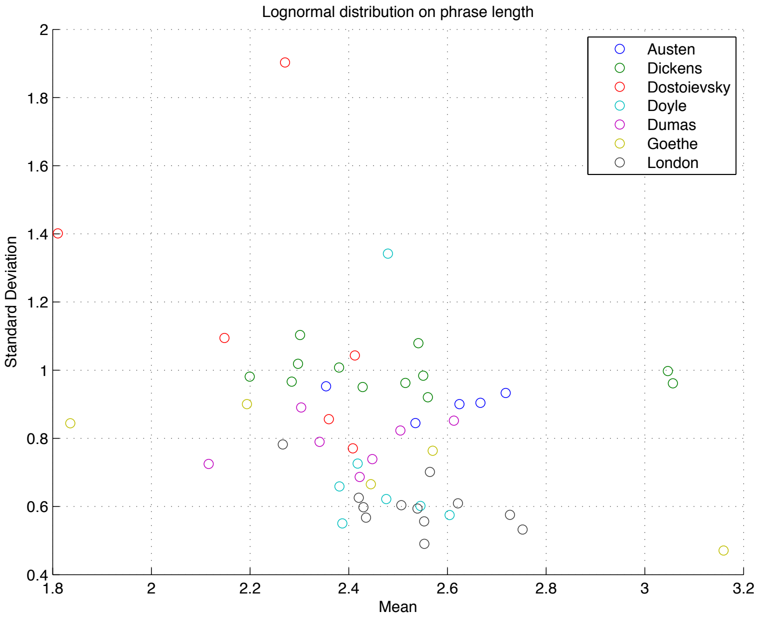

Figure 3

Our language is made of words that are characters combinations. Under certain grammatical rules, words are also combined to make sentences. When we write, we have a personal way to combine words to make sentences that will be long or short depending on our style. For instance, Proust's style is recognizable because of his usage of very long sentences. To some extend, word's length usage might also be an indication of some writer's literary style.

Instead of building a vocabulary for our dataset and then count for each book the words' occurrences, frequencies for each word and sentence lengths were computed for each book, and each book was then represented by two vectors:

- One containing the word length frequencies, with the ith component being the frequency of the words of i characters
- The other containing the sentences length frequencies, with the ith component being the frequency of the sentences made of i words

This approach has a double advantage: first it reduces considerably the size of the data. Word lengths frequencies are null for sizes higher than 20 characters, and sentence lengths frequencies are usually null for length of more than 200 words. But it also makes it possible to actually visualize the data. Figure 1 and Figure 2 show the frequency distributions for respectively word lengths and sentences lengths for Austen's novel *Emma*.

But we can go further and notice that both distributions look like they are log-normally distributed. And since our vectors are discrete, it is very easy to get the mean and the variance, the only two parameters that fully describe a given distribution. If we note X(i) the frequency of the word or sentence of length i, we have:

$$\mu = \sum \log(i)\, X(i) \; \mathit{and} \; \sigma = \left[\sum \log(i)^2\, X(i)\right] - \mu^2$$

This allows us now to represent a book in 4 dimensions:

$$[\text{book}] = [\mu_{\text{word}}, \sigma_{\text{word}}, \mu_{\text{sentence}}, \sigma_{\text{sentence}}]$$

Figure 3 shows a reduced form of the data in 2 dimensions for a reduced set of authors (only 7 of them), using the mean and variance for phrase lengths. This shows that using this criteria, some author can be distinguished, Austen and London for instance, whereas some couple of authors cannot be separated using linear regression or some clustering algorithm, London and Doyle for instance.

Naïve-Bayes and logistic regression were tried on both frequency vectors and interpolated parameters vectors [μ_{word}, σ_{word}, μ_{sentence}, σ_{sentence}]. For each existing author couple, LOOCV was used and the error incremented when the algorithm was making an incorrect prediction.

For Naïve Bayes, surprisingly, the reduced 4-component vector test gave a lower error: 46% compared to 47% for the frequency vectors. Also, the error for each couple was calculated and was highly fluctuating (from 25% to 100%) depending on the couple considered. That confirms the visual intuition we got looking at Figure 3.

For logistic regression, the one versus all approach gives surprisingly better results than the word occurrence approach. The binary classification

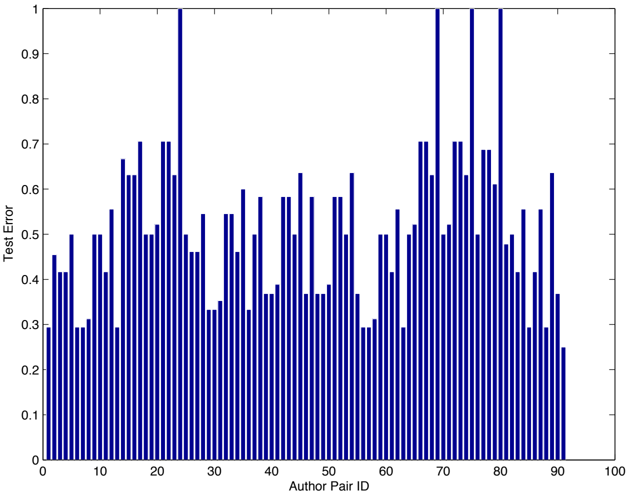

Figure 4

performance strongly depends on the pair of author considered though, as it is shown on Figure 4.

Conclusion and Suggestions for Further Work

The errors for each algorithm and set of data used are displayed below:

Data used	Logistic regression	Logistic regression "multivariate"	Naïve-Bayes	Naïve-Bayes "multivariate"
Word occurrence	29%	100%	0%	18%
Word occurrence with stemming	29%	100%	0%	18%
Word/sentence distribution	51%	89%	25-100% 47% ave.	100%
[book] = [μ_{word}, σ_{word}, $\mu_{sentence}$, $\sigma_{sentence}$]	51%	89%	25-100% 46% ave.	87%

Naïve Bayes for word occurrences works very well to recognize one author in a pool of two. It works well also when trying to recognize an author in a larger pool. Running the algorithm takes a lot of time though since matrices of size of about 40,000 components are manipulated. Also, the other attempt to make shorter vectors showed that phrase lengths and word lengths frequencies were of some significance to characterize one author's style. However, the error found is very high on average. Some authors may be distinguished with these parameters but it cannot be applied universally to all authors. In particular, the 100% error for the one versus all Naïve Bayes can be explained by the way the algorithm works, since it choses between the 13 authors. Also, surprisingly, one-versus-all logistic regression performs better using the frequencies or interpolation parameters than when using word occurrences.

Since input data was entire books here, it appeared early in our work that the information that we chose to extract from them was of high importance for the performance of our algorithms. We chose to work with relatively simple algorithm, logistic regression and Naïve Bayes, to focus on the data extracted and its relevance to the style recognition problem. Finding the relevant data is key here, and should we have more time, we would spend it on this key issue.