EECS 349 Project Final Report
Josiah Evans, Ben Meline
6/8/2016

Problem

Authors typically have a style of writing that is consistent across their various texts. Even though they may write about many different topics they often structure their texts in a similar fashion and use certain parts of speech more frequently than others. Based on this knowledge, our goal for this project was to create a model that can determine which famous author wrote a given text sample. We were able to extract numerous features from text samples, such as part of speech tags, sentiment, word length and sentence length to use in our analysis. We hoped to see if a random text sample could be matched to an author from a list of authors in our training set.

Dataset

For our dataset we collected books from Project Gutenberg, which is an online source of copyright-free books. Books from many famous authors are available, such as William Shakespeare, Mark Twain and Jane Austen. For our training data we first downloaded the top 100 most popular books from the past 30 days. For the test set we wanted to find different books for authors that are included in the training set. We ended up deleting some of the books in the training set, if there was no other book available for the test set. In the end we had 76 books by 56 authors for the training set. For the test set we used 56 books, one for each author. Because certain authors haven't written that many books we weren't able to have a very large dataset.

For the training and test sets the books were then split up into smaller text samples. Each sample was 350 lines of text, which we approximated to be one chapter of a book. We wanted to split up the books because we assumed that there could be variations in style within a book, so we wanted each book to have many different examples. Our training set had 2788 examples and the test set had 1452 examples.

To extract features from the examples we used TextBlob, a natural language processing package for Python. We used the "tags" function to get a count of the number of parts of speech, such as proper nouns, adverbs and adjective, in a given text sample. We determined the percentage of words in a text sample that belonged to each part of speech. We used sentiment analysis to get subjectivity and polarity. Lastly we found average word length and average sentence length in each example.

Approaches

Initially we thought that k nearest-neighbor would be the best approach for this project because we thought there would be clusters of points for each author. We tested this approach as well as others in Weka. When we were testing different classifiers we used 10-fold cross validation to evaluate their accuracy. We found that kNN did not perform as well as other classifiers. We also tried J48, Random Forest, Naive Bayes, Bayes Net and Multilayer Perceptron.The best classifier was the Multilayer Perceptron.

To improve the accuracy we tried removing features in Weka. However, we found that in general removing features only decreased accuracy in 10-fold cross validation. We did find one feature that could be to remove accuracy. This was the “list marker” part of speech, which is used for numbered lists. Very few examples had this parts of speech, so it only added noise.

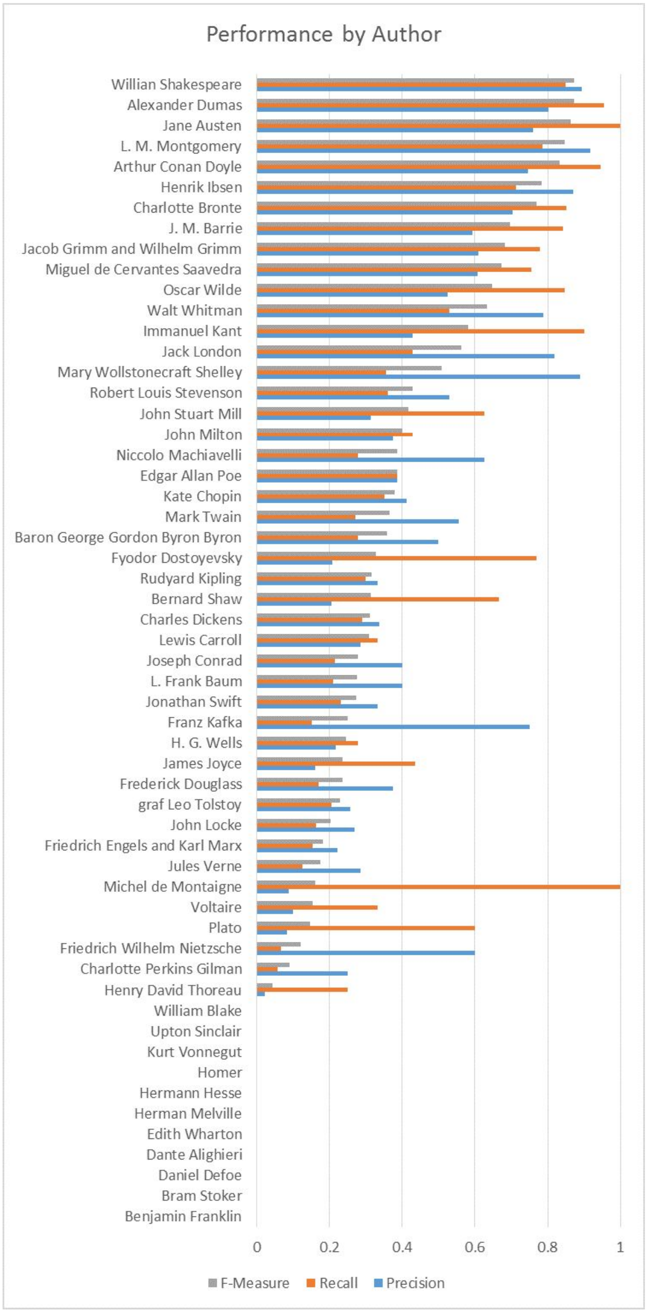
Percentage of correctly classified instances with 10-fold cross validation on training set

Classifier	**Correctly Classified Instances**
ZeroR	10.7604%
J48	57.7834%
Naive Bayes	74.6055%
Bayes Net	75.2511%
Random Forest	81.0617%
Multilayer Perceptron	84.2181%

Results

After determining that multilayer perceptron was the best classifier using 10-fold cross validation, we used the test set we had set aside previously. The test set achieved 44.146% accuracy. For comparison ZeroR had 2.686% accuracy. We found that for certain authors the model achieved very high (90%+) accuracy. For other authors the accuracy was near zero. We looked at the texts for some of the authors with very low accuracy. Benjamin Franklin, for example, had a book in training “Autobiography of Benjamin Franklin” and “Franklin’s Way to Wealth” in testing. The style of the first book was a biography and the second one was a letter. As a result, they had very different styles and we had low test results for this particular author. On the other hand, for William Shakespeare we used “Romeo and Juliet” in training and “Hamlet” in testing and had very high accuracy. This is likely because both these books are tragedies and contain significant amounts of dialog. Certain books had large introductions and notes that were not written by the author of the book, so this also resulted in lower accuracy for those authors.

We believe neural networks to be a good classifier as we have 39 features for examination, and all the features are continuous data. As neural networks are less susceptible to problems of dimensionality and noise, they beat out decision trees, kNN, Naive Bayes, and random forest classifiers.



Future Work

Currently our training and testing sets only include approximately one book for training and one book for testing. Since an author may choose to vary style, vocabulary, and structural composition across books, spending the time to devote more books to the training set would likely improve accuracy.

Conclusion

The output of our model on the test data suggests that authors do indeed have invariant features that can be used to identify unclassified texts. However, the scope of our training data was too small to reliably distinguish between book-invariant and author-invariant features. If the test text was from a book with a significantly different genre, type, or style the model consistently scored lower accuracy. Even with a small data set for training, the results were significantly better than random, suggesting that it is possible to find author-invariant features.

Contributions

Josiah wrote a script to get books from Project Gutenberg and then he sorted these books into training and testing sets. Josiah and Ben worked on code to extract features from the examples. Ben wrote code to sort and output the feature extraction results and tried different classifiers in Weka.