Ewen Wang

Novel Author Classification

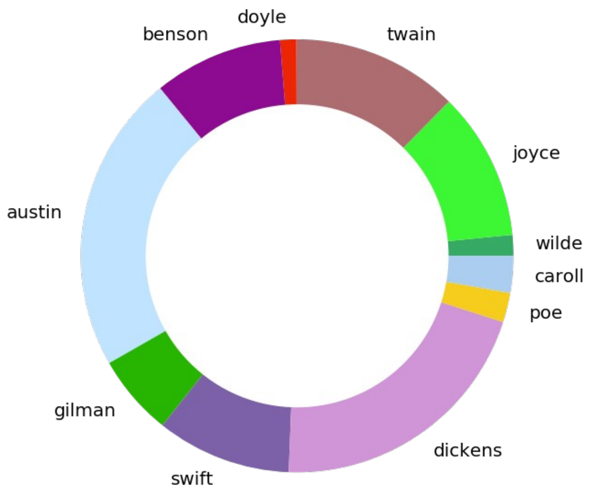
Introduction

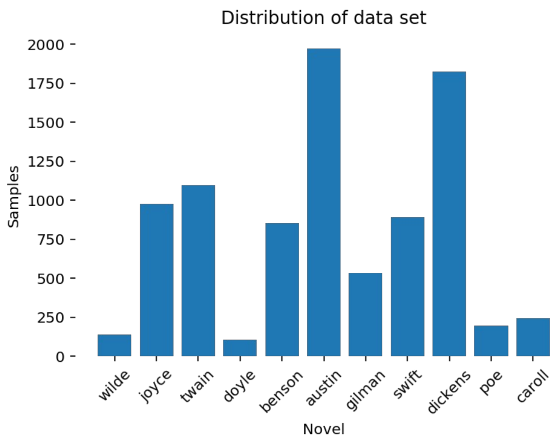
An accurate authorship identification system can be critical to solving authorship dispute problems in texts. Modern text and data mining techniques can automate the attribution of authorship by analyzing language patterns in the author’s past works. This is especially useful for detecting plagiarism detection and labeling ancient documents as the amount of digital literature grows rapidly. Many of the existing research on this topic have been binary classification between two authors. In this project, we look at the effectiveness of techniques for feature selection and classification models in the context of discriminating between 11 authors.

The source code of this project can be found here: https://github.com/ewenw/AuthorML

The Data

The data consists of 54 classical full text novels collected from the Gutenberg project. To imitate the context of classifying a single page of a novel, we divide the raw text into samples by 2,500 characters (including white-spaces), the average in a page of novel, and ending at the nearest words.

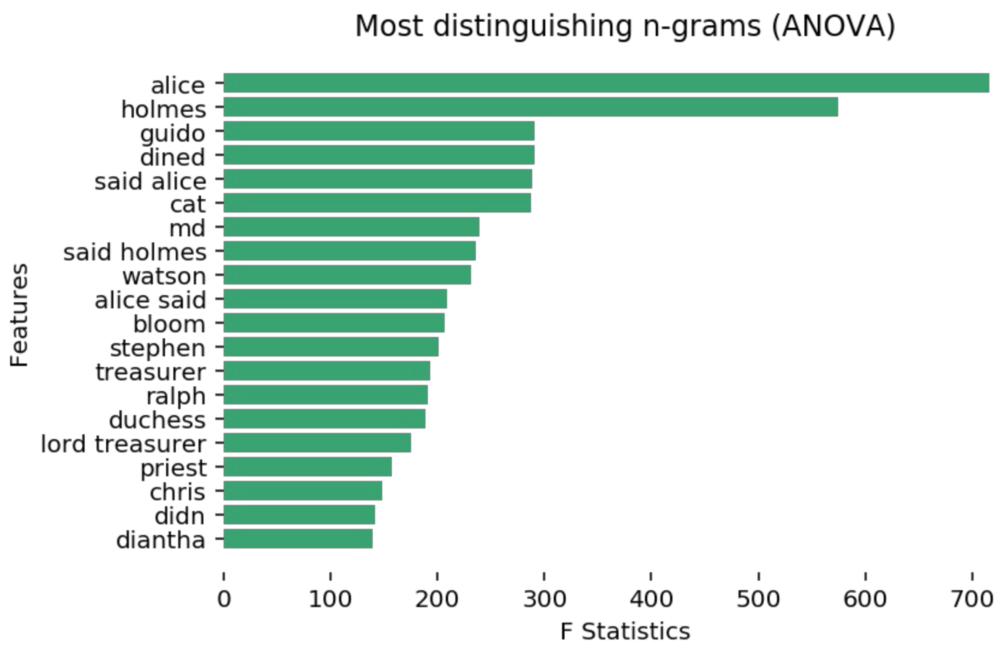


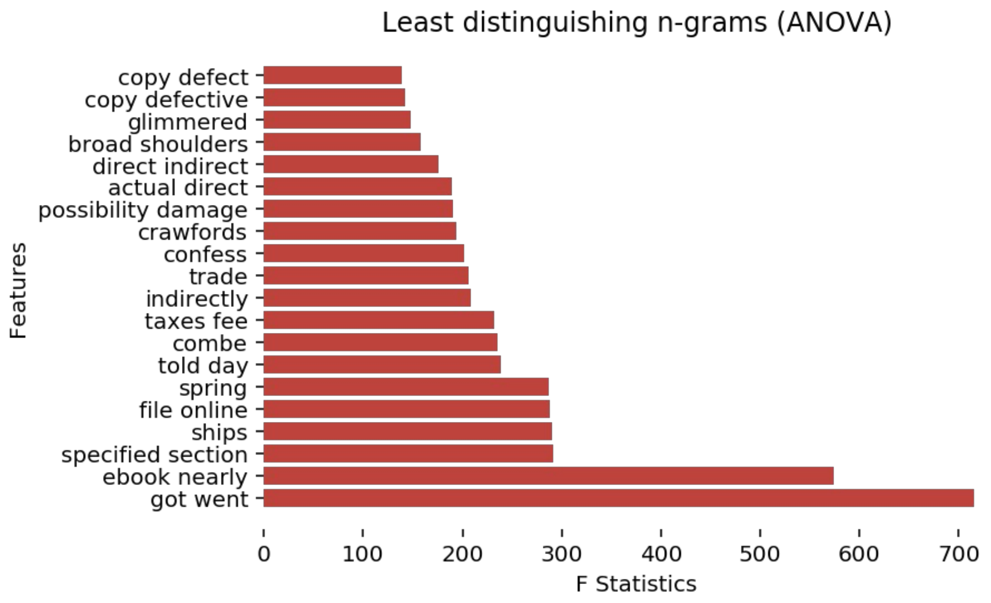


Clearly, the distribution of the authors' samples is highly skewed, but this gives us a more realistic insight into how our models perform when the authors have a varying number of past works that we can use as training samples.

Feature Engineering

To capture the most distinguishing features between the authors, we could capture the commonly used words, phrases, paragraph lengths, and word variety. However, in novels containing many one-liner dialogues, the paragraph lengths and word variety would be uninformative in telling apart the authors. Therefore, my system attempts to capture useful words and phrases by extracting the frequency-inverse document frequency (TFIDF) of uni-grams and bi-grams having 0.001 to 0.1 document frequency to filter out words and phrases that are too rare or common. The vectorizer also ignores common English stop-words. Next, we perform a feature ranking to see the most informative features through an ANOVA test. The mean frequency of each feature between the 11 classes is compared through an F statistic that indicates whether the feature is informative in classification.





Currently, our data is represented in a 8,829x23,379 sparse matrix. Since many phrases and words are used in the same contexts, we can use Principal Component Analysis (PCA) to reduce the dimension of the data by compacting the correlating features in eigenvectors. By capturing 90% of the original variance, we reduce the number of features from 23,379 to 3,953 while sacrificing the interpret-ability of the final results. One challenge of performing PCA on a large data-set is the amount of memory it takes up (23,379 x 23,379 x 8 Bytes ~= 4.37GB). Although using incremental PCA can estimate the eigenvectors through batch-processing, I allocated more memory from disk space to preserve the accuracy of the PCA.

Now that we have reduced the columns drastically by compacting the variances of correlation features, we can further filter out ones that don’t correlate with the author labels by selecting the top 1,000 columns of the PCA results that have the highest F-test statistics. Overall, the number of features has been reduced from 23,379 to 1,000, while maintaining most of the information.

Although cross-validation would yield a slightly more realistic picture of how our models perform on unseen data, training each model multiple times on a matrix of this size on a layman’s laptop would only drive us closer to insanity. Therefore, we randomly split the data into 75% for training and 25% for testing.

Model Performances

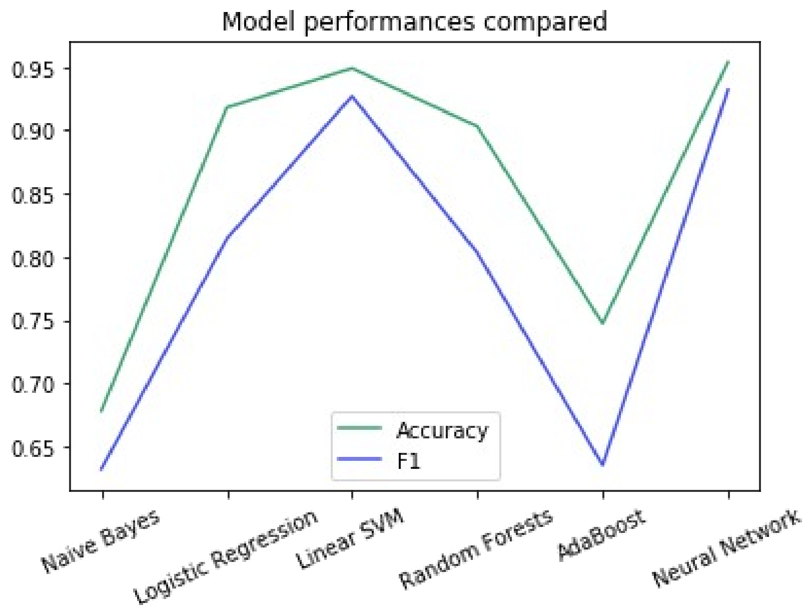
As a basis for comparison, a naive random classifier has an expected accuracy of 1/11 = 0.091. I train six types of classifiers on the training data, and evaluate them on the testing set through four metrics: accuracy, precision, recall, and f-score (macro averaged). To perform hyper-parameter tuning on SVM, Random Forest, and AdaBoost, we train four models of each with varying parameters and select the one that yields the highest accuracy (shown in the Jupyter Notebook code).

Neural network architecture:

1. 1,000 input neurons, ReLU, 25% dropout
2. 100 densely-connected neurons, ReLU, 25% dropout
3. 11 output neurons, Softmax

Model	Accuracy	Precision	Recall	F-score
Feed-forward neural network	0.954	0.967	0.906	0.932
Linear SVM (C=1.8)	0.949	0.968	0.895	0.927

Logistic regression	0.918	0.95	0.759	0.814
Random forests (2,500 trees)	0.903	0.857	0.772	0.803
AdaBoost (2,000 trees)	0.747	0.769	0.596	0.635
Naive Bayes	0.678	0.752	0.7	0.632



Conclusions

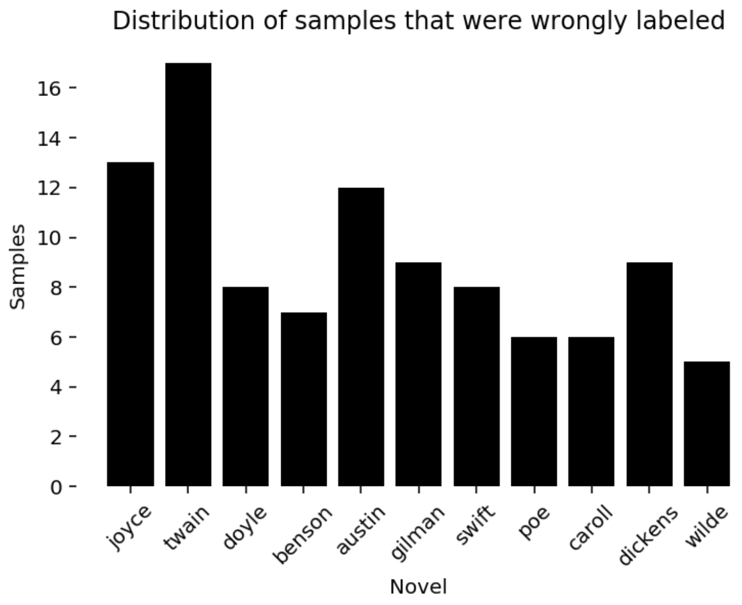
Overall, the Neural Network and linear SVM classifiers yield the highest accuracies and F-scores. This indicates a linear decision boundary in the data and that predicting authors using the features extracted from our process is informative. To further show the process's effectiveness, we train and test a linear SVM model on the raw features before the PCA and filtering with the same model parameters.

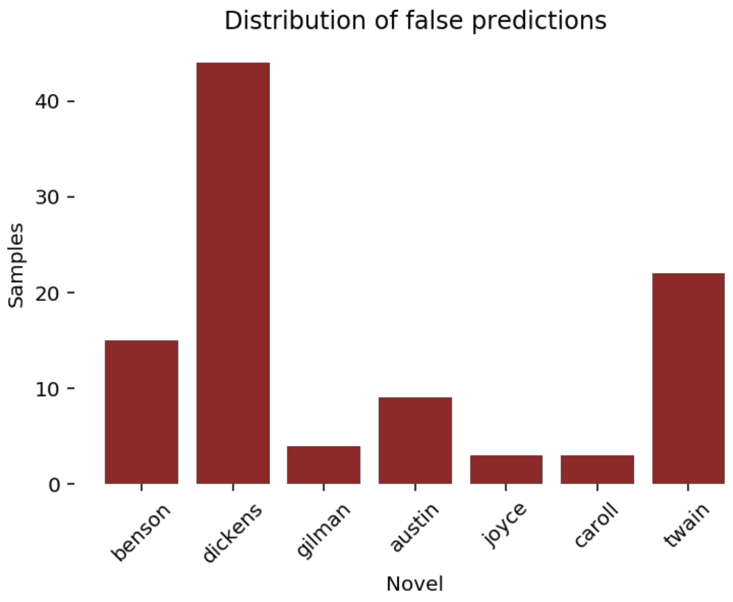
```
Linear SVM (C=1.8) on raw features.

Accuracy, Precision, Recall, F-score
0.952       0.945       0.891    0.914
```

Despite the slightly improved accuracy (0.949 to 0.952), the F-score decreased (0.927 to 0.914), indicating that there's no loss of significant variance from the vectorizer's outputs. To get an understanding of why 4.6% of testing data is mis-classified at best, let's examine small sections of two randomly selected documents:

```
1. YOU HAVE NO REMEDIES FOR NEGLIGENCE OR\nUNDER STRICT LIABILITY, OR FOR
BREACH OF WARRANTY OR CONTRACT,\nINCLUDING BUT NOT LIMITED TO INDIRECT,
CONSEQUENTIAL, PUNITIVE\nOR INCIDENTAL DAMAGES, EVEN IF YOU GIVE NOTICE
OF THE\nPOSSIBILITY OF SUCH DAMAGES.\n\nIf you discover a Defect in this
eBook within 90 days of\nreceiving it...


2. Project Gutenberg Literary Archive Foundation.  Royalty payments\n
must be paid within 60 days following each date on which you\n prepare
(or are legally required to prepare) your periodic tax\n returns. Royalty
payments should be clearly marked as such and\n sent to the Project
Gutenberg Literary Archive Foundation at the\n…
```

Both pages are examples of publisher information that is mutual to the novels and not written by the authors themselves. This raises an inherent problem in our classification task: that some pages in a novel are simply indistinguishable because they embed passages or publishing information written by others. Moreover, the histogram of the of mis-classifications on the left shows that each author is evenly subjected to these flaws, and that the skewed distribution of the training data does not necessarily affect the performance of the models. The histogram of false labels on the right supports this; although “austin” has the second most number of samples, it has the fourth most number of mislabels.





In summary, the neural network’s 95.4% classification accuracy fulfills the goal of discriminating between a small number of authors with varying amounts of available data. Whether this system can withstand a larger database with hundreds or thousands of authors is an intriguing question moving forward. Perhaps new features need to be extracted and online training models used so that new information can be added iteratively.