Language Model for Authorship

Phuong Dinh and Julia Kroll

Data Cleaning

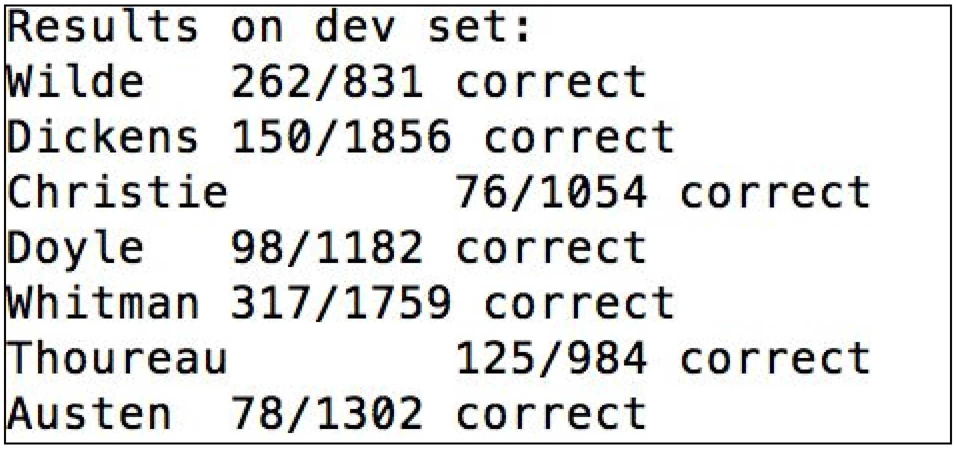
We began cleaning the data by removing most non-alphanumeric characters, including $,;"|`#:%^*_+=~{}<>[](). Many of these characters are rare in prose, and if they do exist, they are not part of a word but rather punctuation that should be removed to isolate tokens. We handled end-of-sentence punctuation (including ?!.) differently because we wanted to preserve sentence boundaries. A series of at least one of those end-of-sentence punctuation characters was tokenized as a single period. This treats abbreviations such as “Mr.” and end-of-sentence punctuation as the same, but taking out periods would have completely erased sentence boundaries, and we wanted to preserve them to the extent it was possible, since we thought that different authors would have different sentence lengths and words that were likely to begin or end a sentence.

We did not add special <unknown> or <name> tokens due to time limitations, although that is one technique that could further improve performance, since names are usually unique to a single book rather than an author’s collected works, and there was a high proportion of unknown words that appeared in the test set but had never been seen in the training set.

Language Model

We used a bigram language model for our final program. We chose a bigram model due to our experimental results after implementing and testing unigram, bigram, and trigram models. Trigrams yielded worse results for every author, possibly due to compounding the high proportion of n-grams containing names and unknown tokens. Unigrams yielded strong results for some authors while performing extremely poorly on others. We suspect unigrams had such variability in performance because it is too simple of a model to provide reliable results. The bigram model was a compromise with more complexity and reliability than the unigram model, without creating an intolerable number of unknown n-grams in the test set.

We implemented backoff, but we removed it from our program as testing demonstrated that it decreased accuracy to below 10% for most authors, which is even worse than chance at 14% for seven authors. Backoff may not have worked because we did not add the <unk> and <name> tokens. These unknown tokens would cause the backoff algorithm to continually back off completely to calculate their probability as the number of n-grams that appeared once divided by our rough probability of words that never appeared, which we approximated by subtracting the number of words in the English language (estimated at 1,000,000) subtracted by the number of n-grams that appeared in the development set, for bigrams and unigrams. While acceptable if unknown n-grams are rare, this estimation is flawed when it is frequently relied upon to calculate probabilities.



```
Results on dev set:
Wilde   262/831 correct
Dickens 150/1856 correct
Christie        76/1054 correct
Doyle   98/1182 correct
Whitman 317/1759 correct
Thoureau        125/984 correct
Austen  78/1302 correct
```

Results of the trigram language model with backoff, 6-32% accuracy

Smoothing

We implemented Good-Turing smoothing. We chose this method because Good-Turing smoothing does not drastically lower the probability of commonly-observed n-grams, as Laplace smoothing does.

Training

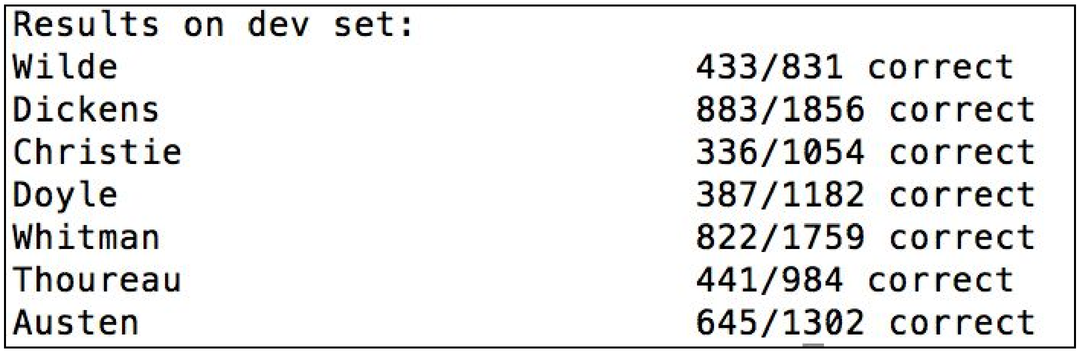
We chose to use 90% of each author's text for training the language models and the remaining 10% for testing. We divided the text into two parts by adding every tenth line to the test set. We decided to select lines rather than sentences because there were already existing line breaks in the input text. If we had attempted to divide the text into sentences, we would have risked improperly treating periods in abbreviations or acronyms as ends of sentences, thereby training our model on inaccurate data. Our choice to use lines rather than sentences also assures that our bigrams and trigrams truly exist in the text, rather than accidentally combining multiple sentences and inadvertently creating n-grams that do not exist in the original text.

Testing

We tested our language models on the 10% of the lines reserved for testing as explained in the Training section above. A major piece of our testing involved handling unknown words that were in the test set but had not been in the training set. We calculated the probability of unknown words by adding 1 to its count of 0 and multiplying the count by N(1)/N(0), where N(c) is the number of bigrams that occur c times. Here the problem of estimating the number of never-seen bigrams arises again, in which case we use the same flawed but adequate estimate of the number of English words (1,000,000) subtracted by the number of bigrams seen in the training text.

Results

Our program correctly identifies the author for between 34-55% of lines in the test set, identifying Wilde best at 55% and Dickens worst at 34%. The aggregated accuracy on lines written by all authors was 3700/7914 correct, or 47%, which is over three times better than the chance rate of 14%.



```
Results on dev set:
Wilde                          433/831 correct
Dickens                        883/1856 correct
Christie                       336/1054 correct
Doyle                          387/1182 correct
Whitman                        822/1759 correct
Thoureau                       441/984 correct
Austen                         645/1302 correct
```

Results of the final bigram language model, 34-55% accuracy