MAT 400 Final Report

Nick Dieffenbacher-Krall

Introduction

Many well known, cannonical, texts have suprising questions when it comes to their authorship and origin. For instance, readers have questioned for hundreds of years whether or not Shakespeare actually wrote all of his poems, and whether or not the identity of the man we know as The Bard is correct at all. These questions are more likely to be asked when looking at ancient, and poetic, literary texts, such as "Beowulf".

A recent innovation in the investigation of these questions has been the application of the statistical subfield of natural langauge processing. This work involves using a variety of techniques to transform text into numerical and categorical data which we can then analyze using statistical models.

The goal of this project is the try and replicate some of this work by creating a system for finding the author of poem. We will use statistical learning models, and natural langauge techniques, in the approach. We hope to attain high accuracy in finding the author of a poem taken from a random subset of poetry from a few famous authors.

Obtaining data

The data for this project is from a collection of 18th and 19th century poets whose work is in the public domain and freely available on Project Gutenberg.

The following authors were included:

Author	Book
William Blake	Poems of William Blake
Lord Byron	The Works Of Lord Byron, Vol. 3 (of 7)
Emily Dickinson	Poems: Three Series, Complete
John Keats	Poems 1817
Rudyard Kipling	Songs from Books
Walt Whitman	Leaves of Grass
William Wordsworth	Lyrical Ballads, With Other Poems, 1800, Vol. I.

There were two main reasons for the selections of these authors. The first was that there work was freely available in a format that was easy to parse and process, namely large text file. The second is that they represent the types of literary works on which statistical analysis has been performed to answer questions of authorship and origin.

The books were processed with a Python script, which can be found in the `./parser` directory of this repository. The Python script collected poems from the books by looking for text in the form of a heading followed by a number of short lines. In was relatively successful, with a total of 1774 different poems collected from the 7 authors.

Environment Setup

All analysis was performed using RStudio, as well as the pacakges listed below.

- `tidyverse`
- `stringi`
- `tm`
- `topicmodels`

- `e1071`
- `randomForest`
- `xgboost`

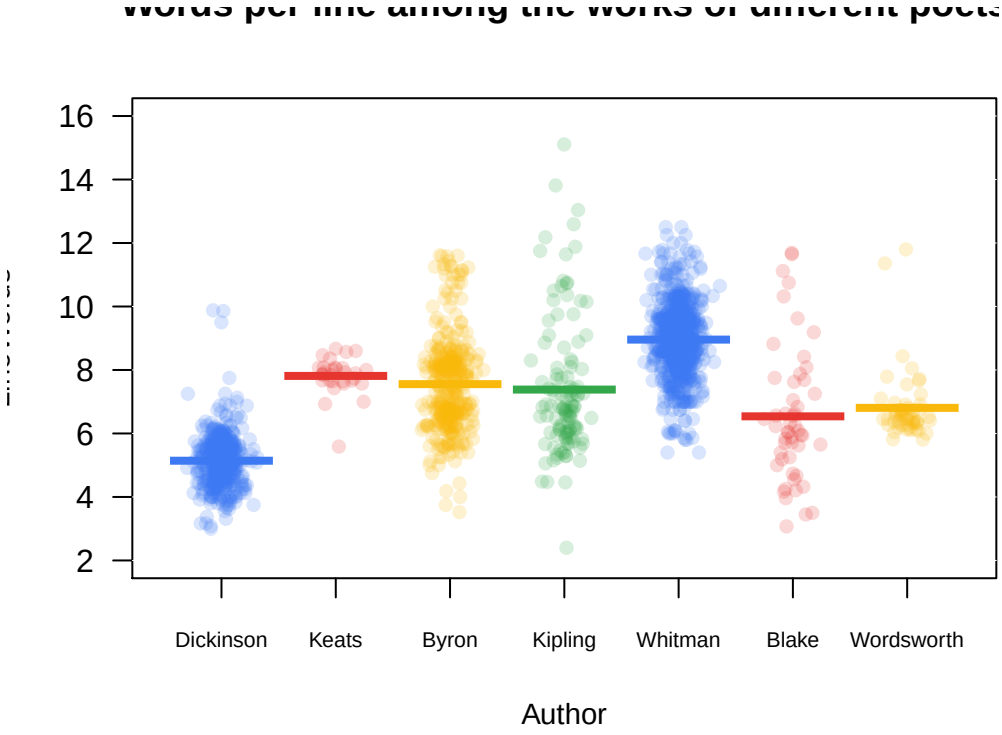
Identifying Important Variables

Before we can attempt to predict the author of a poem based on its text, we need to determine what information we can extract from that text that can be used in a statistical learning model. Although specialized models might be able to handle other inputs, we will attempt to create simple quantitative variables from each poetry document. This will allow a variety of models to be easily adapted for testing.

Basic Variables

One easy variable that we can obtain right from the start is the number of words per line in the poems. Often, the number of words per line is an important part of poetic form. Different poets tend to write poetry in different style and meter, even if they wrote at relatively the same time.

We can see from the figure below that there are clear differences between the words per line used by the different poets, with Dickinson having the poems with the shortest average lines, and, unsuprisingly considering his free verse style, Whitman having the longest.



The data were enriched with a numberical column `LineWords`, representing the mean words per line in each poem, and the `PoemLines` variable, representing the total number of lines in the poem.

Processing and cleaning documents

Most natural langauge processing relies on "cleaning" text documents so that they consist of only base English words. By eliminating punctuation, spelling variations, simple words ("a", "and", "the", etc.) similar it because easier look at the types of words the authors use. To find more variables for our analysis, we will want to perform this cleaning process.

The following exact transformations are performed: - All text is made lowercase - All symbols, numbers, and

punctation is removed - English stopwords are removed - Some poetry specific words, including “O”, and “’T”, are removed - extra whitespace is removed

For an example of the transformations, we can look at the Dickinson poem “This is my letter to the world” before and after the text cleaning process.

```
c(poems[1, ]$Poem, poems[1, ]$CleanText)
```

```
## [1] "That never wrote to me, --\nThe simple news that Nature told,\nWith tender majesty.\nHer messag
## [2] " never wrote simple news nature told tender majesty message committed hands see love sweet coun
```

Sentiment Analysis

Sentiment analysis is a common way to examine documents in NLP. We will be using sentiment analysis to look at the frequncy of words that are associated with different emotions and feelings in the poems, which seems a natural fit for poetry that often deals with emotion. Because of the length of our poems, some of which are only a few words, it is necessary to carry out sentiment analysis using a limited scope. The sentiment analysis performed on the data is based on a publicly available lexicon. Each word is analyzed based on its attributes looked up from the lexicon.

The lexicon used here is released as part of the R package `tidytext`. In the context of the package, the lexicon is known as “AFINN”, referring to its creator Finn Årup Nielsen. It features 2476 English words with their positivity given as an integer between -5 and 5. The words were given their scores manually by the creator of the lexicon. See below for an example.

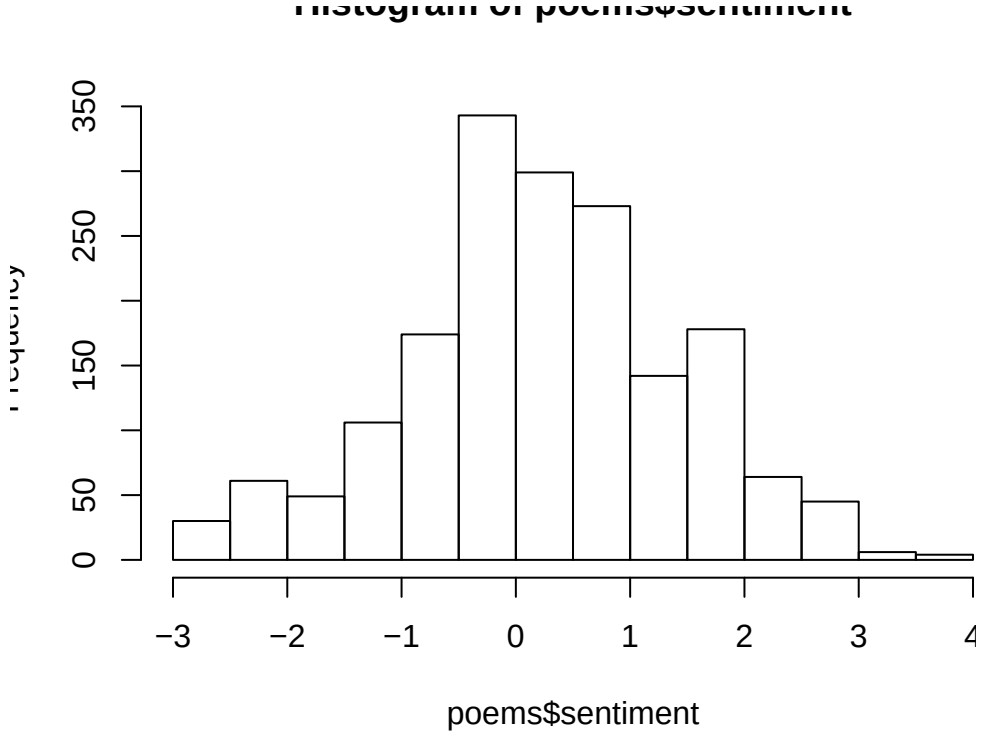
```
lex <- get_sentiments("afinn") # from tidytext
head(lex)
```

```
## # A tibble: 6 x 2
##   word       score
##   <chr>      <int>
## 1 abandon       -2
## 2 abandoned     -2
## 3 abandons      -2
## 4 abducted      -2
## 5 abduction     -2
## 6 abductions    -2
```

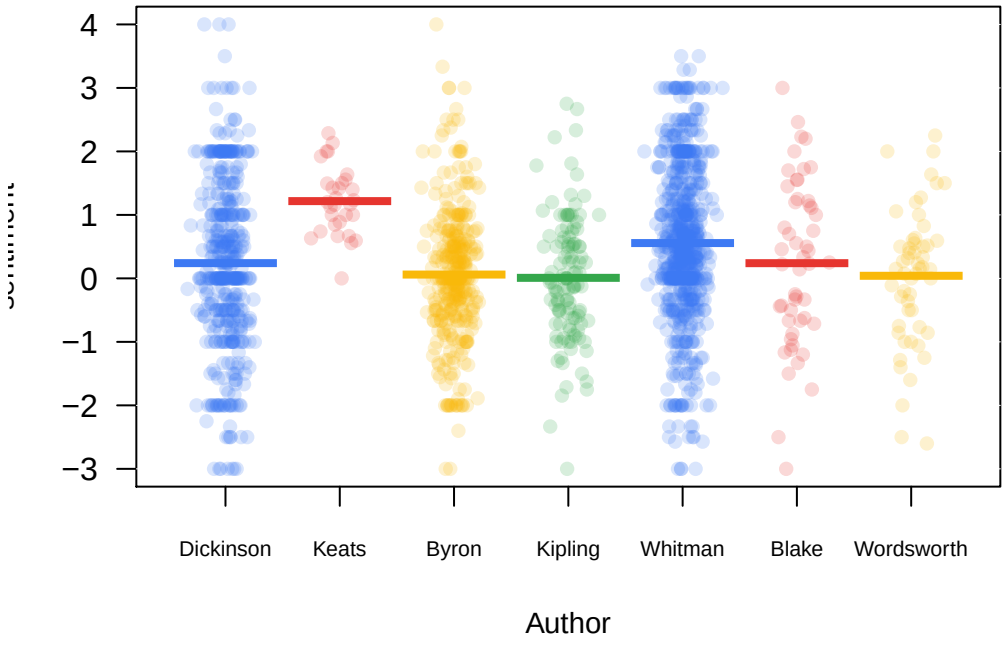
To enrich our poetry data with a sentiment rating, each poem underwent the following procedure: First, the `CleanText` version of the poem was split into words. For each word, the sentiment lexicon was checked. If it was present, the word was given a score from the lexicon. Otherwise, the word recieved a score of 0. Finally, the average was computed of the score of each word in the poem. The sentiment scores were saved as the column `sentiment` in the `poems` data frame (in `setup.R`).

The overall distribution has a roughly normal shape. There are particular spikes in the data at each integer, because the scores began as integers and some poems had few scoring words.

```
hist(poems$sentiment)
```



We can also compare the distribution of sentimt among different authors to see whether or not this will be a useful metric for later classification.



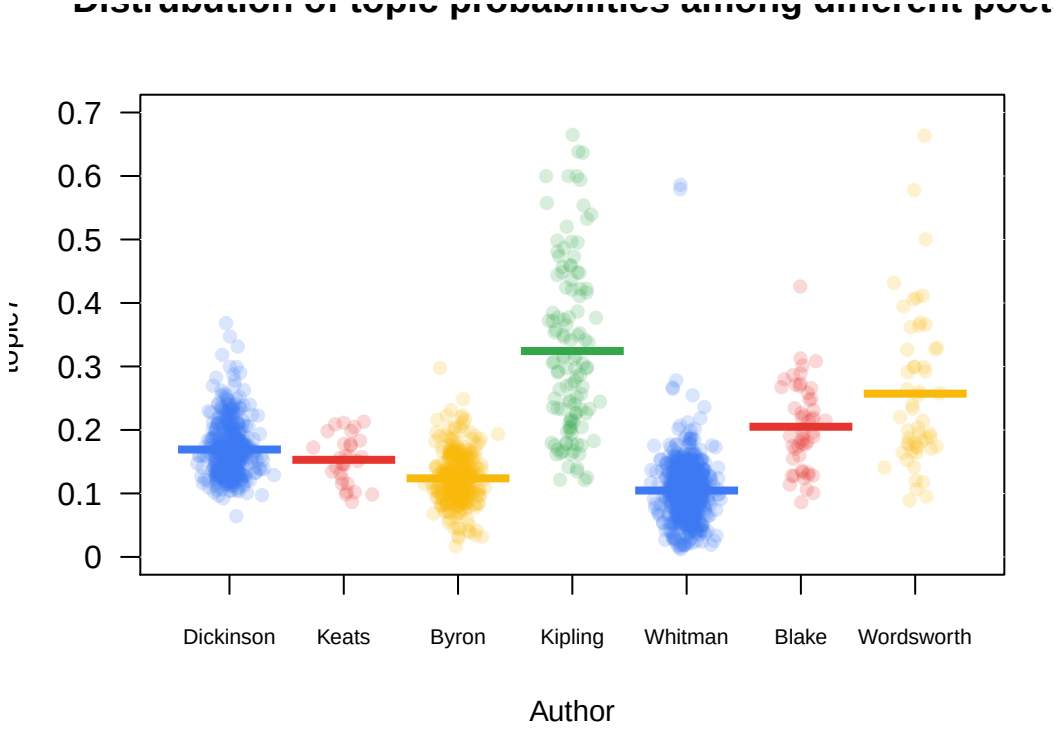
It appears that there is some different in the distrubution of sentiment rating, but these differences are not huge. Notably, the poetry of Keat's seems to have a higher sentiment rating than the others.

Topic Models

Another potential source for more numerical variables derived from the poem corpus is topic modeling. This technique in natural langauge processing involves finding "topics", which are abstract concepts present in documents that generate varius words. In poetry, a topic might be "springtime", if the spring moves the author to include words and phrases often associated with spring. The particular model used in this examination is latent Dirichlet allocation.

The number of topics, k, in the LDA model was chosen by looking the results generated by the LDA model with different values of k, from 1 to 10. The best value of k was the one that would yield topic probabilities for each poem that had the most contrasting distributions between different authors.

For example, an LDA model where $k = 7$ yeilded 7 sets of topic probabilities for each of the poems in the corpus. The figure below shows that topic 7 seems to produce distributions that are somewhat differentiated between poets. The differentiation certainly seems greater than that created by the sentiment analysis.



Ultimately, it seemed as though 7 topics produced a good balance between creating new quntitative variables for the poetry data set, and spending too much time in computation. At least of few of the generated topics seemed to create different distributions betwen authors, and so may be helpful in classification and clustering.

Modeling

Since the poetry corpus has been analyzed, and multiple variables been extracted, it is now possible to create models to try and determine the authorship. The models will include both clustering models and classification models.

Each model will be evaluation for performance by creating a confusion matric, such as in the following R code assessing the accuracy of a random forest model:

```
cm <- table(predict(forest.model, test), test$Author)
forest.accuracy = (sum(diag(cm)))/sum(cm)
forest.accuracy
```

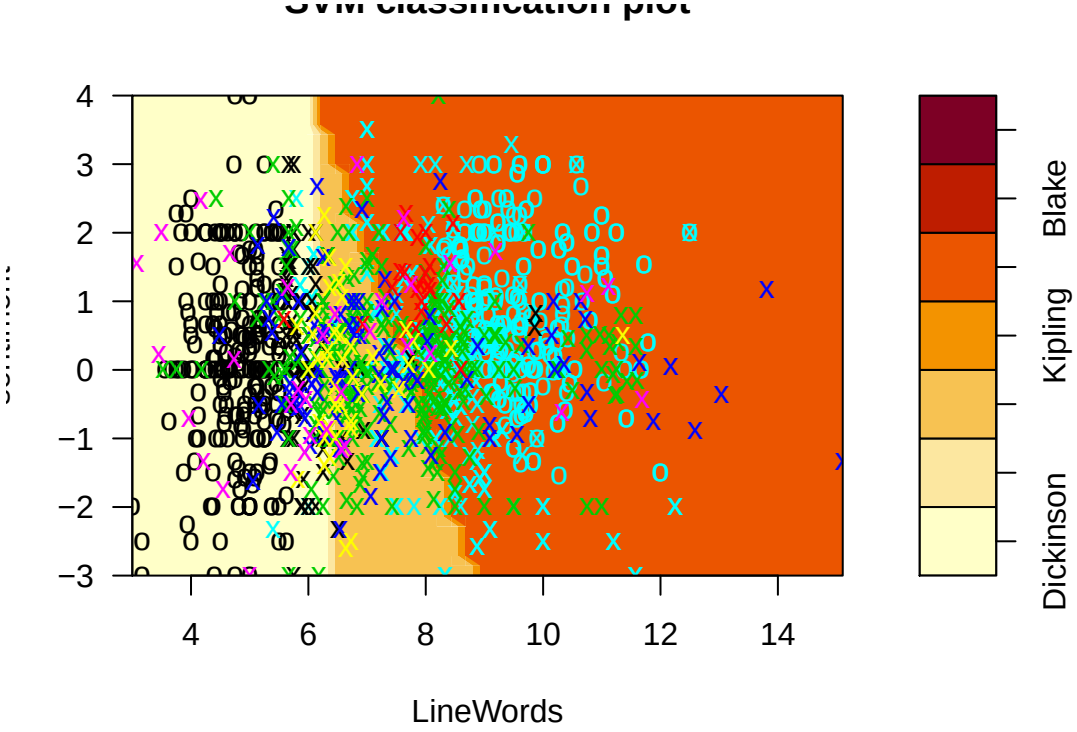
```
## [1] 0.8806306
```

The accuracy reported for other models will use a similar pattern

SVM

An initial attempt to create some sort of model that can find the author of a poem was done with an SVM-based clustering model. This was a natural first choce as SVM can produce simple clusters that are easy to examine visually and understand.

```
plot(svmfit, train[, c("sentiment", "LineWords", "Author")])
```



The prediction error of this SVM model was 0.7162162.

Many of the variables created from the poetry documents have only small differences between the various authors, such as "sentiment". Correctly classifying the authors will likely take a combination of these variables. Since there is little difference between them, it is unlikely that a clustering- based approach, like SVM, will be very successful. No matter in how many dimensions the data is represented, there will exist no exact contiguous spaces, or any shape, where a cluster could be found that correctly represents the work of an author. This finding makes the case for using some different models.

Random Forest

Since the SVM methods do not seem to be promising for these data, it may make sense to use a supervised learning method instead.

Random forests are a common classifier method that we can use with all of our existing variables.

```
forest.model <- randomForest(Author ~ sentiment + topic1 + topic2 + topic3 +
                               topic4 + topic5 + topic6 + LineWords + PoemLines,
                             data = train,
                             ntree=100 )
```

Tuning the number of trees with 5 fold cross-validation on a few significant values of `ntree` yielded 100 as the value with the lowest mse on the trainning data.

The prediction error of the final tuned random forest model was 0.8806306.

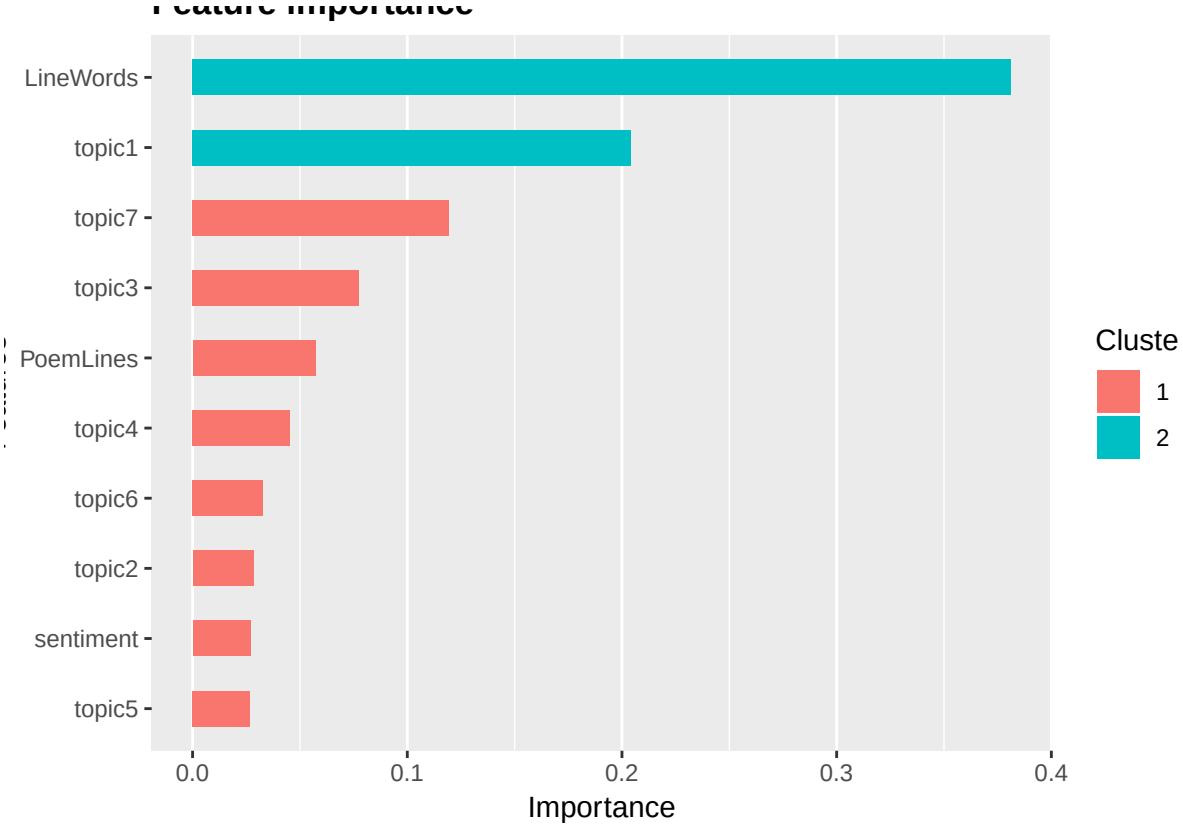
Extreme Gradient Boosting

A variation of trees based classification methods that could be helpful for our problem is gradient boosting. This method will build stronger models out of weaker decision trees.

The `xgboost` library helpfully provides facilities to create various grapics. Below is a diagram of a decision tree used by a variant of the model.

It is also interesting to see which variables were given the highest importance rating by the model. This can be seen in the plot below.

```
boost.importance = xgb.importance(model = boost.model)
gp = xgb.ggplot.importance(boost.importance)
print(gp)
```



Unfortunitely, it seems as though the variable `LineWords`, the average words per line in a poem, is the most important in author classification, which seems rather trivial.

The prediction error of the final tuned `xgboost` model was 0.8671171

Conclusion

The results of this project show that the most difficult part of the problem is creating adequite data from literary and poetic text. The goal was to develop a system for the classification of authors of poetry, but the difficult piece of this task was not finding the correct model to use but turning the poems into numbers that could be used by a particular model.

This difficulty is shown by the fact that the figures comparing the distributions of various quantitative variables in the "Identifying Important Variables" show that there tended to be only small differences in averge values per author. For instance, the `sentiment` variable created by our lexicon-based sentiment analysis failed to show major differentiation between authors. Keats showed a slightly higher mean, and others may have had higher standard varying standard deviations. The differences seemed small enough, however, that they were unlikely to be very helpful to our classifying models. This is confirmed by the random forest model, which rated `sentiment` as one of the least important variables in the final classification.

It seems plausible that there are many indicators of authorship that are difficult to analyze with code and statistics but that might be easily noticed by an a literary expert. These might include the overall message of a poem, the emotional content therein, or the attitude its author had toward certain other topics. Were there more variables, the tree based classification methods that were attempted might have had higher success rates.