Shakespeare & Gender

Lillian Gao and Dashiell Stander

December 14, 2015

Introduction

In this report we will detail, from start to finish, our methodology for using a logistic regression model to classify the gender of of characters in Shakespeare's plays based solely on their dialogue. Our focus will be on the langaueg that they use, but we will also look at other variables and compare their effectiveness.

Data Extraction & Cleaning

We downloaded the complete works of Shakespeare from MIT's website (that was produced in conjunction with the Gutenberg Project).

```
shakesURL <- 'http://ocw.mit.edu/ans7870/6/6.006/s08/lecturenotes/files/t8.shakespeare.txt'
directory <- '~/ShakespeareProject/_rawData'; fileName <- '/Shakespeare.txt';
download.file(shakesURL, paste0(directory, fileName));
```

This is *very* dirty data and cleaning it is going to be extremely laborious. After taking out the license information, we extracted each play and wrote each one to a different text file. Then we had to separate out each character's dialogue. This had many steps: getting the names of all the characters in the play, getting the starting locations of their dialogue, finding where each line of dialogue ends, and then aggregating it all into one text file with the appropriate name. This required a lot of coding infrastructure and was made especially difficult because some of the plays were formatted differently and some characters' names changed mid-play. A sample of the functions we wrote to perform this all this can be found below.

```
#Finds the names of all the characters in a play (as they're identified)
#before their dialogue and returns it as a list.
getCharacters <- function(text){
  chars <- unique(str_trim(na.omit(str_match(text, '^  [A-Z ]{3,}\\.'))));
  if (length(chars) < 5){
    chars <- unique(str_trim(na.omit(str_match(text, '^  [:alpha:]{3,}\\.'))))
  }
  if (length(chars) < 5){
    chars <- unique(str_trim(na.omit(str_match(text, '^  [A-Za-z ]{3,}\\.'))))
  }
  dramPersonae <- removePunctuation(chars);
  dramPersonae <- unique(str_trim(dramPersonae));
  dramPersonae <- dramPersonae[which(dramPersonae != '')];
  return(dramPersonae);
}


#Takes in the Dramatis Personae, and outputs a list.
#Each element of the list stands in for a single character and
#is a vector with all of the locations where a line of that character's
#dialogue begins
getDialogueLoc <- function(chars, play){
  names <- removePunctuation(str_trim(na.omit(chars)));
```

```
  chars <- names;
  personae <- list(); n <- length(chars);
  for (i in 1:n) {
    name <- chars[i]; t <- nchar(name);
    period <- str_sub(name, t, t)
    chars[i] <- paste0(name, '\\.');
  }

  for (i in 1:n){
    name <- chars[i]; pattern <- paste0('^  ', name);
    locations <- grep(pattern, play);
    if (length(locations) != 0) {
      personae[[i]] <- grep(pattern, play);
    } else {
      pattern <- paste0('^', name);
      personae[[i+ 0]] <- grep(pattern, play);
    }
  }
  names(personae) <- names;
  return(personae)
}
```

We made other changes to the dataset as well, removing punctuation, setting everything to lower case, etc... in order to simplify the analysis. We also painstakingly went through each character, checked a third party resource (opensourceshakespeare.org) to assign each character a gender.

Setup

To set up our analysis we made extensive use of the tm package in R. It allowed us to aggregate all of the data into distinct corpuses for men and women.

These document term matrices can be used to do a lot of preliminary analysis of the data. For example:

```
#Outputs the terms used in the range specified
findFreqTerms(dtmMen, lowfreq = 20, highfreq = 50 )[1:30];
```

```
##  [1] "abhorson"     "abide"        "able"         "aboard"
##  [5] "absence"      "absent"       "absolute"     "abuse"
##  [9] "accept"       "according"    "account"      "accuse"
## [13] "acquaintance" "acquainted"   "acts"         "actsc"
## [17] "adam"         "add"          "addition"     "advance"
## [21] "advice"       "advisd"       "advise"       "aegeon"
## [25] "afeard"       "affairs"      "affections"   "afoot"
## [29] "afraid"       "afternoon"
```

```
findFreqTerms(dtmWomen, 20, 50)[1:30];
```

```
##  [1] "about"    "above"    "age"      "air"      "alack"    "alice"
##  [7] "almost"   "alone"    "antony"   "arms"     "ask"      "bad"
## [13] "base"     "beauty"   "because"  "become"   "behind"   "behold"
## [19] "believe"  "benedick" "beseech"  "between"  "bianca"   "black"
## [25] "bless"    "bloody"   "body"     "born"     "bosom"    "bound"
```

Or it can find the words that are correlated.

```
findAssocs(weightedDtmWomen, 'madam', .5)
```

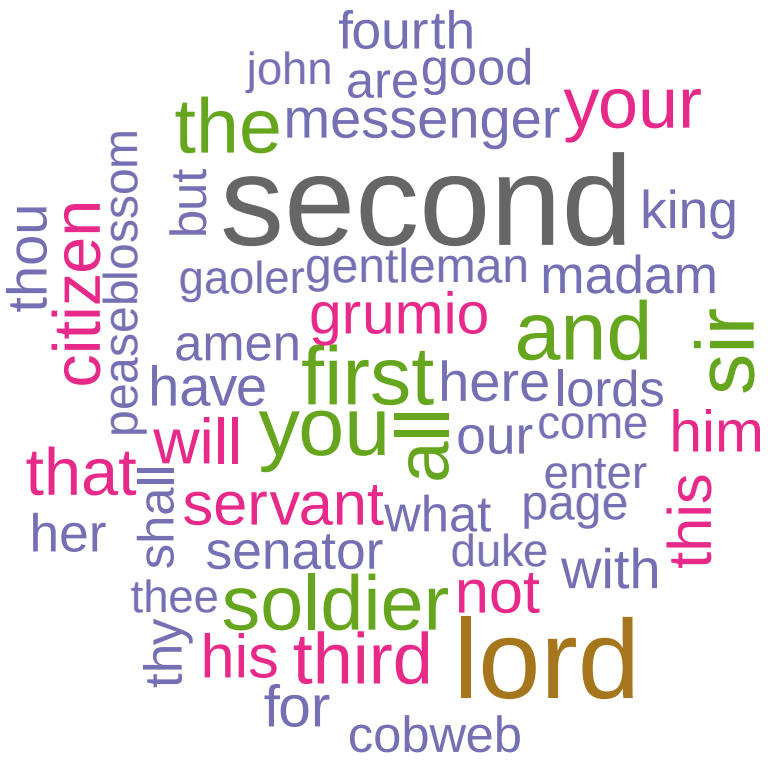
```
## $madam
##  bowls  tales   weep  dance   play  could   well either   sing   lady
##   0.83   0.82   0.82   0.81   0.80   0.78   0.78   0.73   0.70   0.66
```

```
findAssocs(weightedDtmMen, 'madam', .5)
```

```
## $madam
## porter
##   0.72
```

We were also able to make a few nice visualizations.

```
wordCountsMen <- sort(colSums(as.matrix(weightedDtmMen)), decreasing = T)[1:50];
wordCountsWomen <- sort(colSums(as.matrix(weightedDtmWomen)), decreasing = T)[1:50];

set.seed(142)
dark2 <- brewer.pal(8, "Dark2");
menWCPlot <- wordcloud(names(wordCountsMen), wordCountsMen,
                  max.words = 50, rot.per = 0.2, colors = dark2);
```



Challenges

After the most basic analysis had been done persistent challenges started to appear. To start, their are many more men then women in Shakespeare's plays.

```
#Number of men
nrow(as.matrix(dtmWomen));
```

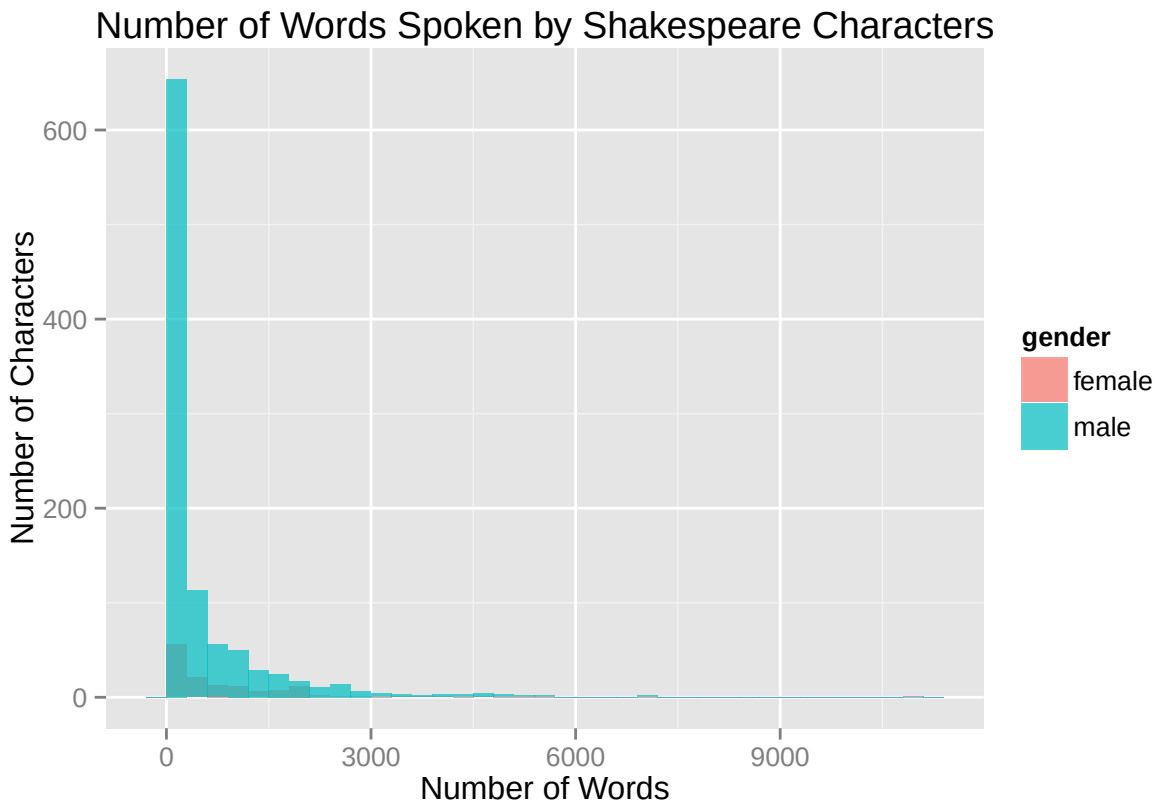
```
## [1] 136
```

```
#Number of women
nrow(as.matrix(dtmMen));
```

```
## [1] 1005
```

Men also talk *a lot* more than women.

```
maleCharCounts <- rowSums(as.matrix(dtmMen));
femaleCharCounts <- rowSums(as.matrix(dtmWomen));
gender <- c(rep('male', length(maleCharCounts)), rep('female', length(femaleCharCounts)));

wordcountDF <- data.frame('wordcount' = c(maleCharCounts, femaleCharCounts),
                          'gender' = gender);

wordcountPlot <- ggplot(wordcountDF, aes(x = wordcount, fill = gender)) +
  geom_histogram(position = 'identity', alpha = .7, binwidth = 300) + labs(title = 'Number of Words Spo
  scale_y_continuous();

wordcountPlot;
```



As you can see, men totally dominate the conversation in the works of Shakespeare.

In addition to this, there is a lot of noise in the data. There is a large amount of common words used by both men and women and many of the words that are used differently by each gender are sparse in the dataset as a whole. As you will see, this makes it difficult to zero in on what makes womens' dialogue distinct.

Fitting Models

We weighted term frequency by TFldf to account for attempt to account for the vastly different size of the two corpuses. The formula for the scheme, if Tf is the term frequency of a word, n is the number of documents it appears in, and N is the total number of documents, is given by:

$$TfIdf = Tf \times log(\frac{n}{N})$$

We then identified a number of key terms that we would use as predictor variable in our logistic regression model.

They were: 'thrust', 'private', 'yourselves', 'whos', 'madam', 'visit', 'dare', 'courtesan', 'luce', 'lord', 'rot', 'lips', 'thankfully', 'madam', 'sir', and 'ready'.

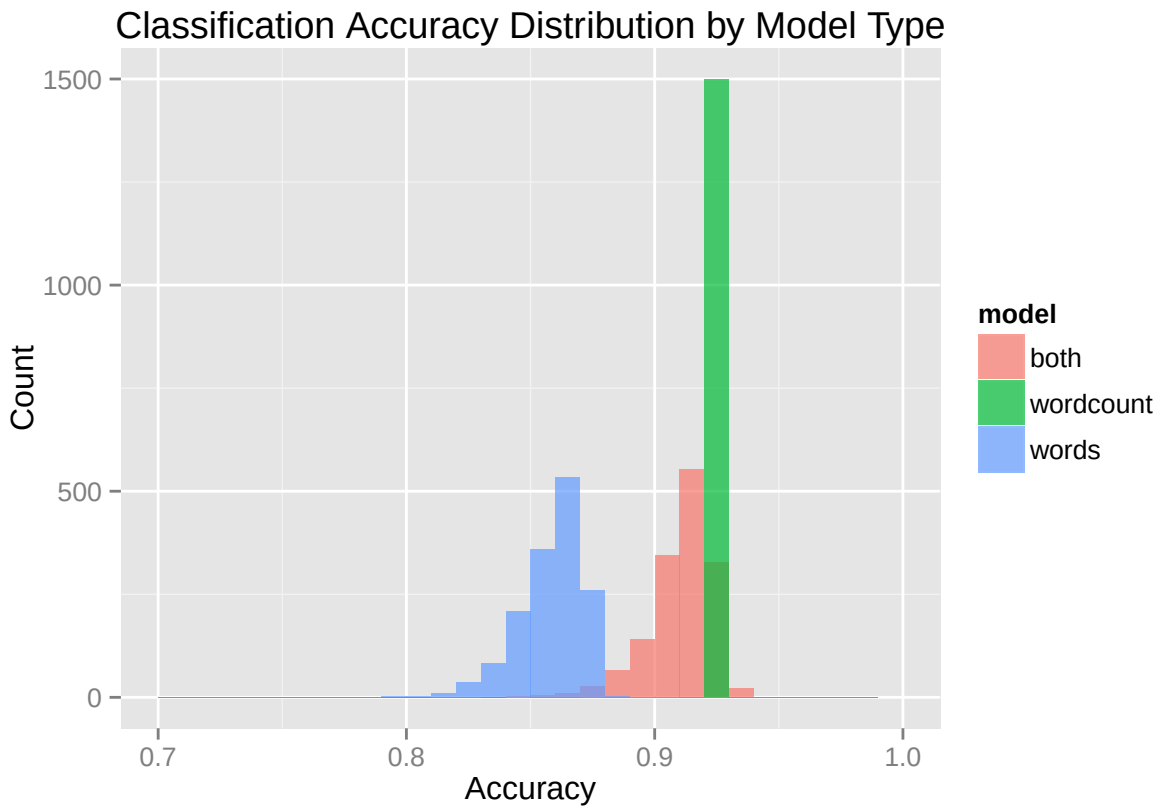
As a benchmark to test the effectiveness of the "words" model, we also built one that only used the number of words a character has in their dialogue ("wordcount") as a predictor variable. And then, for completeness's sake, we built a model that uses both.

To fit each model we took a random sample of characters from the set of men and women and used that as a training set. The training set was kept small in an attempt to downsample and, hopefully, limit the effects of the death of female data. Then we tested its effiency on the entire corpus and checked how accurate its classifications were (after picking an optimal threshold).

To try to limit the effect of the extreme variability of the random sampling of training data we performed a bootstrap, fitting each model to 1500 randomly selected training sets, and used the distribution to compare them.

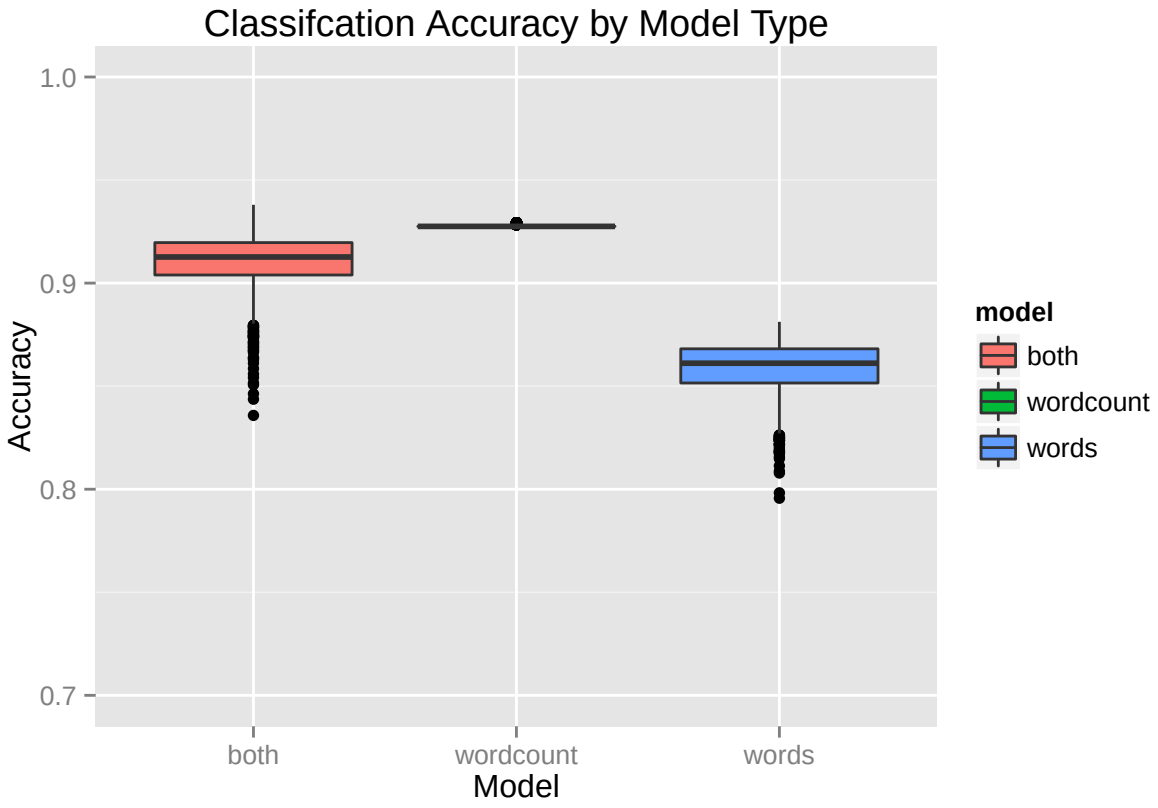
```
resultsDF <- read_csv('~/ShakespeareProject/_cleanData/_modelResults/bootStrapResults.csv');

resultsPlot <- ggplot(data = resultsDF, aes(fill = model))
resultsHist <- resultsPlot + geom_histogram(aes(x = results), , position = 'identity',
              alpha = .7) + scale_y_continuous() + labs(title = 'Classification Accuracy Distribution by
resultsBox <- resultsPlot + geom_boxplot(aes(x = model, y = results)) +
  labs(title = 'Classifcation Accuracy by Model Type', x = 'Model', y = 'Accuracy') +
scale_y_continuous(limits = c(.7, 1))

resultsHist;
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```




```
resultsBox
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



Summary & Results

The exact numeric results are as follows:

```
justwords <- filter(resultsDF, model == 'words')$results;
wordcount <- filter(resultsDF, model == 'wordcount')$results;
both <- filter(resultsDF, model == 'both')$results;

summary(justwords); summary(wordcount); summary(both);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2515  0.8515  0.8611  0.8581  0.8681  0.8812

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.9275  0.9275  0.9275  0.9278  0.9275  0.9293

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8358  0.9039  0.9127  0.9105  0.9197  0.9380
```

The single most predictive variable was wordcount. Using term frequency beat the naive benchmark of 87% that you would get from guessing randomly if you knew the proportions ahead of time. It is, however, not nearly as effective as just looking at how much a character speaks.

We are, at this point, unsure *why* term frequnecy is less predictive. It is possible that the differences in language are simply not distinct enough for the model to capture the differences, or that the differences that are distinct aren't common enough for them to be well represented in the training sets. It may also simply be due to the small sample size of female characters.

Also of note is the extraordinarily low variance of the wordcount model. We hypothesize that this is because the random sampling does not affect the comparative distribution of wordcount as much as it does the distribution of term frequency.

Conclusion

This was an extraordinarily rewarding and interesting project. Though the term frequency model was not as effective as we might have liked, the extreme effectiveness of wordcount is still an interesting result. This also leaves open many questions for us to explore further. Would a different model type (SVM, Lasso, etc...) perform better? Is there any pattern to the characters that the model was classifying incorrectly? After these preliminary investigations we have a good base from which to try and find out.