MSDS680 - Week 3 - Naive Bayes Classification

Benjamin Siebold

Wilhelm-Wundt-University

MSDS680 - Week 3 - Naive Bayes Classification

Introduction

In this project, the Naive Bayes (NB) classification model will be used to classify a set of texts between spam and ham based off words within the text messages. The data used for this analysis can be found by downloading the zip file here. The dataset used in this analysis is simple to start, with only two attributes, a type and text attribute.

Methodology and Results

The first step in the analysis is to read the data in, and set up the environment for analysis. In order to complete this analysis, libraries for the model itself, as well as cleaning and visualizing the data will be loaded. The model did create issues in R when applying to the entire dataset, thus for sake of performing the model, a sample of 4000 records was taken.

```r
library(e1071)
library(caret)
library(tm)
library(SnowballC)
library(wordcloud)
library(gmodels)
library(magick)
setwd("/cloud/project/week_3")
set.seed(486)

texts <- read.delim("SMSSpamCollection", quote = "", header = FALSE,
    stringsAsFactors = FALSE, col.names = c("type", "text"))
```

```r
texts <- texts[sample(nrow(texts), 4000), ]
texts$type <- factor(texts$type)
```

Prior to applying any classification to the texts, it is first necessary to clean the text data so as to unify the data. This will ensure the amount of features is reduced because various tenses of the same word are counted as different features. The steps necessary to clean the data are to ensure all words are lower case, remove all punctuation and numbers, remove stopwords that don't have any inherent meaning such as "the", "and" etc. and then stem the words in each column to change like words to the same word. These steps will enable the texts to be similar data with words such as "learning" and "learned" being recognized the same as their root word, "learn."

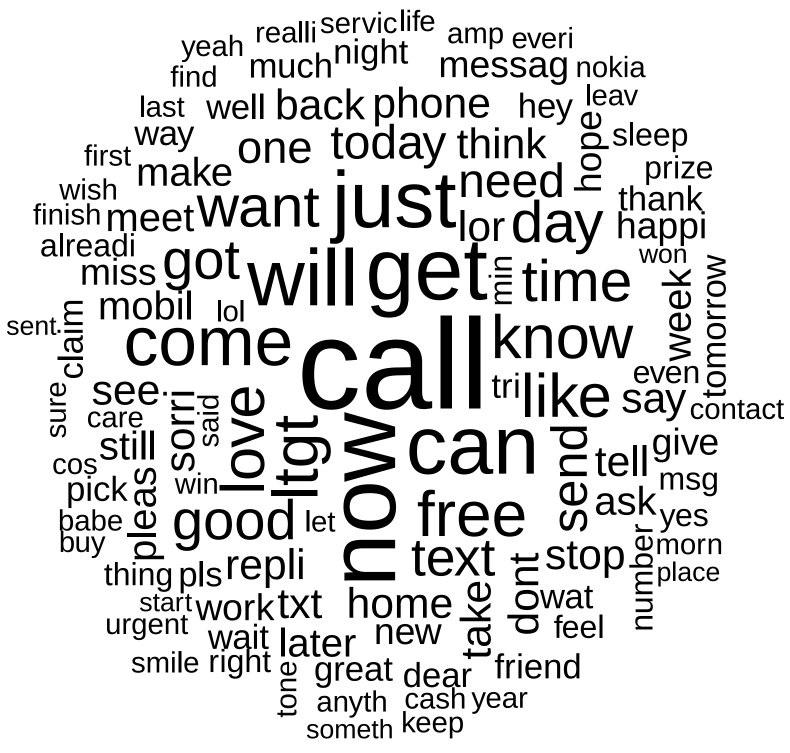
```r
text_corpus <- VCorpus(VectorSource(texts$text))
as.character(text_corpus[[1]])
```

```
## [1] "Your opinion about me? 1. Over 2. Jada 3. Kusruthi 4. Lovable 5. Silent 6. Spl c
```

```r
text_corpus_clean <- tm_map(text_corpus, content_transformer(tolower))
text_corpus_clean <- tm_map(text_corpus_clean, removeNumbers)
text_corpus_clean <- tm_map(text_corpus_clean, removeWords, stopwords())
text_corpus_clean <- tm_map(text_corpus_clean, removePunctuation)



replacePunctuation <- function(x) {
    gsub("[[:punct:]]+", " ", x)
}
```

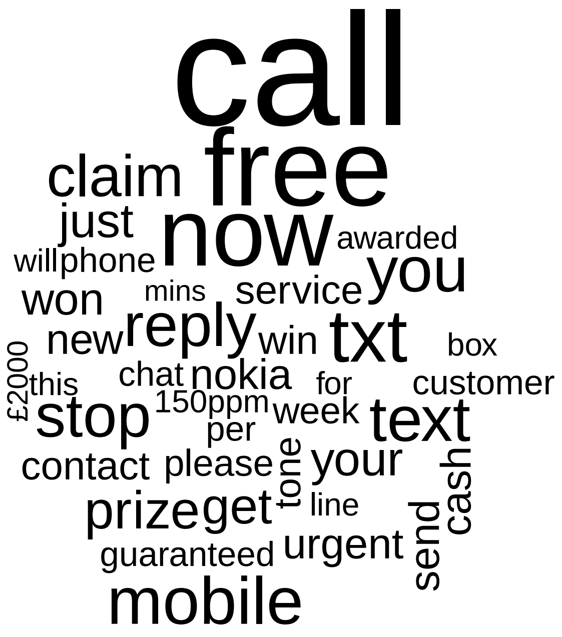
```r
text_corpus_clean <- tm_map(text_corpus_clean, stemDocument)

text_corpus_clean <- tm_map(text_corpus_clean, stripWhitespace)
```

With the text data now cleaned, some general data inspection will be done such as creating word clouds of the whole dataset, the spam and ham portions of the dataset. This inspection will allow for potential differences between the spam and ham texts to be made, as well as an understanding of if there are words that dominate the dataset.

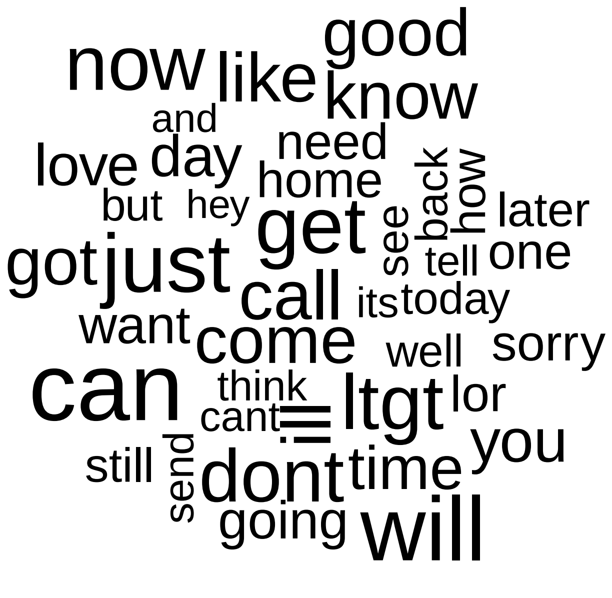


```r
spam <- subset(texts, type == "spam")

ham <- subset(texts, type == "ham")
```

Spam



Ham



From the three word clouds above, it can be seen certain words appear very frequently in the spam texts that don't appear frequently in the ham texts, but there is also a bit of overlap. For example call. free, claim, and stop are all very visible in the spam wordcloud, but not very visible in the ham; however, now and you are both used frequently in both. With this knowledge, distinguishing between the two should be a task achievable with an NB classifier. In order to apply a NB model, a few more steps are necessary. The data will now be turned into a document matrix which will allow the words to each be viewed as a feature, and create train and test data for the model to evaluate.

```
text_matrix <- DocumentTermMatrix(text_corpus_clean)

split <- nrow(text_matrix) * 0.7
max_rows <- nrow(text_matrix)

text_matrix_train <- text_matrix[1:split, ]
text_matrix_test <- text_matrix[(split + 1):max_rows, ]
text_train_labels <- texts[1:split, ]$type
text_test_labels <- texts[(split + 1):max_rows, ]$type
dim(text_matrix)
```

```
## [1] 4000 5485
```

```
attributes(text_matrix)
```

```
## $names
## [1] "i"        "j"        "v"        "nrow"     "ncol"     "dimnames"
##
## $class
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
##
## $weighting
## [1] "term frequency" "tf"
```

Now that the data has been split into training and test data, it can be seen above there are a few potential issues with the training and testing data. The first is the amount of features in the data. over 6,000 features will make for a difficult classification. These features can be reduced by omitting words that appear in less than x amount of documents. The other issue is in the weighting of the data. Currently the terms are looked at as "term frequency" rather than a boolean indicator on whether or not the terms exist. The NB model is mostly used with categorical features opposed to what is occurring in the frequency matrix. To ensure the data is best prepared for the NB model to be applied, the last steps of cleaning (removing words that appear infrequently, and converting the data to categorical) are necessary.

```r
text_freq_words <- findFreqTerms(text_matrix_train, 5)
text_matrix_freq_train <- text_matrix_train[, text_freq_words]
text_matrix_freq_test <- text_matrix_test[, text_freq_words]

convert_counts <- function(x) {
    x <- ifelse(x > 0, "Yes", "No")
}

text_train <- apply(text_matrix_freq_train, MARGIN = 2, convert_counts)
text_test <- apply(text_matrix_freq_test, MARGIN = 2, convert_counts)

prop.table(table(text_train_labels))
```

```
## text_train_labels
##       ham      spam
## 0.8635714 0.1364286
```

```r
prop.table(table(text_test_labels))
```

```
## text_test_labels
##       ham      spam
## 0.8766667 0.1233333
```

The above steps have removed all features that don't appear in at least 5 of the 5,000 documents to increase the efficiency of the NB model, converted variables to categorical data giving a label of "Yes" if a word exists in the text and "No" if it doesn't, and lastly shown the ratio of spam and ham in both the train and test datasets to ensure the ratios are similar for the models accuracy. Now with clean training and test data, a NB model will be applied.

```r
prop.table(table(text_train_labels))
```

```
## text_train_labels
##       ham      spam
## 0.8635714 0.1364286
```

```r
prop.table(table(text_test_labels))
```

```
## text_test_labels
##       ham      spam
## 0.8766667 0.1233333
```

```r
text_freq_words <- findFreqTerms(text_matrix_train, 5)
text_matrix_freq_train <- text_matrix_train[, text_freq_words]
text_matrix_freq_test <- text_matrix_test[, text_freq_words]
```

```r
convert_counts <- function(x) {
    x <- ifelse(x > 0, "Yes", "No")
}

text_train <- apply(text_matrix_freq_train, MARGIN = 2, convert_counts)
text_test <- apply(text_matrix_freq_test, MARGIN = 2, convert_counts)

text_classifier <- naiveBayes(text_train, text_train_labels)
text_pred <- predict(text_classifier, text_test)
```

```
c <- confusionMatrix(text_pred, text_test_labels)
c
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  ham spam
##       ham  1048   19
##       spam    4  129
##
##                Accuracy : 0.9808
##                  95% CI : (0.9714, 0.9878)
##     No Information Rate : 0.8767
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9073
##
##  Mcnemar's Test P-Value : 0.003509
##
##             Sensitivity : 0.9962
##             Specificity : 0.8716
##          Pos Pred Value : 0.9822
##          Neg Pred Value : 0.9699
##              Prevalence : 0.8767
##          Detection Rate : 0.8733
##    Detection Prevalence : 0.8892
##       Balanced Accuracy : 0.9339
```

```
##
##        'Positive' Class : ham
##
```

```
f1score <- as.numeric(c$byClass["F1"])
f1score
```

```
## [1] 0.9891458
```

Above it can be seen that there were a total of 23 texts incorrectly labeled. 4 of the 1052 ham messages were labeled incorrectly, and 19 of the 148 spam texts were incorrectly labeled as ham. The accuracy of the model was really high with an f1 score of .989. The models accuracy is high enough that the study could be finished, but for sake of being thorough the model will be repeated with a Laplace estimator. The Laplace estimator allows for features that only appeared in one of the labels (spam or ham) to not be definite in determining of whether or not a text is labeled.

```
text_classifier2 <- naiveBayes(text_train, text_train_labels,
    laplace = 1)
text_pred2 <- predict(text_classifier2, text_test)
```

```
c2 <- confusionMatrix(text_pred2, text_test_labels)
c2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  ham spam
##       ham  1033    6
##       spam   19  142
##
##                Accuracy : 0.9792
##                  95% CI : (0.9694, 0.9865)
##     No Information Rate : 0.8767
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9072
##
##  Mcnemar's Test P-Value : 0.0164
##
##             Sensitivity : 0.9819
##             Specificity : 0.9595
##          Pos Pred Value : 0.9942
##          Neg Pred Value : 0.8820
##              Prevalence : 0.8767
##          Detection Rate : 0.8608
##    Detection Prevalence : 0.8658
##       Balanced Accuracy : 0.9707
```

```
##
##        'Positive' Class : ham
##
```

```
f1score2 <- as.numeric(c2$byClass["F1"])
f1score2
```

```
## [1] 0.988044
```

From above, it can be seen the laplace estimator impacted the model by creating an inverse of inaccuracy. With the estimator more ham messages were labeled incorrectly but less spam messages were. The overall f1 score of the model with the laplace estimator was .988, almost identical to the original model.

Conclusion

Due to the nature of messaging, the model built without the laplace estimator will be better to use because it is one more accurate, and two preferable for spam messages to slip through a filter than for messages that are intended to be received filtered out. Overall, Naive Bayes proved to be very reliable for filtering spam text messages with the given dataset. Further steps would be to look at the features in more depth and decide if further, or less feature reduction is necessary to either increase the models accuracy or speed at which it performs. This concludes the Naive Bayes classification problem.

References

Lantz, B. (2015). Machine Learning with R : Expert Techniques for Predictive Modeling to Solve All Your Data Analysis Problems: Vol. Second edition. Packt Publishing.