Classifying Amazon Fine food reviews based on Naive Bayes Algorithm

Varsha Reddy Duvvuru

February 26, 2017

Exploring and preparing the data: Amazon Fine Foods reviews data set is from Stanford Large Network Dataset Collection. This dataset consists of around 500,000 reviews of fine foods from amazon for more than 10 years, including all reviews up to October 2012. Reviews include product and user information, ratings, and a plaintext review.

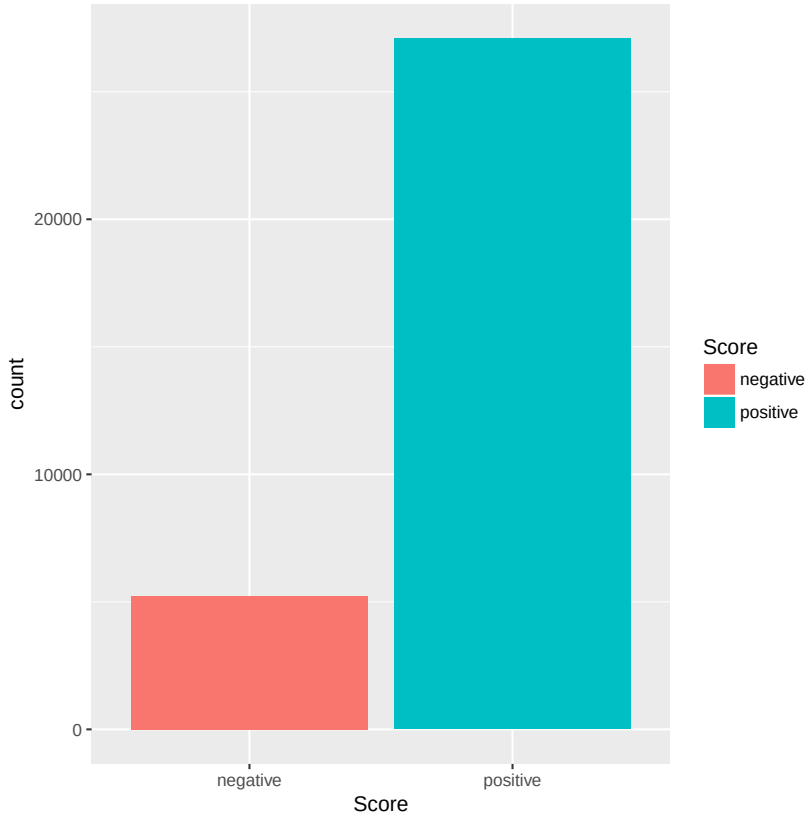
Collecting data: To begin with,we import the data in csv format to data frame. We only consider two attributes, which are Score and Summary. Since Score is a categorical variable, it would be better to convert it into a factor.

```
> foodreviews<-read.csv("C:/Users/varsh/Desktop/Lectures/R/Project/Reviews.csv")
> reviews<-foodreviews[,c("Score","Summary")]
> reviews<-reviews[!(reviews$Score==3),]
> reviews$Score<-ifelse(reviews$Score>3, "positive", "negative")
> reviews<-na.omit(reviews)
> str(reviews)
```

```
'data.frame':        32312 obs. of  2 variables:
 $ Score  : chr  "positive" "negative" "positive" "negative" ...
 $ Summary: Factor w/ 28196 levels "","'Healthy chips'",..: 9084 17971 18 4726 113
 - attr(*, "na.action")=Class 'omit'  Named int 32313
  .. ..- attr(*, "names")= chr "NA"
```

```
> reviews$Score<-as.factor(reviews$Score)
> table(reviews$Score)
```

```
negative positive
    5224    27088
```



Cleaning text data: We use text mining package named tm for removing numbers and punctuation, breaking apart sentences into individual words and Stemming which involves reducing words to their root form.The first step in processing text data involves creating a corpus, which is a collection of text documents. To create a corpus, we'll use the VCorpus() function in the tm package, which refers to a volatile corpus.

```
> library(tm)
> reviews_corpus <- VCorpus(VectorSource(reviews$Summary))
> print(reviews_corpus)

<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 32312

> inspect(reviews_corpus[1:2])
```

```
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 21

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 17
```

```
> as.character(reviews_corpus[[1]])
```

```
[1] "Good Quality Dog Food"
```

```
> lapply(reviews_corpus[1:2], as.character)
```

```
$`1`
[1] "Good Quality Dog Food"

$`2`
[1] "Not as Advertised"
```

```
> library(SnowballC)
> reviews_corpus_clean <- tm_map(reviews_corpus, content_transformer(tolower))
> as.character(reviews_corpus[[1]])
```

```
[1] "Good Quality Dog Food"
```

```
> as.character(reviews_corpus_clean[[1]])
```

```
[1] "good quality dog food"
```

```
> reviews_corpus_clean <- tm_map(reviews_corpus_clean, removeNumbers)
> reviews_corpus_clean <- tm_map(reviews_corpus_clean, removeWords, stopwords())
> reviews_corpus_clean <- tm_map(reviews_corpus_clean, removePunctuation)
> reviews_corpus_clean <- tm_map(reviews_corpus_clean, stemDocument)
```

Splitting text documents into words: The tm package provides functionality to tokenize the review message corpus. The DocumentTermMatrix() function will take a corpus and create a data structure called a Document Term Matrix (DTM) in which rows indicate documents and columns indicate terms.

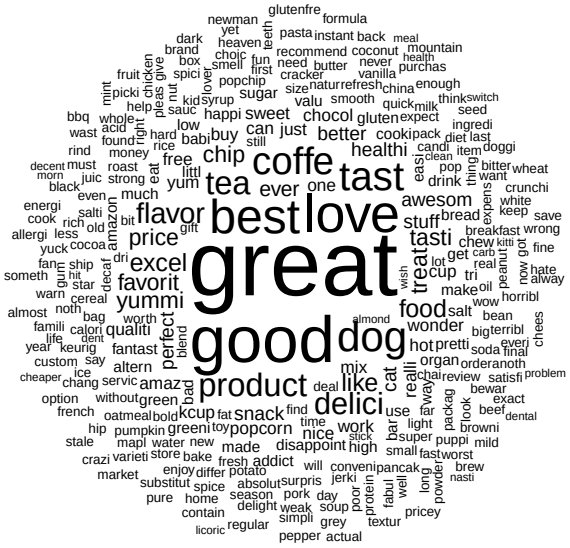
```
> reviews_dtm <- DocumentTermMatrix(reviews_corpus_clean)
> reviews_dtm

<<DocumentTermMatrix (documents: 32312, terms: 6831)>>
Non-/sparse entries: 89826/220633446
Sparsity           : 100%
Maximal term length: 35
Weighting          : term frequency (tf)
```

Creating training and testing data sets: I've divided the data into 75 percent for training and 25 percent for testing.

```
> reviews_dtm_train <- reviews_dtm[1:24234, ]
> reviews_dtm_test <- reviews_dtm[24235:32312, ]
> reviews_train_labels <- reviews[1:24234, ]$Score
> reviews_test_labels <- reviews[24235:32312, ]$Score
> prop.table(table(reviews_train_labels))

reviews_train_labels
 negative  positive
0.1630767 0.8369233

> prop.table(table(reviews_test_labels))

reviews_test_labels
 negative  positive
0.1574647 0.8425353
```

Visualizing text data - Word Clouds: A word cloud is used to visualize the frequency at which words appear in text data. The cloud is composed of words scattered somewhat randomly around the figure. Words appearing more often in the text are shown in a larger font, while less common terms are shown in smaller fonts. The wordcloud() of wordcloud package is used visualize this type of diagrams.

```
> library(wordcloud)
> wordcloud(reviews_corpus_clean, min.freq = 50, random.order = FALSE)
```




```
> positive <- subset(reviews, Score == "positive")
> negative <- subset(reviews, Score == "negative")
> par(mfrow=c(1,2))
> wordcloud(positive$Summary, max.words = 40, scale = c(3, 0.5))
> wordcloud(negative$Summary, max.words = 40, scale = c(3, 0.5))
> cat("Positive reviews include words such as great and good,
+     where these words do not appear in the negative review cloud.")

Positive reviews include words such as great and good,
    where these words do not appear in the negative review cloud.
```

Creating indicator for frequent terms:

```
> reviews_freq_words <- findFreqTerms(reviews_dtm_train, 5)
> str(reviews_freq_words)

 chr [1:1517] "abl" "absolut" "accept" "acid" "action" "actual" ...

> reviews_dtm_freq_train<- reviews_dtm_train[ , reviews_freq_words]
> reviews_dtm_freq_test <- reviews_dtm_test[ , reviews_freq_words]
> convert_counts <- function(x) {
+     x <- ifelse(x > 0, "Yes", "No")
+   }
> reviews_train <- apply(reviews_dtm_freq_train, MARGIN = 2,
+                                          convert_counts)
> reviews_test <- apply(reviews_dtm_freq_test, MARGIN = 2,
+                                         convert_counts)
```

Training a model on the data: We create
reviews$_c$*lassifierobjectwhichisanaiveBayesclassifierobjectthatcanbeusedtomakepredictions.As*

```
> library(e1071)
> reviews_classifier2 <- naiveBayes(reviews_train,
+               reviews_train_labels, laplace = 1)
```

Evaluating model performance: To compare the predictions to the true values, we'll use the CrossTable() function in the gmodels package.

```
> reviews_test_pred2 <- predict(reviews_classifier2, reviews_test)
> library(gmodels)
> CrossTable(reviews_test_pred2, reviews_test_labels,
+     prop.chisq = FALSE, prop.t = FALSE, prop.r = FALSE,
+     dnn = c('predicted', 'actual'))
```

```
   Cell Contents
|-------------------------|
|                       N |
|           N / Col Total |
|-------------------------|


Total Observations in Table:  8078


             | actual
   predicted |  negative |  positive | Row Total |
-------------|-----------|-----------|-----------|
    negative |       600 |       228 |       828 |
             |     0.472 |     0.033 |           |
-------------|-----------|-----------|-----------|
    positive |       672 |      6578 |      7250 |
             |     0.528 |     0.967 |           |
-------------|-----------|-----------|-----------|
Column Total |      1272 |      6806 |      8078 |
             |     0.157 |     0.843 |           |
-------------|-----------|-----------|-----------|
```

```
> (6578+600)/(228+672+6578+600)

[1] 0.8885863

>
```

Conclusion: A total of 6578 + 600 = 7178 of the 8078 reviews are correctly classified which is 88.8 percent accuracy. Positive reviews include words such as great and good, where the word "not" is most frequently appeared in the negative review cloud.

References: https://www.kaggle.com/snap/amazon-fine-food-reviews R Data Analysis and Visualization by Jaynal Abedin;