Predicting Business Yelp Ratings through the Text Analysis of Reviews

Lucas Neo

22nd November 2015

Introduction

In recent years, the rapid expansion of technology into social life is marked by the emergence of online platforms that allow individuals from across the world to interact and share their experiences. One such platform is Yelp, a service which crowd-sources reviews about local businesses. Focussing on data about lowly-rated and highly-rated restaurants, this report will analyse the most common terms used in reviews and aim to predict if a business will be rated as such.

Methods and Data

Data Preparation

First, the category column is cleaned and the dataset is segmented by businesses that contain "restaurant" in their category.

```
biz$categories = sapply(biz$categories, toString)
biz$categories = sapply(biz$categories, tolower)
restaurants = biz[grep("restaurant",biz$categories),]
```

Next, the businesses table is merged with the review table and businesses that are not restaurants are removed from the dataset.

```
biz.reviews = merge(reviews, restaurants, by="business_id", all.x=T)
biz.reviews = biz.reviews[!is.na(biz.reviews$stars.y),]
```

Before analysing the review data, the text has to be transformed into a usable format. The `tm_map` function from the `tm` package was used to store the text in a document matrix, change all the text to lowercase, remove punctuation, strip the whitespace, and remove all english stop words. A function for cleaning corpuses was created to make this process easily reproducible.

```
corpuscleaner <- function(corpus){
  clean <- tm_map(corpus, content_transformer(tolower))
  clean <- tm_map(clean, removePunctuation)
  clean <- tm_map(clean, stripWhitespace)
  clean <- tm_map(clean, removeWords, stopwords("english"))
  return(clean)
}
```

Utilising the function described on the `tm` project page, unigrams, bigrams, and trigrams can be extracted into document term matrices for analysis.

```
# Unigrams
unigramTokenizer = function(x){
  unlist(lapply(ngrams(words(x), 1), paste, collapse = " "), use.names = FALSE)
}

# Bigrams
bigramTokenizer = function(x){
  unlist(lapply(ngrams(words(x), 2), paste, collapse = " "), use.names = FALSE)
}

# Trigrams
trigramTokenizer = function(x){
  unlist(lapply(ngrams(words(x), 3), paste, collapse = " "), use.names = FALSE)
}
```

Initially, these ngrams were generated for each business star rating i.e. 1, 1.5, 2, 2.5, 4.5, 5 stars. However, an inspection of the document matrices revealed that term frequencies were very low when n is too small. To counter this problem, 1 and 1.5 business star reviews were combined as were 2 and 2.5, and 4.5 and 5 star reviews.

```
corpus.1star <- Corpus(VectorSource(biz.reviews[biz.reviews$stars.y ==1 |
                                                biz.reviews$stars.y==1.5,]$text))

corpus.2star <- Corpus(VectorSource(biz.reviews[biz.reviews$stars.y ==2 |
                                                biz.reviews$stars.y==2.5,]$text))

corpus.5star <- Corpus(VectorSource(biz.reviews[biz.reviews$stars.y==5 |
                                                biz.reviews$stars.y==4.5,]$text))

all.corpus <- Corpus(VectorSource(biz.reviews.subset$text))
```

For each corpus, a sparsity threshold of 0.99 was applied and for cases when applying this threshold resulted in 0 terms, then `findFreqTerm` was used to gather terms that had a minimum frequency of 100. Finally, a single corpus is also generated for training the prediction models. Here, a 0.95 sparsity was applied to unigrams, 0.99 for bigrams, and a minimum frequency of 200 for trigrams.

Prediction

Training and test datasets are created from the main dataset and a seed is set to ensure that the results can be reproduced.

```
set.seed(2102)
df_trainrandom <- createDataPartition(df$star.rating, p=0.7, list=F)
df_train <- df[df_trainrandom,]
df_test <- df[-df_trainrandom,]
```

Then, 3 models are trained on the training dataset. Rpart, nnet, and LogitBoost are used.

```
model.rpart <- train(star.rating ~.,data=df_train,method="rpart")
model.nnet <- train(star.rating ~.,data=df_train,method="nnet")
model.logitboost <- train(star.rating~.data=df_train,method="LogitBoost")
```

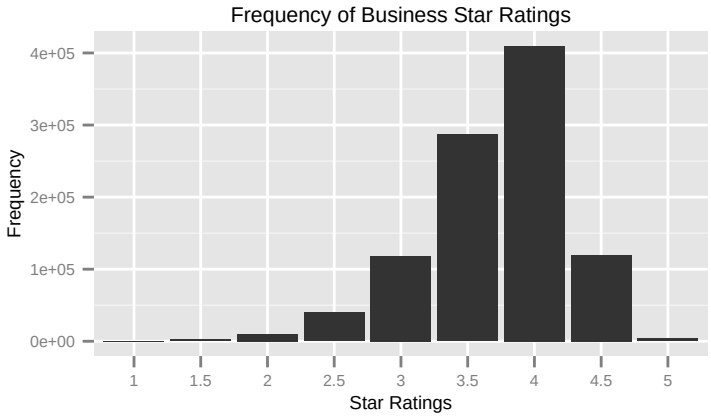
Once the models are trained we use the `predict` function to run them against the test dataset. This gives us an estimate of how accurate these models are when applied to previously unseen data.

```
predict.rpart <- predict(model.rpart, df_test)
predict.nnet <- predict(model.nnet, df_test)
predict.logitboost <- predict(model.logitboost,df_test)
```

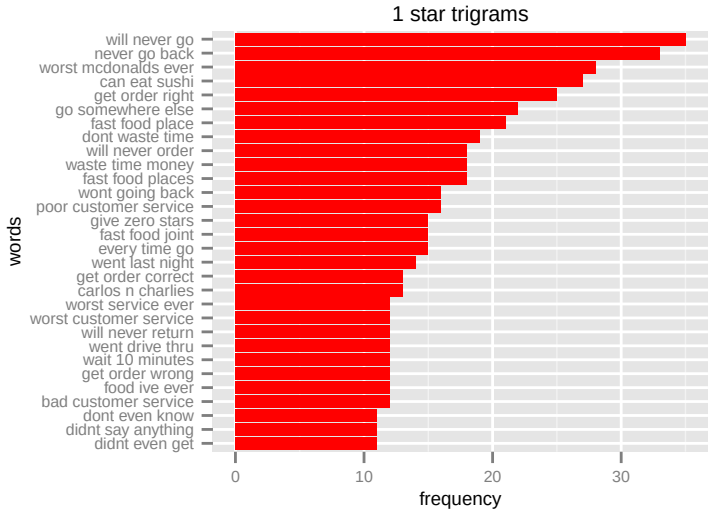
Results

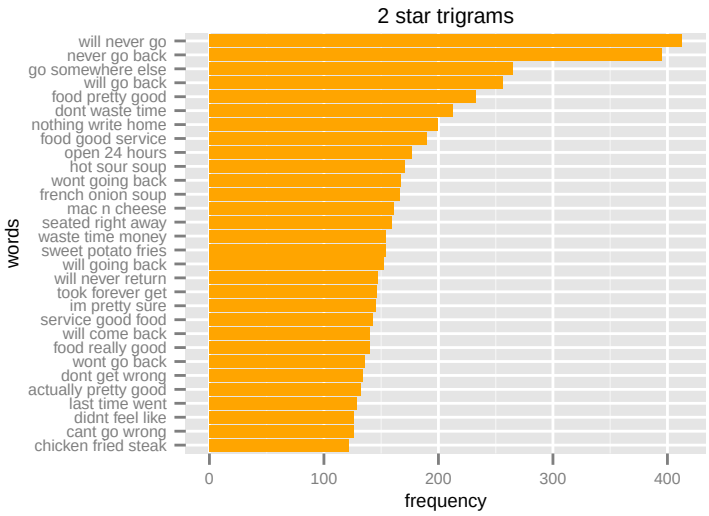
Exploratory Analysis

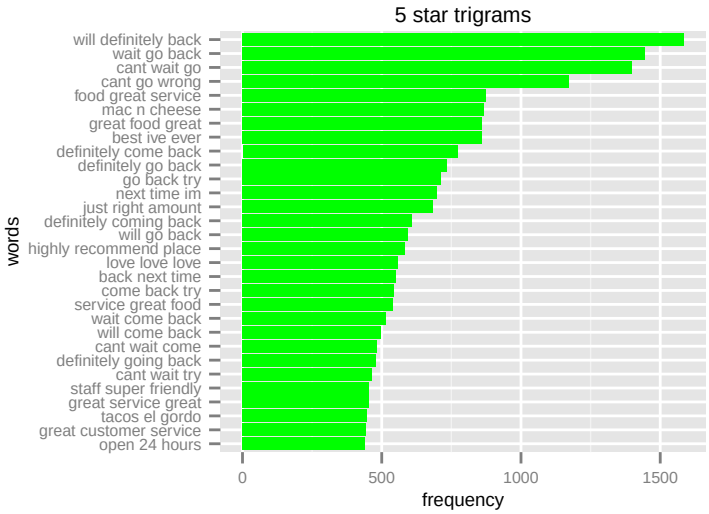
Beginning with an analysis of business star ratings, it was uncovered that Yelp reviewers are skewed towards more positive ratings.



This finding sparked the cuorisity to find out if reviews on businesses above the 3rd quantile (4.5 and 5) differed from those below the 1st quantile (1, 1.5, 2, 2.5). By comparing the top 30 trigram term frequencies between these star ratings, some patterns have emerged.







From the graphs, it is evident that reviews of restaurants with 5 star business ratings are typically more postive than 1 or 2 star restaurants. For example, the top words in reviews for such restaurants were "will never go" and "never go back". Also within the top 30 terms were sub topics such as poor customer service and fast food. Conversely, 5 star rated restaurants had "great food" and "great service" within the top 30. Based on these findings, it could be possible to predict the business rating of a restaurant based on the presence of certain words in their reviews.

Models

As described in the "Methods and Data" section, 3 models were first trained against 70% of the data (training set). Then, the trained models are used to predict the outcomes on the test data set. The results of these predictions gives an estimate of the model's accuracy.

```
results.rpart <- confusionMatrix(df_test$star.rating,predict.rpart)
results.nnet <- confusionMatrix(df_test$star.rating,predict.nnet)
results.logitboost <- confusionMatrix(df_test$star.rating,predict.logitboost)

results.rpart$overall[1]
```

```
##  Accuracy
```

```
## 0.7339519
```

```
results.nnet$overall[1]
```

```
##  Accuracy
## 0.7754213
```

```
results.logitboost$overall[1]
```

```
##  Accuracy
## 0.7353124
```

Discussion

All 3 models are fairly accurate in predicting the business star rating of restaurants. The neural networks model performed the best with a 78% accuracy. However, it should be noted that all 3 also had problems with classifying 1 star business ratings and sensitivity was largely not applicable for all the models. Even though 1 and 1.5 star businesses were combined, the total sample size is still too small for there to be an accurate prediction.

Drawing from the findings here, it is likely that the models can be improved on to increase their accuracy and predictive value. One immediate use for these models is for Yelp businesses to apply them to other channels where they may receive qualitative feedback. This would allow them to gauge how users on those platforms feel about their service and food; this provides them with a standardized score to measure their performance ~ the Yelp business rating, and allows them to improve on their future offerings.