Attributes favourable to Restaurant Success

Edwin Seah

21 November 2015

1. Introduction

1.1 Task Preamble In the Yelp! dataset challenge, participants are provided with rich real-life data from Yelp! and asked to present and answer open-ended questions about the list of businesses, ratings and user behaviour. In the project here, data is taken from the sixth series of the Yelp! challenge. Due to time limits, the depth of the analysis will be limited and far less ambitious than some of the models presented in previous winning submissions.

1.2 Research Questions: Large numbers of eating establishments are represented in the dataset; Yelp!'s raison d'etre and format tends to favour the review of such businesses and its use by a community of users. The project is therefore interested in the following questions:

- What characteristics does a highly rated restaurant possess that are roughly consistent across different locations (states/cities)?
- Do such restaurants obtain more positive tips that are useful to potential customers than those rated lowly?
- Is it possible to generate a reasonable and useful model that predicts their rating from these characteristics/review/tip types?

The response variable is `stars` (business rating) which has levels from 1 to 5 in 0.5 steps. A classification attempt is made to determine the most favourable characteristics by location(state, city), then a check is made on the sentiment reflected in the review text to see if they help provide a there is a clear distinction between sentiment scores and what users assigned as ratings. For brevity, not all code is shown although the the full code is available from the Rmd document.

2. Methods

2.1. Data Sources The dataset used is the academic dataset provided from the Yelp! dataset challenge. For consistency, the dataset used is downloaded from https://d396qusza40orc.cloudfront.net/dsscapstone/dataset/ and read into a series of three data frames (business, tip, review, and user) using `readr::read_lines()` and `jsonlite::fromJSON()`, using the snippet `lapply(filesToRead, function(x) fromJSON(sprintf("[%s]", paste(read_lines(x), collapse=",")), flatten=TRUE))`. Since the dataset is large and converting from JSON into a usable format is excruciatingly slow, the data frames are cached into .rds files and read from there.

```
filenames <- c("business", "tip", "review", "user")
vecRDS <- paste0("../data/yelp/", filenames, ".rds")
business <- readRDS(vecRDS[1])
tip <- readRDS(vecRDS[2])
review <- readRDS(vecRDS[3])
user <- readRDS(vecRDS[4])
```

For usage in sentiment analysis of review/tip texts, a combined list of positive/negative words from A.Finn(2011) `dict.afinn` and Hu,Liu (2004) `dict.huliu` is used. They are merged after scaling the former AFINN-111 (containing nuanced scores) to between -1 and 1 to match the latter, then joined into one

set of positive and negative words. The enlarged set provides better coverage over a wider variety of English terms. `vec.pos` and `vec.neg` are kept as vectors of the final postive and negative words.

```
# Merge them into one single set, using only -1 or 1 to denote +/- sentiment
dict.terms <- merge(dict.huliu, dict.afinn, by.x="word", by.y="word", all=TRUE) %>% mutate(score=ifelse
vec.pos <- as.vector(dict.terms[dict.terms$score>0,]$word)
vec.neg <- as.vector(dict.terms[dict.terms$score<0,]$word)
```

2.2. Cleaning and Transforming To filter for businesses which are restaurants or serve food, `grep` is run on the category column from the business dataset for "Food|Restaurants", and subsets are taken accordingly by intersecting with `business_ids` from reviews and tips. A series of transformations is applied as follows:

- factorizing stars variable
- flattening of nested lists
- removing redundant and irrelevant variables/attributes, such as "Hair|Insurance"
- turn column names R-compatible using `make.names()`
- converting city names from unicode to ascii equivalent

```
# Subset the other data frames for only restaurants (subset is fastest)
b.rest <- business[grep("Food|Restaurants", business$categories),]
b.rest$categories <- sapply(b.rest$categories, toString)
b.rest <- subset(b.rest, select=-c(type, grep("Hair|Insurance", names(b.rest)))) # remove irrelevant
t.rest <- subset(tip, business_id %in% b.rest$business_id, select=-c(type))
r.rest <- subset(review, business_id %in% b.rest$business_id, select=-c(type))
names(b.rest) <- make.names(names(b.rest)) # make R-compatible colnames
b.rest <- b.rest %>% mutate(stars=as.factor(stars)) # Transform stars into character
```

NA

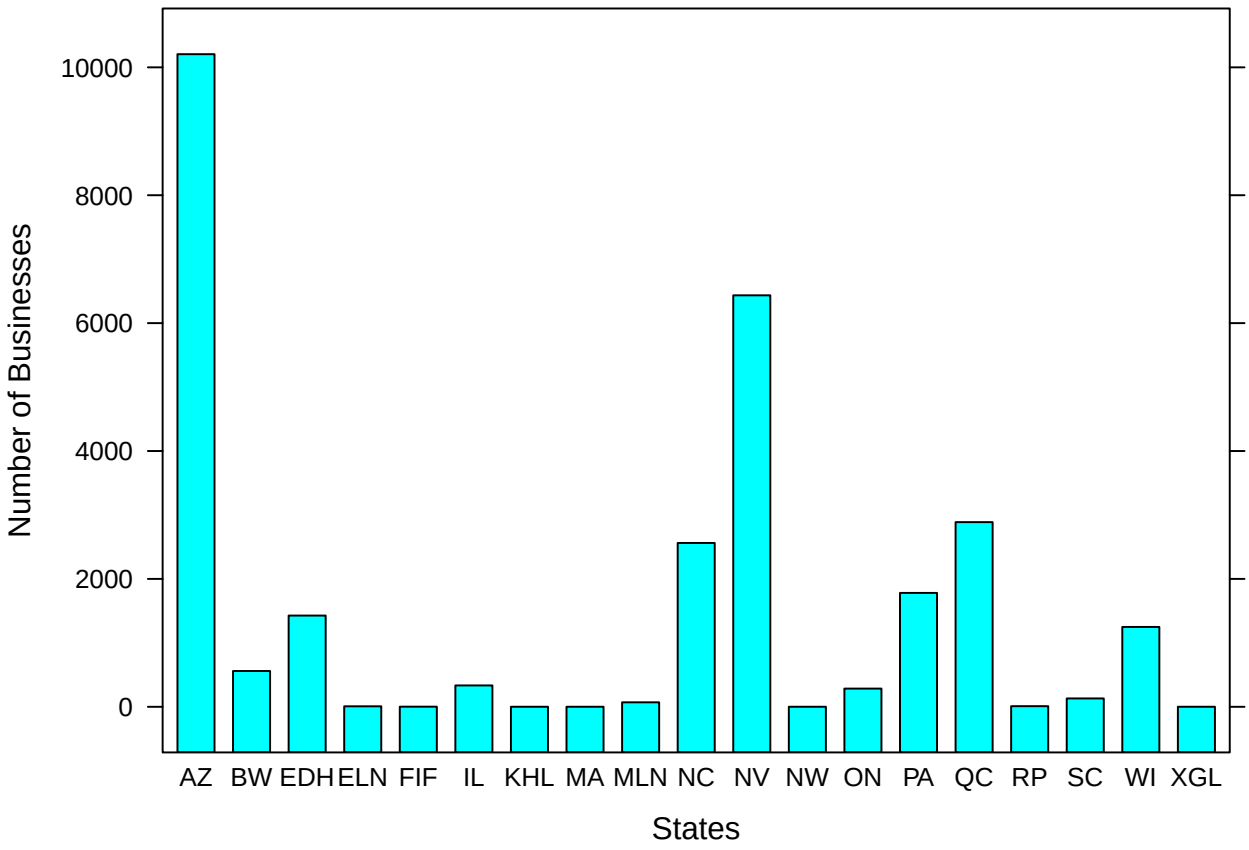
```
b.open <- b.rest[,grep("hours.+open", names(b.rest),)]
b.open <- !is.na(b.open)
colnames(b.open) <- gsub("(hours.)(.+).(open)", "\\3.\\2", colnames(b.open))
#mx.open <- Matrix::Matrix(as.matrix(b.open), sparse=TRUE)
```

The text in reviews is also grouped according to how many stars the reviewer accorded the business along with their reviews, which can the provide a baseline sentiment to use for our classification task.

```
# Split up into 1 to 5 star reviews
rev1 <- r.rest[r.rest$stars==1, "text"]
rev2 <- r.rest[r.rest$stars==2, "text"]
rev3 <- r.rest[r.rest$stars==3, "text"]
rev4 <- r.rest[r.rest$stars==4, "text"]
rev5 <- r.rest[r.rest$stars==5, "text"]
```

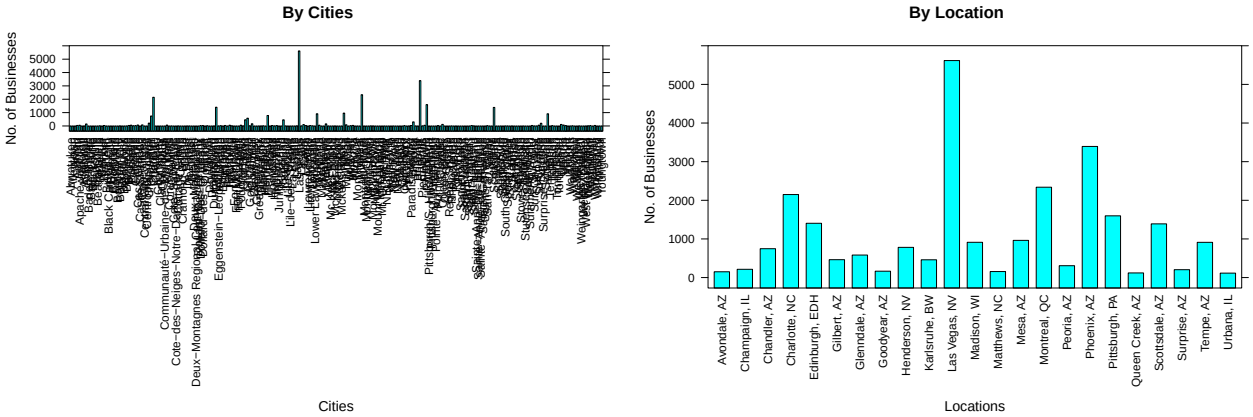
2.3. EDA Checking the distribution of restaurants brings us into an issue with our research question; inconsistency in numbers of restaurants across states/cities within the dataset may derail attempts at classification. Indeed, some states are represented by only one entry.

```
b.loc <- subset(b.rest, select=c(business_id, stars, state, city, neighborhoods, latitude, longitude))
locS <- b.loc %>% group_by(state) %>% summarize(n())
lattice::barchart(locS$n~as.factor(locS$state), xlab="States", ylab="Number of Businesses")
```



We arrive at the following candidate cities for location-specific restaurants after restricting to states with having at least the median number of businesses amongst states, and at least the mean number amongst cities. Plotting with `lattice` yields:

```
library(gridExtra); library(lattice)
vec.state <- locS[locS$n>median(locS$n),]$state
b.loc <- b.loc[b.loc$state %in% vec.state,]
locC <- b.loc %>% group_by(state, city) %>% summarize(n=n())
locC1 <- locC[locC$n>mean(locC$n),] %>% mutate(loc=paste0(city,", ",state))
vec.city <- unique(locC1$city)
scaleCtl <- list(x=list(rot=90))
plot1 <- barchart(locC$n~as.factor(locC$city), xlab="Cities", ylab="No. of Businesses", scales=scaleCtl
plot2 <- barchart(locC1$n~as.factor(locC1$loc), xlab="Locations", ylab="No. of Businesses", scales=scal
grid.arrange(plot1, plot2, ncol=2)
```



Our plots show the large variance in numbers; some states/cities are actually represented by only one business, let alone individual neighbourhoods.

Nevertheless, to proceed the analysis will restrict the dataset then to these cities at the resolution of city-level, and also simplisitcally consider different parts of the city to be part of the same city. The data set used for classification/prediction of star ratings viz their business attributes and opening days then are for the following states and cities, denoted as `B`:

```
b.loc <- b.loc[b.loc$city %in% vec.city,]
B <- select(b.rest, c(business_id, state, city))
B <- cbind(B, b.att, b.open) # combine with attributes, opening days
B <- B[B$business_id %in% b.loc$business_id,]
```

2.4. Establishing an error estimate baseline by classifying attributes/opening days Bearing in mind that there are significant numbers of NAs in the data, an attempt is still made to classify and predict the star ratings of the restaurants based on their attributes and days open for business. The previous data is split into train (60%), test (20%) and validation sets (20%) using a custom-defined function `createSets`.

A variety of fits for predicting `stars` is attempted, namely classification tree via the `rpart` method from `caret`, a boosted regression tree via `gbm`. Training is performed on the train set, and estimates obtained from the test set, with the validation set untouched. The fits will help give an idea of the error estimate from simply classifying by just these attributes.

```
set.seed(350);
data <- B[,4:length(B)]; sets <- createSets(data, 0.6)
trainSet <- sets[[1]]; testSet <- sets[[2]]; validationSet <- sets[[3]]
```

```
## RPart
modRpart <- train(as.factor(stars)~., method="rpart", control=rpart.control(xval=5, minsplit=15, minbuc
## GBM with 3-fold CV
fitControl <- trainControl(method = "repeatedcv", number = 3, repeats = 3)
modGBM <- train(stars~., method="gbm", data=trainSet, na.action=na.pass, trControl=fitControl, verbose=
```

2.5. Sentiment scoring review text Using the positive and negative word lists combined from A.Finn(2011) and Hu,Liu (2004), we grade the sentiment for every restaurant review from the review dataset and generate a score to represent the nominal sentiment per star grouping. The review text is transformed to lower-case with newlines, stopwords, punctuation and numbers removed, using functions from `tm`. However, since corpus building with `tm` is slow for the size of the text, a custom function is used instead.

```
# Returns a score of +/- words from vector "txt" using positive and negative word vectors "pos" and "ne
library(tm)
scoreReviewText <- function(txt, pos, neg) {
    t <- tolower(txt)
    t <- removeWords(t, words = stopwords("en"))
    t <- gsub("\\.", " ", t)
    t <- gsub("\\\n", "", t)
    t <- removePunctuation(t)
    t <- removeNumbers(t)
    s <- lapply(strsplit(t, " ", fixed=TRUE), function(x) ifelse(!duplicated(x), x, ""))
    sapply(s, function(x) sum(pos %in% x)-sum(neg %in% x))
}
```

```
# Split up into 1 to 5 star reviews and score them for sentiment
rev1 <- r.rest[r.rest$stars==1, "text"]; score1s <- scoreReviewText(rev1, vec.pos, vec.neg)
rev2 <- r.rest[r.rest$stars==2, "text"]; score2s <- scoreReviewText(rev2, vec.pos, vec.neg)
rev3 <- r.rest[r.rest$stars==3, "text"]; score3s <- scoreReviewText(rev3, vec.pos, vec.neg)
rev4 <- r.rest[r.rest$stars==4, "text"]; score4s <- scoreReviewText(rev4, vec.pos, vec.neg)
rev5 <- r.rest[r.rest$stars==5, "text"]; score5s <- scoreReviewText(rev5, vec.pos, vec.neg)
```

3. Results

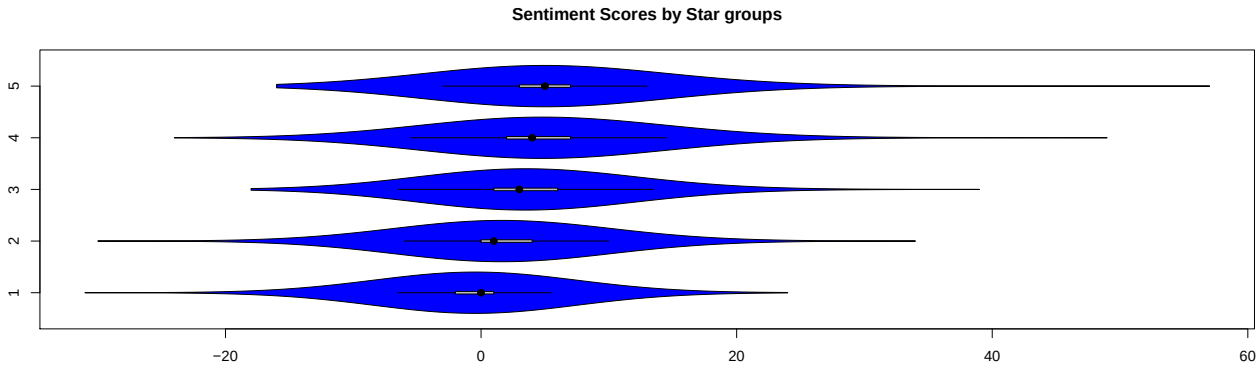
Classifying using attributes and opening days, the classification error estimates (calculated with a user-defined function `metrics(model, newdata)`) from both methods indicate clearly that a coin-flip works better. Our results from omitting opening days also seem to indicate opening days hardly matter either in influencing the outcome, at least based on the error estimates:

```
modError <- rbind(Rpart=metrics(modRpart, testSet)[[3]], GBM=metrics(modGBM, testSet)[[3]])
modErrorAtt <- rbind(Rpart=metrics(modRpartAtt, testSet)[[3]], GBM=metrics(modGBMAtt, testSet)[[3]])
modError <- cbind(modError, modErrorAtt)
colnames(modError) <- c("ErrEst(%).all", "ErrEst(%).attributesOnly")
modError
```

```
##       ErrEst(%).all ErrEst(%).attributesOnly
## Rpart          71.5                    70.6
## GBM            69.5                    68.2
```

Plotting the results from our scoring of the positive/negative sentiment of review text shows that ratings do reflect the reviews given. It indicates that reviews do have a tendency to match the ratings given, although the variability of the sentiment is rather large. This indicates that the word lists used are possibly insufficient to cover the breadth of sentiment, while saying nothing about whether there are large numbers of shill reviews within the lot.

```
library(vioplot)
vioplot::vioplot(score1s, score2s, score3s, score4s, score5s, col="blue", horizontal=TRUE, rectCol="gra
title("Sentiment Scores by Star groups")
```



4. Discussion

4.1 Assessing the results The results from the attempt at classification were rather dismal, characterised by high error estimates. Our classification task was hampered by large numbers of NAs within the attributes, not least the fragmented nature of the data ill-suited to the question, or rather, vice versa. Insufficient data was usable to train any classification model based simply on business attributes or any combination thereof. The disappointing conclusion may be that classification by attribute is the wrong approach to use in this case.

On a positive note, there is distinct banding evident from the sentiment scores quantitatively measured from review text. This implies that the worth of the dataset (indeed the Yelp data) is most likely in the text rather than the business characteristics (which merely serve a purely categorization role with little use for prediction tasks).

```
as.data.frame(cbind(stars=1:5, rbind(summary(score1s), summary(score2s), summary(score3s), summary(scor
```

```
##   stars Min. 1st Qu. Median    Mean 3rd Qu. Max.
## 1     1  -31      -2      0 -0.4275       1   24
## 2     2  -30       0      1  1.7730       4   34
## 3     3  -18       1      3  3.8210       6   39
## 4     4  -24       2      4  5.1930       7   49
## 5     5  -16       3      5  5.4410       7   57
```

Perhaps restaurants already know that beyond simple local discovery, attributes and taking time to enhance the listing is not nearly as important as concentrating on their core function; GOOD FOOD. The result is reflected in the review texts; good food draws effusive reviews and higher ratings that tend to accumulate over time.

4.2. Ideas for follow-ups: Generalising the approach to non-food related businesses may not help us find a good model of attributes either. Approaches which focus on the review text itself should work better than business characteristics; after all, the reviews are continuously updated with newer reviews/ratings and grows, while business attributes are inherently more static. Given that there sentiment scores look like they fall into bands, albeit overlapping, making use of reviews upvoted by users may be helpful in providing a scaling factor to use, or simply to predict ratings.

More interesting follow-ups would include investigation into applying further NLP techniques on the text data, along with additional qualitative factorization methods on the user-contributed corpus. Perhaps an investigation into multiple ngram tokenization of phrases will provide better quality sentiment scores, or provide a proxy voting mechanism as additional explanatory variables, rather than just one-word sentiment alone.

5. Project Repo and References

- All files and full code used are available from the Github Project Repository
- Yelp Dataset used
- Build out valid JSON array from given pseudo-JSONforum discussion link
- Building out valid JSON array from given pseudo-JSON by first constructing valid JSONstackoverflow link :
- Yelp dataset challenge data dictionary NA NA
- Yelp Dataset Challenge 2014 Submission Kevin Hung and Henry Qiu, University of California, San Diego.
- Collective Factorization for Relational Data: An Evaluation on the Yelp Datasets Nitish Gupta, Indian Institute of Technology, Kanpur and Sameer Singh, University of Washington.
- Naive bayes in R

NA + Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, Washington, USA. Links: Paper, Page, Word list