Mysteries of the (Yelp) Orient

Eugene Woo

March 2017

Background and Summary

For the Springboard Capstone project, I analyzed Yelp's public dataset to explore hacks that could enhance the customer experience and ultimately user retention.

The Yelp app does a reasonably good job of recommending restaurants using a top-down approach comprising proximity, star ratings and review counts.

During a trip to San Francisco, however, I had a hard time locating a Chinese restaurant that served a dish called 'Ee-Fu Noodles'. The search results seem to favour highly-rated Chinese restaurants whose reviews merely contain 'noodles' but do not precisely match the actual dish name.

First-time visitors to a city might be in the same situation as me but now the problem is picking the right restaurant out of a ton of listings with similar ratings. Even after all of that, they might be disappointed to arrive at the restaurant and find out the dish isn't on the menu!

In his excellent Springboard capstone 'Eat, Rate, Love', Robert Chen proposed upweighting frequent reviewers and generating authenticity ratings based on the reviewers' names. His approach cracks the 'restaurant authencity' problem from the top-down; my solution is bottom-up as it helps users narrow their search quickly according to their favorite dishes.

With that I came up with a Shiny App to query Chinese restaurants by popular dishes ranked according to how often they were mentioned. I focused on reviews from Arizona and Nevada as they contributed 83% of the total reviews in the dataset.

The following report details my approach to extracting and contextualising the insights from the Yelp dataset. Hope you enjoy reading it as much as I had fun working on it!

Data Exploration and Cleaning

Importing Datasets

Normally I would run `sapply` on the vector of package names but this doesn't work in RMarkdown due to error messages.

```r
library(jsonlite)
library(dplyr)
library(tidyr)
library(stringr)
library(qdap)
library(ggplot2)
library(scales)
library(tm)
```

Next I import the Business and Reviews datasets using `stream_in` function from the 'jsonlite' package, after which I save in .Rds format for faster importing.

```r
## Load JSON files and save in .Rds format
#biz_json <- stream_in(file("yelp_academic_dataset_business.json"))
#review_json <- stream_in(file("yelp_academic_dataset_review.json"))
#saveRDS(biz_json, "biz.Rds")
```

```r
#saveRDS(review_rds, "review.Rds")

## Import .Rds files
biz <- readRDS("biz.Rds")
biz_df <- as.data.frame(biz)
review <- readRDS("review.Rds")
review_df <- as.data.frame(review)
```

Inspecting the data with `str` and `summary` show the Reviews dataset contains votes, user ids, star ratings and the review texts. The Business dataset - with `summary` hidden due to length - shows the business details such as shop names & addresses, opening hours, attributes such as 'Good for Kids', 'Wi-Fi' or 'Takes Reservations'.

The common variable between the tables is **Business ID**, which I will use as the matching criteria to join both datasets later on.

```r
str(review_df, max.level = 1)
```

```
## 'data.frame':    2225213 obs. of  8 variables:
##  $ votes      :'data.frame': 2225213 obs. of  3 variables:
##  $ user_id    : chr  "PUFPaY9KxDAcGqfsorJp3Q" "Iu6AxdBYGR4A0wspR9BYHA" "auESFwWvW42h6alXgFxAXQ" "uK8
##  $ review_id  : chr  "Ya85v4eqdd6k9Od8HbQjyA" "KPvLNJ21_4wbYNctrOwWdQ" "fFSoGV46Yxuwbr3fHNuZig" "Di3
##  $ stars      : int  4 5 5 5 3 1 4 5 5 3 ...
##  $ date       : chr  "2012-08-01" "2014-02-13" "2015-10-31" "2013-11-08" ...
##  $ text       : chr  "Mr Hoagie is an institution. Walking in, it does seem like a throwback to 30 y
##  $ type       : chr  "review" "review" "review" "review" ...
##  $ business_id: chr  "5UmKMjUEUNdYWqANhGckJw" "5UmKMjUEUNdYWqANhGckJw" "5UmKMjUEUNdYWqANhGckJw" "UsF
```

```r
summary(review_df)
```

```
##      votes.funny         votes.useful         votes.cool      
##  Min.   :  0.00000   Min.   :  0.00000   Min.   :  0.00000  
##  1st Qu.:  0.00000   1st Qu.:  0.00000   1st Qu.:  0.00000  
##  Median :  0.00000   Median :  0.00000   Median :  0.00000  
##  Mean   :  0.44123   Mean   :  1.02101   Mean   :  0.54964  
##  3rd Qu.:  0.00000   3rd Qu.:  1.00000   3rd Qu.:  1.00000  
##  Max.   :142.00000   Max.   :167.00000   Max.   :138.00000  
##    user_id           review_id             stars            date          
##  Length:2225213     Length:2225213     Min.   :1.000   Length:2225213    
##  Class :character   Class :character   1st Qu.:3.000   Class :character  
##  Mode  :character   Mode  :character   Median :4.000   Mode  :character  
##                                        Mean   :3.756                     
##                                        3rd Qu.:5.000                     
##                                        Max.   :5.000                     
##      text               type           business_id       
##  Length:2225213     Length:2225213     Length:2225213    
##  Class :character   Class :character   Class :character  
##  Mode  :character   Mode  :character   Mode  :character  
##                                                          
##                                                          
##                                                          
```

```r
str(biz_df, max.level = 1)
```

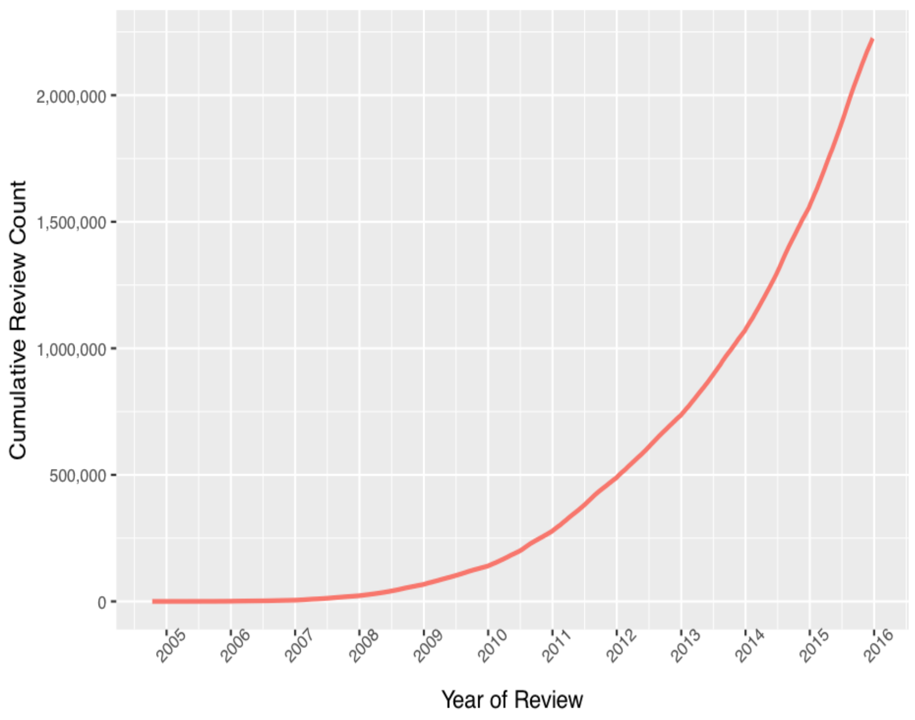
```
## 'data.frame':    77445 obs. of  15 variables:
##  $ business_id  : chr  "5UmKMjUEUNdYWqANhGckJw" "UsFtqoBl7naz8AVUBZMjQQ" "3eu6MEFlq2Dg7bQh8QbdOg" "cl
##  $ full_address : chr  "4734 Lebanon Church Rd\nDravosburg, PA 15034" "202 McClure St\nDravosburg, P
## $ hours        :'data.frame':   77445 obs. of  7 variables:
## $ open         : logi  TRUE TRUE TRUE FALSE TRUE TRUE ...
## $ categories   :List of 77445
##   .. [list output truncated]
## $ city         : chr  "Dravosburg" "Dravosburg" "Dravosburg" "Bethel Park" ...
## $ review_count : int  4 4 3 5 5 20 3 21 7 4 ...
## $ name         : chr  "Mr Hoagie" "Clancy's Pub" "Joe Cislo's Auto" "Cool Springs Golf Center" ...
## $ neighborhoods:List of 77445
##   .. [list output truncated]
## $ longitude    : num  -79.9 -79.9 -79.9 -80 -80.1 ...
## $ state        : chr  "PA" "PA" "PA" "PA" ...
## $ stars        : num  4.5 3.5 5 2.5 2.5 5 2.5 4 2.5 4 ...
## $ latitude     : num  40.4 40.4 40.4 40.4 40.4 ...
## $ attributes   :'data.frame':   77445 obs. of  36 variables:
## $ type         : chr  "business" "business" "business" "business" ...
```

Exploratory Data Analysis

Do we have enough reviews to analyse?

I'd like to first get a handle on the volume of reviews since the start date of the dataset. It seems we do have a large sample size (N = 2,225,213) amassed over 10 years, with the latter half coming in exponentially after 2013:

```r
## Arranging into two columns comprising date and cumulative review count
ts_reviews <- review_df %>%
  select(date, review_id) %>%
  group_by(date) %>%
  summarise_all(funs("reviews" = n())) %>%
  mutate(dates = as.Date(date), cumulative = cumsum(reviews)) %>%
  arrange(dates)

# Plot time series chart of review count
ggplot(ts_reviews, aes(x = dates, y = cumulative, col = "red")) +
  geom_line(size = 1) +
  scale_y_continuous(labels = comma) +
  labs(x = "Year of Review", y = "Cumulative Review Count") +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") +
  theme(legend.position = "none", axis.text.x = element_text(size = 8, angle = 45), axis.text.y = eleme
```



Scrubbing Inactive Users

Quick statistics on the reviews - there are users who posted only ONCE over the past 10 years!

```
ratings_stats <- review_df %>%
  select(user_id, stars) %>%
  group_by(user_id) %>%
  summarise_all(funs("count" = n(), "mean" = mean, "median" = median)) %>%
  arrange(count) %>%
  ungroup()

head(ratings_stats)
```

```
## # A tibble: 6 <U+00D7> 4
##                  user_id count  mean median
##                    <chr> <int> <dbl>  <dbl>
## 1 --0mI_q_0D1CdU4P_hoImQ     1     5      5
## 2 --20-1jZD5NnAnkwBBC_uQ     1     5      5
## 3 --37T3V10ZeoJ1I17D2Wzw     1     3      3
## 4 --52YqcuRttZN62TCKQdbw     1     5      5
## 5 --82_AVgRBsLw6Dhy8sEnA     1     4      4
## 6 --BGW_TY55SH-90iHmtitg     1     1      1
```

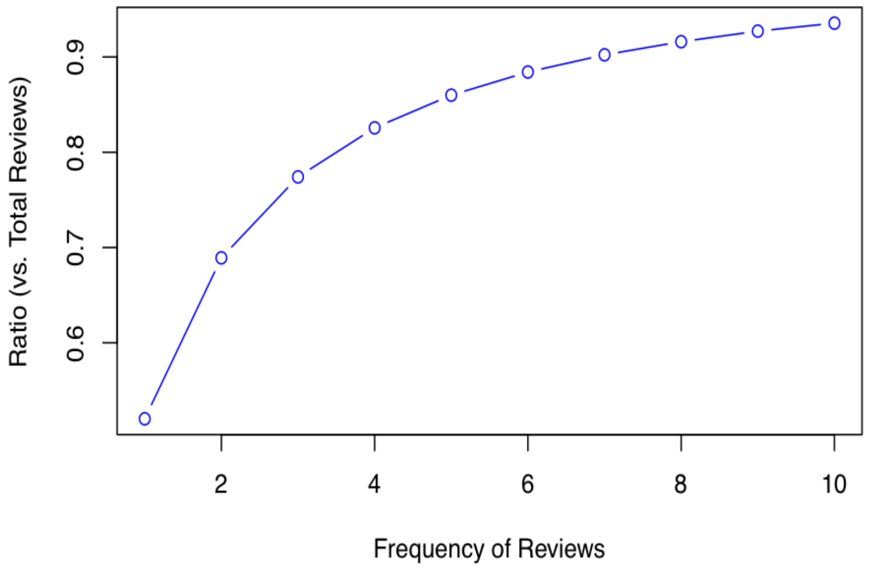
This isn't surprising given that social platforms generally maintain many inactive accounts. Yelp is the largest user-generated review platform in the US and inevitably collects large swathes of inactive users (lose interest after leaving 1 to 2 reviews) or fake accounts maintained by unscrupulous businesses.

Yelp's recommendation engine ensures that only ratings from friends, foodies and credible users (e.g. an

Italian rating Italian cuisine) are counted toward the aggregate Recommended Ratings. Assuming the ratings in the dataset are already clean, I'll need to scrub inactive users once more to improve reliability of the reviews.

My initial hypothesis is users with 2 or fewer reviews belong to the Inactive bucket and should be removed. Let's create a frequency chart to verify this:

```
inactive_ratio <- function(criteria){
  ratio <- rep(0, as.numeric(criteria))
  for (i in 1:as.numeric(criteria)){
    ratio[i] <- nrow(filter(ratings_stats, count <= i)) / nrow(ratings_stats)
  }
  plot(ratio, type = "b", xlim = c(1,i), xlab = "Frequency of Reviews", ylab = "Ratio (vs. Total Review
}

inactive_ratio(10)
```



Filtering out users who posted 1 to 2 reviews would remove ***70%*** of our observations! Unfortunately this confirms my suspicion that a large number of users are casual (albeit thoughtful) reviewers; that leaves me with no choice but to remove the single-review users!

```
## Remove these users from the Review dataset
inactive_raters <- ratings_stats %>%
      filter(count == 1)

review_clean_df <- review_df[!(review_df$user_id %in% inactive_raters$user_id),]
```

The cleaning removed 13% of our observations.

```
round(nrow(review_clean_df) / nrow(review_df), 2)
```

```
## [1] 0.87
```

Another point to consider is **Review Recency** such that restaurants reviewed more than 5 years ago may not exist today. In our case, the bulk of reviews came during the last 4 years plus Chinese restaurants tend to operate a very long time. For the purpose of the analysis, we'll assume that all reviewed restaurants are still operating.

Selecting States

Summary

I decide to focus on the reviews in Nevada and Arizona for my project and this section walks you through how I arrived at that conclusion.

The first step was merging the Business and Review data to produce a table of all reviews with matching business information. Parsing by business category gave me a showed Chinese contributing 15% of the Top 5 Restaurant categories. I'd definitely consider this a sizeable quantity to justify our focus on Chinese food reviews.

A view by state showed Nevada and Arizona contributing a overwhelming 83% of reviews. The count of total reviews and Chinese food reviews were much lower in subsequent states so they were dropped from analysis.

Step 1: Merging the Business and Review data

First things first, I consolidated business details, ratings and reviews into the same table while removing extraneous variables to trim the file:

```
review_biz <- merge(review_clean_df, biz_df, by = "business_id")
rev_biz_tidy <- review_biz %>%
                  select(-starts_with("hour"), -starts_with("attribute"), -contains("votes"), -contains
```

Step 2: Confirming Restaurant Category has Sufficient Chinese Reviews

We see approximately 190K reviews covering the Top 5 restaurant categories - Mexican, Italian, Pizza, American and Chinese. 15% of these were for Chinese and this justifies our focus.

```
cat_count <- rev_biz_tidy %>%
              group_by(as.character(categories)) %>%
              summarise(Count = n()) %>%
              arrange(desc(Count))

head(cat_count[, 1:2])
```

```
## # A tibble: 6 <U+00D7> 2
##                   `as.character(categories)` Count
##                                        <chr> <int>
## 1                c("Mexican", "Restaurants") 71496
## 2                  c("Pizza", "Restaurants") 33265
## 3                c("Restaurants", "Italian") 28811
## 4         c("American (New)", "Restaurants") 28779
## 5                c("Chinese", "Restaurants") 28187
## 6 c("Sushi Bars", "Japanese", "Restaurants") 27196
```

Step 3: Zooming in on Arizona and Nevada Chinese Restaurants

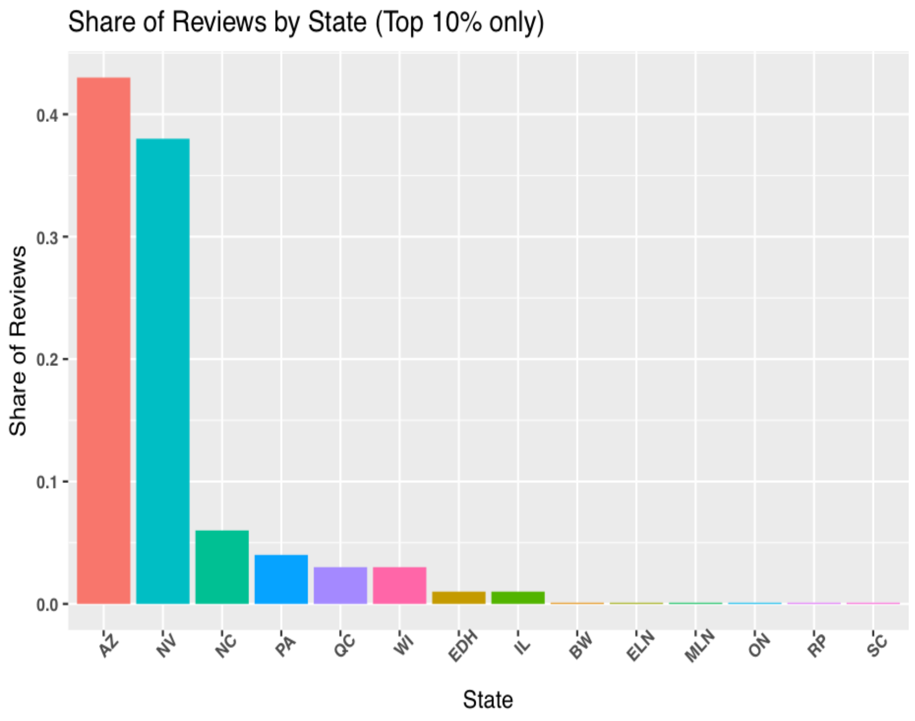
You'll notice the **Categories** column is made up of lists of descriptions tagged by users. The problem with this is an American restaurant could be categorised wrongly as a bar or a lounge. To be doubly sure we're not barking up the wrong tree, I separated the values in the lists:

```r
genre_count <- rev_biz_tidy %>%
                select(state, categories) %>%
                filter(str_detect(categories, "Restaurant")) %>%
                unnest(categories) %>%
                group_by(state) %>%
                count(categories) %>%
                arrange(desc(n))

genre_count[1:10,]
```

```
## Source: local data frame [10 x 3]
## Groups: state [3]
##
##    state             categories      n
##    <chr>                  <chr>  <int>
## 1     NV            Restaurants 480798
## 2     AZ            Restaurants 477563
## 3     AZ              Nightlife  80805
## 4     NC            Restaurants  80673
## 5     AZ                   Bars  77834
## 6     AZ         American (New)  69778
## 7     AZ                Mexican  67071
## 8     NV American (Traditional)  64480
## 9     NV              Nightlife  61227
## 10    AZ American (Traditional)  59497
```

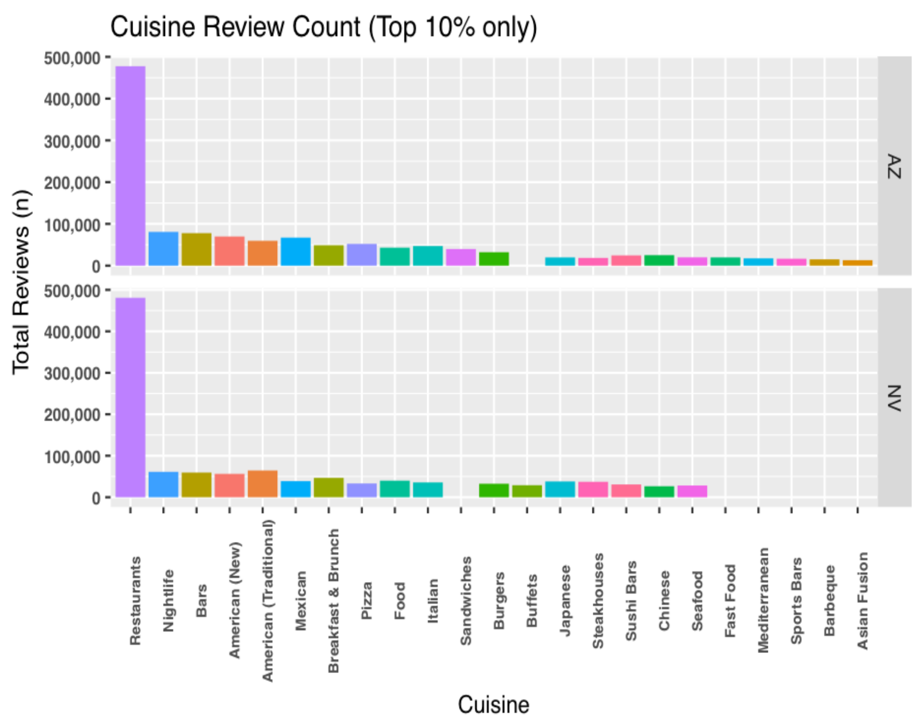
A view by state show Nevada and Arizona dominating with a combined 83% of counts:

```r
## Remove non-essential categories AND filtering the 90th percentile
state_rest_count <- genre_count %>%
  group_by(state) %>%
  filter(categories != "Restaurants" || categories != "Nightlife" || categories != "Bars") %>%
  filter(n > quantile(n, 0.9))

## Visualise Share of Reviews by State (90% percentile)
state_table <- state_rest_count %>%
                select(state, n) %>%
                group_by(state) %>%
                summarise_all(funs("count" = sum(n))) %>%
                arrange(desc(count)) %>%
                mutate(proportion = round(count / sum(count), 2))

plot_state_table <- state_table %>%
    ggplot(aes(x = reorder(state, -proportion), y = proportion, fill = state)) +
    geom_bar(stat = "identity") +
    scale_y_continuous(labels = comma) + # Requires 'scales' package
    ggtitle("Share of Reviews by State (Top 10% only)") +
    labs(x = "State", y = "Share of Reviews") +
    theme(legend.position = "none", axis.text.x = element_text(face = "bold", size = 8, angle = 45), ax

plot_state_table
```



A final inspection reveals 20K to 30K reviews per state in Nevada and Arizona for Chinese. The same charts for the other states (not shown here) reveal much lower counts for Chinese and supports our decision to drop them from the analysis. Furthermore, the sample sizes for those states are too small to yield meaningful dish insights, let alone mentions of the dish names!

```
## Visualise cuisine review counts in Arizona and Nevada
library(scales) #add comma separators to Y-axis labels
plot_aznv_cuisine <- state_rest_count %>%
        filter(state == "AZ" | state == "NV") %>%
        ggplot(aes(x = reorder(categories, -n), y = n, fill = categories)) +
        geom_bar(stat = "identity") +
        facet_grid(state ~., scales = "free") +
        scale_y_continuous(labels = comma) +
        ggtitle("Cuisine Review Count (Top 10% only)") +
        labs(x = "Cuisine", y = "Total Reviews (n)") +
        theme(legend.position = "none", axis.text.x = element_text(face = "bold", size = 7, angle = 90)
        axis.text.y = element_text(face = "bold", size = 8))

plot_aznv_cuisine
```



Creating The Corpus

I will focus on **positive** reviews since our goal is recommending the best Chinese restaurants that serve a specific dish. To reduce computational time on my ancient Macbook, I sampled 30% of positive reviews:

```
## Filtering Chinese reviews from Nevada and Arizona
aznv_ch <- rev_biz_tidy %>%
    filter(state == "AZ" | state == "NV") %>%
    filter(str_detect(categories, "Chinese"))

# Filtering only positive reviews and converting to matrix
aznv_ch_text <- aznv_ch[aznv_ch$stars.x >= 4,]$text
aznv_ch_matrix <- as.matrix(aznv_ch_text)
# Randomised sampling
random.rows <- sample(1:nrow(aznv_ch_matrix), 0.3 * nrow(aznv_ch_matrix), replace = FALSE)
aznv_ch_sample <- aznv_ch_matrix[random.rows,]
```

Next we'll create a corpus from the matrix of reviews, essentially a structured set of texts that will be the input for the topic mining and sentiment analysis. As shown, the corpus of sampled reviews contains 9,400 documents. Each document here corresponds to a single review.

```
## Creating the corpus
aznv_ch_corpus <- VCorpus(VectorSource(aznv_ch_sample))
aznv_ch_corpus
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 9400
```

Extracting Names of Popular Dishes

Preprocessing the Review Texts

Human-written sentences contain punctuation, abbreviations, numbers and English stopwords such as "my", "your", "to" and "from". Also, people tend to use abbreviations liberally and get sloppy with capitalisation; we need the word counting algorithm to know that "shouldn't" and "Chicken" are identical to "should not" and "chicken" - essentially reducing duplicated words.

The end goal here is to count only the words that matter - dish names and sentiment words. The cleaning function below removes the noise from the corpus by deduplicating words and removing stopwords:

```
clean_corpus <- function(corpus){
  corpus <- tm_map(corpus, content_transformer(function(x) iconv(x,to='UTF-8-MAC', sub='byte')),  mc.co
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, content_transformer(replace_abbreviation))
  corpus <- tm_map(corpus, removeNumbers)
  corpus <- tm_map(corpus, removeWords, c(stopwords("en"), "food"))
  corpus <- tm_map(corpus, stripWhitespace)
  return(corpus)
}
```

Counting N-Grams

After cleaning the corpus, we'll list the 20 most frequently occurring single words or 'unigrams' in the reviews as a first attempt at extracting popular dish names. A term document matrix is a table with unigrams as rows and review number as columns.

The unigram counts per document (or review) will thus be the values in the table. Summing the rows gives us the total count of every unigram in the corpus:

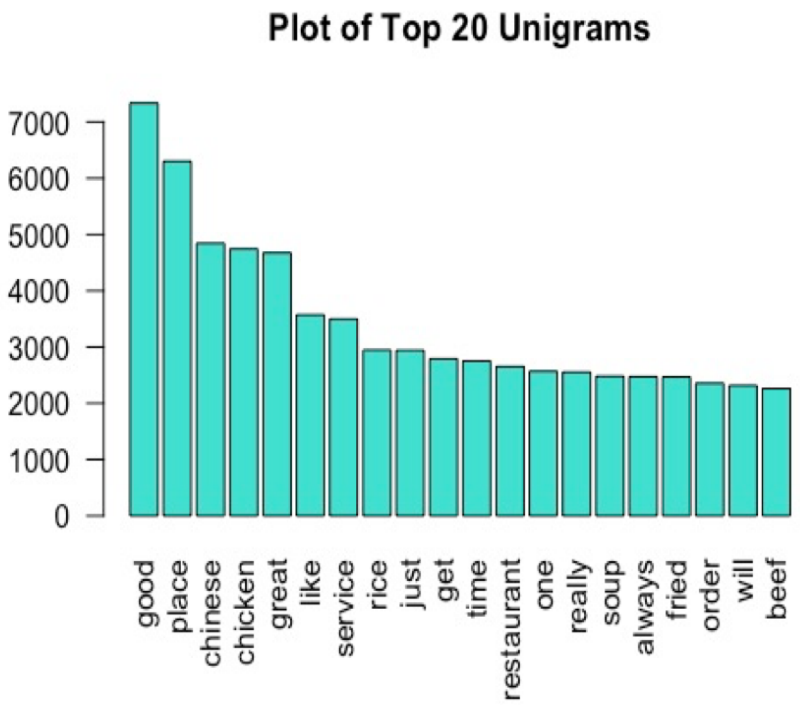
```
unigram_count <- function(cleaned_corpus) {
                tdm <- TermDocumentMatrix(cleaned_corpus)
                tdm_matrix <- as.matrix(tdm)
                term_freq <- rowSums(tdm_matrix)
                term_freq <- sort(term_freq, decreasing = TRUE)
                barplot(term_freq[1:20], col = "red", las = 2, main = "Plot of Top 20 Unigrams")
}
```

```
clean_aznv_ch <- clean_corpus(aznv_ch_corpus)
unigram_count(clean_aznv_ch)
```

Two things are clear on closer inspection - we need to get better at removing stopwords AND we'll need longer phrases to get any meaningful dish insights. From personal experience, Chinese dish names tend to be three words or longer i.e. 'wanton noodle soup', 'black pepper beef' so we should be on the right track.

Expanding the stopword list in the corpus cleaner to capture superlatives and unnecessary nouns:

```
clean_corpus2 <- function(corpus){
  corpus <- tm_map(corpus, content_transformer(function(x) iconv(x, to = "UTF-8-MAC", sub = "byte")), m
  corpus <- tm_map(corpus, removePunctuation)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, content_transformer(replace_abbreviation))
  corpus <- tm_map(corpus, removeNumbers)
```



Plot of Top 20 Unigrams
7000
6000
5000
4000
3000
2000
1000
0
good
place
chinese
chicken
great
like
service
rice
just
get
time
restaurant
one
really
soup
always
fried
order
will
beef

Figure 1:

```
  corpus <- tm_map(corpus, removeWords, c(stopwords("en"), "food", "good", "place", "great", "service",
  corpus <- tm_map(corpus, stripWhitespace)
  return(corpus)
}
## Cleaning the corpus with the new and improved function
clean_aznv_ch2 <- clean_corpus2(aznv_ch_corpus)
```

The next step of preprocessing is to tokenize the cleaned corpus. What we are doing here is breaking up the text review strings into groups of keywords, or phrases, called 'tokens'. As mentioned earlier, the unigrams do not contain meaningful information about the Chinese dishes and we are now listing the most frequently occurring phrases containing 2 words ('Bigrams') or 3 words ('Trigrams').

```
## Create term document matrix for two-word (bigram) and three-word (trigram) phrases
install.packages("SnowballC")
library(SnowballC) #required by latest version of 'tm' package
library(tm)
update.packages("tm",  checkBuilt = TRUE) #updating 'tm' package

# Create bigram and trigram tokenizer functions
BigramTokenizer <- function(x) unlist(lapply(ngrams(words(x), 2), paste, "", collapse = " "), use.names
TrigramTokenizer <- function(x) unlist(lapply(ngrams(words(x), 3), paste, "", collapse = " "), use.name

## Create term document matrix (tdm) of bigrams and trigrams
bigram_ch_tdm <- TermDocumentMatrix(clean_aznv_ch2, control = list(tokenize = BigramTokenizer))
trigram_ch_tdm <- TermDocumentMatrix(clean_aznv_ch2, control = list(tokenize = TrigramTokenizer))

## Convert tdm into data frames of bigram and trigram counts
ngram_freq <- function(tdm){
                freq <- sort(rowSums(as.matrix(tdm)), decreasing=TRUE)
                freq_df <- data.frame(word=names(freq), freq=freq)
                return(freq_df)
}

bigram_ch_freq <- ngram_freq(bigram_ch_tdm)
trigram_ch_freq <- ngram_freq(trigram_ch_tdm)
```

Plotting frequencies of the Bigrams and Trigams reveals the information gained as the phrase length increases:

```
plot_bigram_ch <- bigram_ch_freq[1:20,] %>%
                ggplot(aes(x = reorder(word, -freq), y = freq)) +
                geom_bar(stat = "identity", fill = "green", col = "red") +
                scale_y_continuous(labels = comma) +
                ggtitle("Histogram of 20 Most Frequent Bigrams") +
                labs(x = "Words / Phrases", y = "Frequency") +
                theme(legend.position = "none", axis.text.x = element_text(face = "bold", size = 9, ang

plot_bigram_ch
```

```
plot_trigram_ch <- trigram_ch_freq[1:20,] %>%
                    ggplot(aes(x = reorder(word, -freq), y = freq)) +
                    geom_bar(stat = "identity", fill = "green", col = "red") +
                    scale_y_continuous(labels = comma) +
                    ggtitle("Histogram of 20 Most Frequent Trigrams") +
                    labs(x = "Words / Phrases", y = "Frequency") +
                    theme(legend.position = "none", axis.text.x = element_text(face = "bold", size = 8,
```

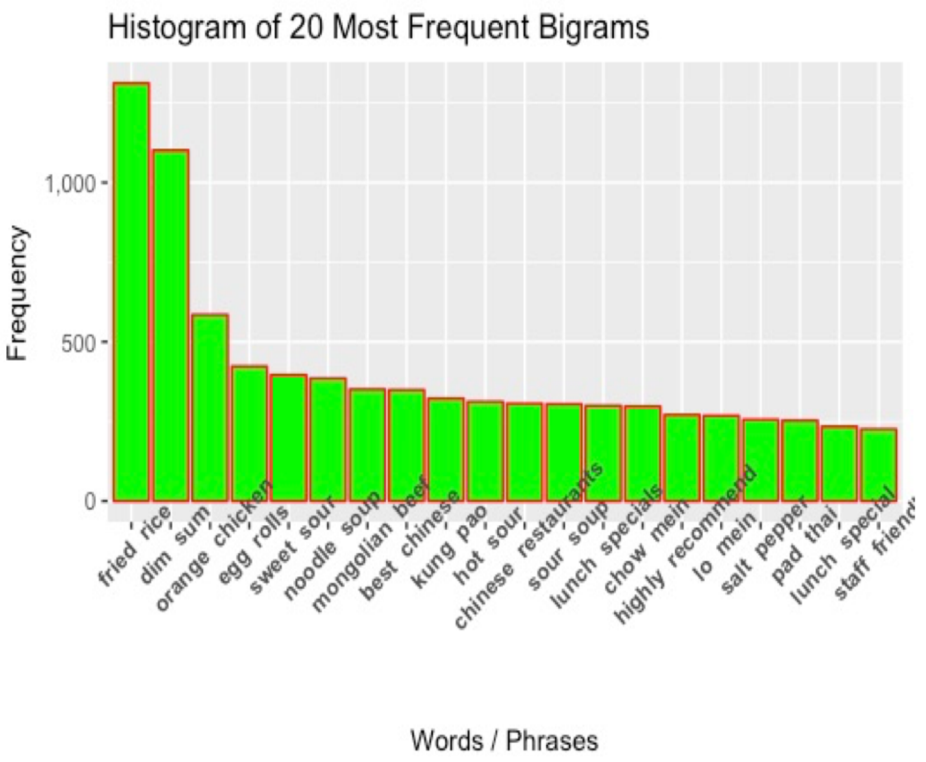

Histogram of 20 Most Frequent Bigrams
Frequency
1,000
500
0
fried rice
dim sum
orange chicken
egg rolls
sweet sour
noodle soup
mongolian beef
best chinese
kung pao
hot sour
chinese restaurants
sour soup
lunch specials
chow mein
highly recommend
lo mein
salt pepper
pad thai
lunch special
staff friend
Words / Phrases

Figure 2:

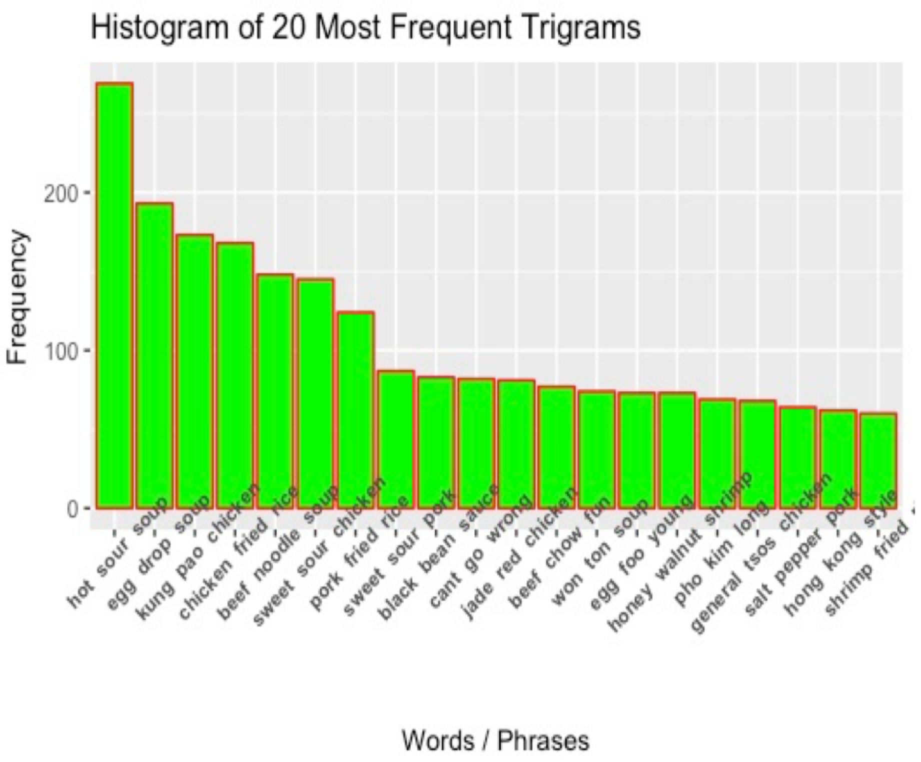


Figure 3:

```
plot_trigram_ch
```

Initial Findings

The perennial favorite, hot & sour soup, topped the list followed by egg drop soup, kung pao chicken, fried rice, and beef noodle soup. Bearing in mind that these reviews originated from Nevada and Arizona, Part 2 allows users to query their favorite dishes by state.

Deploying the Mini Search Engine

The Shiny App allows users to select state and dish from the dropdown menus. The app searches for exact matches with positive reviews and finally returns the names of the top 5 corresponding restaurants with the most reviews.

In summary, the user is now able to query a restaurant serving his or her favourite dish!

Results indicate that **P. F. Chang** is a popular chain for many well-known Chinese dishes in Nevada and Arizona.

Click on this LINK to try out the Shiny App

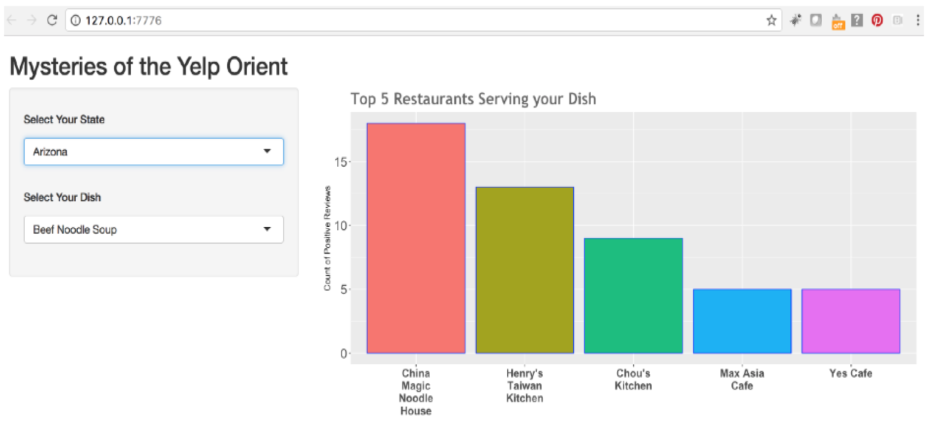


Figure 4:

Limitations

Due to computational resource limitations, I took a sample of the reviews and focused on Nevada and Arizona. In a perfect world where I had more time and every data mining trick in the book, I would have calculated the sentiment for the dish by examining the words immediately before and after instead of looking at the overall rating.

In another words, the customer might have enjoyed the meal at P.F. Chang's but hated the Sweet & Sour Soup. My framework does not capture that nuance.

Lastly, I would have liked to have data around user reliability and engagement which would allow me to filter reviews by quality and relevance.

Key Stakeholders

The Yelp Search team implements the feature into their search algorithm, and the UX team adds a Google Map window to provide directions to the nearest restaurant and creates a 'Make Reservation' button to accompany each result.

The Ads Product team could allow advertisers to bid for dish names within users' search queries; doing so will improve the precision of their ad targeting and drive more high-intent traffic to their advertisers' pages.

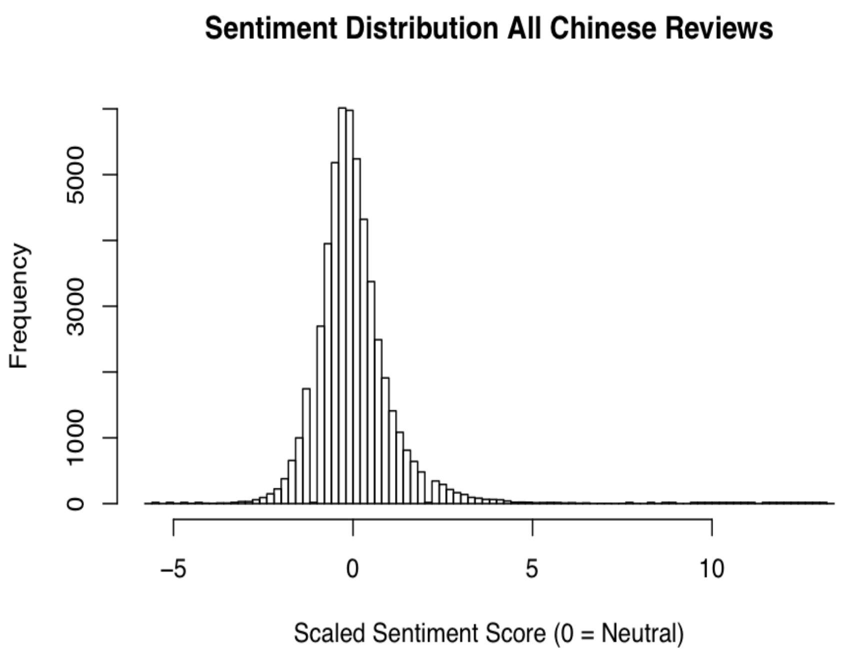
One More Thing . . .

The Yelp dataset challange has produced excellent white papers on predicting votes and ratings from sentiment and restaurant attributes. This is a Machine Learning project which is next on my to-do list; I'll leave you with these charts as a preview.

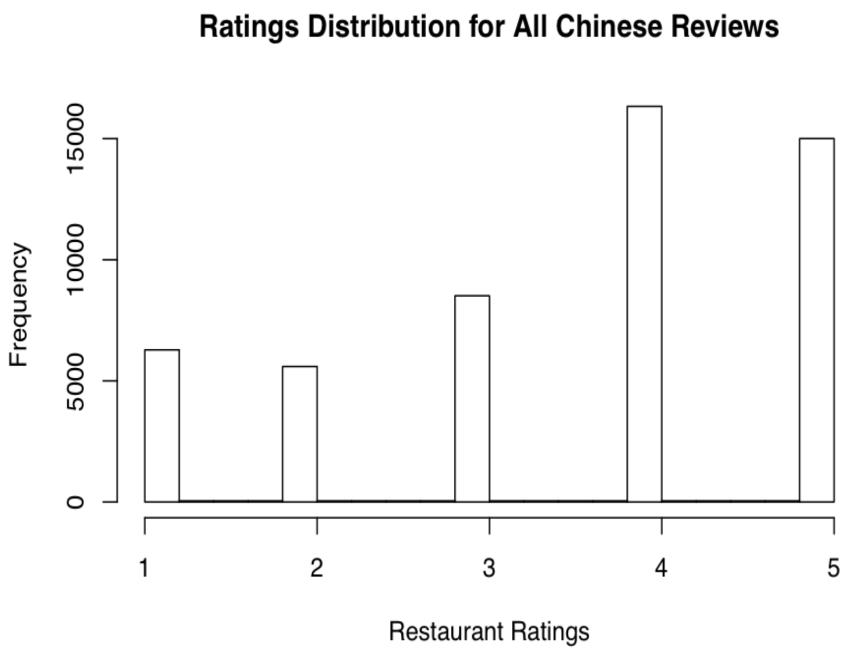
[TL;DR] Review sentiment is likely not a good predictor of ratings. The first 2 charts show that ratings tend to skew toward the positive while review sentiment is normally distributed:

```
chreviews <- aznv_ch$text

## Sentiment lexicon from Bing Liu (cs.uic.edu/~liub/FBS/sentiment-analysis.html)
pos_words <- scan("positive-words.txt", what='character', comment.char=';')
```

```r
neg_words <- scan("negative-words.txt", what='character', comment.char=';')

## Sentiment analysis function
score.sentiment <- function(text_vector, pos.words, neg.words, .progress='none')
{
  require(plyr)
  require(stringr)

  scores <- ldply(text_vector, function(text_vector, pos.words, neg.words) {

    # clean up text with regex
    text_vector <- gsub('[[:punct:]]', '', text_vector)
    text_vector <- gsub('[[:cntrl:]]', '', text_vector)
    text_vector <- gsub('\\d+', '', text_vector)
    text_vector <- tolower(text_vector)

    # split into words using 'stringr' package
    word.list <- str_split(text_vector, '\\s+')
    words <- unlist(word.list)

    # compare our words to the dictionaries of positive & negative terms
    pos.matches <- match(words, pos.words)
    neg.matches <- match(words, neg.words)

    # match() returns the position of the matched term or NA
    # we just want a TRUE/FALSE:
    pos_matches <- !is.na(pos.matches)
    neg_matches <- !is.na(neg.matches)
    score <- sum(pos_matches) - sum(neg_matches)

  }, pos.words, neg.words, .progress=.progress)

  scores_final <- data.frame(sentiment_score = scores, text = text_vector)
  return(scores_final)
}

## Score sentiment for all Chinese restaurants reviews from Nevada and Arizona
ch_sentiment <- score.sentiment(chreviews, pos_words, neg_words)
ch_starsentiment <- cbind(aznv_ch[, c(4,6)], ch_sentiment[,1])
colnames(ch_starsentiment) <- c("stars", "text", "sentiment_score")

## Compare distribution of ratings and review sentiment; ratings are dominated by 4 and 5 stars whereas
hist(scale(ch_starsentiment$sentiment_score), breaks = 100, main = "Sentiment Distribution All Chinese
```

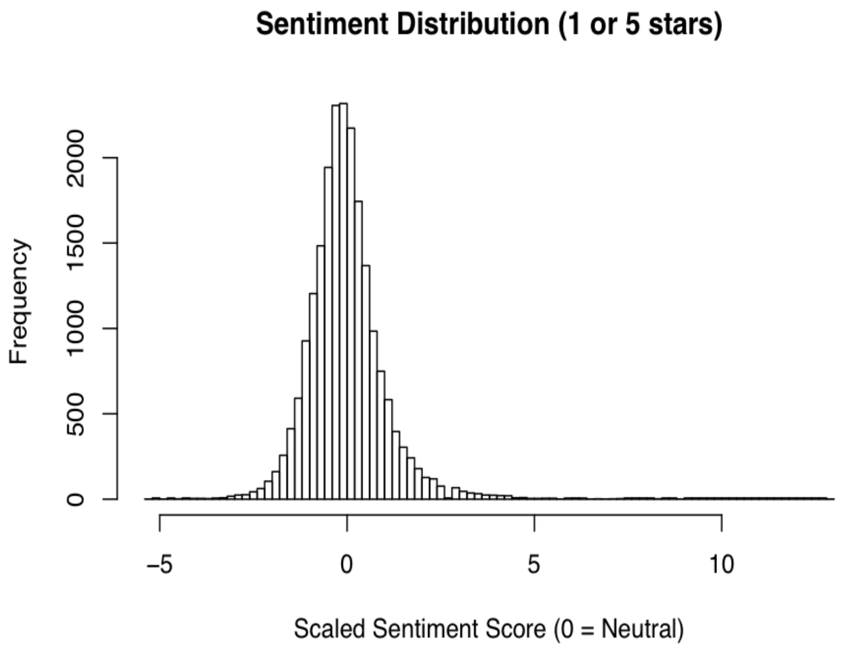


```
hist(aznv_ch$stars.x, main = "Ratings Distribution for All Chinese Reviews", xlab = "Restaurant Ratings'
```



The normality of review sentiment persists for 1 and 5 star ratings - this implies low correlation between sentiment and ratings even at extreme values:

```
# Normality persists for 1 and 5 star ratings - sentiment are not predictive of ratings
extreme_stars <- filter(ch_starsentiment, stars == 1 | stars == 5)
hist(scale(extreme_stars$sentiment_score), breaks = 100, main = "Sentiment Distribution (1 or 5 stars)"
```



Please send your feedback to eugene.hw.woo@gmail.com Full code available on Github