Text Mining: Sentiment Analysis

Code

I Introduction

The Yelp Dataset provided on kaggle is a collection of more than 5 million customer reviews of 174,000 businesses across four countries. We chose to analyse the data with regards to Seboca's upcoming projects in the food sector. The created case study will be used for sales purposes.

To begin with we are starting to read in the data sets (~3GB). If you are interested to run the analysis just change the working directory in the folloing code. Note that this can take a while due to the large dataset.

Goal & Approach

The goal of the analysis is to look at reveiws for some of the restaurants and to deduct insights for the businesses. To do so we need to get insight into work frequencies, combinations and an indication on their sentiment. At the moment there are three common general-purpose lexicons: AFINN, Bing and NRC. Each lexicon classifies words slightly differently. AFINN: Gives each word a score where positive scores indicate positive sentiment and negative scores, negative sentiment. Bing: Classification into positive and negative NRC: Assigns sentiments to words such as negative, sadness, trust etc.

Limitations

The chosen lexicon doesn't cover every single English word because a lot of them are neutral words. They would not pick up if word combinations change the sentiment of the phrase.

II Analysis

Data exploration

To start, we want to know which language we are dealing with as our lexicon will check for english words. We randomly extract 100 entries and let the textcat function pick up the laguage of the reviews.

Hide

```
#library(textcat)
review_row_number = nrow(review)
random = sample(x = 1:review_row_number, size = 100)
textcat(review[random,]$text)
```

The majority of reviews is in English, however some are classified "scots". This should not interfer with our model.

Next, let's have a brief look at the data types and structures. The data is a combination of factors and integers. For the following analysis we need the reviews in a character format which is why we specify "stringsAsFactors = FALSE" when reading in the review table.

I like to create a libraries chunk at the top of the code so I can quikcly copy and paste it into a new project.

Poweruser analysis

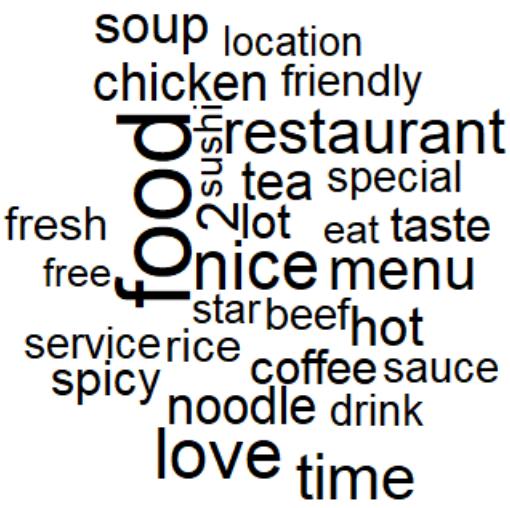
Let's have a look at the most active user. We count the unique userIDs and return the 6 most reviewing users and then filter to find the one with the largest count.

The user with ID "CxDOIDnH8gp9KXzpBHJYXw" made 3569 contributions. Let's get a feel for the content of his or her reviews.

Hide

```
createWordCloud = function(data)
{
  data %>%
  unnest_tokens(word, text) %>%
  filter(!word %in% stop_words$word) %>%
  count(word,sort = TRUE) %>%
  ungroup()  %>%
  head(30) %>%

  with(wordcloud(word, n, max.words = 30))
}
createWordCloud(review %>%
  filter(user_id == "CxDOIDnH8gp9KXzpBHJYXw"))
```



Using the wordcloud, we instantly get a feel for the sentiment of the user's reviews. The wordcloud function counts the frequency of each work in the reviews and prints the words such that the word with the biggest font has the highest count. The first insights we gain from this basic analysis is the poweruser is writing a lot about the quality of food and drinks in the places he went to. While other users might go more into detail about the waiters and interior. We can suspect that the sentiment is overall positive as the words "love", "nice" and "friendly" pop up.

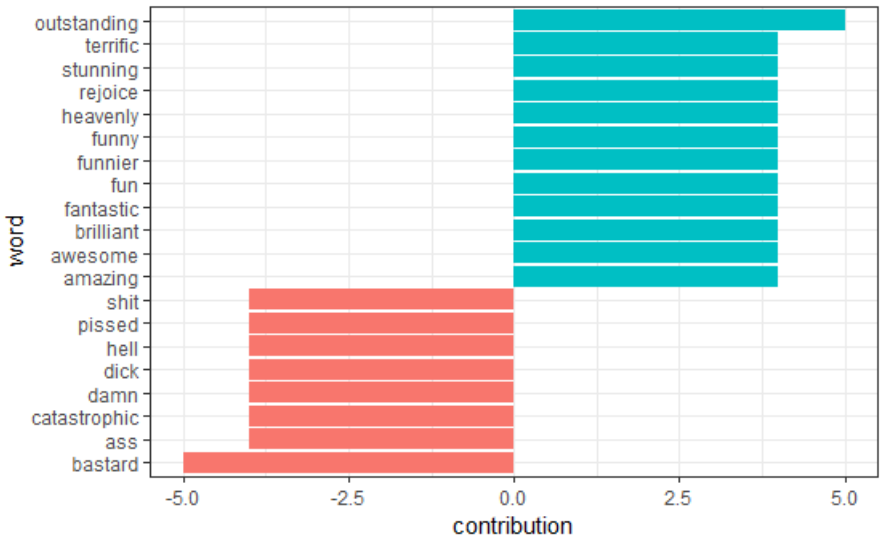
Let's use the AFINN lexicon to find our more about the sentiment scores.

Hide

```
SentimentScores <- function(data, count) {
  contributions <- data %>%
    unnest_tokens(word, text) %>%
    count(word,sort = TRUE) %>%
    ungroup() %>%

    inner_join(get_sentiments("afinn"), by = "word") %>%
    group_by(word) %>%
    summarize(occurences = n(),
              contribution = sum(score))

  contributions %>%
    top_n(count, abs(contribution)) %>%
    mutate(word = reorder(word, contribution)) %>%
    head(count) %>%
    ggplot(aes(word, contribution, fill = contribution > 0)) +
    geom_col(show.legend = FALSE) +
    coord_flip() + theme_bw()
}
SentimentScores(review %>%
                        filter(user_id == "CxDOIDnH8gp9KXzpBHJYXw"), 20)
```



The visulalization shows us 20 words with particularly high and low sentiment scores according to the AFINN lexicon. Not that there are

more positively outstanding words than negative ones. When we increase the number of words to plot we see that the result get's less accurate and the words with letters a,b,c are prioritized.

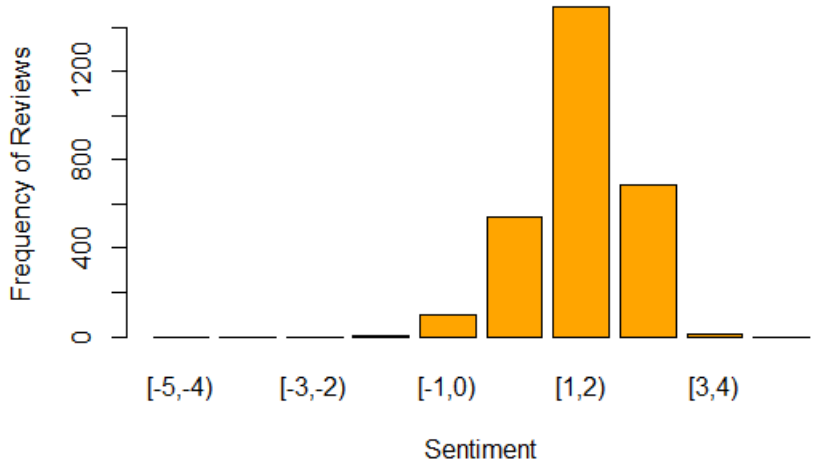
Now we can have a look at the sentiment scores of each review the poweruser made. Therefore we simple draw the average of sentiment scores of every review in order to create a histogram.

Hide

```
calculate_sentiment <- function(review_text) {
  sentiment_lines  =  review_text %>%
                  filter(textcat(text) == "english") %>%  # excluding "scots"" and other languages
                  unnest_tokens(word, text) %>%
                  inner_join(get_sentiments("afinn"), by = "word") %>%
                  group_by(review_id) %>%
                  summarize(sentiment = mean(score),words = n()) %>%
                  ungroup() %>%
                  filter(words >= 5)
  return(sentiment_lines)
}
sentiment_lines = calculate_sentiment(review %>%
                        filter(user_id == "CxDOIDnH8gp9KXzpBHJYXw"))
head(sentiment_lines)
```

Hide

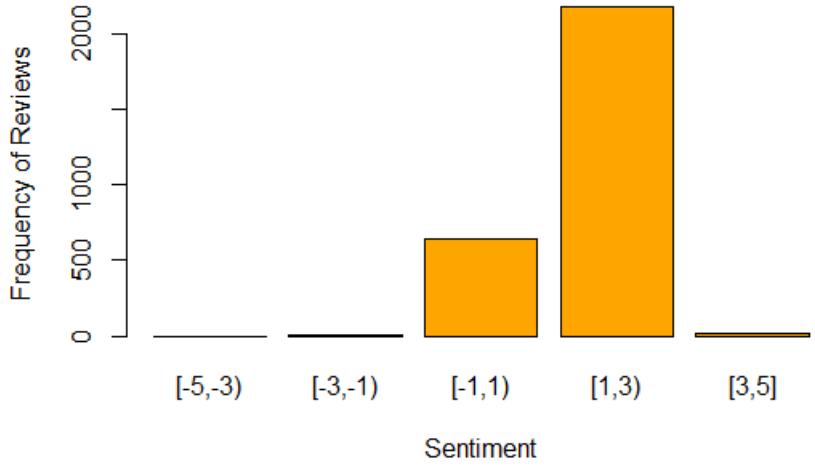
```
# plot frequency of sentiment scores
breaks <- c(-5,-4,-3,-2,-1,0,1,2,3,4,5)
bins <- cut(sentiment_lines$sentiment, breaks, include.lowest = T, right=FALSE)
# summary(bins)
plot(bins, xlab="Sentiment", ylab="Frequency of Reviews", col="orange")
```



The histogram allows us to see the sentiment our algorithm is picking up in each review. Now we are wondering if the sentiment score matches thestars the user has to submit. The sentiment score is rated from -5 to 5 but there are only 5 stars to allocate. We can group the sentiment scores to compare the given stars with the evaluated sentiment.

Hide

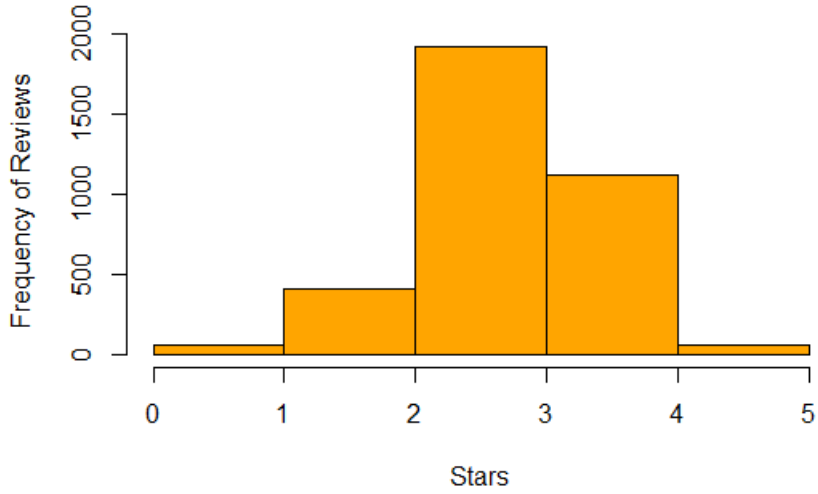
```
breaks <- c(-5,-3,-1,1,3,5)
bins <- cut(sentiment_lines$sentiment, breaks, include.lowest = T, right=FALSE)
# summary(bins)
plot(bins, xlab="Sentiment", ylab="Frequency of Reviews", col="orange")
```



According to our grouping the poweruser would have mainly allocated 3 and 4 stars to his or her reviews. Now we can compare how many stars were actually given.

Hide

```
power_reviews <- filter(review, user_id == poweruser$user_id)
# table(power_reviews$stars)
hist(power_reviews$stars, breaks = c(0,1,2,3,4,5), col="orange", main = "", xlab = "Stars", ylab = "Frequen
cy of Reviews")
```

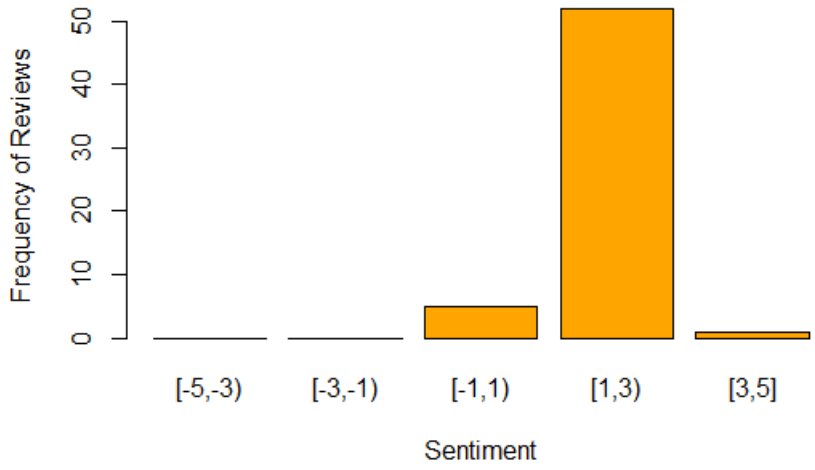


The above histogram shows us that the user did mainly allocate 3 and 4 star reviews but indeed more 3 stars than 4 which differs from the sentiment we analyzed. Reasons for the discrepancy could be that many used words are not scored in the AFINN lexicon or that the star scale does not reflect the user sentiment to the full extent. For instance although my review sounds like I had a good time at the restaurant, I might still think it is only worth 3 stars instead of four. This could be an effect of our particular poweruser, who has a lot to compare with, writing a good review but then comparing if it is actually as good as a place he rated 4 or 5 stars before.

Indeed, the review with the best sentiment index was only rated 3 out of 5 stars. But which reviews are now the ones the poweruser actually rated 5 stars?

Hide

```
beststars = filter(power_reviews, stars == 5)
# beststars$text
sentiment_lines_five = calculate_sentiment(power_reviews %>%
                        filter(stars == 5))
# head(sentiment_lines_five)
# plot frequency of sentiment scores that were rated 5 stars
breaks <- c(-5,-3,-1,1,3,5)
bins <- cut(sentiment_lines_five$sentiment, breaks, include.lowest = T, right=FALSE)
# summary(bins)
plot(bins, xlab="Sentiment", ylab="Frequency of Reviews", col="orange")
```



5 star ratings were given to reviews with mainly reviews with a sentimentscore of 1-3.

Poweruser Analysis Wrapup

We learned that the sentiment score and the star rating do not necessarily correlate.

The more insightful aspects of this analysis is to learn about what the user mentiones most. According to the wordcloud, the poweruser talks most about food (chicken, steak, etc), flavors (spicy, taste, hot) and positive words like nice, friendly and love. This gives us an indication that the user writes more positive then negative reviews. For some users we could use this analysis together with the star ratings to detect if there are users that only write positive (or negative) reviews. There is a chance that some users are getting paid to write reviews and we could pick up patterns accordingly.

The wordcloud and word contribution to the sentiment scores could also be used to create banners that speak to the user, using their language. We've learned through Donald Trump that speaking the same language as the receiver of a canpaign is extremely powerful in marketing.

Business Analysis

Data about individual users is most interesting for the platform yelp itself. Businesses know how reviews on popular platforms can impact their revenue. Therefore they are extremely eager to learn about their reviews and what makes cusotmer write positive reviews.

Let us have a look which businesses have the best average star ratings as the star rating is the first indication for users to judge the quality of the restaurant. Afterwards we want to explore the topics users talk about in the reviews of those businesses.

As a starting point we take out businesses with less than 10 reviews. Let's be honest everybody finds 10 friends to write positive a nice review when starting their restaurant.

Hide

```
business_reviewcount = count(review, business_id)
business_reviewcount_filter <- filter(business_reviewcount, n > 9)
ten_reviews_up <- review[review$business_id %in% business_reviewcount_filter$business_id,]
# head(ten_reviews_up)
```

We are left with more than 80,000 businesses and 4.8 million reviews. Now we will have a look at their average star ratings, rank them and select the top 10,000 restaurants. At the end we have to filter all reviews by the top 10,000 business_id's.

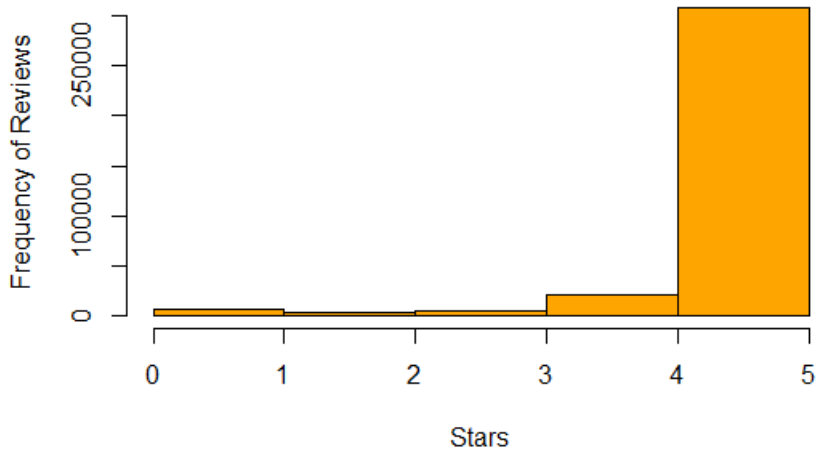
Hide

```
#we aggregate the unique business_ids and calculate the average rating per business
av_stars <- aggregate(stars ~ business_id, ten_reviews_up, mean)
#head(av_stars)
#we select the top 10000 restaurants with the best average star rating
av_stars <- av_stars[with(av_stars, order(-stars)),]
av_stars <- av_stars[1:10000,]
#check if we also have some businesses with an average rating below 5.0
#tail(av_stars)
#filter all reviews by the businesses with the best ratings
best_av_star_reviews <- review[review$business_id %in% av_stars$business_id,]
#nrow(best_av_star_reviews)
#[1] 343132
#head(best_av_star_reviews)
```

As a result we are left with more than 340,000 reviews that were given to the restaurants that have the best average ratings. To get more familiar with the distribution of the stars and numbers of reviews, we do a bit more exploration.

Hide

```
hist(best_av_star_reviews$stars, breaks = c(0,1,2,3,4,5), col="orange", main = "", xlab = "Stars", ylab = "
Frequency of Reviews")
```

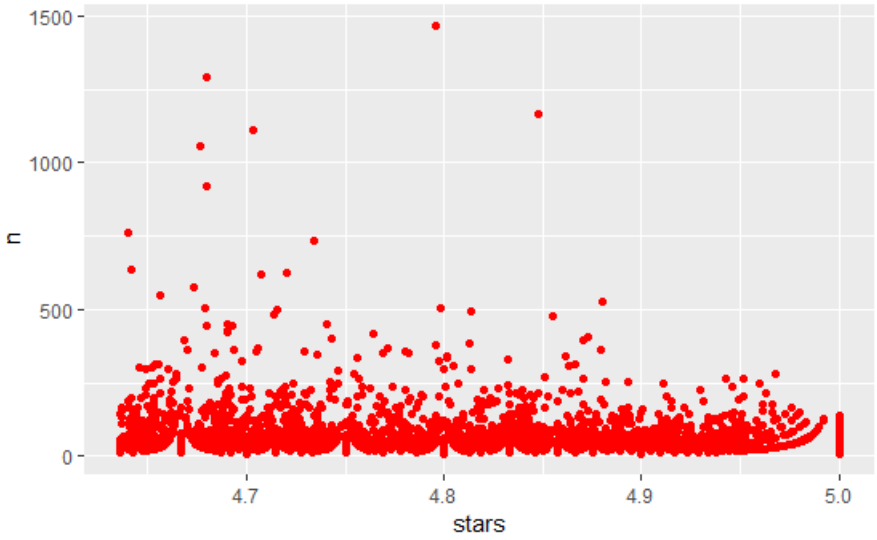


As expected most of the reviews are 4 and 5 stars. But we also have some 1,2 and 3 star reviews.

Now let's have a look at how the number of reviews per restaurant is linked to the number of stars given.

Hide

```
best_av_star_reviewcount = count(best_av_star_reviews, business_id)
#nrow(best_av_star_reviewcount)
#nrow(av_stars)
business_star_review = merge(best_av_star_reviewcount, av_stars)
#nrow(business_star_review)
#head(business_star_review)
ggplot(business_star_review, aes(x=stars, y=n)) + geom_point(col="red")
```



In the above scatterplot each red dot represents one of the 10,000 selected businesses and they are mapped according to their number of reviews as well as their average star rating. We can spot a few outliers with more than 500 reviews but most are between 10 (which we stated as the minimum above) and 500 reviews. The average ratings vary from just over 4.6 to 5.0. We did consider to take out the 5.0 ratings but to get a clearer result we leave them in for this analysis.

Next we want to learn about topics and the content of those top reviewing restaurants so we know what distinguishes them and can advise other lower rated businesses on what they might want to improve.

As it is visually attractive and easy to understand we first have a look at the wordcloud.

Hide

```
#we are re-using the word cloud function defined above
createWordCloud(best_av_star_reviews[1:10000,])
```

The 5 largest words and therefore the most frequently used ones besides the excluded stopwords, are “time”, “service”, “recommend”, “friendly” and “amazing”. What can businesses learn from this word cloud? 1. Service is a huge topic. People are using words like friendly, professional, nice, staff, customer, home and experience. Creating a friendly environment with friendly staff members and a homely atmosphere generated great reviews. 2. Think beyond the customer. Poeple “recommend” places they love to their peers. Getting a review which includes something like “I’ll definitely recommend this place!” is very powerful. 3. Other potential insights: - Price: the word price shows up in our wordcloud and we can expect that the value for money relation is mentioned in top reviews. - Time: Time on its own is a bit tricky to analyze. It can be short waiting times, having a great time and else. We hope to learn more about the word in relation with other words. - Clean: Another word that people should keep in mind. Top rated reviews often include the word clean and as a restaurant owner it should be part of the top priorities.

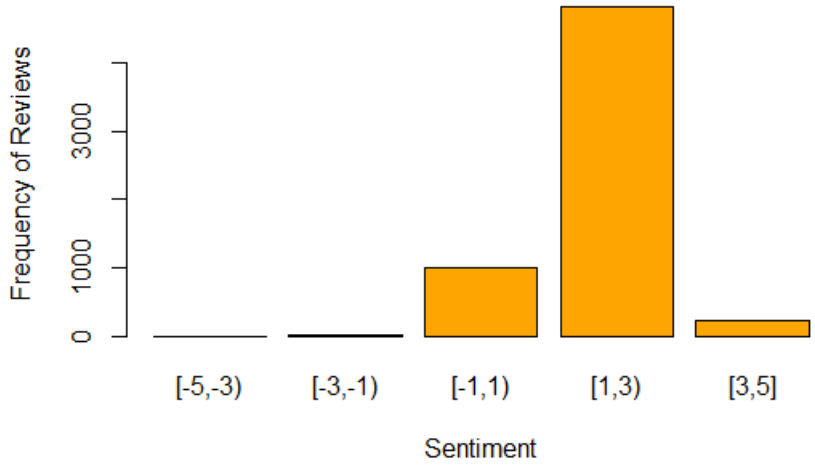
Some of the words in the cloud are not adding great value, for instance “called”, “found”, “dr”, “shop” etc. A way to overcome this would have been to filter the data right in the beginning and exclude all categories except for restaurants such as Art & Entertainment or Health & Medical. If we compare the top rated reviews with the power user we see that people talk much more about service than about food and taste.

Now we have a look at the sentiment lines again to see whether the star ratings are accurate or whether we would have to adjust our sentiment scale.

Limitation: From now on we use the first 10,000 reviews only to reduce processing time.

Hide

```
# we reinitiate the function to be able to make changes if needed
calculate_sentiment1 <- function(review_text) {
  sentiment_lines_best  =  review_text %>%
                unnest_tokens(word, text) %>%
                inner_join(get_sentiments("afinn"), by = "word") %>%
                group_by(review_id) %>%
                summarize(sentiment = mean(score),words = n()) %>%
                ungroup() %>%
                filter(words >= 5)
  return(sentiment_lines_best)
}
# the analysis is requiring more storage space than available on my computer, therefore we are only using th
e first 10000 rows for now.
sentiment_lines_best = calculate_sentiment1(best_av_star_reviews[1:10000,])
#breaks <- c(-5,-4,-3,-2,-1,0,1,2,3,4,5)
breaks <- c(-5,-3,-1,1,3,5)
bins <- cut(sentiment_lines_best$sentiment, breaks, include.lowest = T, right=FALSE)
#summary(bins)
plot(bins, xlab="Sentiment", ylab="Frequency of Reviews", col="orange")
```



We see that it would be more accurate to set a sentiment of 1-4 equal to a star rating of 5. If businesses are analyzing review data from other sources they need to make sure they understand the discrepancy between the sentiment and the star rating to make it comparable.

Next, we want to look into words that are used together frequently. Above we found some frequently used words such as “price” which on their own do not tell us much about their sentiment. We evaluate bigrams (pairs of consecutive written words).

Hide

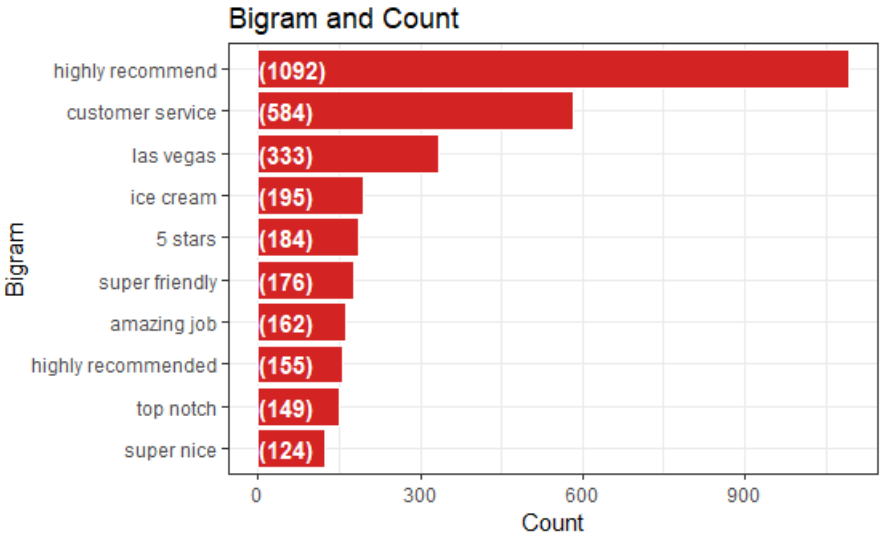
```
count_bigrams <- function(review_text) {
  review_text %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
    filter(!word1 %in% stop_words$word,
           !word2 %in% stop_words$word) %>%
    count(word1, word2, sort = TRUE)
}
best_av_star_reviews[1:10000,] %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  select(bigram,review_id) %>%
  head(10)
```

Looking at the first bigrams we see that we have to get rid of top words such as “is”, “i”, “my” etc. to gain insights. Our second attempt show us more insightful results.

Hide

```
best_av_star_reviews[1:10000,] %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2) %>%
  separate(bigram, c("word1", "word2"), sep = " ") %>%
  filter(!word1 %in% stop_words$word,
         !word2 %in% stop_words$word) %>%
  unite(bigramWord, word1, word2, sep = " ") %>%
  group_by(bigramWord) %>%
  tally() %>%
  ungroup() %>%
  arrange(desc(n)) %>%
  mutate(bigramWord = reorder(bigramWord,n)) %>%
  head(10) %>%

  ggplot(aes(x = bigramWord,y = n)) +
  geom_bar(stat='identity',colour="white", fill = "#D32323") +
  geom_text(aes(x = bigramWord, y = 1, label = paste0("(",n,")",sep="")),
            hjust=0, vjust=.5, size = 4, colour = 'white',
            fontface = 'bold') +
  labs(x = 'Bigram',
       y = 'Count',
       title = 'Bigram and Count') +
  coord_flip() +
  theme_bw()
```

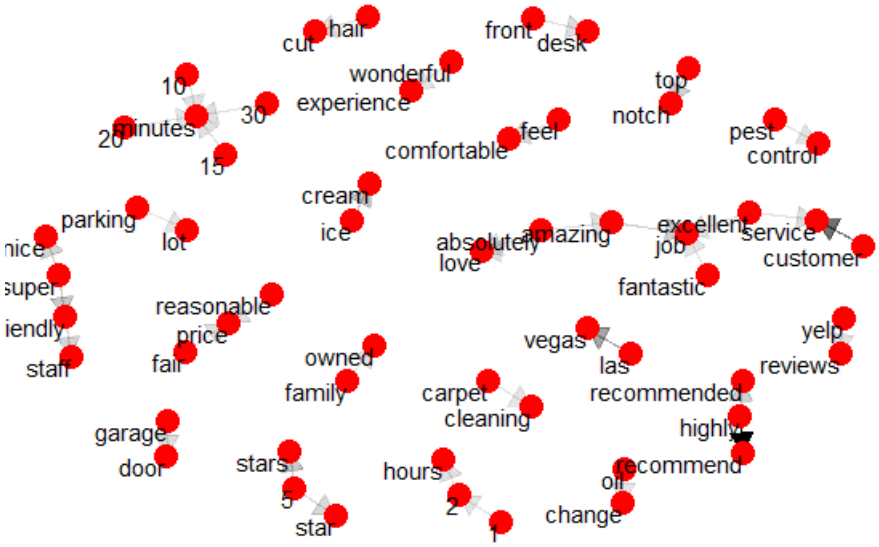


The Bigram and Count diagram shows us words frequently used together such as highly recommended, customer service and 5 stars. Again we learn more about the sentiment of the top rated locations. Nevertheless we are still lacking more insights into e.g. price. Let us take it a step further and visualize bigrams in a graph that indicates relationships between words.

Hide

```
best_av_star_bigrams <- best_av_star_reviews[1:10000,] %>%
  count_bigrams()
visualize_bigrams <- function(bigrams) {
  set.seed(123)
  a <- grid::arrow(type = "closed", length = unit(.15, "inches"))

  bigrams %>%
    graph_from_data_frame() %>%
    ggraph(layout = "fr") +
    geom_edge_link(aes(edge_alpha = n), show.legend = FALSE, arrow = a) +
    geom_node_point(color = "red", size = 5) +
    geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
    theme_void()
}
best_av_star_bigrams %>%
  filter(n > 50) %>%
  visualize_bigrams()
```



There we go! The visualization shows us that price is mostly associated with reasonable and fair in the top reviews. Moreover people talk about (probably waiting) times, wonderful customer experience and excellent customer service. The different nodes in the word trees are connected with arrows so we can also read which way around words are combined. On the left hand side the words “nice”, “super”, “friendly” and stuff are combined in two ways: either “super nice” or “super friendly staff”. Another expression that sticks out is “family owned” and knowing that we are only looking at the top rated places this is an indication that family owned places often do well.

III Conclusion

Sentiment Analysis is a great technique to quickly gain insights into a large set of data. Instead of reading millions of reviews, we can already get a good feel for the content by analyzing it with techniques such as wordclouds, sentiment scores and bigram counts. The yelp data set is a collection of much more data than we have actually considered in this example but even with this limited amount of data we can generate business relevant insights. Some of the key findings are that the sentiment scores are not in line with the star ratings users assign to rate restaurants. The wordcloud we created for the reviews of the poweruser told us instantly that he or she is writing about food quality whereas the best average star rated reviews are talking more about experience and customer service. Beyond the word cloud, bigrams are extremely helpful because they put neutral words such as “price” in relation which in this case is needed to understand whether is has a positive or negative sentiment. The word tree graph shows that price is used with words like “reasonable” and “fair” and is thereby positive.

Resources

“Text Mining with R” - Julia Silge and David Robinson

https://www.kaggle.com/yelp-dataset/yelp-dataset (data)

Stack Overflow