Chipotle Restaurant Review Mining

By: Syed Multani, Alvin Lee, Shamit Singh, Andrew Dugal

**Abstract:** The goal of this project is to build an app that mines customer reviews for Chipotle using a Yelp review dataset. It will provide a sentiment analysis on generated reviews and provide a targeted response based on whether the review is positive or negative. This app could be utilized by restaurants review and interpret business reviews, and will have potential for future development in terms of accurately responding to reviews and customer comments.

**Introduction and Discussion:**

Businesses gather data and feedback from many sources. From social media- facebook, Instagram, twitter, to review sites like google and yelp, along with probably thousands of other websites where consumers provide various levels of feedback. Our goal with this project, is to provide an app that uses consumer reviews to analyze and determine a sentiment analysis on randomly generated reviews and provide an instant and accurate response to the review based on whether the review is positive or negative.

In this mining project, the dataset is refined from the Yelp Dataset. For the purposes of this project, we chose to use two of the datasets, yelp\_review.csv and yelp\_business.csv. We merged those two datasets and filtered the data to specifically extract data on Ontario Chipotle restaurants. Our app will mine Chipotle Yelp reviews, run sentiment analysis, and determine the relevant response to the generated review.

**Dataset:**

Our review mining will be completed using the dataset accessed here: <https://www.kaggle.com/ambarish/a-very-extensive-data-analysis-of-yelp/data>

The dataset was compiled by Yelp as a tool for students to conduct research and analysis on Yelps business, review, and user data. The Yelp dataset is made up of seven CSV files containing Yelp data from 5.2 Million reviewers on 174 thousand restaurants spanning 11 metropolitan areas.

As stated above, we merged the yelp\_review.csv and yelp\_business.csv files to create the chipotleinON.csv file containing 22 columns and 568 rows.

**Ethical ML Framework:**

The goal of our report and app is to accurately determine the sentiment of individual reviews, and then be able to provide a relevant response pop-up visual breakdown of the sentiment analysis.

This data is collected and compiled from Yelp’s database. Yelp’s terms of service for review collection and data usage can be found here: <https://www.yelp.com/static?p=tos>.

There are no personal identifying factors in our dataset, therefore consumer and reviewer privacy is maintained completely. There are no socio-economic implications for this review mining app, we can only assume that there are a wide range of individuals who provided reviews to the dataset but have no way of knowing the demographic breakdown.

Our access and usage of the dataset is bound by the dataset agreement, which can be accessed at the following link:

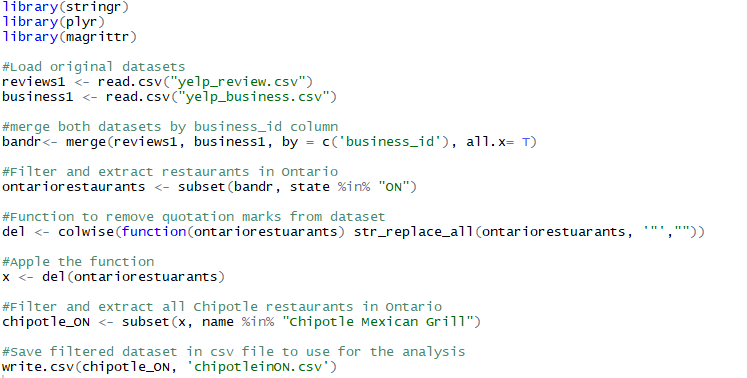
<https://s3-media2.fl.yelpcdn.com/assets/srv0/engineering_pages/af4b9cebfb4f/assets/vendor/dataset-challenge-dataset-agreement.pdf>

**Assumptions:**

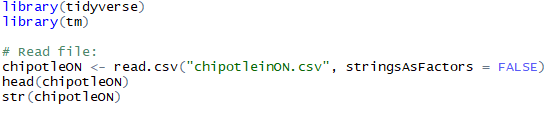
We assume that the datasets are compiled using a wide distribution of opinion, background. Since we are using Chipotle data from across Ontario, we are assuming most of the reviews to be from Ontarians. There is no way to assume the socio-economic background of the reviewers, we are only able to assume that the reviews come from a diverse representation of the province. We are assuming that Chipotle places their restaurants based on their target consumers, so our reviews would be assumed to come from that same consumer breakdown.

**Data Preparation:**

Load original dataset, and merge to create dataset isolating only Chipotle restaurants in Ontario:



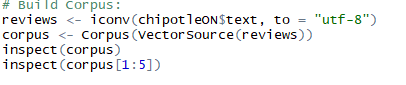
Load created dataset and explore. Please note that we excluded a screenshot of head(chipotleON) simply because of the fact that it is too large of a result to display in this paper. Please refer to our coding files and run code to explore data.



**Word Analysis:**

Next set of code is the building and cleaning of a corpus. The corpus is a collection of text documents from our reviews. We must perform some cleaning on the corpus such as transforming text to lower case letters, removing punctuation/numbers/stop words/URLs, transforming some content, and stripping white space.

Build Corpus:

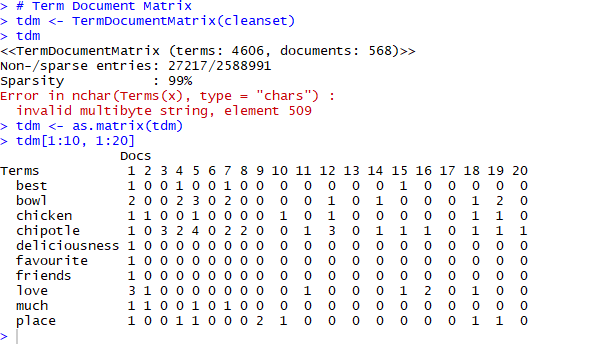


Clean text:



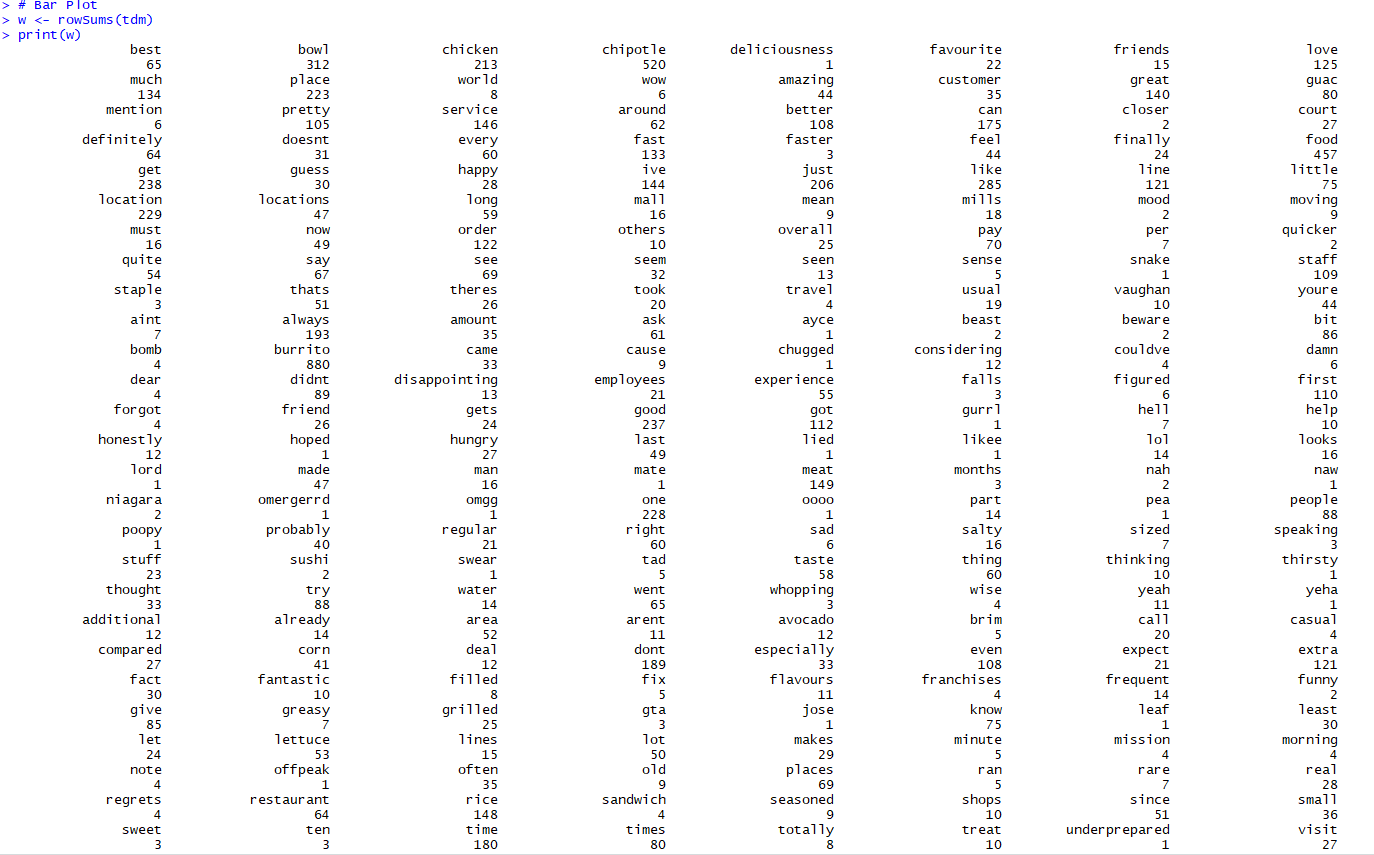
**Term document matrix:**

This will give a visual insight into the frequency of terms that occur in our collection of reviews.

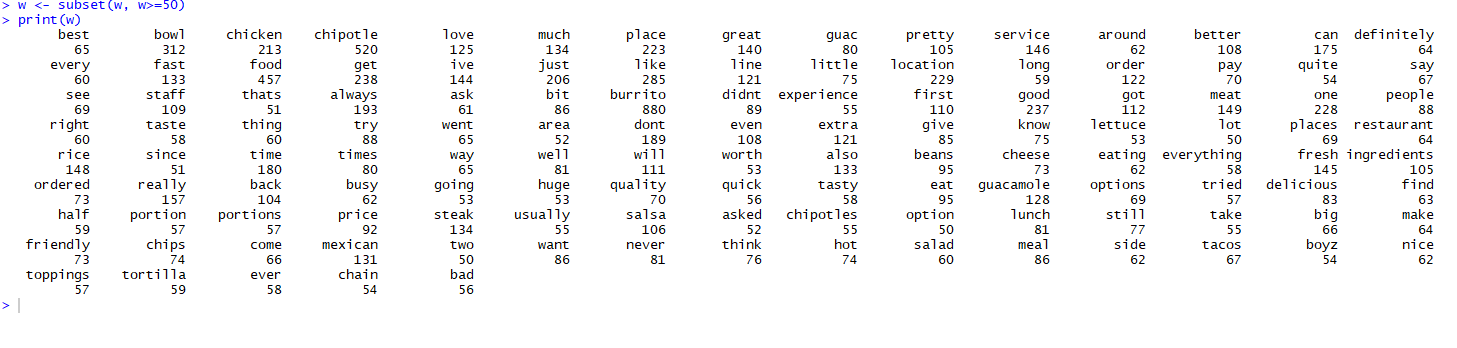


**Bar Plot Visualization:**

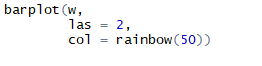
For the sake of space, only a partial view of the bar plot data will be shown below. This visual will give a good idea of the frequency and types of words that will appear in the graphical representation. The screenshot shown below represents roughly one quarter of the bar plot data. You will notice the cleaning of our corpus (in one of the previous sections) results in grammatical consistency amongst all the words mined from our reviews.

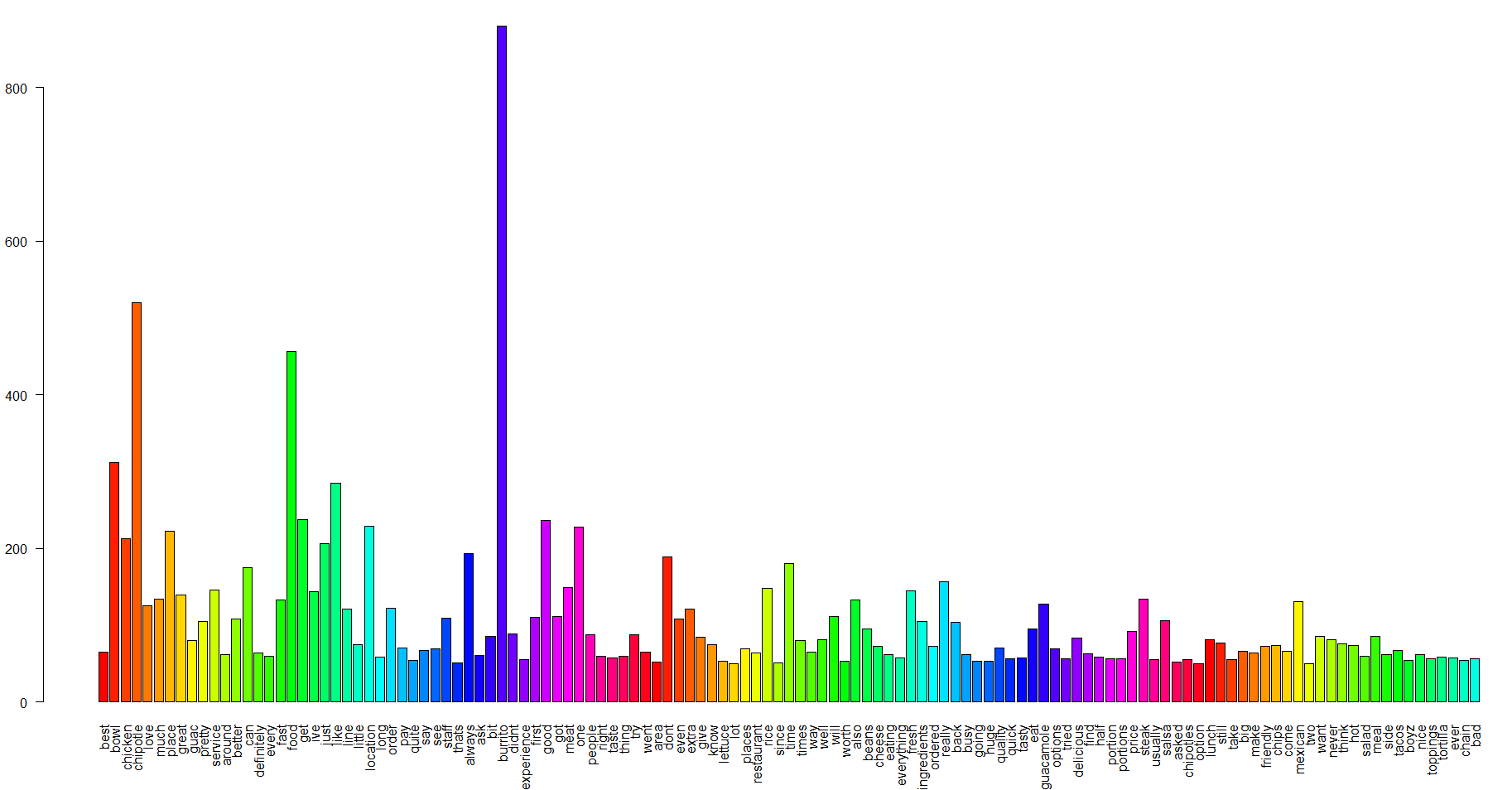


The bar plot data below is a subset of our data of words occurring 50 times or more in our reviews:



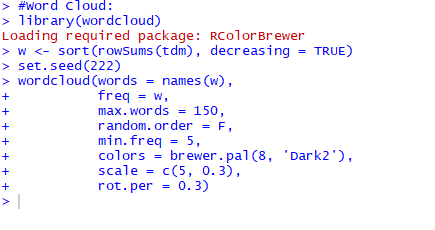
Below you will see the code and graph of our bar plot for all entries occurring with a frequency of 50 times or more:

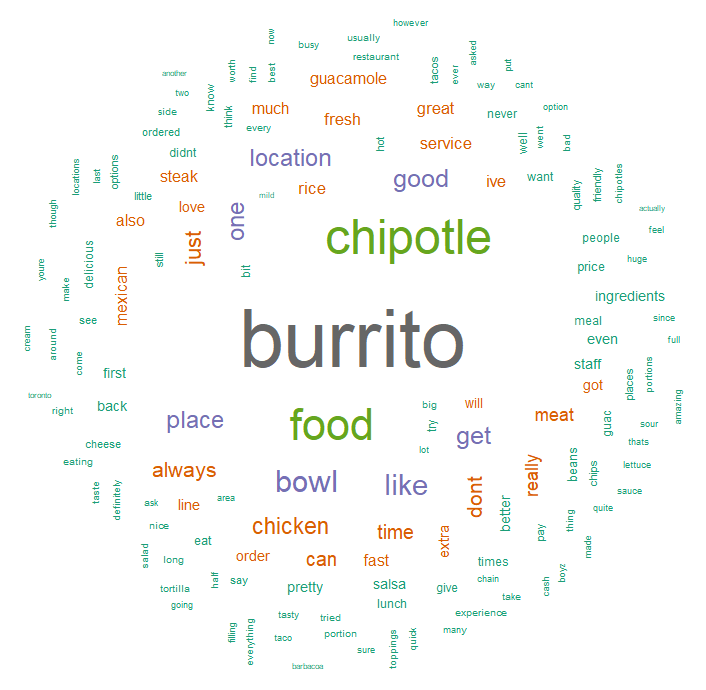




**Word cloud:**

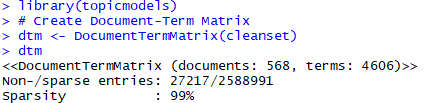
Below you will see the word cloud code and visualization for the 150 most frequent words in our reviews:



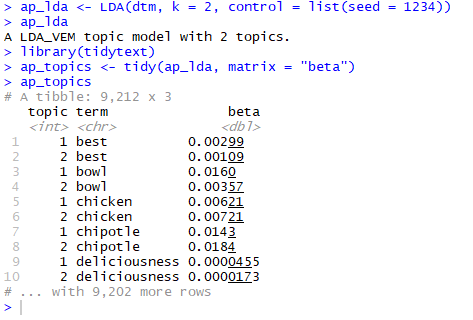


**Topic Modeling - Latent Dirichlet Allocation (LDA)**

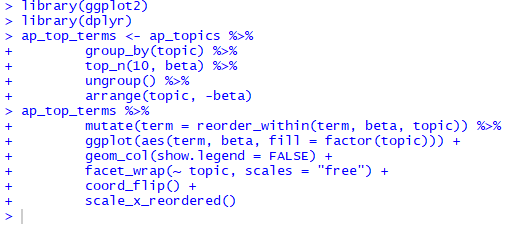
Create document term matrix:

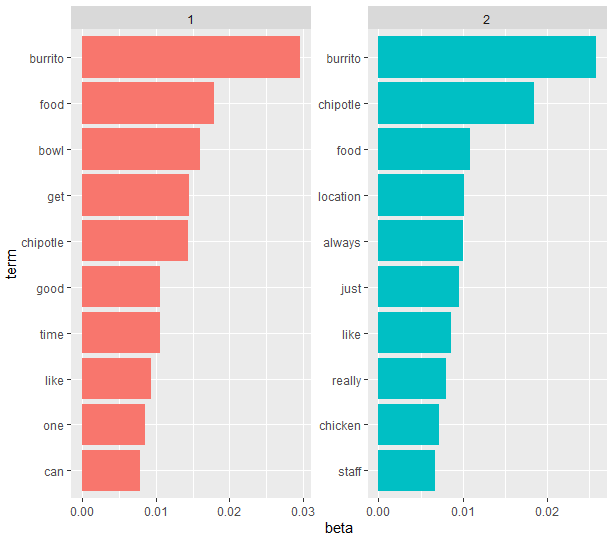


Using LDA() function from the topicsmodels package, setting k=2, to create a two - topic LDA model. Then, using tidytext package to extract the per-topic-per-word probabilities from the model:

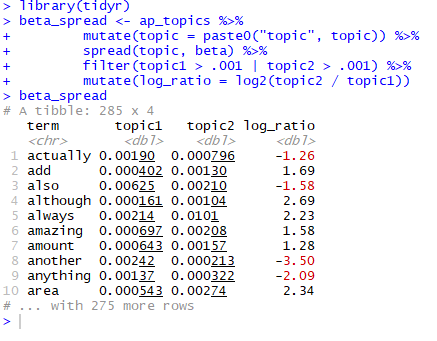


Notice that the above has turned the model into a one-topic-per-term-per-row format. Using dplyr's top\_n() we can find the 10 terms that are most common within each topic:

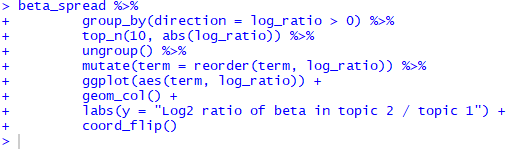


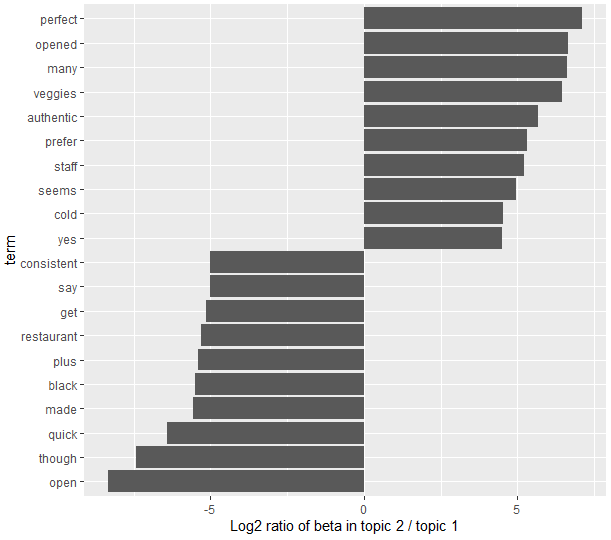


To constrain it to a set of especially relevant words, we can filter for relatively common words:

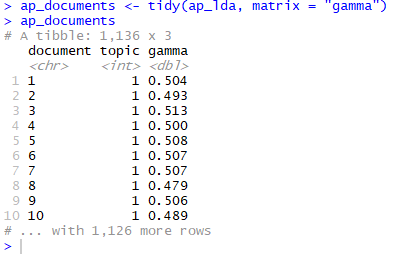


The words with the greatest difference b/w the two topics are visualized in the following:

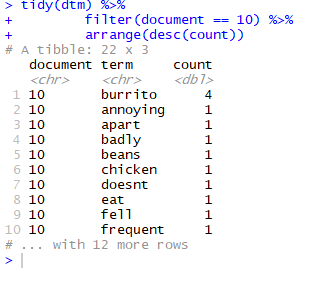




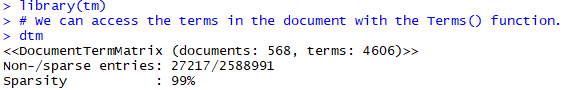
Document-Topic Probabilities: using LDA to model each document as a mixture of topics. We can examine the per-document-per-topic probabilities, called "gamma", with the matrix = "gamma" argument to tidy():

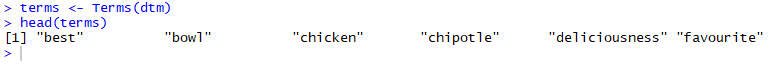


Use following word to check what are the most common words in any given document:



Tidying DocumentTermMatrix objects for Data Analysis. We can access the terms in the document with the Terms() function:

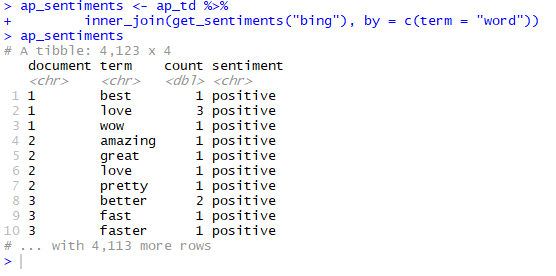




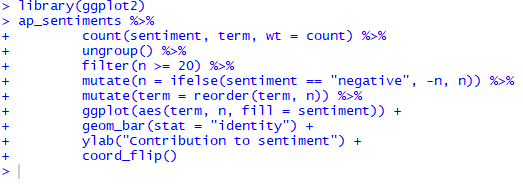
Use the following to analyze the data:

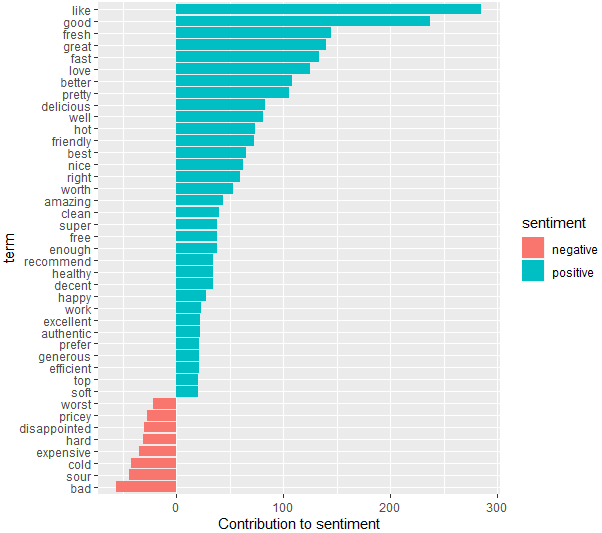


Using the following we can perform sentiment analysis on the documents:



Words from Chipotle reviews with the greatest contribution to positive or negative sentiments, using the Bing sentiment lexicon:

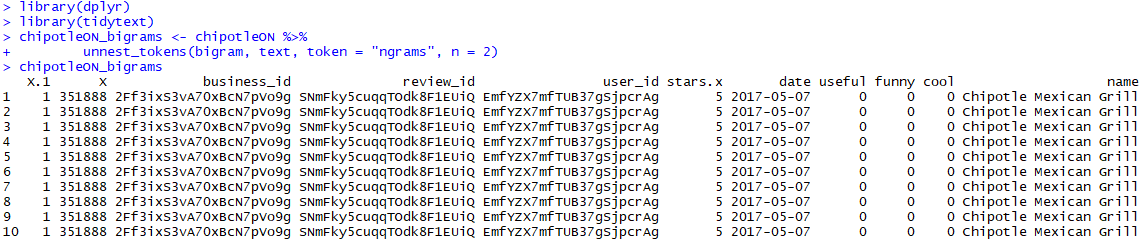


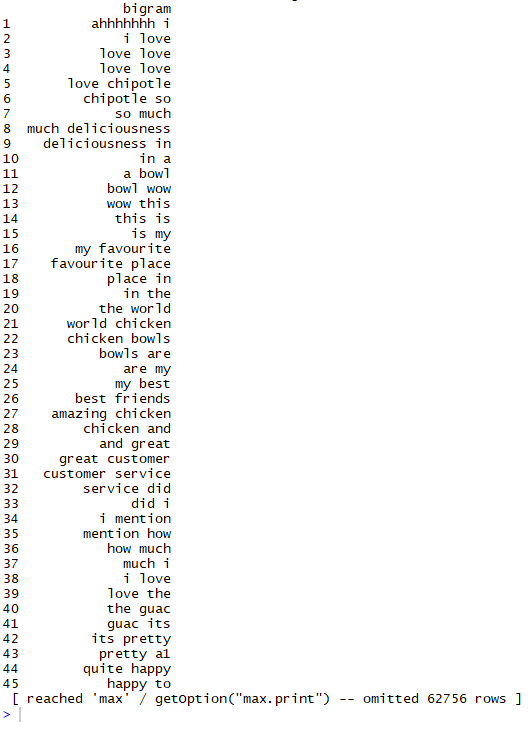


**Relationships between words: n-gram:**

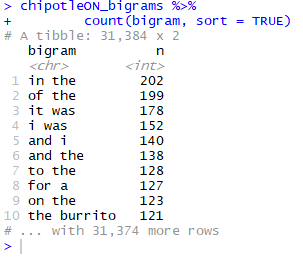
N-grams function is used to tokenize words into consecutive sequences of words. By seeing how often word X is followed by word Y, we can then build a model of the relationships between them. We do this by adding the token = "ngrams" option to unnest\_tokens(), and setting n to the number of words we wish to capture in each n-gram. When we set n to 2, we are examining pairs of two consecutive words, often called "bigrams". In some analyses you may be interested in the most common trigrams, which are consecutive sequences of 3 words. We can find this by setting n = 3.

The following code and image shows partial results (for the sake of space) of our Chipotle bigram, and a list of 45 bigrams below:

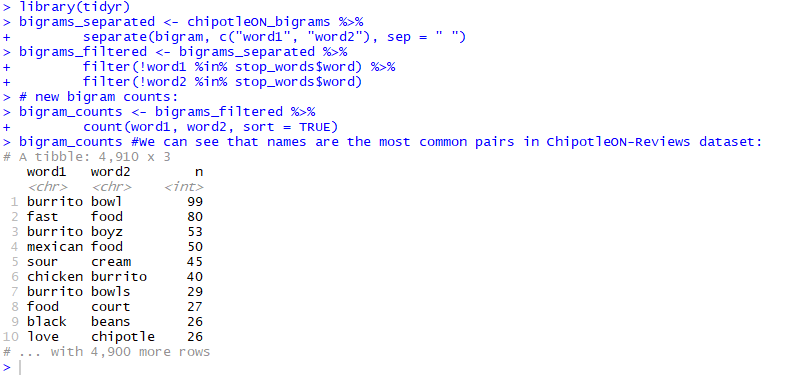




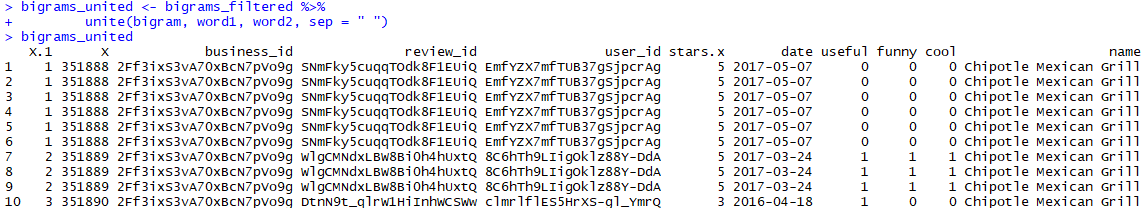
Counting and filtering n-grams- we can examine the most common bigrams using dplyr's count():

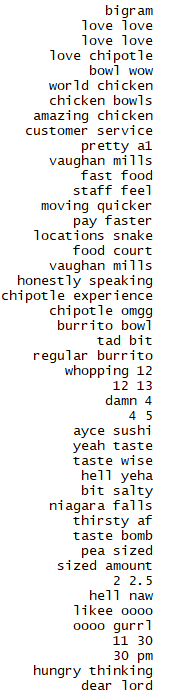


As one might expect, a lot of the most common bigrams are pairs of common (uninteresting) words, such as "in the" and "of the": what we call "stop-words". This is a useful time to use tidyr's separate(), which splits a column into multiple based on a delimiter. This lets us separate it into two columns, "word1" and "word2", at which point we can remove cases where either is a stop-word. You can see that names are the most common pairs in ChipotleON-Reviews dataset:

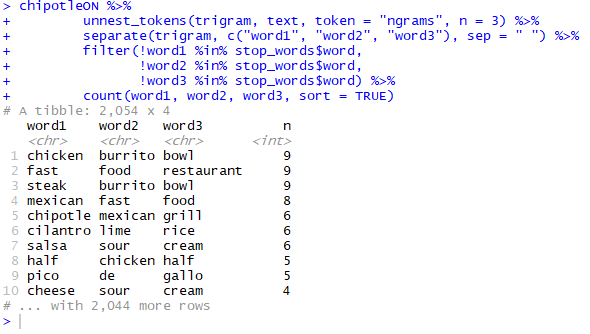


Tidyr's unite() function is the inverse of separate(), and lets us recombine the columns into one. Thus, "separate/filter/count/unite" let us find the most common bigrams not containing stop-words. The following code and image shows partial results (for the sake of space) of our Chipotle bigram, and a list of 45 united bigrams below:



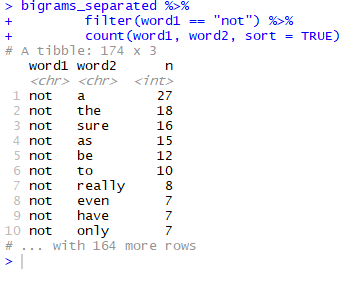


Let's find out the most common TRIGRAMS, which are consecutive sequences of 3 words. We can find this by setting n = 3:

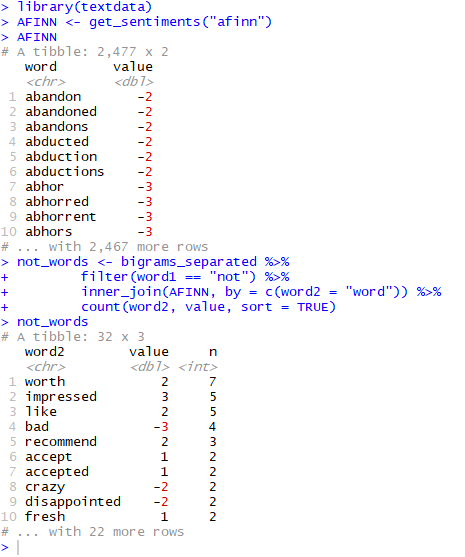


Using bigrams to provide context in sentiment analysis. For example, the words "happy" and "like" will be counted as positive, even in a sentence like "I'm not happy and I don't like it!"

Now that we have the data organized into bigrams, it's easy to tell how often words are preceded by a word like "not". By performing sentiment analysis on the bigram data, we can examine how often sentiment-associated words are preceded by "not" or other negating words. We could use this to ignore or even reverse their contribution to the sentiment score.

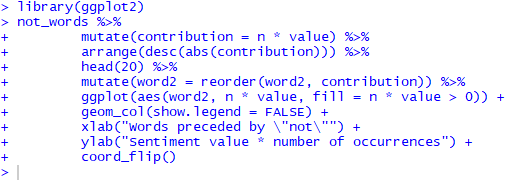


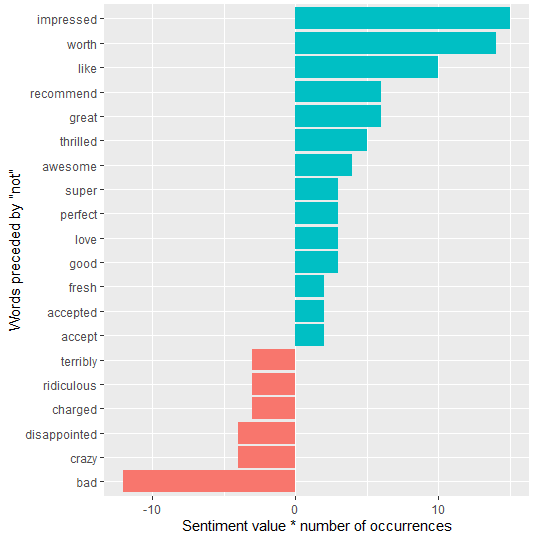
Let's use the AFINN lexicon for sentiment analysis, which gives a numeric sentiment value for each word, with positive or negative numbers indicating the direction of the sentiment. We can then examine the most frequent words that were preceded by "not" and were associated with a sentiment:



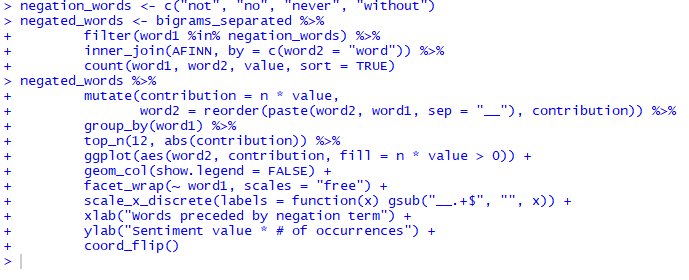
Above you can see the most common sentiment-associated word to follow "not" was "worth", which would normally have a (positive) score of 2.

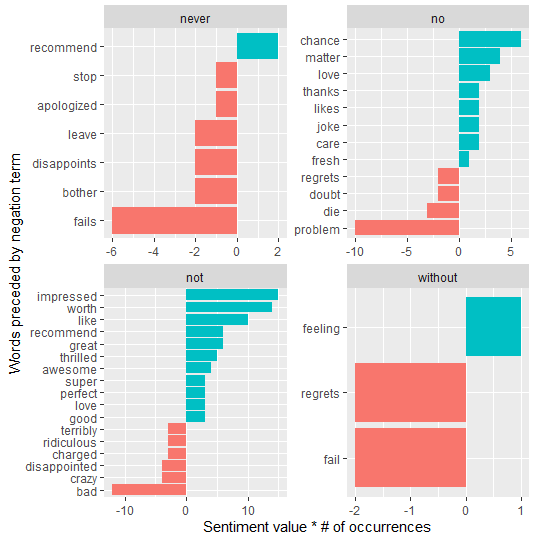
A bar plot to visualize which words contributed the most in the "wrong" direction:





"Not" isn't the only term that provides some context for the following word. We could pick four common words (or more) that negate the subsequent term, and use the same joining and counting approach to examine all of them at once. We can visualize what the most common words to follow each particular negation:

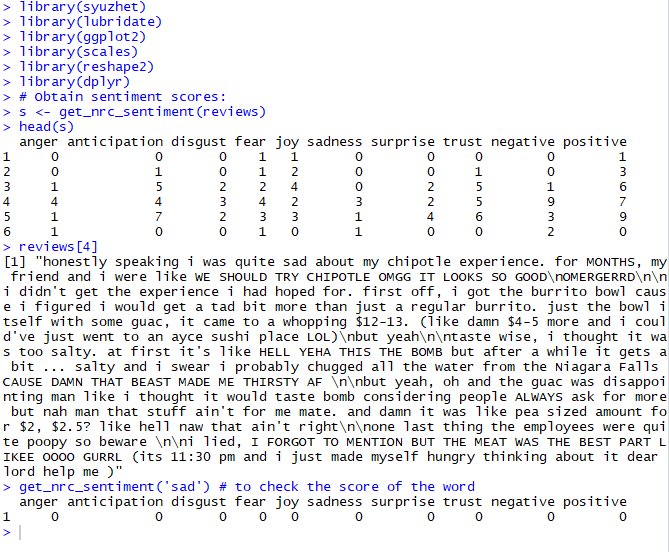




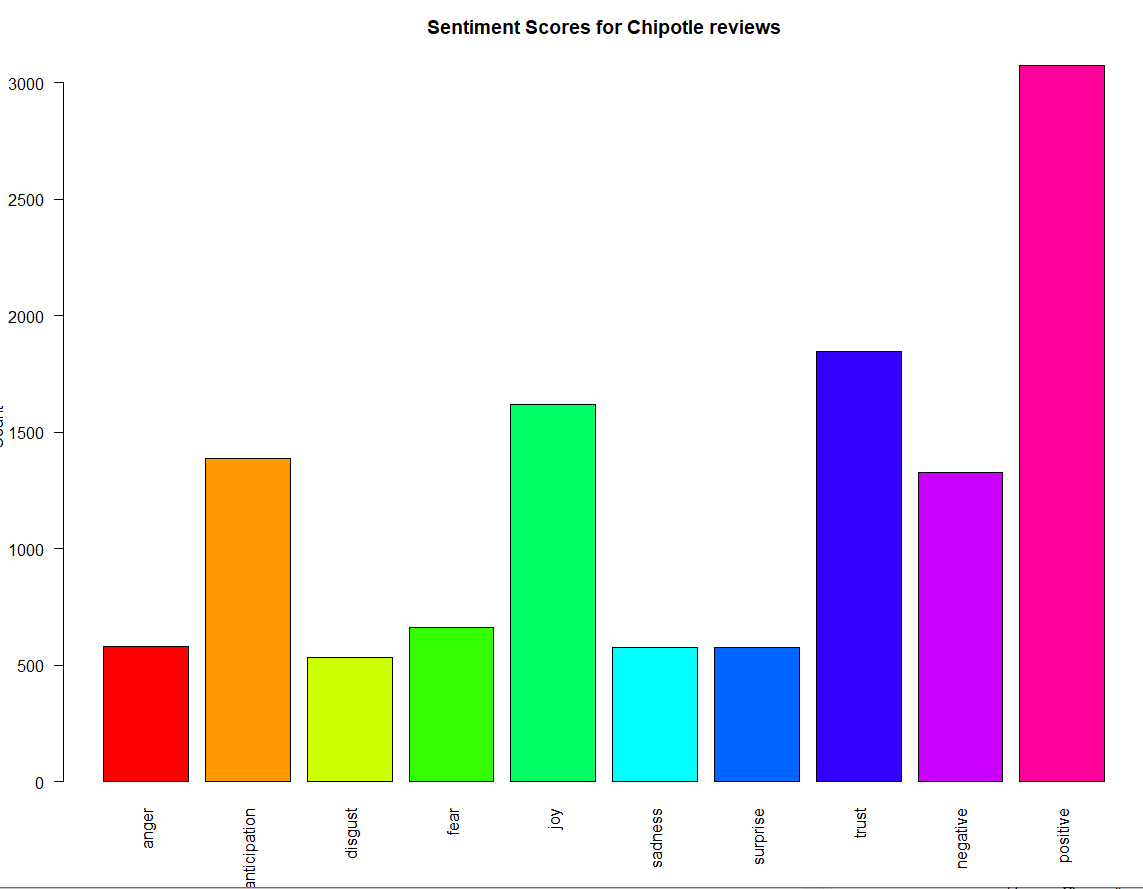
Note - Using bigram we actually double the total size of our matrix! Sparsity problem - Most of the cell in your matrix are going to be empty. This is because most of the time, most of the documents are not going to share the same bigram. Therefore, as you add more powerful feature like trigram & fourgram etc., it is even less likely that particular word order is going to be shared across multiple documents. Sparsity problem exists where matrix is huge because of lots of columns.

**Sentiment Analysis:**

The code below is our sentiment analysis of the Chipotle Ontario reviews. You will see we checked the sentiment analysis on review #4. As you can see through reading the review, sentiment can be difficult to determine. This customer had many things they seemingly didn’t like, but they did like the meat. Overall, however, the review was more negative than positive.



Below you will see a bar graph visualization of the overall sentiment analysis across all Chipotle Ontario reviews. You will notice there are both positive and negative sentiments discovered through the reviews, however, overall the scores for their reviews in general are overwhelmingly positive:



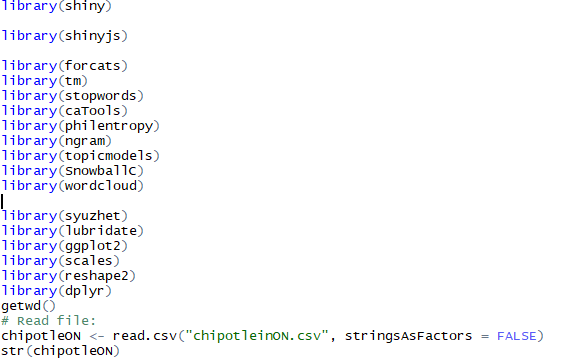
**Prepare Shiny App for Deployment:**

App can be found at: <https://alvin20142.shinyapps.io/Chipotle_Restaurant_Review/>

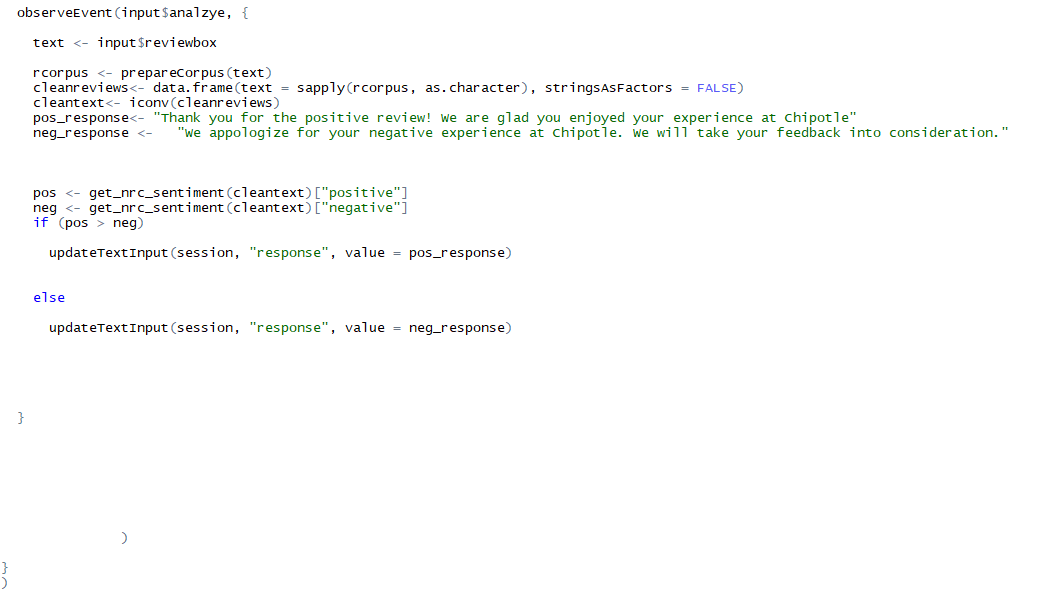
Build and prepare ui.R (user interface):



Build and prepare server.R file:







**App Deployment Discussion:**

As you will see by our app (you can run the code, or click the link in the previous section of this paper), we have created a review generator which analyzes the review and gives either a thank you response, or an apology depending on whether the review is determined positive or negative in our sentiment analysis.

Although we are pleased with the accuracy, you will notice that once in a while, analysis of the generated review does turn up an incorrect response. Further exploration and work can be done to improve the accuracy of the analysis before this app would be ready for market implementation.

The app provides strong proof of concept. Ideally, future iterations of this app would require a deeper analysis and hopefully be able to identify key concepts in the reviews as well as sentiment. The app would then be able to provide more personalized and accurate machine generated responses to the consumer at the point of submission.

**Resources:**

<https://www.kaggle.com/ambarish/a-very-extensive-data-analysis-of-yelp/comments>