IDS 572 Homework 4: Text Mining

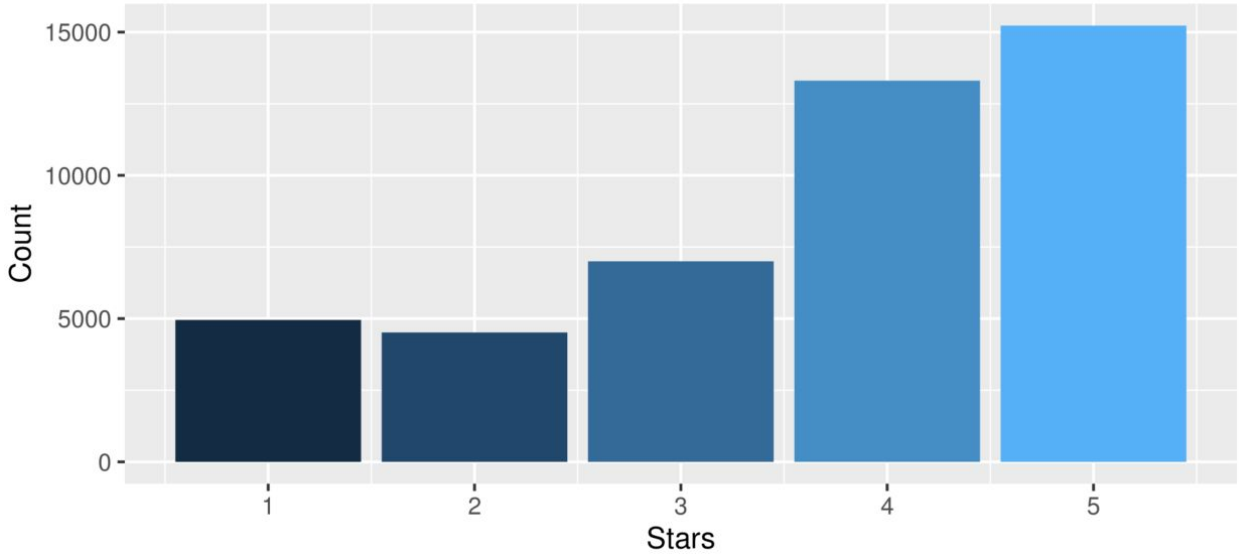
Abdullah Saka, Britney Scott

Introduction

Yelp is a platform through which users can review their experiences with a wide variety of businesses. Each review consists of a text portion, as well as a star rating using a 1 to 5 scale. This project takes a subset of the restaurant reviews on Yelp and attempts to draw conclusions about the relationship between various words in the text portion of the review and the star rating through using text mining. Specifically, we will utilize the “bad of words” approach to text mining and apply three individual dictionaries. The objective is to identify which reviews are positive and negative based on the context of the text portion of the review.

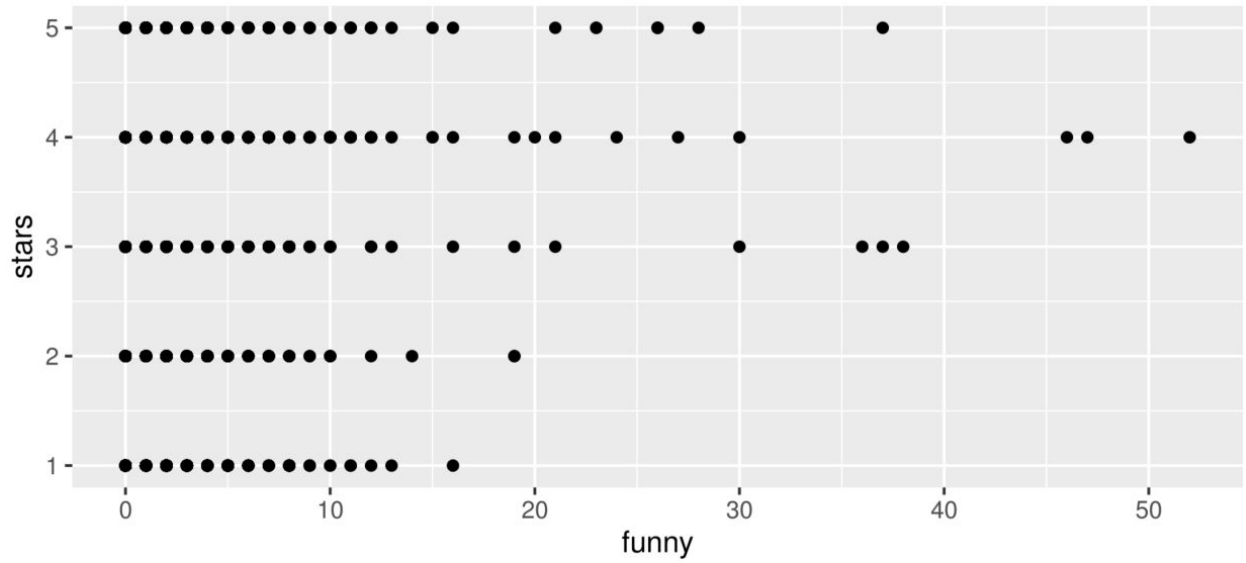
Data Exploration and Preparation

To begin, we explored the data in order to determine some basic information about the ratings in the provided dataset. The star ratings are distributed somewhat unevenly throughout the dataset, as demonstrated in the following histogram.

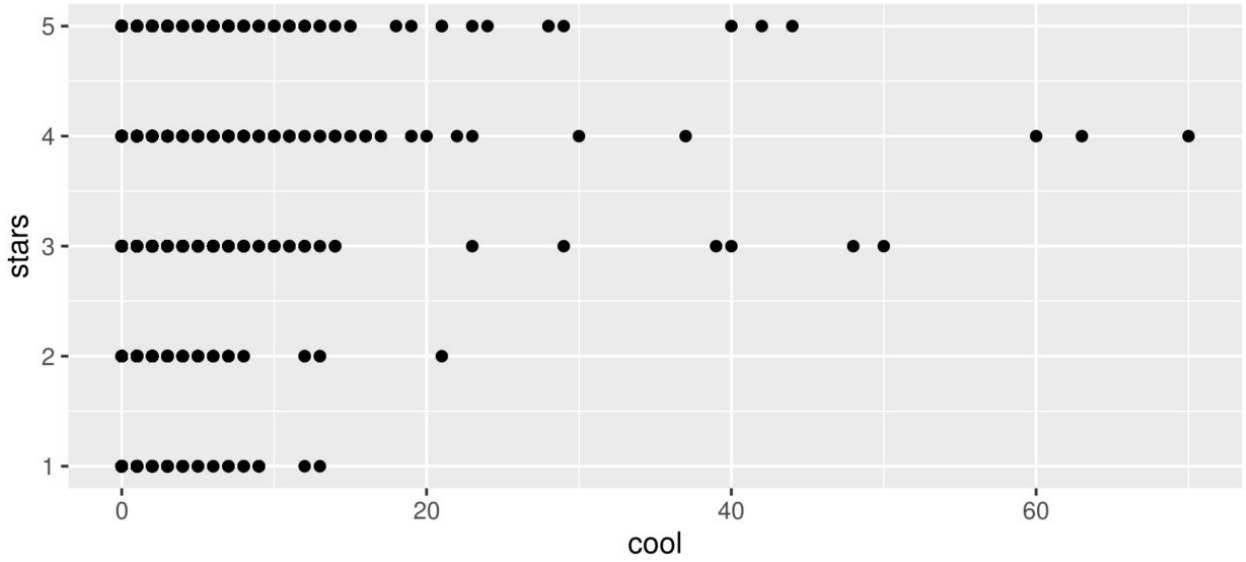


In order to perform sentiment analysis, the star ratings must be transformed into a binary classification. Two classes indicating positive and negative reviews will be required. To do so, we will eliminate all reviews which fall into the 3 star ratings. These reviews are considered neutral rather than positive or negative. Ratings of 1 and 2 stars will be considered negative, while ratings of 4 and 5 stars will be considered positive.

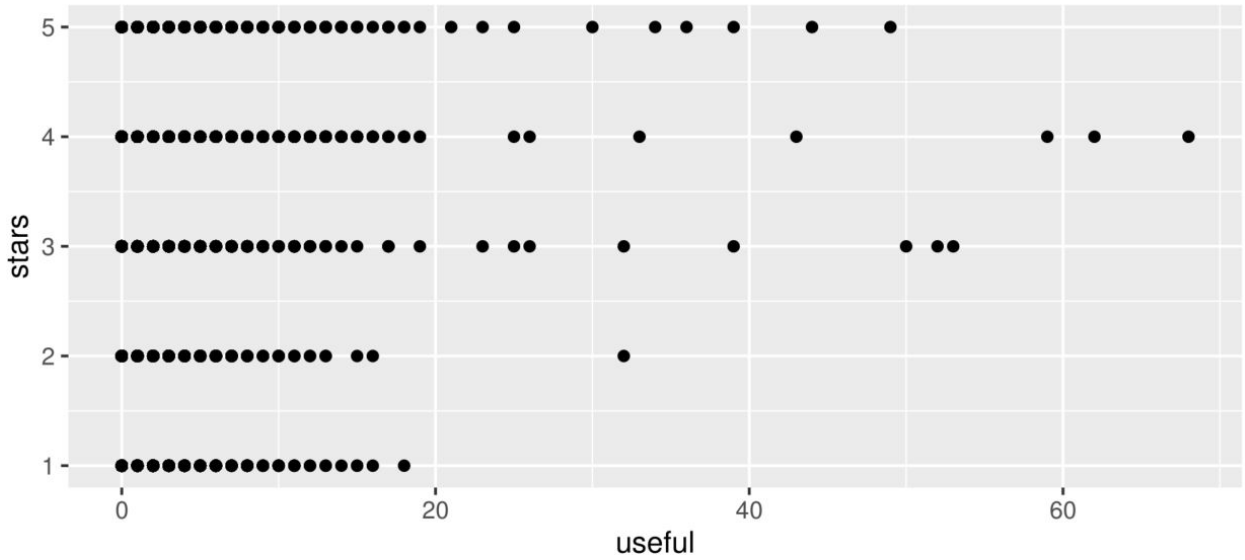
Before continuing to the sentiment analysis, though, we will examine a few words which are present in the text reviews and see if they relate to specific star ratings. Specifically, we will focus on the words 'funny', 'cool' and 'useful', all of which we would expect to be related to the positive reviews.



It's evident in the above plot that the word 'funny' is most commonly used in 4 star reviews. It's not very common in the negative reviews, which makes sense considering funny is generally a positive quality.



Similarly, 'cool' is generally related with positive reviews. It's very interesting that this word seems to be used in 3 star ratings even more than 5, but it's clearly most common in the 4 star ratings.



Finally, the word useful is also commonly used within the positive reviews. The patterns are similar to the previous words.

Before performing sentiment analysis, we also did some modification to the data. We removed all reviews which did not come from a 5-digit postal code. We then tokenized the reviews, which converts the reviews from one long string of text to individual words. This is done in order to prepare the data for sentiment analysis, as each individual word from the review will need to be compared to the dictionary. The order of the words is not relevant since we will use the "bag of words" approach.

Next, we removed stop words because they are not helpful in understanding the meaning behind the review. Stop words include words such as 'and', 'a', and 'the' which are present across all reviews regardless of the review's content. Removal of the stop words decreased the total number of tokens (unique words) from 68,204 to 67,505

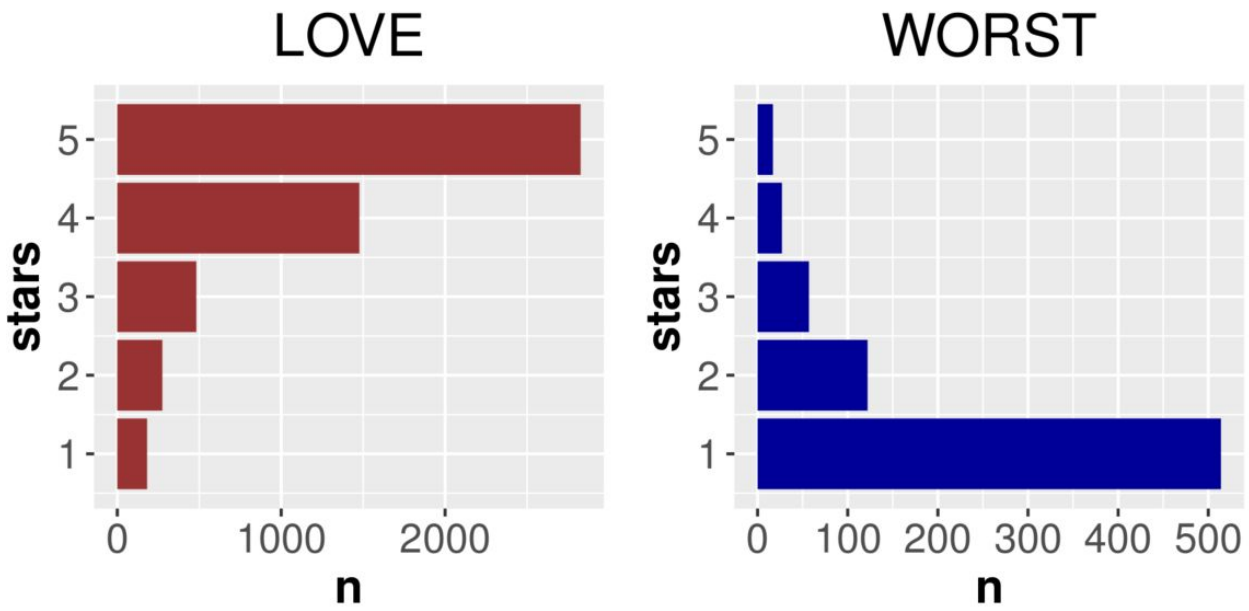
We also wanted to remove any additional words which are either present in most reviews, or in very few reviews. Rare words included several words with numbers such as '12pm', as well as some words in other languages and words that are not very relevant to restaurants such as 'courthouse.' The most common word used in over 20,000 reviews is 'food', which does not indicate a positive or negative sentiment since the review could either compliment or complain about the food. Therefore, we removed all words used in less than 10 reviews or more than 15,000 reviews.

We also removed any other numeric words since they don't have much meaning in terms of sentiment analysis. All of this brought down the number of tokens to 8,649. This is a significant reduction in tokens from the initial set of 68,204.

Data Analysis

Then, we analyzed the frequency of words in each rating and then calculated the proportion of word occurrence by star ratings. We checked the proportion of 'love' (positive sentiment) and 'worst' (negative sentiment) among reviews with rating stars. We can clearly say that while rating 4 and 5 represent positive reviews, rating 1 and 2 are more related to negative reviews.

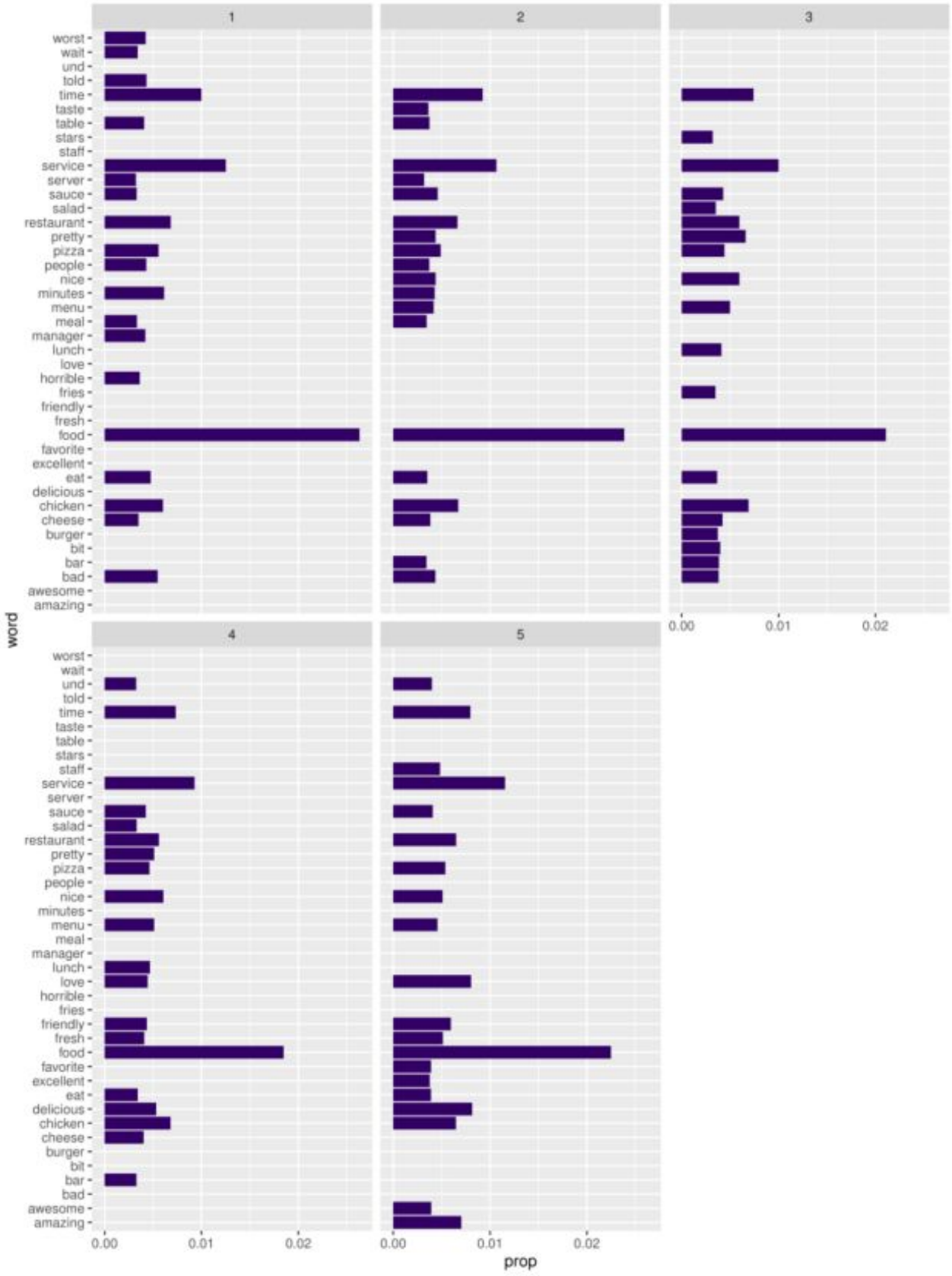
.



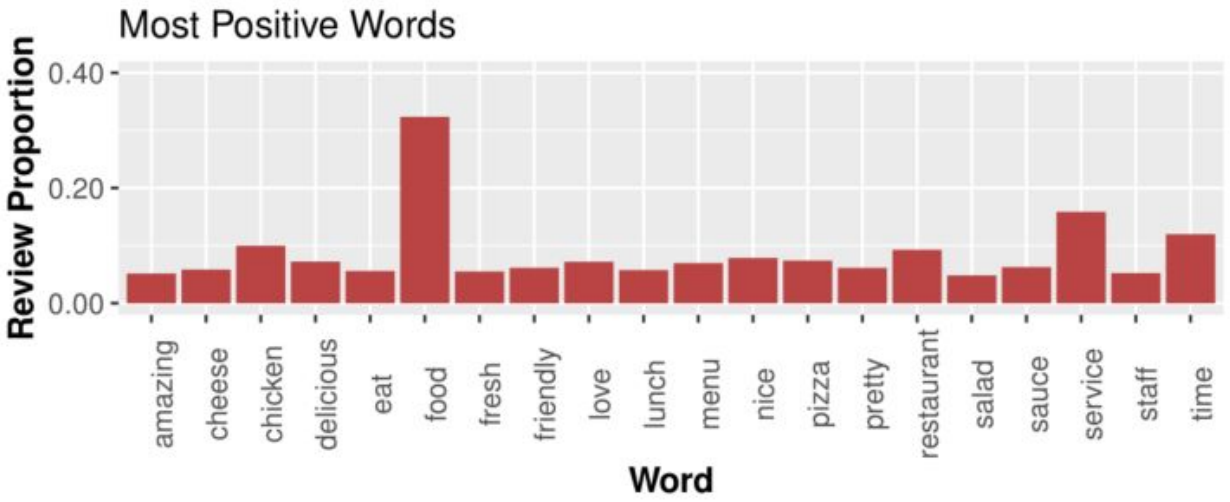
Afterwards, we computed the number of occurrences and probability of each word by rating. We plotted the top 20 words in each rating to understand the difference between ratings. According to the plot shown below, there are common words among ratings which are 'service', 'restaurant', 'menu', 'table', 'people' and 'time'. We have to prune a set of common words from the token list since these words are not useful for understanding differences among reviews. As expected, ratings 1 and 2 include negative words such as 'bad', 'worst', 'horrible' and 'wait'. On the other hand, higher ratings (4 and 5) consist of positive words which are 'delicious', 'amazing', 'pretty' and 'nice'.

To understand which words are generally related to higher and lower ratings, we calculated the average stars associated with each word and then summed up the star ratings with reviews where each word occurs in. Based on that, top 20 words with the highest ratings include general words ('restaurant', 'service', 'menu') and positive words ('nice', 'delicious', 'love' and 'friendly').

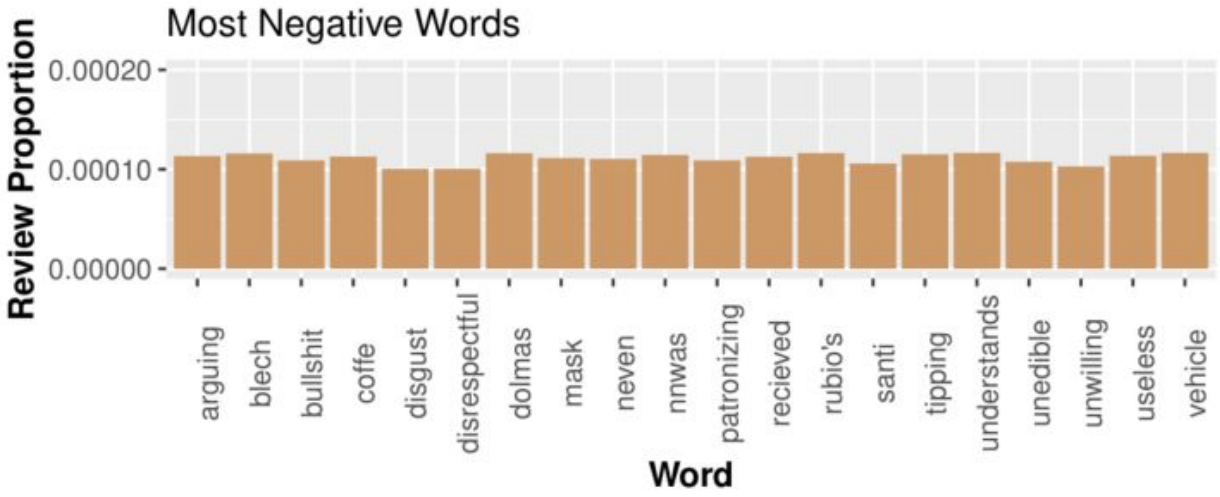
Nevertheless, review of lowest star rating generally obtains negative words which are 'disgust', 'disrespectful', 'unwilling', 'bullshit'. This analysis states the fundamental difference between ratings.


1
2
3
4
5
worst
wait
und
told
time
taste
table
stars
staff
service
server
sauce
salad
restaurant
pretty
pizza
people
nice
minutes
menu
meal
manager
lunch
love
horrible
fries
friendly
fresh
food
favorite
excellent
eat
delicious
chicken
cheese
burger
bit
bar
bad
awesome
amazing
word
0.00
0.01
0.02
prop

To understand which words are generally related to higher and lower ratings, we calculated the average stars associated with each word and then summed up the star ratings with reviews where each word occurs in. Based on that, the top 20 words with the highest rating includes general words ('restaurant', 'service', 'menu') and positive words ('nice', 'delicious', 'love' and 'friendly').



Nevertheless, review of lowest star rating generally obtains negative words which are 'disgust', 'disrespectful', 'unwilling', 'bullshit'. This analysis states the fundamental difference between ratings.



As a result of exploratory data analysis, we removed common words from the token list such as ‘food’, ‘chicken’, ‘service’, ‘time’ and ‘restaurant’. The reason why we eliminate is to enhance the performance of sentiment analysis. These words do not make a difference in identifying positive and negative sentiments among documents.

Stemming and Lemmatization

After tokenizing and removing stopwords, we are able to perform stemming or lemmatization processes in text mining. When stemming helps us achieve the root forms of inflected words. Moreover, lemmatization is the process of grouping together the diverse inflected forms of a word so they can be analyzed as a single item. While converting any word to the root-base word, stemming can create non-existent work but lemmatization generates real dictionary words. This table shows difference between stemming and lemma words:

Original	Stemming	Lemma
salad	salad	salad
beware	bewar	beware
men’s	men’	men’s
bathroom	bathroom	bathroom
portions	portion	portion
imagine	imagin	imagine
night	night	night
friends	friend	friend
typical	typic	typical
improved	improv	improve
servers	server	server
attentive	attent	attentive
previous	previou	previous
experience	experi	experience
recommended	recommend	recommend
clams	clam	clam
lemon	lemon	lemon
pepper	pepper	pepper
seasoning	season	season
salty	salti	salty

Term Frequency

We carried out lemmatization instead of stemmed words and filtered out less than 3 characters and more than 15 characters to decrease the number of tokens. Then, we computed tf-idf scores in order to run sentiment analysis. Tf-idf is a statistic which reflects how important a word is to a document in a collection of groups. Term frequency (tf) identifies the frequency of individual terms within a document. Also, we need to understand the importance that words provide within and across documents. Inverse document frequency (idf) which decreased the weight for commonly used words and increased the weight for words that are not used very much in a collection of documents.Tf-idf score calculated by multiplying these two scores. The table below demonstrates tf-idf scores of the first review.

review_id	stars	word	n	tf	idf	tf_idf
–9qM_dRW4rrKTWO_SX_qQ	1	buffet	1	0.1	4.075686	0.4075686
–9qM_dRW4rrKTWO_SX_qQ	1	copper	1	0.1	6.862459	0.6862459
–9qM_dRW4rrKTWO_SX_qQ	1	kettle	1	0.1	7.373285	0.7373285
–9qM_dRW4rrKTWO_SX_qQ	1	price	1	0.1	1.919389	0.1919389
–9qM_dRW4rrKTWO_SX_qQ	1	service	1	0.1	1.184336	0.1184336
–9qM_dRW4rrKTWO_SX_qQ	1	soo	1	0.1	6.048359	0.6048359
–9qM_dRW4rrKTWO_SX_qQ	1	star	1	0.1	2.466600	0.2466600
–9qM_dRW4rrKTWO_SX_qQ	1	suck	1	0.1	4.652616	0.4652616
–9qM_dRW4rrKTWO_SX_qQ	1	tempe	1	0.1	5.565777	0.5565777
–9qM_dRW4rrKTWO_SX_qQ	1	top	1	0.1	2.769067	0.2769067

Sentiment Analysis

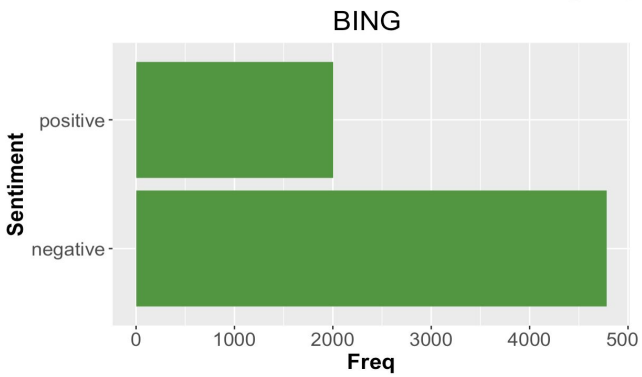
In the sentiment analysis, we applied 3 different dictionaries as follows of:

- Bing
- Nrc
- Afinn

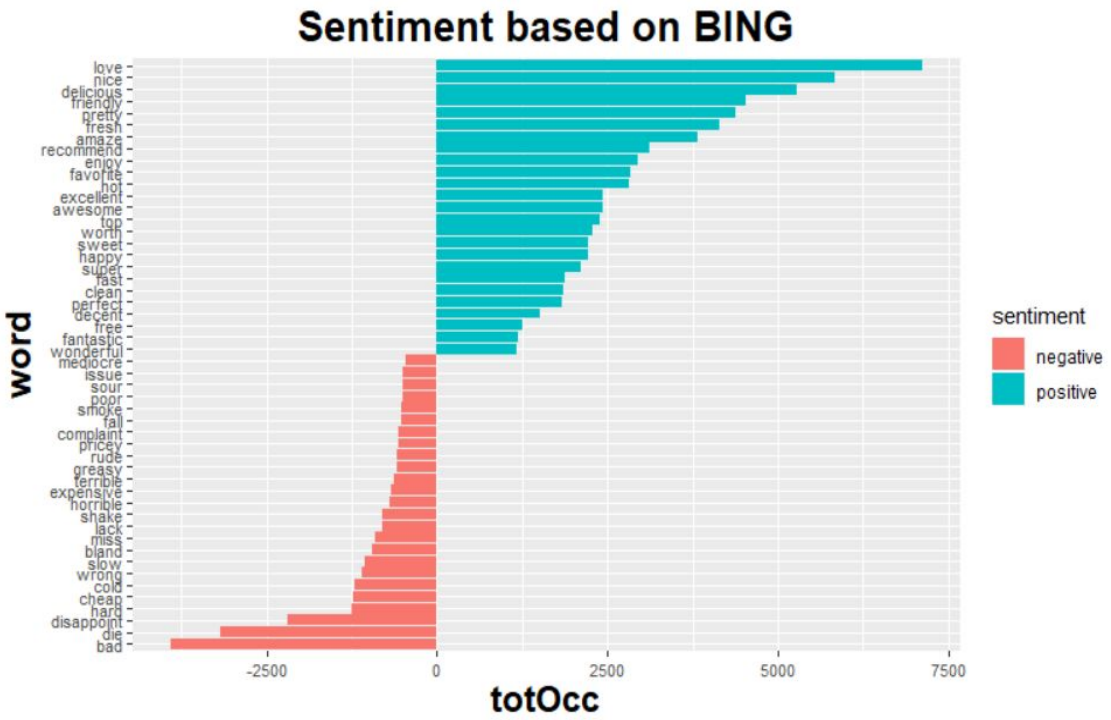
1. BING Dictionary

Bing with Inner Join

We focused on the 'bing' dictionary which includes 6786 words and their sentiments. Sentiments are described as positive or negative. Most of the words (4781) belong to negative sentiments.



To determine sentiments to words in documents, we applied 'bing' dictionary by doing inner join. There were 935 individual words in the review data matching with the ‘bing’ dictionary. Then, we also added occurrences of positive and negative sentiment words in reviews. The plot demonstrates the most positive and negative words in reviews. While, 'love', 'nice', 'delicious', 'friendly', 'pretty' are the most popular positive words, 'bad', 'disappoint', 'die', 'hard', 'cold’ represents negative sentiments.



We have analyzed overall sentiment across reviews until this point, now we concentrate on sentiment by review to understand how review relates to review's star ratings. For each review, we computed positive and negative words, then created a probability of being positive and negative. Lastly, we created a sentiment score by taking the absolute value of difference of positive and negative score of review.

stars <dbl>	avgPos <dbl>	avgNeg <dbl>	avgSentiSc <dbl>
1	0.2924336	0.7075664	-0.4151328
2	0.4464544	0.5535456	-0.1070913
3	0.5964308	0.4035692	0.1928615
4	0.7373631	0.2626369	0.4747263
5	0.8109949	0.1890051	0.6219898

By using the sentiment score of reviews, we computed the average of positive and negative scores for each rating. According to the table above, star ratings 1 and 2 represent negative reviews since average sentiment score is below than zero. Nonetheless, star rating 4 and 5 points out positive reviews.

We built a document-term matrix of reviews dataset. In a document-term matrix, rows show reviews and columns correspond to words. Then, we filtered out reviews whose rating is 3 since sentiment score of rating 3 is positive, but lower (includes both negative and positive reviews). Then, when the star rating of review is less than 2, we assigned these reviews as class -1 and others belong to class 1. Based on the table, the most of reviews are assigned to class 1 and only 6671 reviews correspond to negative reviews. The document-term matrix has 27473 rows and 937 columns.

hiLo <dbl>	n <int>
-1	6670
1	20803

Bing with Left Join

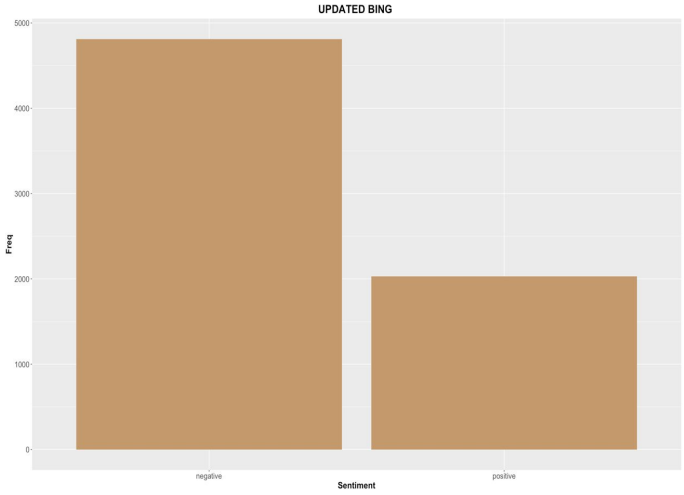
We used the same dictionary by applying left join operation. There are lots of worlds which do not exist in the bing dictionary. Firstly, we removed neutral words which do not have any effect on sentiment (popular words in reviews).

	words
1	travel
2	drink
3	eat
4	life
5	pay
6	money
7	customer
8	mexican
9	salad
10	sandwich
11	staff
12	location

After, we created a small dictionary and then added this dictionary to the original ‘bing’ dictionary. This shows a part of the dictionary in which we created:

	word	sentiment
1	unprofessional	negative
2	tasty	positive
3	caution	negative
4	recommend	positive
5	advice	positive
6	stop	negative
7	cancel	negative
8	large	positive
9	survive	negative
10	tremendous	positive
11	wait	negative
12	leave	negative
13	hit	negative
14	hungry	negative

Then, an updated dictionary which includes 6786 words and their sentiments. Sentiments are described as positive or negative. Most of the words (4811) belong to negative sentiments. Remaining 2030 words corresponds to positive sentiments.



Then, we calculated the average of positive and negative scores for each rating. According to the table below, star ratings 1 and 2 still represent negative reviews since average sentiment score is below than zero. Nonetheless, star rating 4 and 5 points out positive reviews.

stars <dbl>	avgPos <dbl>	avgNeg <dbl>	avgSentiSc <dbl>
1	0.2905854	0.7094146	-0.4188291
2	0.4283059	0.5716941	-0.1433881
3	0.5615179	0.4384821	0.1230359
4	0.6961578	0.3038422	0.3923156
5	0.7690244	0.2309756	0.5380488

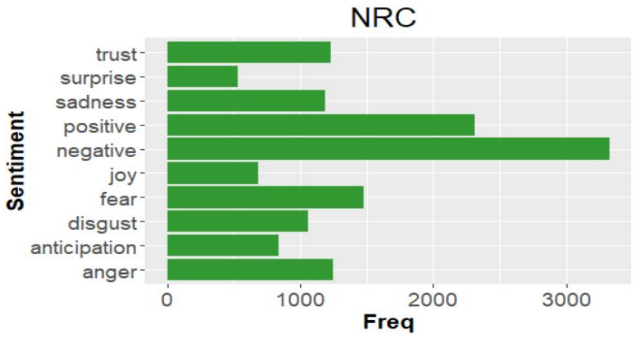
When we compared to the previous one, sentiment score of rating 2 increased slightly and sentiment score of 4 and 5 decreased gradually.

hiLo <dbl>	n <int>
-1	6801
1	21062

This indicates distribution of classes. In comparison with the inner join of a dictionary, this includes more reviews since we can lose many words and reviews while applying inner join.

2.NRC Dictionary

For a second dictionary choice, we used the NRC dictionary and an inner join. Rather than just identifying words as positive and negative, this dictionary assigns a more specific sentiment to each word. For example, some words may portray 'anger', 'trust' and 'negative'. Within the dictionary, the most common sentiment class for the words in ‘negative’.



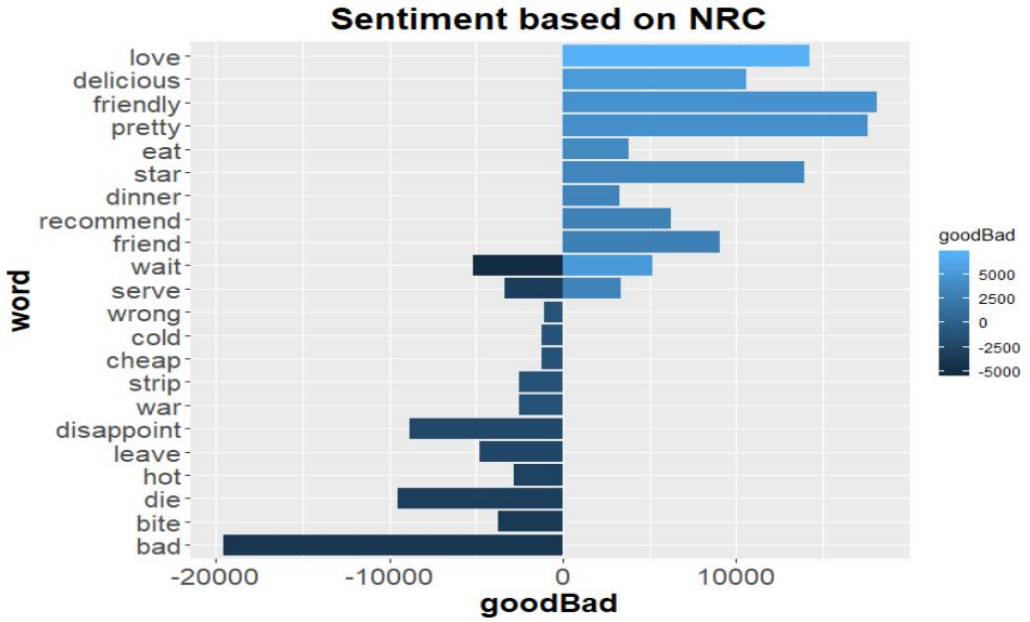
The dictionary had 2831 matching terms with the review data. The occurrences of these sentiments in the data set are summarized in the following table. ‘Count’ is the number of unique words associated with that sentiment in the dataset, and ‘sum n’ is a sum of all of the occurrences of those words across the different reviews.

sentiment	count	sumn
anger	186	30079
anticipation	249	71521
disgust	164	23847
fear	201	27001
joy	237	87548
negative	499	76956
positive	636	168393
sadness	182	31661
surprise	145	33290
trust	332	85883

We checked in more detail the words which were associated with the different emotions in the dataset. Here are the top three words for each of the sentiments within the reviews. It's obvious that some of the words are associated with multiple sentiments.

- Anger: bad, hot, disappoint
- Anticipation: wait, friendly, pretty
- Disgust: bad, disappoint, finally
- Fear: bad, die, war
- Joy: love, delicious, friendly
- Negative: wait, bad, bite
- Positive: love, delicious, friendly
- Sadness: bad, die, leave
- Surprise: amaze, leave, sweet
- Trust: friendly, pretty, star

We can consider some of these emotions such as anger, disgust, fear, sadness and negative to be 'bad', while positive, joy, anticipation and trust are 'good.' This way, we can group together all of the positive and negative emotions to determine which words in the review are most commonly good and bad. After grouping the emotions together, we can view the most positive and negative reviews in the same way as with the Bing dictionary.



It can be seen that some of the words are both positive and negative. This is possible with the NRC dictionary since words can fall into more than one sentiment, as demonstrated above. For example, the word 'wait' could be used in multiple contexts. One review could say 'the restaurant had a long wait' and leave a negative star rating. Alternatively, someone could say 'I cannot wait to eat here again!' and leave 5 stars. This dictionary helps to acknowledge this fact.

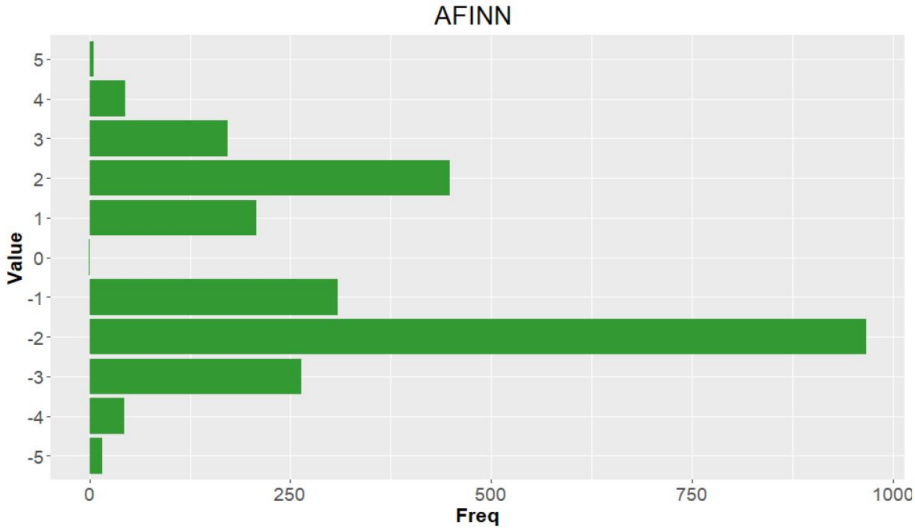
Once again, we created a document-term matrix using the terms in the NRC dictionary and filtered out the reviews with the rating of 3. We again assigned ratings of 1 and 2 stars to the -1

class and of 4 and 5 stars to the 1 class. Below is the distribution of the classes now, which is still highly imbalanced as in the Bing dictionary. The document-term matrix has 28061 rows and 1310 columns.

hiLo <dbl>	n <int>
-1	6893
1	21168

3.AFINN Dictionary

The third dictionary that we will consider is the AFINN dictionary. Rather than simply labeling the words as positive or negative in this dictionary, AFINN assigns a score from -5 to 5 to every word. This way, a word scored at -5 can be thought of as more negative than -3, and 5 as more positive than 3. For example, the word 'breathtaking' is very positive with a rating of 5, and 'ability' is still positive with a score of 2, but less positive than the 5.
As is the case with the other two dictionaries, this one contains more words in the negative scores than the positive ones, with -2 being the most frequent score within the dictionary.



We used an inner join for this dictionary, and there were 518 matching terms with the reviews. Within these specific reviews, we can see which values are most commonly used in the following table. Despite the fact that the dictionary contains the most words with the -2 score, words with a score of 2 are most commonly used within the reviews with a total of 40,681 occurrences. This is expected, since the distribution of the reviews is skewed as shown earlier, and there are more positive reviews than negative within the dataset.

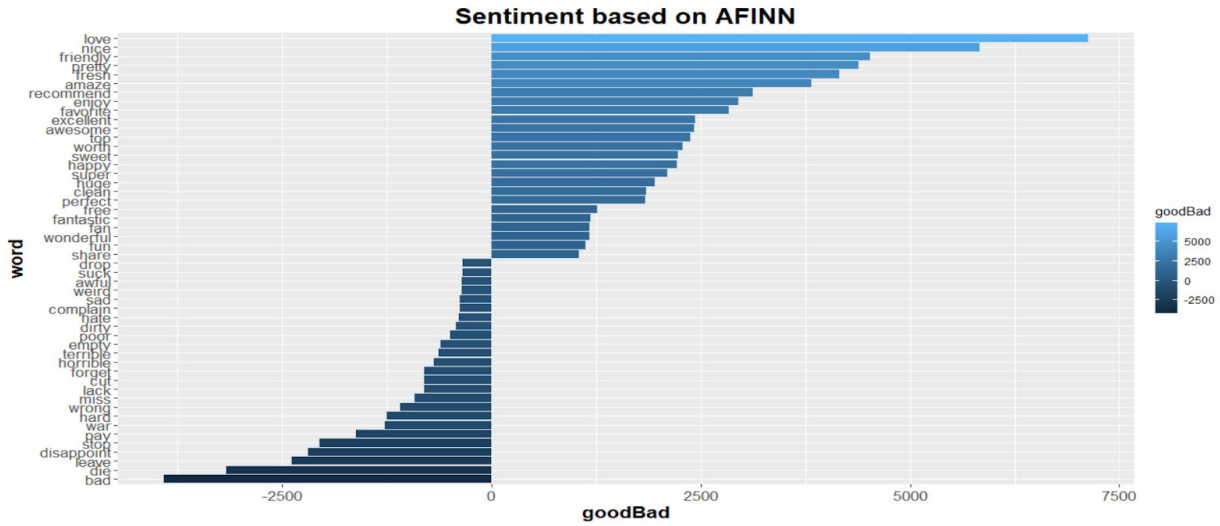
value	count	sumn
-5	1	13
-4	9	927
-3	46	11867
-2	134	17098
-1	67	14476
1	64	20967
2	125	40681
3	57	31282
4	13	7747
5	2	696

The values can be used to create an overall score for the review. This can occur by adding up the values associated with each of the individual words within the review. Then, we can examine the average score for each of the star ratings.

stars <dbl>	avgLen <dbl>	avgSenti <dbl>
1	4.003976	-2.3493326
2	4.313061	0.7138584
3	4.314566	3.1422384
4	4.246678	5.5467348
5	3.963570	6.4570960

Something surprising is that the 2-star reviews have a very slightly positive average score. As expected, though, the average score increases consistently from 1 to 5 star reviews.

Like the other dictionaries, we checked which words were associated with the most positive and negative sentiments. To do so, we designated all words with a value of -1 to -5 as ‘bad’, and all reviews with a value between 1 and 5 as ‘good’. Then, the sentiment analysis is below. One interesting observation is that ‘love’ is the most positive and ‘bad’ is the most negative word across all three of the dictionaries.

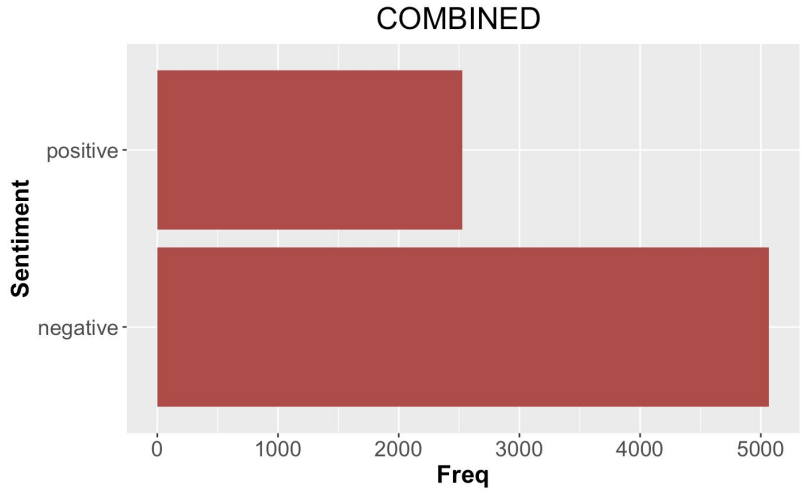


Finally, we completed one more document-term matrix and assigned reviews to the two classes. For AFINN, we removed the 2-star reviews, as their average sentiment score was rather neutral at only 0.71. Then, we assigned the 1-star reviews to the '-1' class, and the 3, 4, and 5-star reviews to the '1' class. This leads to a less balanced dataset than with the other dictionaries, but is the best choice considering the average sentiment score across the different star ratings. The document-term matrix has 28297 rows and 520 columns.

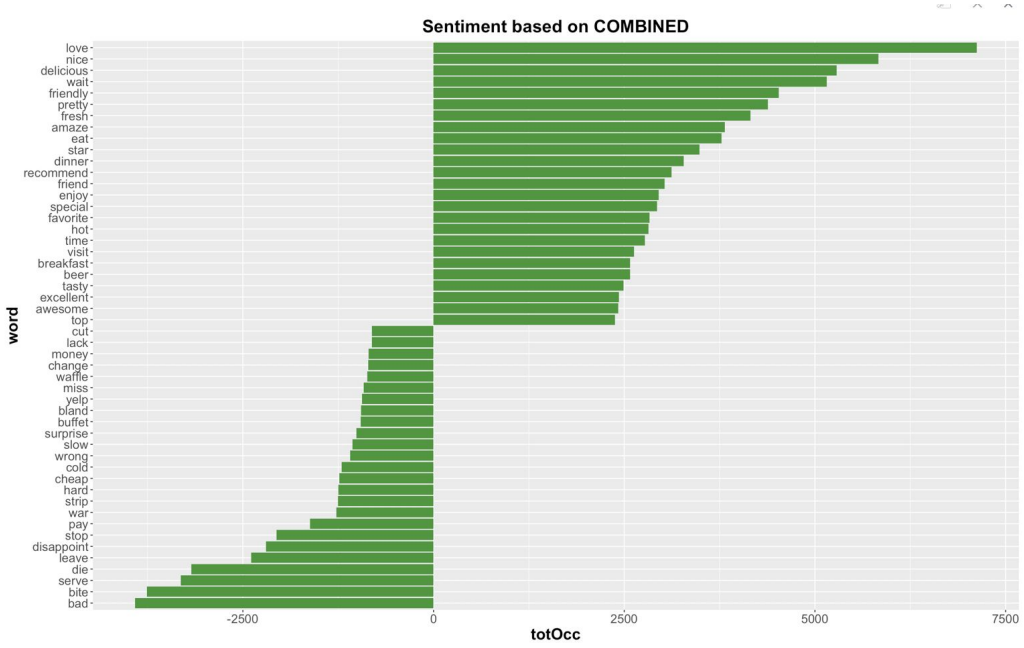
hiLo <dbl>	n <int>
-1	3521
1	24776

4. Combined Dictionary

For the next approach, we combined all three dictionaries based on the inner join into one to see if it will improve the performance of the models. To do this, we combined the words from all three and converted them to a structure similar to the Bing dictionary, where each word has a negative or positive sentiment association. Finally, 5068 words correspond to negative words and the remaining 2527 words represent positive words.



The top 3 most positive words in this combined dictionary are ‘love’, ‘nice’, and ‘delicious.’



The combined dictionary has 1747 matching terms with the review dataset when using an inner join. Then, we recalculated the sentiment score, which leads to the following average sentiment scores distributed across the star ratings. It can be clearly seen that the sentiment score of rating 4 and 5 is higher than other dictionaries since this dictionary obtains more positive words.

stars <dbl>	avgPos <dbl>	avgNeg <dbl>	avgSentiSc <dbl>
1	0.4714587	0.5285413	-0.05708269
2	0.5646776	0.4353224	0.12935526
3	0.6458829	0.3541171	0.29176590
4	0.7351485	0.2648515	0.47029707
5	0.7899933	0.2100067	0.57998668

It can be noted that the 1 star ratings are the only ones with a negative average sentiment score. Therefore, we assigned only the 1-star reviews to the -1 class. We removed the 2-star reviews and then assigned the 3, 4, and 5-star ratings to the 1 class since they have high average sentiment scores. The document-term matrix has 29832 rows and 1749 columns.

hiLo <dbl>	n <int>
-1	3729
1	26103

Model Design

For each dictionary, we inner join the dictionary with the terms in the ratings to form a document-term matrix, which we used to assign the data to two classes. We used tf-idf scores of each word in each review as independent variables.Tf-idf is a statistic which represents how important a word is to a document in a collection of groups. This score is a multiplication of tf and idf scores. We considered both '*frequency of individual terms within a document'* and '*importance of words across documents'* by applying tf-idf score. The importance grows proportionally to the number of times a word appears in the document but it is offset by the frequency of the word in the corpus. Then, we then took a sample from the original datasets to decrease complexity. We used a sample size of 12000. Then, we divided sample data into train and test data at the ratio of 65:35.

We built several models including random forest, generalized linear model and support vector machine. For each model, we present results of the model on train and test dataset.

Models Based on 'Bing' (Inner Join)

1. Random Forest Model

We implemented a random forest model by using different numbers of trees such as ***70, 120, 180*** trees. Then, we realized that there is no significant improvement (*just 0.5%*) when we increase the number of trees so we decided on **120 trees**. The table shows confusion matrix of train dataset (thresh=0.5):

	preds	
actual	FALSE	TRUE
-1	1661	229
1	74	5836

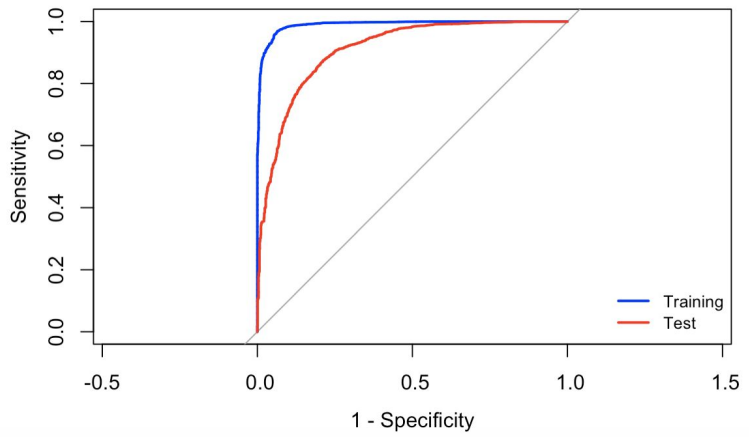
The table shows confusion matrix of test dataset:

	preds	
actual	FALSE	TRUE
-1	639	396
1	152	3013

The table demonstrates performance metrics of test dataset:

Metric <fctr>	Value <fctr>
Accuracy	0.87
Recall	0.952
Precision	0.884

This plot below indicates the ROC curve of the random forest model on train and test dataset. When the blue line represents the model performance on training data, the red line corresponds to test data.



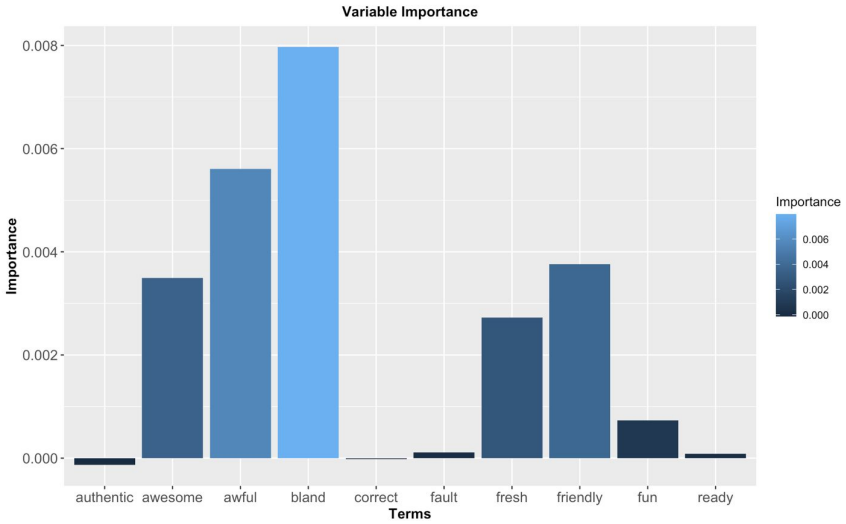
Then, we determined the best threshold (*0.6193*) by using the ROC curve and so created the confusion matrix on test data. It can be clearly seen that a random forest model with the optimal thresholds separates reviews which have different classes much better.

	preds	
actual	FALSE	TRUE
-1	746	289
1	267	2898

The table demonstrates performance metrics of test dataset based on optimal threshold:

Metric <fctr>	Value <fctr>
Accuracy	0.868
Recall	0.916
Precision	0.909

Besides, we checked which variables carried more weights on the model to predict class of review. According to importance specified by random forest, the most 10 significant words comprise both positive and negative words,but the majority of them are related to positive words. (‘bland’, 'fresh', 'friendly','fun','awesome',etc...)



2. Generalized Linear Model

We ran a generalized linear model by using different parameters. When we apply different regularization parameters, we observed that the model with *Lasso* performs slightly better than the model with *Ridge*. Also, we executed this model by using **5 fold** cross validation to avoid overfitting. Then, we selected the best regularized model in the test data. The table demonstrates confusion matrix of the train dataset:

	preds	
actual	FALSE	TRUE
-1	1278	612
1	105	5805

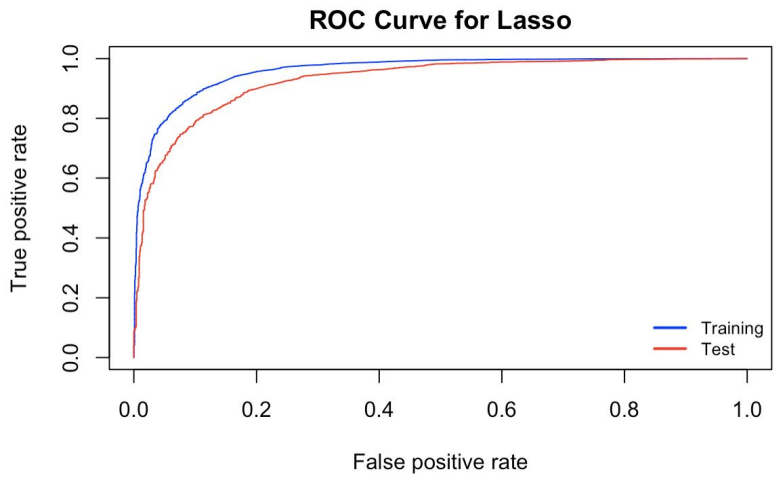
The table demonstrates confusion matrix of the test dataset:

	prediction	
actual	-1	1
-1	613	422
1	115	3050

The table demonstrates performance metrics of the model on test dataset:

Metric <fctr>	Value <fctr>
Accuracy	0.872
Recall	0.964
Precision	0.878

This plot below indicates the ROC curve of the generalized linear model on train and test dataset. When the blue line represents the model performance on training data, the red line corresponds to test data.



3. Naive Bayes Classifier

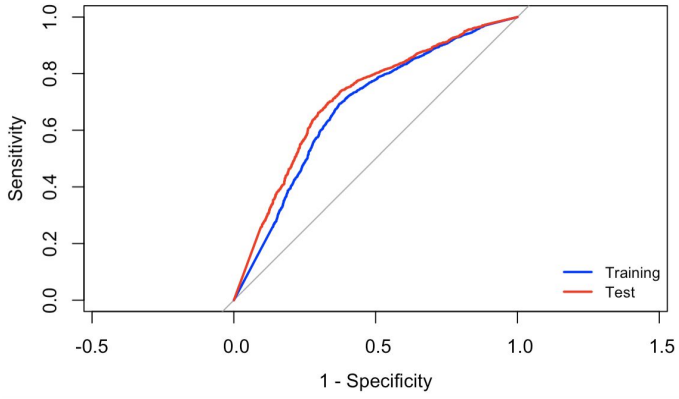
The Naive Bayes Classifier is a simple probabilistic classifier which is based on Bayes theorem. We implemented Naive Bayes classifier with Laplace smoothing. The table demonstrates confusion matrix of the train dataset (thresh=0.5):

	pred	
actual	FALSE	TRUE
-1	1600	290
1	4179	1731

The table demonstrates confusion matrix of the test dataset:

	preds	
actual	FALSE	TRUE
-1	916	119
1	2241	924

This plot below indicates the ROC curve of the Naive Bayes Classifier on train and test dataset. When the blue line represents the model performance on training data, the red line corresponds to test data.



Then, we determined the best threshold () by using the ROC curve and so created the confusion matrix on test data. It can be clearly seen that a Naive Bayes model with the optimal threshold separates reviews which have different classes much better.

actual	preds FALSE	TRUE
-1	685	350
1	945	2220

The table demonstrates performance metrics of the model on test dataset:

Metric <fctr>	Value <fctr>
Accuracy	0.692
Recall	0.701
Precision	0.864

<u>Models Based on 'Bing' (Left Join)</u>

We also run only a random forest model by using a bing dictionary with left join to understand the difference between inner and left join.

1. Random Forest

We implemented a random forest model by using different numbers of trees such as ***50, 80, 120*** trees. Then, we realized that there is <u>no significant improvement</u> (*just 0.6%*) when we increase the number of trees so we decided on **80 trees**. The table shows confusion matrix of train dataset (thresh=0.5):

actual	preds FALSE	TRUE
-1	1724	190
1	58	5829

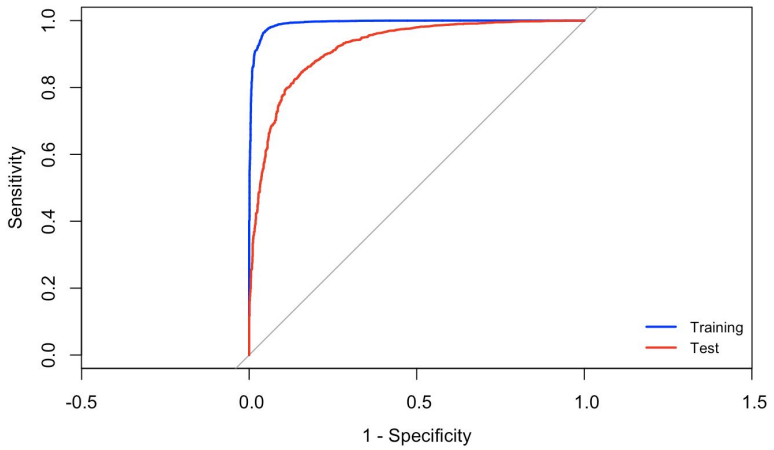
The table shows confusion matrix of test dataset:

actual	preds FALSE	TRUE
-1	702	338
1	162	2998

The table demonstrates performance metrics of test dataset:

Metric <fctr>	Value <fctr>
Accuracy	0.881
Recall	0.949
Precision	0.899

This plot below indicates the ROC curve of the random forest model on train and test dataset. When the blue line represents the model performance on training data, the red line corresponds to test data.



Then, we determined the best threshold by using the ROC curve and the table demonstrates performance metrics of test dataset based on optimal threshold:

Metric <fctr>	Value <fctr>
Accuracy	0.868
Recall	0.903
Precision	0.92

2. Generalized Linear Model

We ran a generalized linear model , but we found that lasso had better performance across all of the performance metrics that we used.Using lasso, and 5 fold cross validation, the resulting training confusion matrix:

```
        preds
actual  FALSE  TRUE
    -1   1233   681
    1     122 5765
```

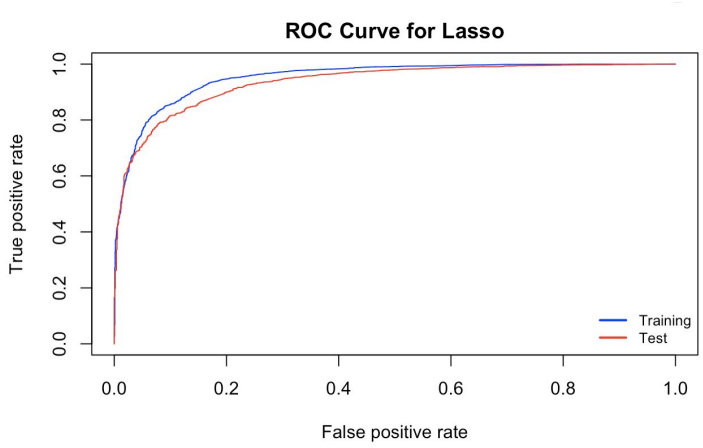
And the testing confusion matrix:

	prediction	
actual	-1	1
-1	617	423
1	101	3059

The performance metrics on the testing data for this model:

Metric <fctr>	Value <fctr>
Accuracy	0.875
Recall	0.968
Precision	0.879

We also plotted the ROC curves for this model for both the training and testing datasets. The blue line represents the training, and the red line the testing data.



It can be clearly seen that there is a small difference between model with the left join and model with inner join. However, when we increase size of dictionary more, there would be vital change

Models Based on 'NRC'

1. Random Forest Model

We once again tried the random forest model with three sizes: **70, 120 and 180** trees in order to compare the performance between the three. The performance between the three is essentially the same, with accuracy differences of only 0.3%, so we selected the **70** trees as the best model because it requires less computation time than the others. The confusion matrix of this model on the training data when using the 0.5 threshold for the cutoff between classes:

actual	preds FALSE	TRUE
-1	1801	221
1	39	5907

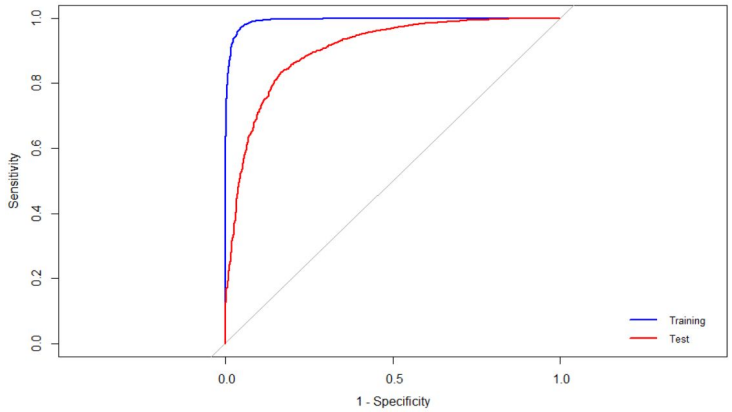
The confusion matrix from this model on the testing data:

actual	preds FALSE	TRUE
-1	605	402
1	169	3113

On the testing data, the model with 70 trees has the following performance metrics:

Metric <fctr>	Value <fctr>
Accuracy	0.865
Precision	0.95
Recall	0.883

In order to determine the best threshold value rather than just using 0.5, we can plot the ROC curve and use that to determine the optimal value to use. We plotted together the ROC curves of the training and testing:



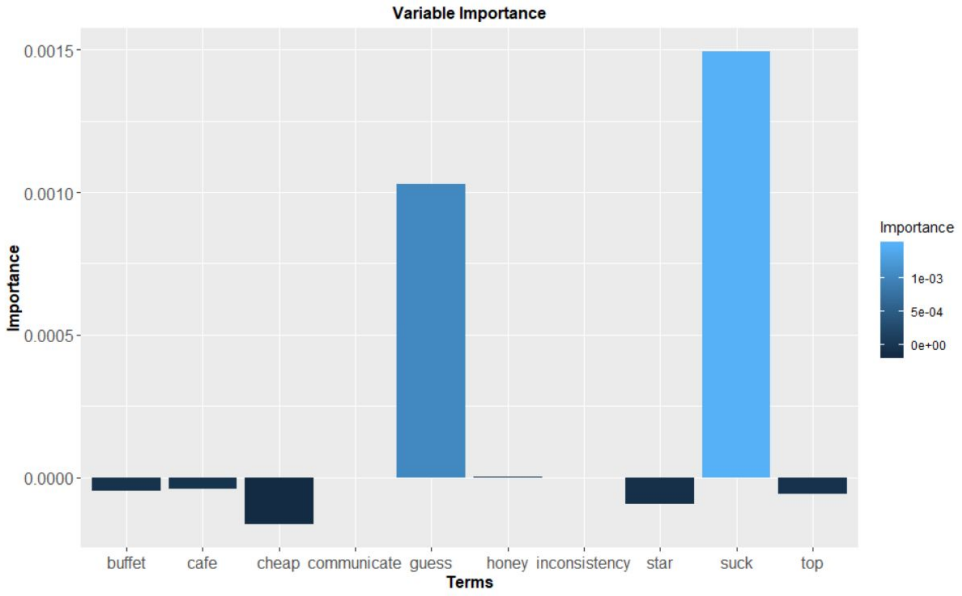
This determines that the optimal threshold value is *0.6356*, so we will use this value as the cutoff in the confusion matrix rather than 0.5 as we did originally. When using the optimal threshold on the testing data, we get the following precision matrix:

actual	preds FALSE	TRUE
-1	752	255
1	359	2923

And the performance metrics change:

Metric <fctr>	Value <fctr>
Accuracy	0.857
Recall	0.891
Precision	0.92

The precision is increased heavily now with the new threshold. Finally, we checked the variable importance for the best random forest model of 70 trees. The most important word is ‘suck.’



2. Generalized Linear Model

We next ran the GLM model on the NRC dictionary. We ran both ridge and lasso, but we found that lasso had better performance across all of the performance metrics that we used. Therefore, this is the best GLM model of the two.

Using lasso, and 5 fold cross validation, the resulting training confusion matrix:

	preds	
actual	FALSE	TRUE
-1	1233	789
1	120	5826

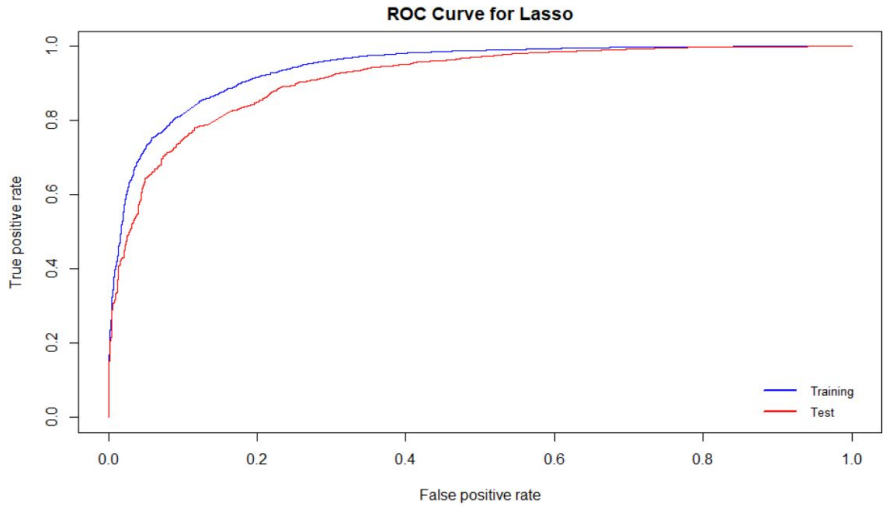
And the testing confusion matrix:

	prediction	
actual	-1	1
-1	548	459
1	124	3158

The performance metrics on the testing data for this model:

Metric <fctr>	Value <fctr>
Accuracy	0.864
Recall	0.962
Precision	0.873

We also plotted the ROC curves for this model for both the training and testing datasets. The blue line represents the training, and the red line the testing data.



3. Naive Bayes Classifier

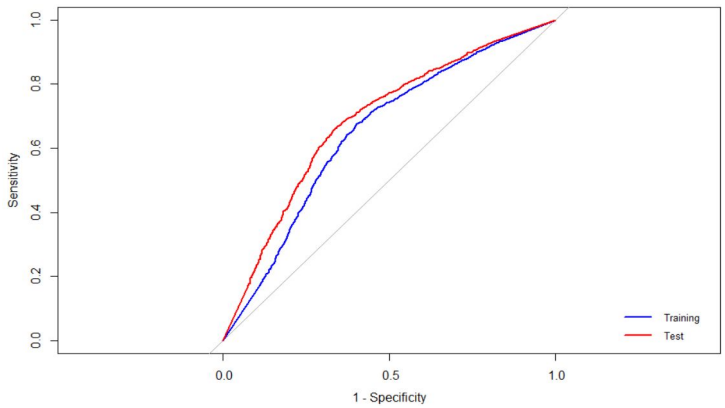
Finally, we ran Naive Bayes as a third model for the NRC dictionary. Using the threshold of 0.5, and including Laplace smoothing, this is the confusion matrix for the training data:

actual	pred FALSE	TRUE
-1	1769	253
1	4792	1154

And for the testing data:

actual	preds FALSE	TRUE
-1	920	87
1	2630	652

Upon looking at the testing confusion matrix, it seems that the model predicts the negative class well, but is not very good as predicting the positive class. We can further observe the ROC curves for training and testing sets, shown in blue and red respectively.



We can use the ROC curve to determine a better threshold in an attempt to improve the performance of the classification. The new confusion matrix for the testing data using the optimal cutoff threshold is:

	preds	
actual	FALSE	TRUE
-1	646	361
1	1058	2224

And the performance metrics:

Metric <fctr>	Value <fctr>
Accuracy	0.669
Recall	0.678
Precision	0.86

<u>Models based on 'AFINN'</u>

1. Random Forest Model

For the AFINN dictionary, we tried three random forest models: **70, 120, and 180** trees in order to compare the performance between the three. The accuracy of all three models is the same, and the precision and recall vary only by 0.1% between the three, so we chose **70** trees as the best.

For the training data, here is the confusion matrix when using a 0.5 threshold:

	preds	
actual	FALSE	TRUE
-1	765	260
1	45	6964

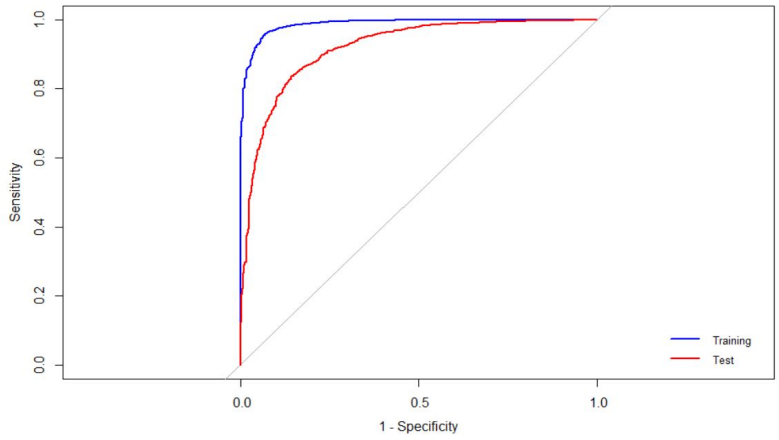
And for the testing data:

	preds	
actual	FALSE	TRUE
-1	260	291
1	65	3710

The performance metrics are very high. This is likely because we have more information from this dictionary than we do with the other ones, as instead of just a sentiment, we have a sentiment score indicating how strong the sentiment is.

Metric <fctr>	Value <fctr>
Accuracy	0.918
Precision	0.983
Recall	0.927

And the ROC curves:



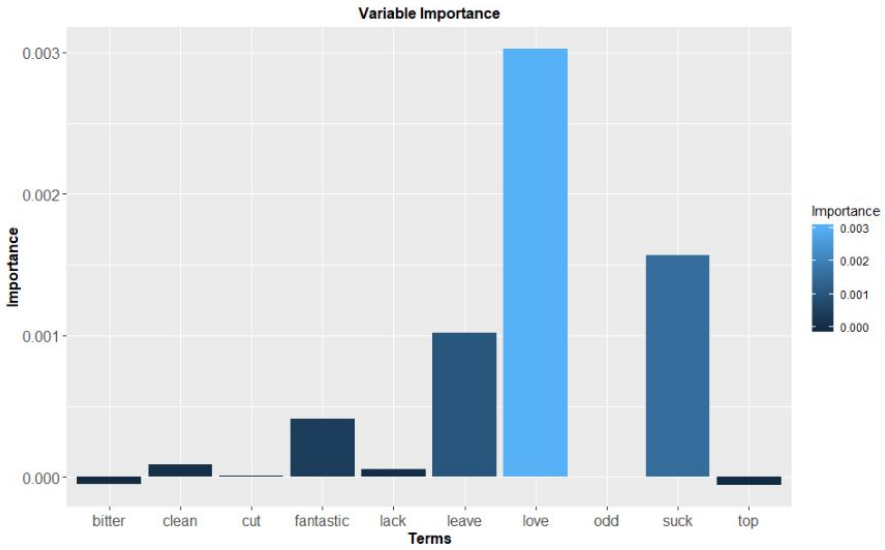
We still can try to improve the performance, though, by using the optimal threshold obtained from the ROC curves, which is *0.747* and changes the testing confusion matrix to be:

	preds	
actual	FALSE	TRUE
-1	384	167
1	273	3502

This improved the precision of the model, resulting in the following metrics:

Metric <fctr>	Value <fctr>
Accuracy	0.898
Recall	0.928
Precision	0.954

We also checked the variable importance for this model, and found that the most important variables were ‘love’, ‘suck’, and ‘leave’.



2. Generalized Linear Model

We also ran a GLM model on the third dictionary. We found again that lasso has better performance than ridge, so we used the lasso regression with 5 fold cross validation. This resulted in the following confusion matrix using the training data.

actual	preds FALSE	TRUE
-1	499	526
1	78	6931

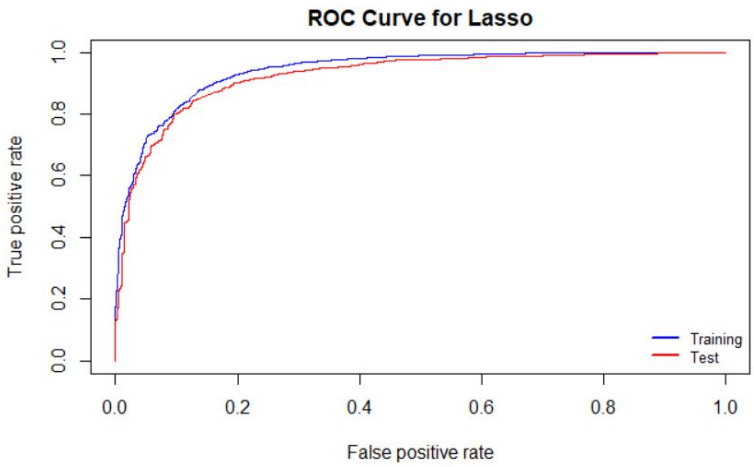
And using the testing data:

actual	prediction FALSE	TRUE
-1	207	344
1	77	3698

On the validation data, the GLM performance metrics for the AFINN dictionary are also very high.

Metric <fctr>	Value <fctr>
Accuracy	0.91
Recall	0.983
Precision	0.92

You can see the corresponding ROC curves for the training model in blue and the testing model in red.



3. Naive Bayes Classifier

Finally, we ran a third model of naive bayes on the AFINN dictionary to compare the performance. We chose to use Laplace smoothing again in order to achieve the best results. The confusion matrix using the training data and a threshold of 0.5:

actual	pred FALSE	TRUE
-1	837	188
1	5101	1908

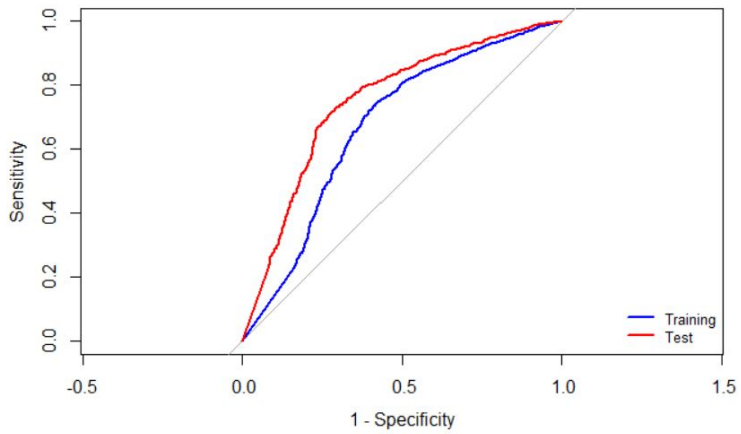
And using the same threshold of 0.5, the confusion matrix for the testing data:

actual	preds FALSE	TRUE
-1	499	52
1	2746	1029

The accuracy and recall of this model using the cutoff threshold value of 0.5 are very low, demonstrating the low performance of the model using this threshold.

Metric <fctr>	Value <fctr>
Accuracy	0.353
Recall	0.273
Precision	0.952

Instead, we can use the ROC curve to determine a better threshold value in order to reclassify some of the cases in the testing set. In the graph below, the ROC curve of the training set is shown in blue and the ROC curve of the testing set in red.



Because the optimal threshold value based on the ROC curve is so different from 0.5, and is in fact *1.195512 e-68*, the new confusion matrix for the testing data using this cutoff is very different from the original one.

actual	preds FALSE	TRUE
-1	378	173
1	970	2805

The correctly identified members of the negative class have gone down slightly which lowers the precision a little bit, but many more occurrences of the positive class are now correctly identified, bringing up the overall accuracy and recall scores a lot.

Metric <fctr>	Value <fctr>
Accuracy	0.736
Recall	0.743
Precision	0.942

<u>Models Based on Combination of Dictionaries</u>

We updated the data by using a combination of dictionaries to train all three models once again: random forest, GLM, and Naive Bayes.

1. Random Forest Model

For random forest, we again trained three individual random forest models using **70, 120 and 180 trees** to compare the performance. We found that the performance metrics (accuracy, precision, and recall) were identical for 70 and 120 trees, and increased only a small amount when using 180 trees. Therefore, we chose **70 trees** as the best model, because the 180 trees is very time intensive to run yet does not lead to much better results (only 0.3% increase in accuracy).

Then, using the 70 trees model and a threshold of 0.5 for the classification, this is the confusion matrix for the training data:

actual	preds FALSE	TRUE
-1	837	172
1	15	7447

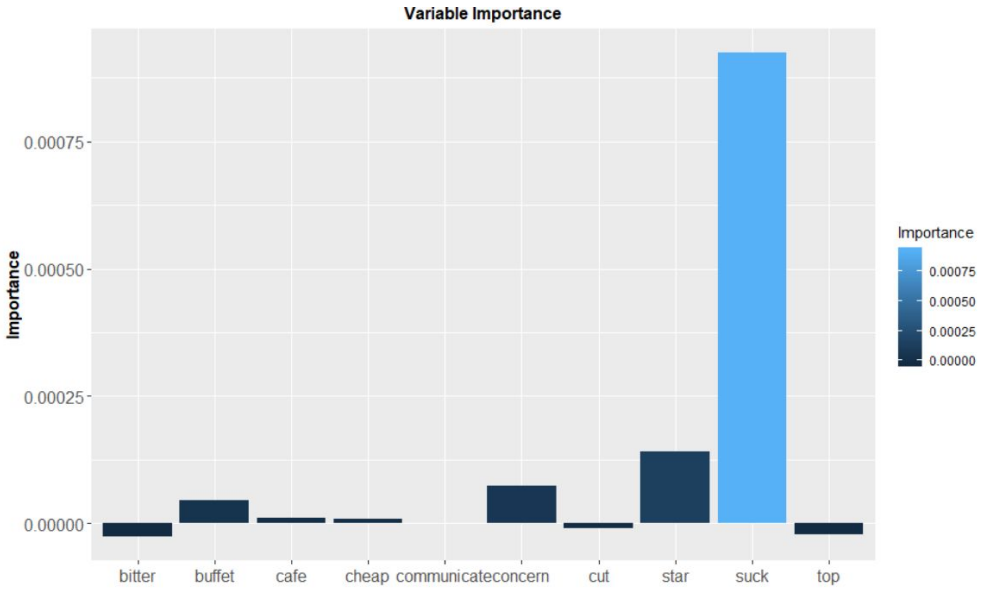
And for the testing data:

actual	preds FALSE	TRUE
-1	212	360
1	47	3941

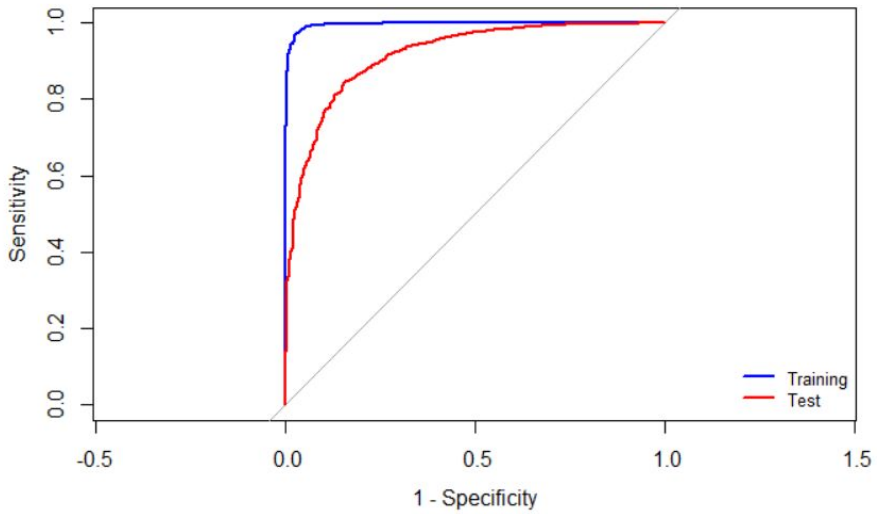
And the performance metrics for this model are very high on the testing data:

Metric <fctr>	Value <fctr>
Accuracy	0.911
Precision	0.988
Recall	0.916

We also checked the variable importance for this random forest model, and we found that the word ‘suck’ is by far the most important word, with words such as ‘star’ and ‘concern’ coming behind.



Next, we wanted to see if we can use a better threshold to improve the performance of the model, even though the performance metrics were already very high. To do this, we used the ROC curves, which are shown below. The ROC curve for training is blue, and the one for testing is red.



We found that the best threshold isn't 0.5 but it is actually *0.7600604,* so we created the testing confusion matrix again using the cutoff with the following result.

actual	preds FALSE	TRUE
-1	399	173
1	295	3693

We can visually see from the confusion matrix that the number of correct -1 class predictions increased, but there are slightly less correct positive class predictions. The data is very imbalanced with few negative class instances to begin with, so this is an improvement to be able to predict the negative class slightly better. You can see this in the performance metrics, because the precision of the model increased.

Metric <fctr>	Value <fctr>
Accuracy	0.897
Recall	0.926
Precision	0.955

2. Generalized Linear Model

Next, we ran GLM on the combined dictionary. We ran both lasso and ridge with 5-fold cross validation, but the performance was better with lasso, so as with the other dictionaries, we used lasso as the best model for GLM. The confusion matrix for the training data:

actual	preds FALSE	TRUE
-1	461	548
1	44	7418

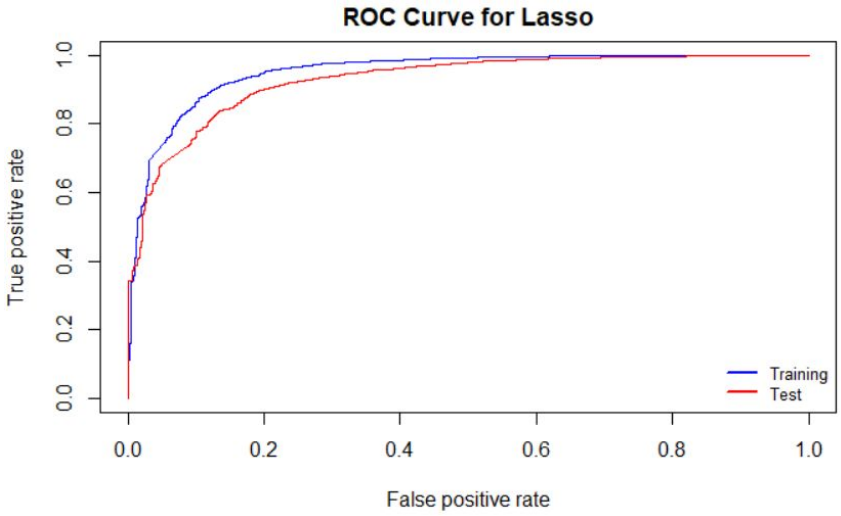
And the testing data:

actual	prediction FALSE	TRUE
-1	226	346
1	49	3939

The performance metrics for the GLM model also seem to be very high when using the combined dictionaries. This makes sense since we have more information from the combined dictionaries than we do from any of the single individual ones, so we would expect for the model trained on the combined dictionaries to outperform the individual dictionaries.

Metric <fctr>	Value <fctr>
Accuracy	0.913
Recall	0.988
Precision	0.919

We also plotted the ROC curves, with the training in blue and the testing in red.



3. Naive Bayes Classifier

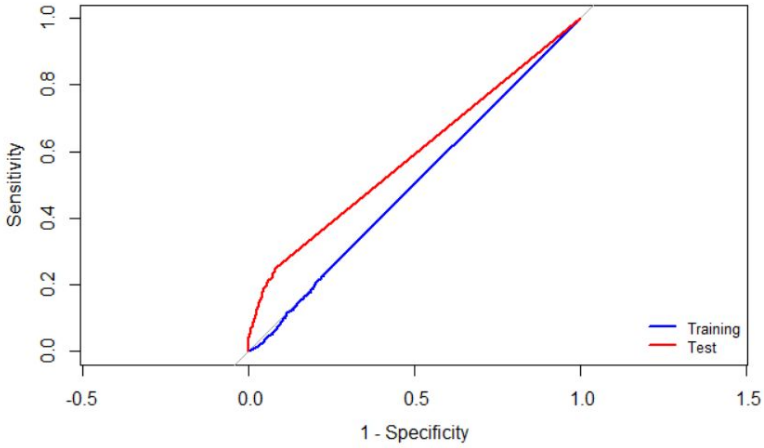
The third model we ran on the combined dictionaries is the Naive Bayes model. We ran the model using Laplace smoothing in order to get the best results, and then obtained the following confusion matrix on the training dataset when using a threshold of 0.5.

actual	pred FALSE	TRUE
-1	975	34
1	7331	131

And using the same threshold on the testing data:

actual	preds FALSE	TRUE
-1	572	0
1	3914	74

Looking at the ROC curves, the testing curve is in red and the training in blue.



We can use the ROC curve to find a better threshold than 0.5, and see if this can increase the performance of the model. The ROC curve determined that the best threshold to use for the cutoff is *3.259095e-307,* which is very different from 0.5. If we use this threshold instead, the confusion matrix for testing changes:

actual	preds FALSE	TRUE
-1	526	46
1	3011	977

The false negative count is slightly better, but as a sacrifice the precision is no longer perfect.

Metric <fctr>	Value <fctr>
Accuracy	0.33
Recall	0.245
Precision	0.955

Models Based on Broader Set of Terms

We follow exactly the same approach in which we built in the previous step (removed stopwords and performed a lemmatization process). *Lemmatization* is the process of grouping together the diverse inflected forms of a word so they can be analyzed as a single item. Hence, lemmatization increases efficiency of text analysis. Besides, the words whose occurrence are more than **3500** and less than **40** are filtered out before creating the document-term matrix. (in total, 2721 words exist in the matrix). We created a document-term matrix whose dimension is 33438 2723.(This is significantly larger than other models).

Then, we take a sample from the original datasets created to decrease complexity. We use a sample sample size of 12000. After, we divide sample data into train and test data at the ratio of 65:35 and develop the same models such as Random Forest, Generalized Linear Model and Naive Bayes Classifier. We apply a **wider range of words** to understand whether there is a difference between a limited set of words (dictionary terms) and a broader set of words.

1. Random Forest Model

We again tried the random forest model with three sizes: **70, 120 and 180** trees in order to compare the performance between the three models. The performance result between the three is essentially the same, with accuracy differences of only 0.1%, so we selected the **70** trees as the best model because the model has less time complexity. The confusion matrix of this model on the training data when using the 0.5 threshold for the cutoff between classes:

	preds	
actual	FALSE	TRUE
-1	1860	51
1	6	5883

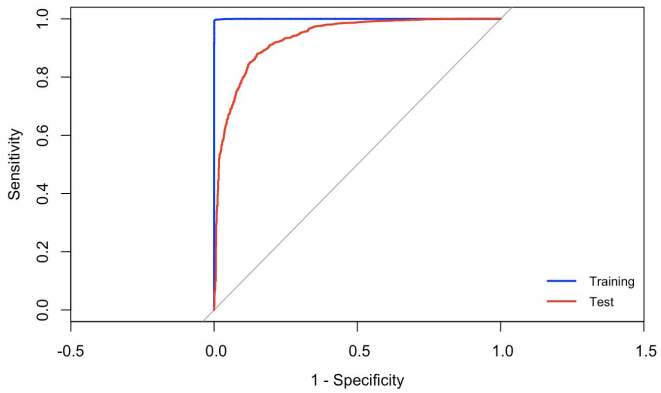
The confusion matrix from this model on the testing data:

	preds	
actual	FALSE	TRUE
-1	661	335
1	105	3099

On the testing data, the model with 70 trees has the following performance metrics:

Metric <fctr>	Value <fctr>
Accuracy	0.895
Recall	0.967
Precision	0.902

In order to determine the optimal threshold value rather than just using 0.5, we can draw the ROC curve and use that to decide the optimal value (0.584) to use. We plotted together the ROC curves of the training and testing:



The confusion matrix of testing data based on optimal threshold:

actual	preds FALSE	TRUE
-1	751	245
1	216	2988

This improved the precision of the model, resulting in the following metrics:

Metric <fctr>	Value <fctr>
Accuracy	0.89
Recall	0.933
Precision	0.924

2. Generalized Linear Model

We also ran a GLM model on the broader set of terms. We observed that lasso has better performance than ridge, so we used the '*lasso'* regression with 5 fold cross validation. This resulted in the following confusion matrix using the training data.

actual	preds FALSE	TRUE
-1	751	245
1	216	2988

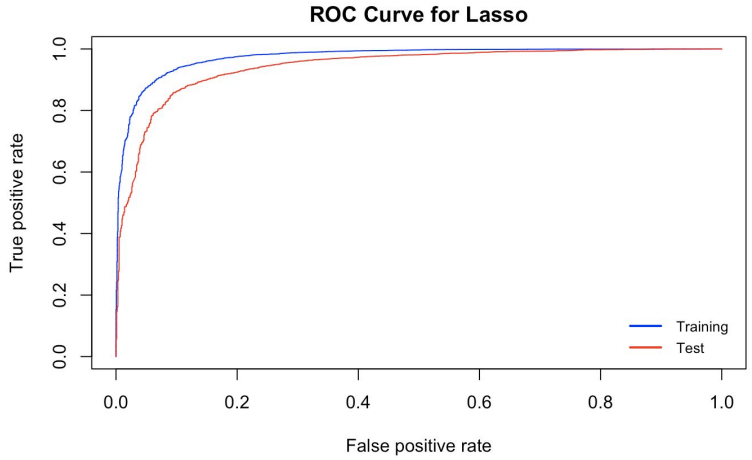
The table demonstrates confusion matrix of the test dataset:

actual	prediction -1	1
-1	671	325
1	112	3092

The table demonstrates performance metrics of the model on test dataset:

Metric <fctr>	Value <fctr>
Accuracy	0.896
Recall	0.965
Precision	0.905

You can see the corresponding ROC curves for the training model in blue and the testing model in red.



3. Naive Bayes Classifier

Lastly, we ran Naive Bayes to compare the performance with other models on a broader set of terms. We chose to use Laplace smoothing again in order to achieve the best results. The confusion matrix using the training data and a threshold of 0.5:

	pred	
actual	FALSE	TRUE
-1	1696	215
1	4480	1409

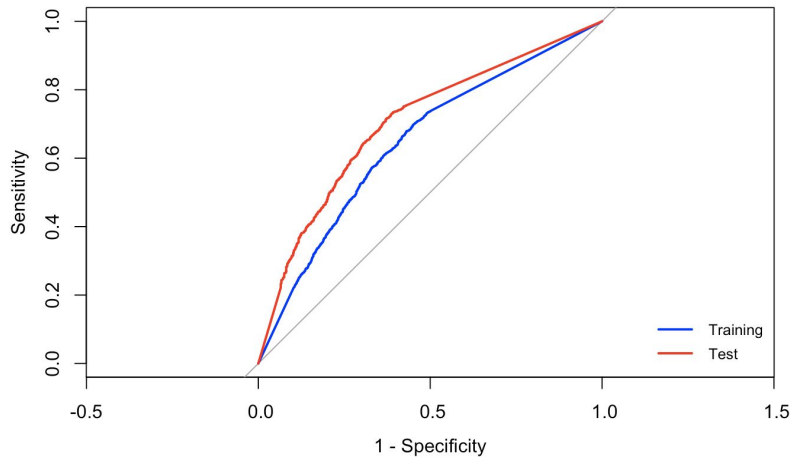
And using the same threshold of 0.5, the confusion matrix for the testing data:

	preds	
actual	FALSE	TRUE
-1	929	67
1	2428	776

The accuracy and recall of this model using the cutoff threshold value of 0.5 are very low, demonstrating the low performance of the model using this threshold.

Metric <fctr>	Value <fctr>
Accuracy	0.406
Recall	0.242
Precision	0.921

Instead, we can use the ROC curve to determine a better threshold value in order to modify some of the cases in the testing set. In the graph below, the ROC curve of the training set is shown in blue and the ROC curve of the testing set in red.



Because the optimal threshold value based on the ROC curve is so different from 0.5, and the new confusion matrix for the testing data using the optimal cutoff is very different from the original one.

	preds	
actual	FALSE	TRUE
-1	637	359
1	970	2234

The correctly identified members of the negative class have gone down slightly which lowers the precision a little bit, but many more occurrences of the positive class are now correctly identified, bringing up the overall accuracy and recall scores a lot.

Metric <fctr>	**Value** <fctr>
Accuracy	0.684
Recall	0.697
Precision	0.862

Performance Evaluation

We worked on a text mining problem and implemented 3 different algorithms in this dataset. Then, we computed several performance evaluation metrics to compare models:

- Accuracy
- Precision
- Recall
- F1 score

Models	Metrics	BING	NRC	AFINN	COMBINED	BROADER
Random Forest	*Accuracy*	86.8%	85.7%	89.8%	89.7%	89.0%
	Recall	91.6%	89.1%	92.8%	92.6%	93.3%
	Precision	90.9%	92.0%	95.4%	95.5%	92.4%
	F1 score	91.2%	90.5%	94.08%	94.02%	92.8%
Generalized Linear Model	*Accuracy*	87.2%	86.4%	91.0%	91.3%	89.6%
	Recall	96.4%	96.2%	98.3%	98.8%	96.5%
	Precision	87.8%	87.3%	92.0%	91.9%	90.5%
	F1 score	91.9 %	91.5%	95.0%	95.22%	93.4%
Naive Bayes Classifier	*Accuracy*	69.2%	66.9%	73.6%	33.0%	68.4%
	Recall	70.1%	67.8%	74.3%	24.5%	69.7%
	Precision	86.4%	86.0%	94.2%	95.5%	86.2%
	F1 score	77.4%	75.8%	83%	39%	77%

Since the data is so imbalanced, instead of accuracy, we considered F1 score which takes into account both the precision and the recall using the following formula:

$$F1 = (2 * Precision * Recall)/(Precision + Recall)$$

Of the three individual dictionaries, the *'AFINN' and 'combined' dictionary* provides the best performance consistently across all three models: random forest, generalized linear model, and naive bayes classifier. These dictionaries have the highest performance metrics for all three models and for all three performance measures that we calculated: accuracy, precision, and recall, with the exception of the Naive Bayes classifier for the combined dictionary which did not perform as well as the others.

Conclusion

In this context, we analyzed Yelp reviews related to restaurants and applied 4 different dictionaries to a sentiment score of the document such as Bing, Afinn, Nrc and combined. Besides, we built Random forest, Generalized Linear model and Naive Bayes classifier. Overall, the *generalized linear model* performs slightly better than others. Performance of the models with *Afinn dictionary* is much better than other dictionaries.

As a next step, we are going to focus other dictionaries to improve performance of the model. We concentrate on improving Naive Bayes classifier. Besides, we will try to work on multi classification (positive, negative and neutral) since many words do not make a significant difference to differentiate positives from negatives.