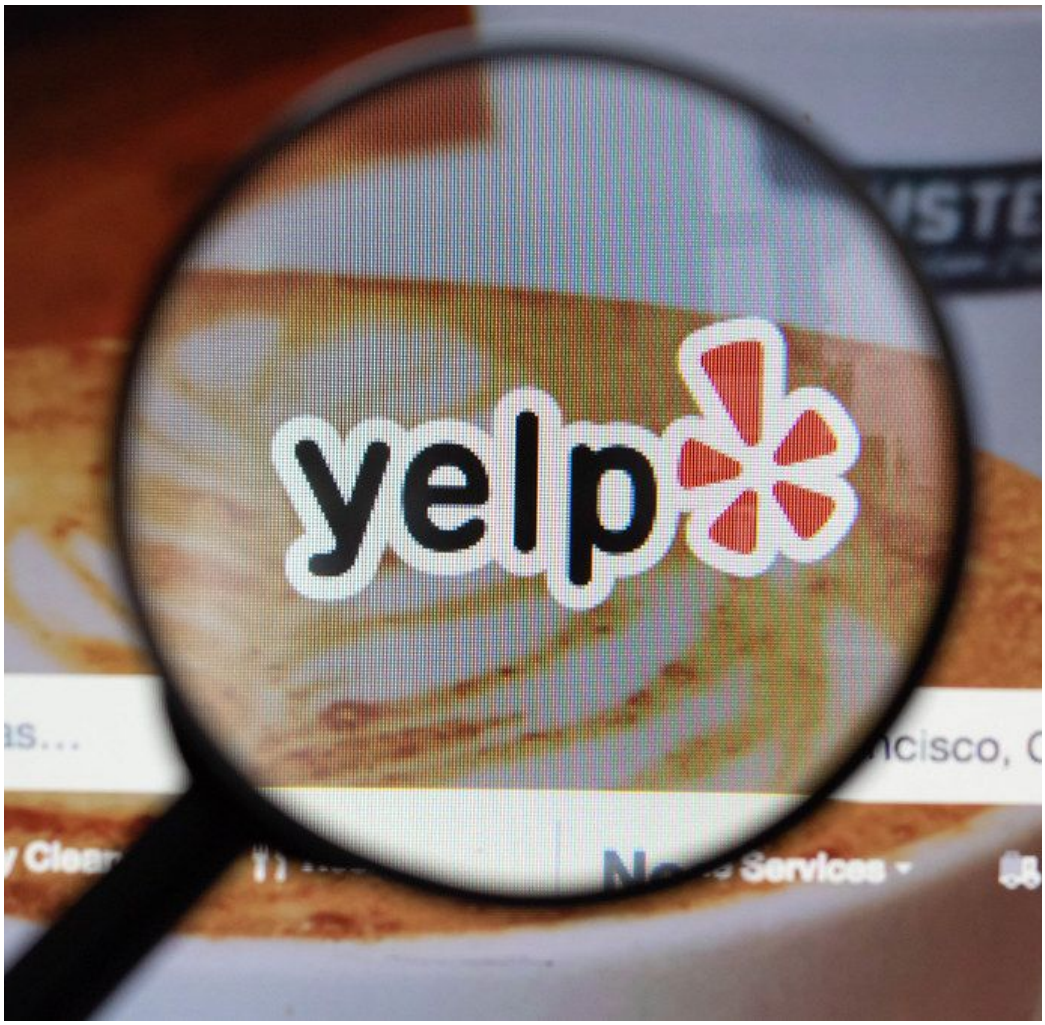
Text Mining

ASSIGNMENT 4

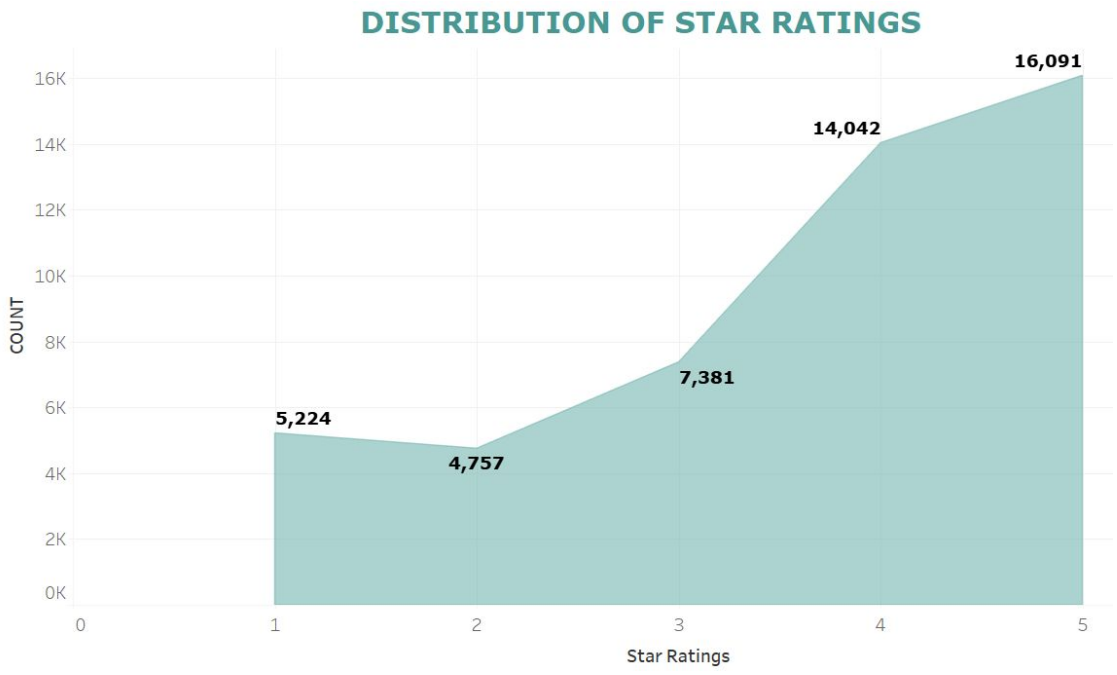


Honey Salve | 669167842

a) Explore the data. How are star ratings distributed?

The given dataset consists of 47,495 user reviews submitted on Yelp.com.
The star ratings can be described as factors from 1 to 5. The distribution is as follows :

Stars	Total	Percentage
1	5,224	10.99%
2	4,757	10.01%
3	7,381	15.54%
4	14,042	29.56%
5	16,091	33.87%



Distribution of star ratings:

- Range - 1 to 5
- Maximum number of star ratings - 4.60 and 5.00 = 14124
- Minimum number of star ratings - 1.00 and 1.40 = 4525

Descriptive Statistics:

Star ratings	1	2	3	4	5
Count	4525	4692	7142	13286	14124

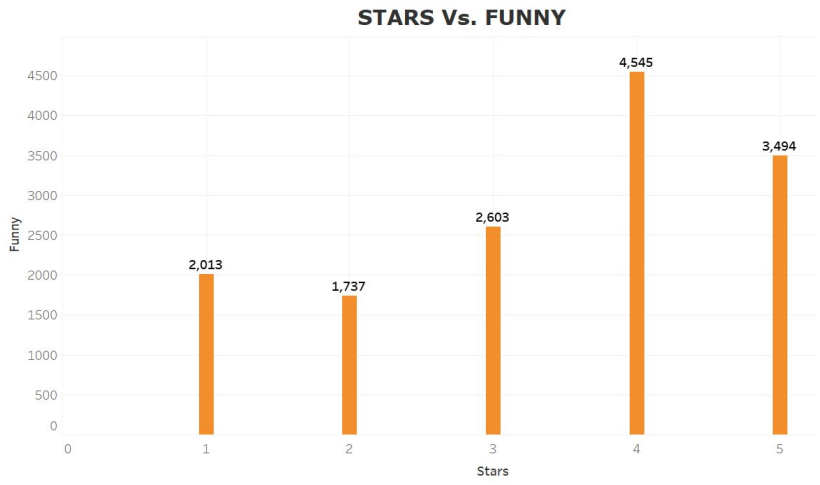
Minimum value	Maximum value	Mean	Median	Standard Deviation
1	5	3.653	4	1.328

Considering the above statistics, we'll be taking anything over 3.65 as positive and others as negative reviews.

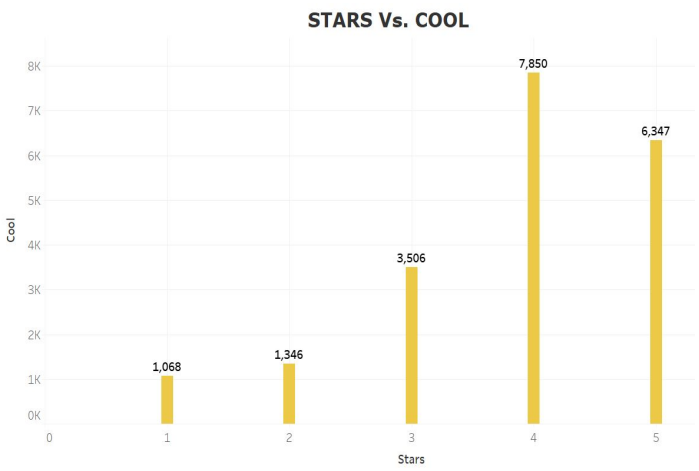
How will you use the star ratings to obtain a label indicating 'positive' or 'negative' – explain using the data, graphs, etc.?

Higher ratings (4 and 5) generally give out a positive feeling about the review. These values constitute of roughly 63% of the dataset. About 15% of the reviews have a rating of '3', which can be said to carry a neutral sentiment. Reviews with lower ratings (1 and 2) can be said to have a negative sentiment. These constitute of about 21% of the total number of reviews.

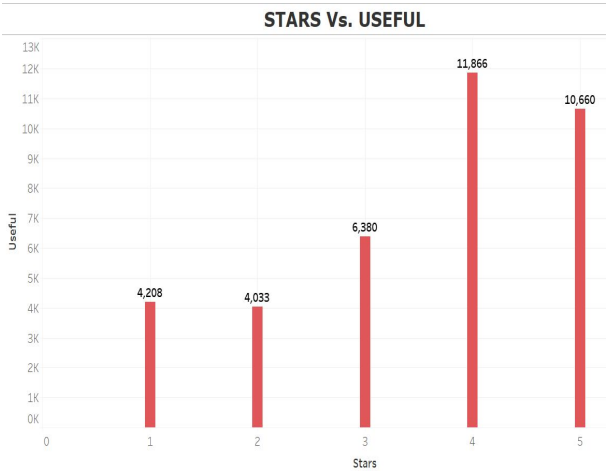
Does star ratings have any relation to 'funny', 'cool', 'useful'? (is this what you expected?)
Our Hypothesis : Since the words 'funny', 'cool', and 'useful' are generally used in a positive context, reviews with higher ratings will have more of these words.



This graph shows how the ratings and reviews with funny words are related.



This graph shows how the ratings and reviews with cool words are related.



This graph shows how the ratings and reviews with useful words are related.

b) What are some words indicative of positive and negative sentiment?

We used the average star rating for a word based on star rating for the reviews in order to determine which words convey positive/negative sentiments.

For this we will be performing the following steps in order to find out the average ratings:

- Read the data, convert it from 'nominal' to 'text', use "process document from data" to tokenize, transform cases, filter stopwords, and filtering tokens. In this, the pruning is set to 1% below level and 98% through above.
- We used the process shown below to get the average % rating.

word	in documents	total	att_1 ↓
und	1293	4397	1.612
pho	894	1846	1.255
pizza	3701	7716	1.227
die	1668	3770	1.109
sushi	1896	3717	1.095
der	980	2483	1.085
ist	929	2014	1.072
sehr	799	1518	1.056
das	935	2287	1.044
burger	2373	4638	1.018
thai	1192	2096	0.985
ich	641	1415	0.903
indian	671	1028	0.901
wings	1090	1771	0.897
tacos	1580	2393	0.874

Positive Words

word	in documents	total	att_1 ↑
worst	991	1073	0.282
horrible	852	961	0.313
terrible	880	991	0.337
disappointing	641	670	0.349
poor	629	696	0.369
paid	632	664	0.373
rude	725	826	0.376
mediocre	771	809	0.380
sorry	525	544	0.387
bland	1287	1431	0.388
waited	1015	1198	0.389
nnoverall	847	853	0.392
understand	649	689	0.393
told	1703	2064	0.395
unfortunately	763	798	0.397

Negative Words

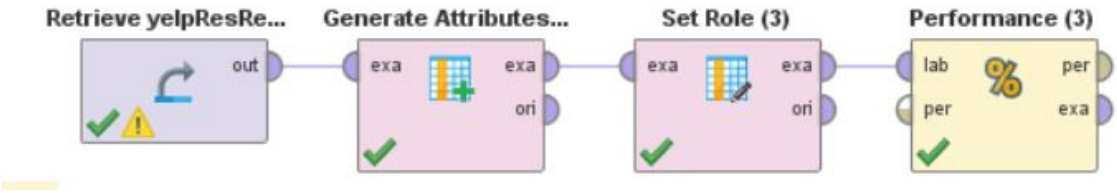
Do these ‘positive’ and ‘negative’ words make sense in the context of user reviews?

From the output of the process above, we can conclude that the words do have a direct correlation with categorization of ‘positive’ or ‘negative’ words. For example, the words that had a lower score tend to be connected to negative values, as compared to the positive reviews, which had relation to higher scores.

c) How many matching terms are there for each of the dictionaries?

We used the process as a guideline and determined the matching words for the dictionaries. Comparing the positive and negative.

- We read the three files (negative words, positive words and the yelp responses)
- Then we assigned weights and set target where the document will go; as well as assign target variables by setting it as a label.
- Then, we identify matching words with yelp response words, based on the available positive and negative dictionaries. We excluded the words that did not match.
- Then new attributes are generated, which are then aggregated by the individual scores (calculated by using term frequency) of matching positive and negative words
- We then merged the files (negative and positive) and stored it into separate sources to run its matrix. Shown as follows :



Sentiment Lexicon Dictionary:
Word match:

- 107 variables - 32 negative, 75 positive

Here, we considered ratings 4 and 5 as ‘1’ and ratings 1 and 2 as ‘0’, whereas rating 3 was considered as ‘neutral’.

accuracy: 68.44%

	true neutral	true 0	true 1	class precision
pred. neutral	0	0	0	0.00%
pred. 0	1491	5488	2893	55.59%
pred. 1	6292	4981	28465	71.63%
class recall	0.00%	52.42%	90.77%	

We got an accuracy of 68.44%. However, this accuracy is considering the the neutral rating, which tends to reduce it. Considering only positive and negative classifications, we can get a much higher accuracy, as shown in the calculation below :

= (28,365 + 5,488) / (5,488 + 28,465 + 2,893 + 4,981) = 81%

Harvard Dictionary :
The process for Harvard dictionary is similar to that of the Lexicon dictionary. However, words need to be converted into lower case for them to be read properly.

Word match:

- 47 variables - 19 negative, 28 positive

accuracy: 55.15%

	true neutral	true 0	true 1	class precision
pred. neutral	0	0	0	0.00%
pred. 0	4538	8577	12576	33.39%
pred. 1	3245	1892	18782	78.52%
class recall	0.00%	81.93%	59.90%	

We get a total accuracy of 55.15%, but again, this is reduced due to the fact that the 'neutral' sentiment is being considered (like the previous case).

Accuracy without 'neutral' = (18,782 + 8,577) / (8,577 + 12,576 + 18,782)
= 65.5%

AFINN Dictionary :

accuracy: 66.40%

	true neutral	true 0	true 1	class precision
pred. neutral	0	0	0	0.00%
pred. 0	1245	4537	2952	51.95%
pred. 1	6538	5932	28406	69.49%
class recall	0.00%	43.34%	90.59%	

Again, as we consider rating '3' as 'neutral', we get a slightly reduced accuracy.

Accuracy without 'neutral' = (38,406 + 4,537) / (28,406 + 2,952 + 4,537 + 5,932)
= 78.8%

Does any dictionary perform better?

We see that the Sentiment Lexicon dictionary performs better as compared to the other models. It has an overall accuracy of 81% for positive and negative sentiments.

It also has a higher accuracy when we consider ‘neutral’ / rating ‘3’.

ii) Compare this approach with the use of SentiWordNet. Describe how you use SentiWordNet.

We’ve divided the dataset into 5 different sets of 10,000 each and created different .csv files as inputs to the SentiWordNet process block.

Performance dataset 1:

accuracy: 74.29%

	true positive	true negative	class precision
pred. positive	316	56	84.95%
pred. negative	70	48	40.68%
class recall	81.87%	46.15%	

Performance dataset 2:

accuracy: 79.19%

	true positive	true negative	class precision
pred. positive	153	20	88.44%
pred. negative	26	22	45.83%
class recall	85.47%	52.38%	

Performance dataset 3:

accuracy: 77.50%

	true positive	true negative	class precision
pred. positive	111	12	90.24%
pred. negative	24	13	35.14%
class recall	82.22%	52.00%	

Performance dataset 4:

accuracy: 86.73%

	true positive	true negative	class precision
pred. positive	152	10	93.83%
pred. negative	16	18	52.94%
class recall	90.48%	64.29%	

Performance dataset 5:

accuracy: 71.21%

	true positive	true negative	class precision
pred. positive	40	15	72.73%
pred. negative	4	7	63.64%
class recall	90.91%	31.82%	

d)

i) Develop models using only the sentiments dictionary terms (you can try individual dictionaries or combine all dictionary terms).

For the modeling process, we use the following :

- We use Lasso Regression, Naive Bayes classification and SVM with minimum advised threshold for the Lexicon Sentiment dictionary, as it gave us the best performance.
- We also run Lasso regression models with Harvard and AFINN dictionaries, as it has the best performance with the Lexicon Sentiment dictionary.
- We split the data in a 70:30 ratio.
- We haven't considered the 'neutral' / rating '3' cases.
- We have roughly 10,000 words examples.

The results of the models are summarised below :

Dictionary used	Models Used	Accuracy	Class 1 Recall
Lexicon Sentiment Dictionary	**Lasso Regression**	77.68%	90.77%
	Naïve Bayes	76.17%	79.31%
	SVM	74.51%	87.68%
Harvard Dictionary	**Lasso Regression**	72.33%	93.55%
AFINN Dictionary	**Lasso Regression**	74.99%	90.32%

Lasso Logistic Regression with Lexicon Dictionary gives us an overall accuracy of about 77%.

Do you use term frequency, tf-idf, or other measures? What is the size of the document-term matrix?

TF - IDF (Term Frequency – Inverse Document Frequency) gives us the importance of a particular term across the reviews (documents). The term frequency is a measure of the importance of a word within a document. It'll help to reduce the effect of words that are repetitive in one review. The document term matrix of (10,418 * 148).

ii) Develop models using a broader list of terms – how do you obtain these terms? Will you use stemming here?

To obtain a broader list of terms, we combined the three dictionaries available.
We combine the multiple .csv files and use stemming to reduce the words to their root words, which makes sentiment analysis more accurate.

Results :

Models Used		Accuracy	Class 1 Recall
Logistic Lasso Regression	**Training**	75.57%	91.88%
	Testing	74.46%	92.15%

Naïve Bayes Classification	**Training**	72.73%	83.29%
	Testing	71.07%	82.84%

The broader dictionary of words had similar results as the Lexicon Sentiment dictionary's results. Logistic Lasso Regression has better performance and overall accuracy. However, the Naive Bayes model gives better true positive and true negative recalls, and can be considered.

Report on the performance of the models. Compare performance with that in part (c) above. For models in (i) and(ii): Do you use term frequency, tf-idf, or other measures, and why? Do you prune terms, and how (also, why?). What is the size of the document-term matrix?

For our models so far :

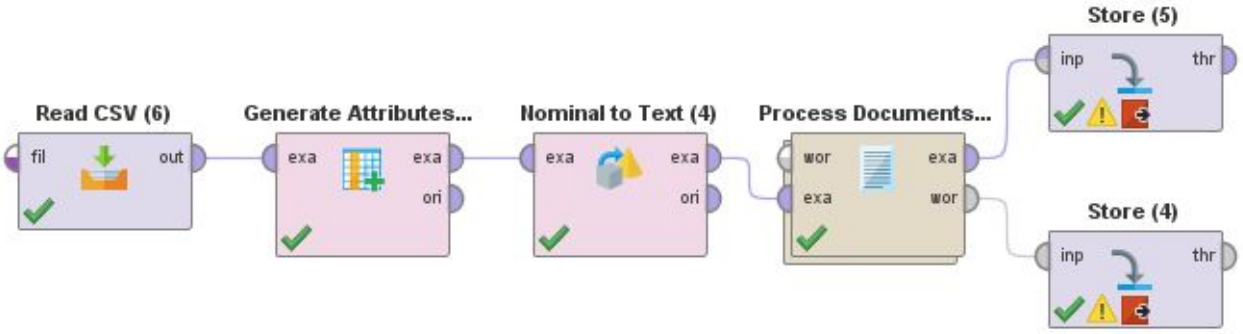
- We used TF - IDF in our analyses
- We used different document - term matrices.

Method	Overall Accuracy	Positive Sentiment Accuracy	Negative Sentiment Accuracy
Lexicon Sentiment Dictionary	81.0%	90.5%	52.4%
Lexicon Sentiment Dictionary - Lasso Logistic Regression	77.7%	90.8%	55.2%
Lexicon Sentiment Dictionary - Naïve Bayes Classification	76.6%	77.8%	74.4%
Stemmed Document - Logistic Regression	74.5%	92.2%	44.1%
Stemmed Document - Naïve Bayes Classification	71.1%	82.8%	50.9%

As we can see, Lexicon Sentiment dictionary (model from (c)) has the best overall accuracy.

Which of the two approaches do you find to be more efficient in terms of processing time?

We carried out the below steps to conclude this:



Process Documents from data with stemming:

The second approach lags in terms of processing time when compared to the first one.