Text Mining and Sentiment Analysis of Yelp Reviews

-Sheethal Heremagalur Sridhar

We have analyzed the data and found out that the Yelp dataset contains 46,978 examples and 26 variables that includes:

1) Type of business
2) Reviews
3) Star Ratings
4) Whether the customer found the comments useful, funny or cool

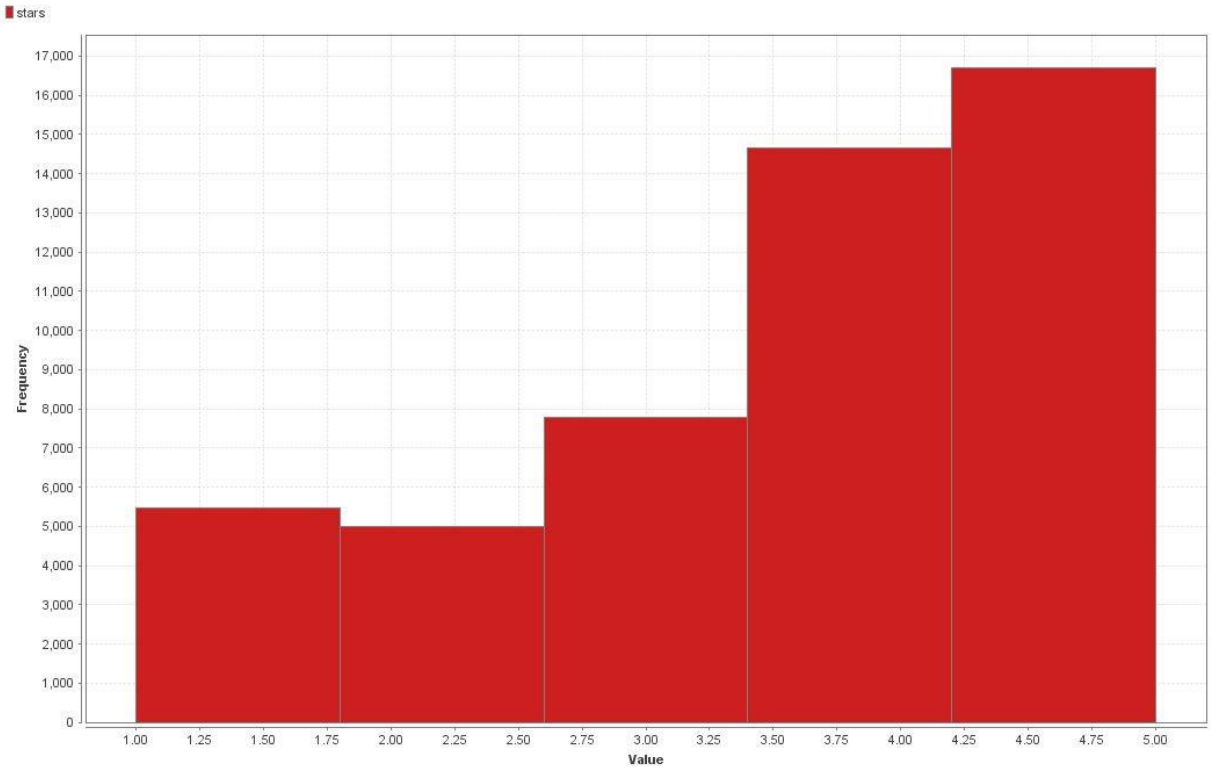
ExampleSet (46,978 examples, 0 special attributes, 26 regular attributes)

Row No.	business_id	cool	date	funny	review_id	stars	text	type	useful	user_id
1	4uiijOUDzc-...	0	Sep 15, 2009	0	-K5z7DzXH...	3	We came he...	review	2	C0jquh-km...
2	4uiijOUDzc-...	2	Sep 6, 2010	1	N42b2u6YS...	4	My boyfrien...	review	2	GDeoUHALg...
3	4uiijOUDzc-...	1	Jul 28, 2010	0	3r40NTxUZ...	4	***for loung...	review	0	d1sLYlfSzHo...
4	4uiijOUDzc-...	0	Feb 13, 2015	0	XQaK1CiSsJ...	1	A Superficial...	review	0	AbC6lu_5Rr...
5	4uiijOUDzc-...	1	Feb 2, 2010	0	Ulb1lFkd2Ej...	2	We stumble...	review	1	wDdh7cLIFK...
6	4uiijOUDzc-...	0	Dec 4, 2009	0	H8SsK_SsLZ...	1	What a wast...	review	0	jljC8F9Xiz7...
7	4uiijOUDzc-...	0	Apr 22, 2011	0	ws3sKm5Mr...	2	Not worth th...	review	0	HvquD8Qal...

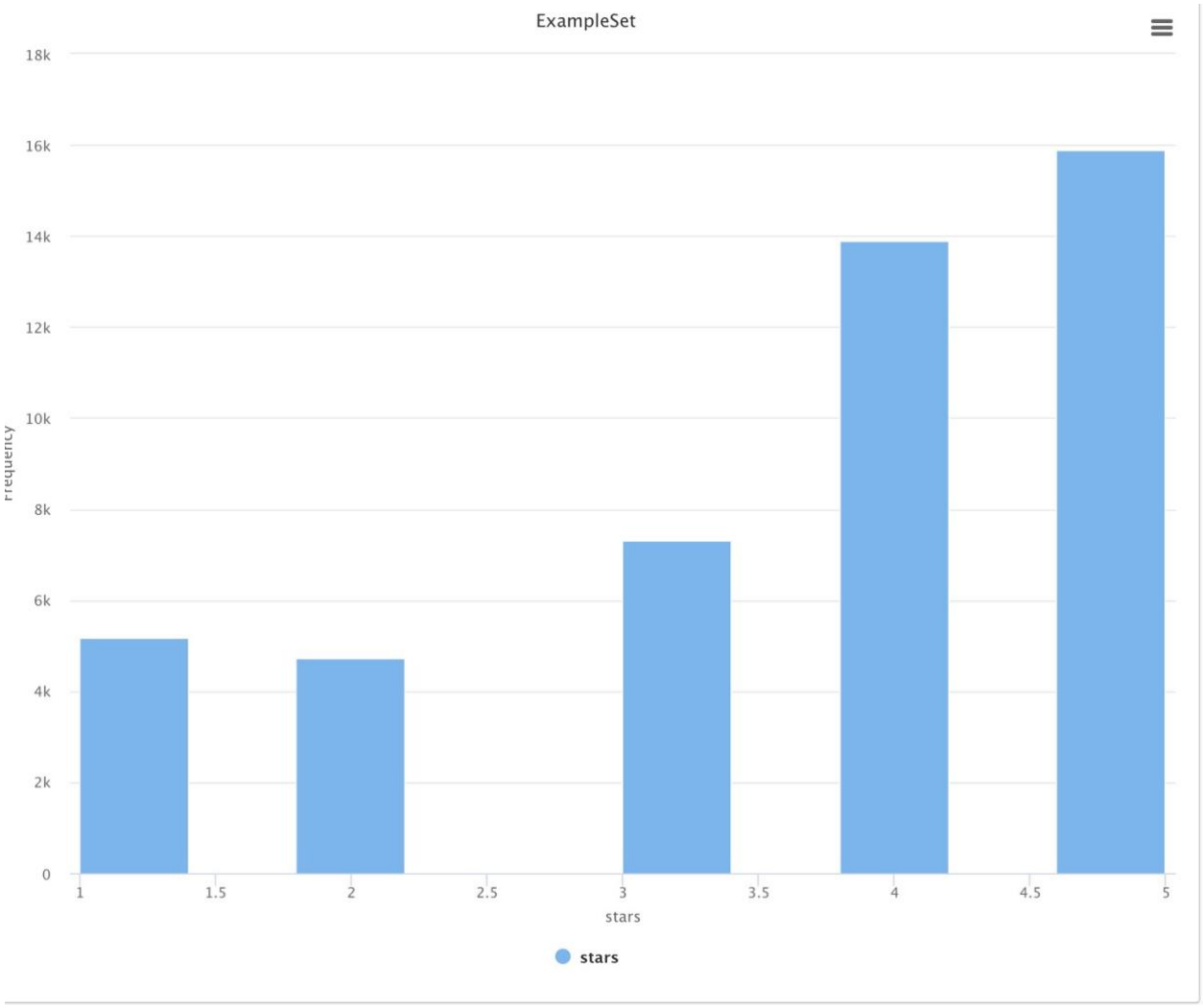
Star Rating distribution across the dataset. In it we can see that the distribution of higher star ratings is significantly larger than low star ratings.

stars
3
4
4
1
2
1
2
2
2
1
2
4
3
4
5
4
3
5

Star Rating Attribute



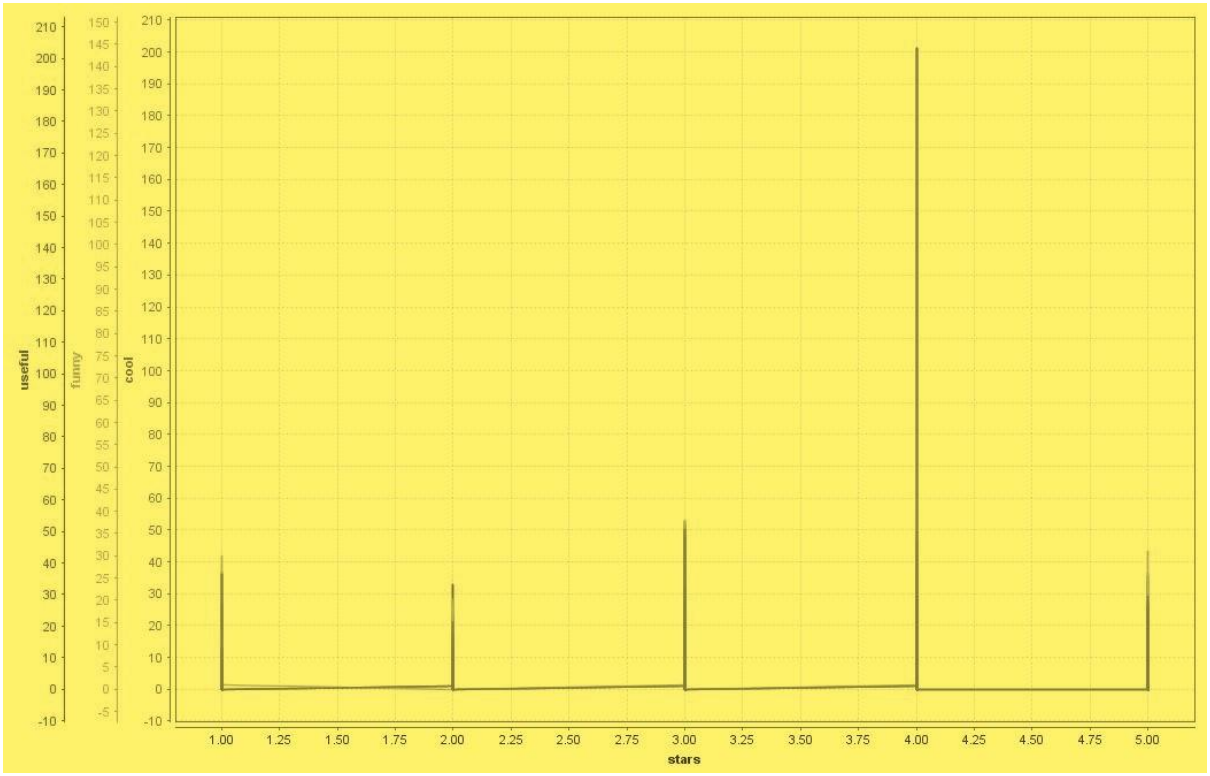
We have done the univariate analysis on the star rating variable and found that it is distributed as given below:



Ratings	1	2	3	4	5
Count	5163	4720	7317	13901	15877

The distribution for the rating is shown above:
It clearly depicts that the rating 4 and 5 is much higher than the 1 and 2.

It can be clearly seen from the histogram that there is skewness in the distribution (on the left side), with a higher percentage of 4 and 5-star ratings overall and comparatively lower percentage of 1, 2 and 3 ratings.



From the box-plot it can be seen clearly that the mean of the star rating is centered at 4-rating.

We did the classification to obtain a label by which we can indicate that whether a rating is ***Positive*** or ***Negative*** based on the value that we have set as threshold value.

The value that we have set as a **threshold value is 2.5.**

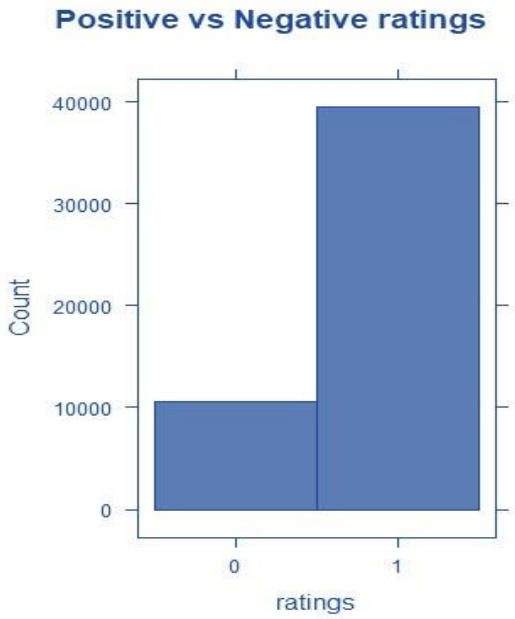
Any rating above our threshold value is taken as Positive (1) and rest as Negative (0), ratings whose values are less than our threshold values.

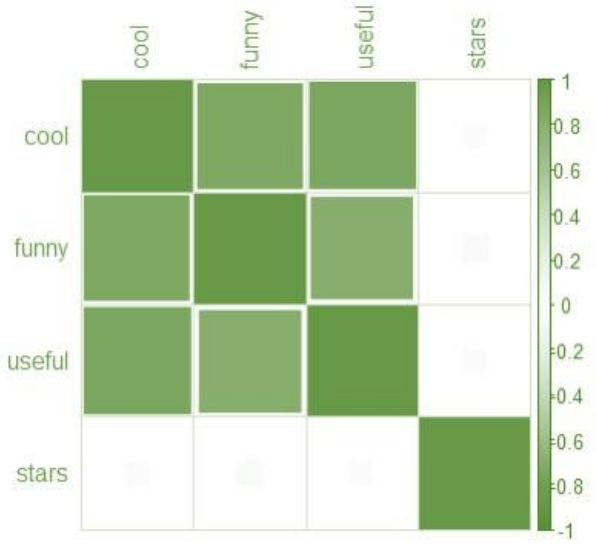
Because of the above assumptions, we found that there is a 1:4 (20:80) distribution for ***negative*** and ***positive*** ratings.

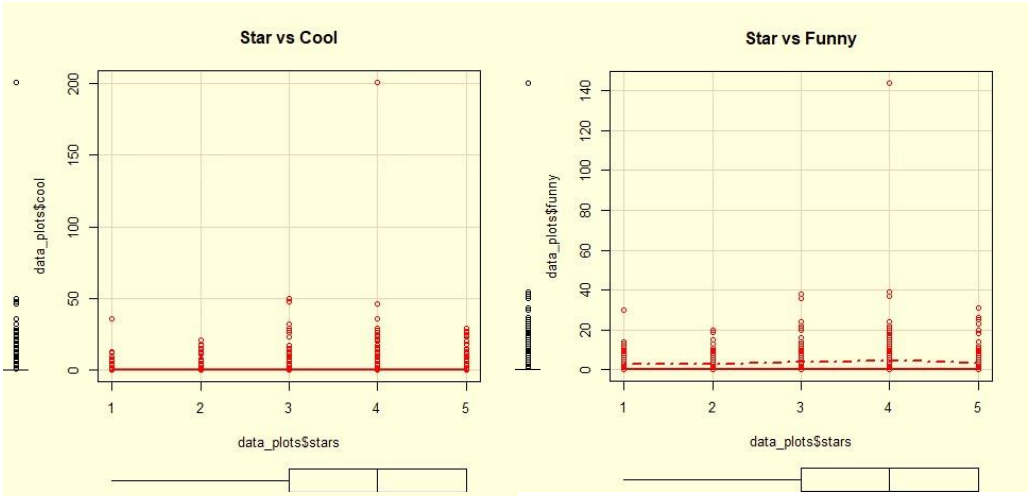
We also plot scatterplot between variables Funny, Useful, Cool each vs the Star Rating and after examining these scatterplots we found that there is no significant relationship between these variables among them.

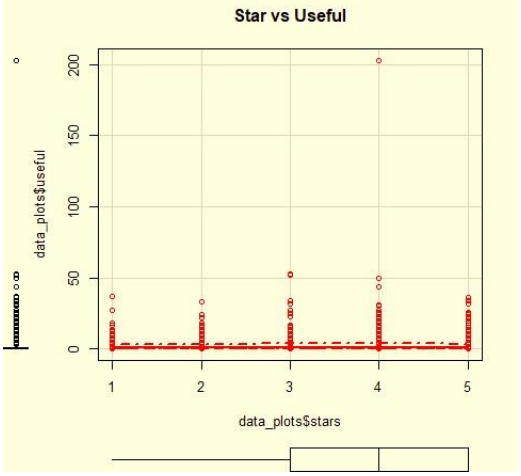
This was completely against to our prior expectation that there might be a strong relation between these variables. However, we can see that there was a visibly high correlation between the variables Funny, Cool and Useful, which was not expected.

The plots are shown below:









Tools used:
Rapid Miner

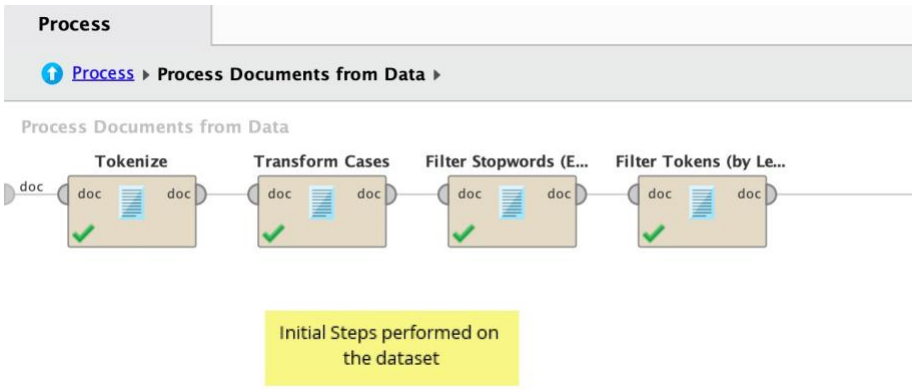
File used:
46,978 Yelp Review Documents

Task Performed:
Text and Sentiment Analysis

Steps used:

1) To start with, we loaded each word within a review as a column within a sparse matrix and performed Lemmatization.
2) Next, we pruned the huge sparse matrix with some conditions as given below.

- First, we performed the Tokenization
- After that we transformed each word in review to lower case
- We filtered Stop words, punctuations, commas, separators, hyphens etc.
- Then we filtered the words by the word length (words less than 3 words and more than 15 words were removed)
- From the above matrix, words occurring with less than 1% frequency were removed



We were left with a word count of the pruned matrix as 649.

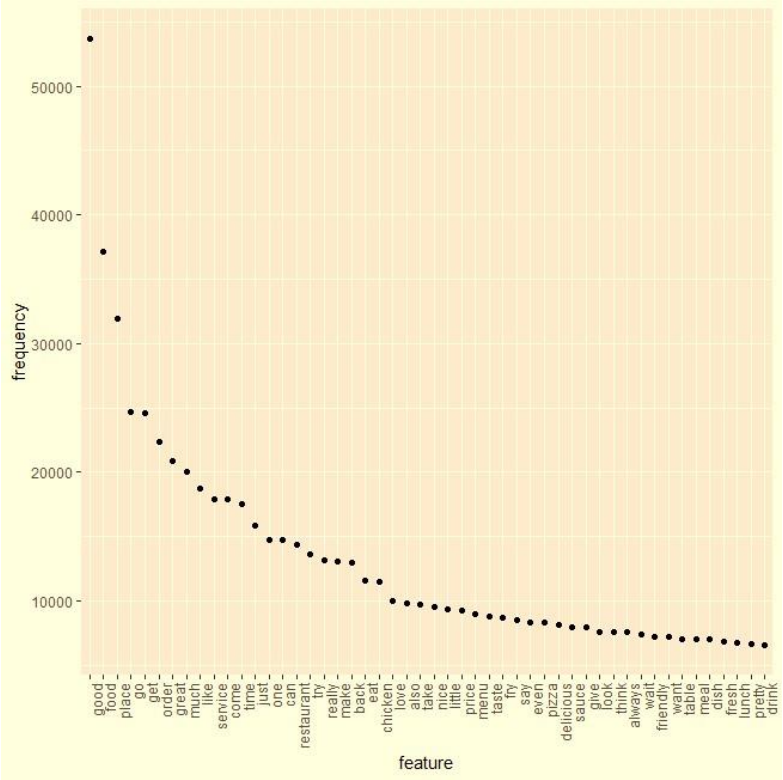
ExampleSet (649 examples, 0 special attributes, 4 regular attributes)

Row No.	word	in docu... ↓	total	att_1
197	food	24177	36485	0.805
229	good	19831	30397	0.819
413	place	19369	28045	0.777
497	service	15277	17667	0.547
233	great	15022	20552	0.813
572	time	10014	13105	0.606

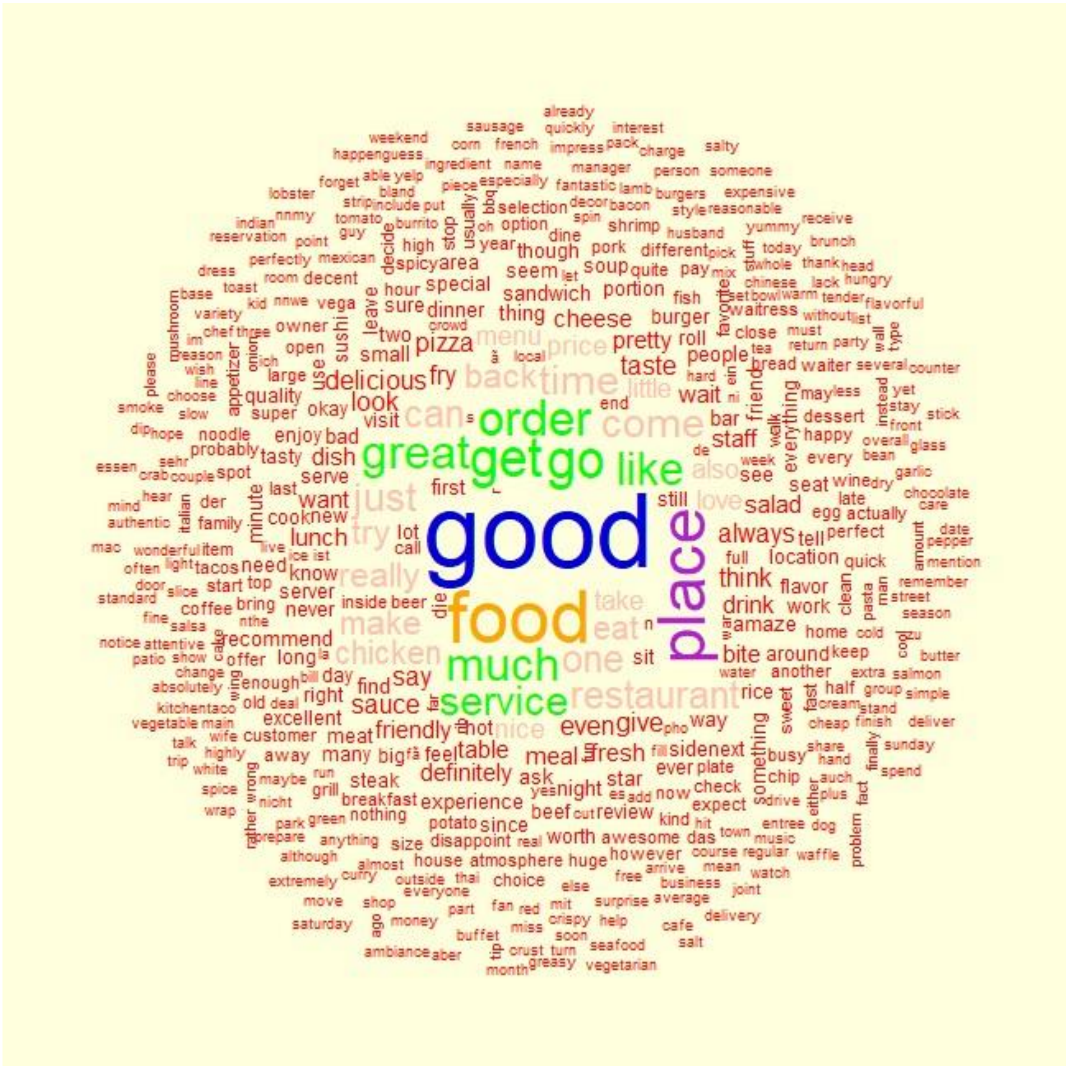
Row No.	word	in docume...	total	att_1
1	aber	588	926	0.719
2	able	882	942	0.491
3	absolutely	1424	1517	0.564
4	actually	1985	2165	0.454
5	add	924	979	0.488
6	added	691	743	0.478
7	amazing	4438	5084	0.706
8	ambiance	1037	1071	0.603
9	amount	1210	1295	0.489
10	anywhere	500	512	0.548
11	appetizer	1209	1362	0.515
12	appetizers	830	915	0.542
13	area	3241	3720	0.557
14	arrived	1079	1244	0.422
15	asian	573	711	0.581
16	ask	1832	2033	0.459
17	asked	2145	2622	0.407
18	ate	1418	1505	0.487
19	atmosphere	2943	3054	0.549
20	attentive	1543	1572	0.546
21	auch	668	1213	0.809

We calculated the frequencies of the top 50 words which are shown below.

Top 50 words Frequency:



Word cloud of the most frequent words:



After performing the above steps, we then assigned some weights to the Document Term Matrix on the basis of the Term Frequency.

After that we took the product of each of the rating of the Document Matrix with the Term Frequency assigned to those terms.

Later we took the mean of the multiplied frequency of the word that we got in the above across all the documents to attain the average star rating of each word.

Review with positive sentiment value will lead to high Average rating.

Review with negative sentiment value will lead to low Average Rating.

Sentiment Value associated with each word can be found out by the Average Rating value that is associated with that word.

List of Top 20 words with their Average Rating

Word	Average Star Rating
go	0.1383
food	0.1237
good	0.1195
little	0.1152
nice	0.1148
menu	0.1145
place	0.1138
take	0.1136
price	0.1133
love	0.1126
also	0.1124
taste	0.1112
say	0.1060
order	0.1052
eat	0.1032
fry	0.1031
chicken	0.1029
get	0.1020
pizza	0.1003
great	0.1003

to differentiate between the positive and the negative words we used a threshold value of 0.7 for the positive words and 0.4 for the negative words.

positive words (greater than 0.7) with high attribute value are – Sushi, Burger, Indian, wings, Pizza, Burritos, Good, Yum, etc. **negative words (less than 0.4)** with the low attribute values are – Terrible , Horrible, Poor, worst, Rude etc

List of some positive and negative words are given below:

Positive	Negative
Good	Hate
Nice	Weird
Love	Dirty
Eat	Numb
Great	Burn

The average star ratings of all the words are attached below.

List of positive and negative words with attribute score:

ExampleSet (649 examples, 0 special attributes, 4 regular attributes)

Row No.	word	in documents	total	att_1 ↓
594	und	1293	4397	1.560
407	pho	894	1846	1.250
412	pizza	3701	7716	1.202
544	sushi	1896	3717	1.072
128	die	1668	3770	1.063
490	sehr	799	1518	1.055
274	ist	929	2014	1.040

ExampleSet (649 examples, 0 special attributes, 4 regular attributes)

Row No.	word	in documents	total	att_1 ↑
639	worst	991	1073	0.279
255	horrible	852	961	0.318
562	terrible	880	991	0.331
421	poor	629	696	0.362
463	rude	725	826	0.372
394	paid	632	664	0.372
331	mediocre	771	809	0.383

After performing the above steps:

1) Tokenizing
2) Transforming to lower case
3) Filtering Stop words
4) Filtering words by length

We were left with 649 examples including 739 attributes.
After performing the above steps, we performed analysis to predict the polarity score (sentiment score) (positive score or negative score)
In preprocessing data, we did it by first setting the “stars” as our label and then we did the matching of it to the positive and negative words for each of the dictionary.

In the below table the count matching score can be found with the respective dictionary:

Dictionaries Used	Match Count
Harvard Positive	29
Harvard Negative	20
UIC BIN LIU Positive	67
UIC BIN LIU Negative	40
AFINN Positive	56
AFINN Negative	17

After that we have created positive and negative sum scores (aggregated) for each of the review.
For that we have created a binary variable that will be equal to “1” if the positive sum > negative sum
Else the value of the binary variable will be “0”.

We have also classified the review data into actual sentiments by transforming the “ star” variable:
We have created a label that will be equal to “1” if stars
== 4 or stars == 5

else the value will be “0” if
stars == 2 or stars == 1

we have removed the stars ratings that are neutral (ratings=3) from our dataset (yelp review dataset) so that we can improve the prediction of our analysis.

From the above analysis the predicted class variable where positive sum> negative sum and actual class label developed above
we have analysed the performance for all the three dictionaries in predicting actual sentiments.

Individual performances for the above dictionaries are given below:

Harvard IV:

Table View Plot View

accuracy: 65.41%

	true 0	true 1	class precision
pred. 0	8578	12575	40.55%
pred. 1	1891	18783	90.85%
class recall	81.94%	59.90%	

Afinn:

Table View Plot View

accuracy: 78.59%

	true 0	true 1	class precision
pred. 0	4461	2946	60.23%
pred. 1	6008	28412	82.55%
class recall	42.61%	90.61%	

UIC Bing Lu:

Table View Plot View

accuracy: 80.95%

	true 0	true 1	class precision
pred. 0	5373	2874	65.15%
pred. 1	5096	28484	84.82%
class recall	51.32%	90.83%	

	Harvard IV	UIC BIN LIU	AFFIN dictionary
Accuracy	0.66	0.81	0.79
Recall	0.91	0.85	0.83
Precision	0.82	0.91	0.90
Specificity	0.41	0.66	0.61

From the above confusion matrices we can clearly see that UIC BING LIU dictionary is performing the best with an accuracy for positive reviews of around 81% and a recall rate of around 91%.

To identify negative reviews Harvard IV dictionary is performing better than the other two dictionaries.

As we want to identify both the negative and the positive reviews , our option is to choose the dictionary which will give best results overall.

ii) SentiWordNet

Sample	Accuracy	Class Precision		Recall	
		Pred. Positive	Pred. Negative	True Positive	True Negative
1	74.39	84.99	40.47	81.89	46.35
2	79.49	88.34	45.63	85.42	52.58
3	81.37	87.14	52.8	90.65	44.1
4	86.53	93.57	52.99	90.43	64.38
5	70.73	72.73	58.53	88.90	31.84

SentiWordNet is a tool that relies on list of words and phrases with positive and negative connotations.
The list of the words was compiled in many years.
The sentiwordnet contains around 6800 positive and negative opinion words.
We have used the sentiword by installing a package called as wordnet in rapid miner and using the inbuilt set of list in it on our dataset to find the accuracy and other parameters.

Steps to use Sentiwordnet:
Extract words from the document
Match the words from the subjectivity score from the document
Generate a dataset of these terms
Used classifier to analyse the document

For our analysis by using the sentiment dictionary terms we have developed three models and the broader list of terms. For this we have used the measure as **TFIDF**.
As TFIDF assigns weights basis the relative importance of the word across all the documents.

TFIDF assigns high weight to the words that have high frequency all over the documents and a lower weight to the words that have lower frequency in lesser number of documents.

We have created the **document-term matrix for broader list of terms** with the size of **46,978 x 647.**

We have created the broader list of terms by performing the below steps:
Here No stemming is done, as we have used the data entirely.
We have increased the range of stars to obtain these terms
Inaccurate results will be obtained if we use stemming over data here.

To create the model, we have considered the below things:

1) TFIDF scores
2) Other variables such as “Cool”, “Useful”, “Categories” and “Funny”

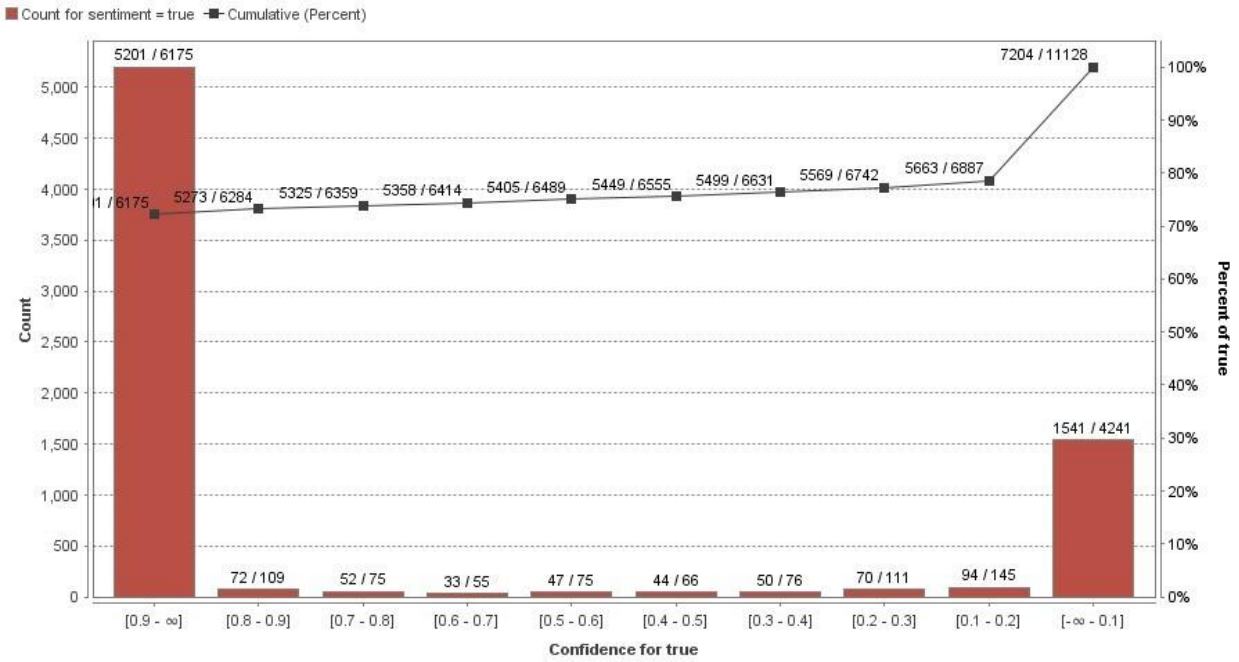
We have built models using “Naïve Bayes”, “Decision Tree” and “SVM” To measure the performance, we have used the parameters such as Precision, Accuracy, Recall and Specificity**.**

UIC BING LIU dictionary			
MODELS	**Decision Tree**	**SVM**	**Naïve Bayes**
Accuracy	0.829	0.859	0.705
Recall	0.975	0.969	0.667
Precision	0.828	0.867	0.939
Specificity	0.249	0.445	0.839

AFFIN dictionary			
MODELS	**Decision Tree**	**SVM**	**Naïve Bayes**
Accuracy	0.815	0.845	0.696
Recall	0.987	0.978	0.658
Precision	0.838	0.848	0.933
Specificity	0.185	0.348	0.825

Harvard IV dictionary			
MODELS	**Decision Tree**	**SVM**	**Naïve Bayes**
Accuracy	0.829	0.862	0.811
Recall	0. 836	0.968	0.837
Precision	0.963	0.895	0.909
Specificity	0.259	0.474	0.699

BROADER LIST OF TERMS			
MODELS	**Decision Tree**	**SVM**	**Naïve Bayes**
Accuracy	0.819	0.899	0.837
Recall	0.979	0.979	0.858
Precision	0.829	0.899	0.929
Specificity	0.229	0.598	0.745



After comparing the above models (Naïve Bayes, Decision Tree, and SVM model) built for broader sample of terms and various libraries, we found that the SVM performs best among them all and consistently for each libraries and broader list of terms.

Document-term matrix for broader list of terms with the size of **46,978 x 647.**

After comparing the model performances in question 3 and question 4, we can clearly see that the SVM model is performing better than the predictions obtained by aggregating negative and positive scores of each document.

In RapidMiner, the Filter Token by content operator requires the dictionary terms specified as (for example): abide|ability|able. Another approach to using only those words matching a sentiment dictionary: first form the documentterm matrix on the broader set of words.Which of the two approaches do you find to be more efficient in terms of processing time?

In Rapid miner, among these two choices for filtering

1) Filter token by using regular expression
2) Using only those words that matches a sentiment dictionary

I would prefer the second approach as compared to 1st as the second is more efficient and saves time by using vectorization technique to check the terms in a document as compared to first one. in first we have to manually add the words using the regular expression and after that these words are searched in the whole document row wise which takes a comparatively large amount of time as done in column approach.

Also, matrices are fast and easy to compute.