Sentiment Analysis of Reviews Posted on Yelp for an American Chain of Fast Food Restaurants – Taco Bell

Jasleenkaur Gorowada (jasleen06@umbc.edu)
Neel Patel (neel4@umbc.edu)
Neetu Menon (mneetu1@umbc.edu)
Nitin Deb (dnitin1@umbc.edu)
Rency Varghese (renva1@umbc.edu)

Introduction

With increasing gourmets, online reviews and ratings act as a major deciding factor in choosing restaurants and cafes. These reviews enable customers to have an idea about not just food but also other factors about a place to be visited, namely its ambience, service. 'Yelp' a multinational company that hosts Yelp.com; a social platform that connects locals through mobile apps which hosts crowdsourced opinions about everything from restaurant to mechanics to local businesses. 'Yelp' also withholds customers trust in terms of integrity of the Yelp reviews and hence, customers rely highly on the ratings and reviews available on Yelp for casual decision making such as food joints, entertainment, shopping. The statistics say that on an average, there are 24 million unique app users who post their reviews about the local business, out of which, 18% reviews are for restaurants and 22% for shopping. Yelp has collected around 121 million reviews which is the highest number of reviews till date.(Source)

Motivation

High reliability on online reviews for selecting food joints further adds to the need to analyze these reviews. 'Fast Food' being commonly available and preferred by all ages in the United States, we decided to analyse the reviews (extract sentiment) on ' Taco-Bell restaurants. ' Taco-Bell a leading fast food chain in the United States, houses a variety of Tex-Mex cuisine (fusion of American and Mexican cuisine) and value-menu" items (items at a fast food restaurant that are least expensive). This speciality makes Taco-Bell, readily preferred by people as it also takes into consideration the budget aspect. Hence, performing sentiment analyses on the Taco-Bell reviews at various Taco-Bell joints is essential. Negative reviews would help Taco Bell make improvements in their services and positive reviews would help them promote their services and get a better rank amongst their competitors.
Along with a review, a yelper also provides rating from 0-5 for every review. Analysing reviews and extracting sentiments becomes crucial as a rating alone wouldn't provide a complete information about the different aspects of a Tacobell joint. Consider a Yelp Review: **"Everything is kept really clean and the inside is really modern. The staff has always been friendly, and service is usually quick as well."** This review has a star rating of 5. But from the rating it is not clear what the Yelper specifically liked about TacoBell. It is for this reason that aspect based analysis becomes important where various aspects of a review can be analyzed to understand what a Yelper specifically likes or dislikes. The mentioned review talks about "Ambience" and "Service" on a positive note. Such analysis also reduce the effort of reading a review to understand what a user specifically likes or dislikes as the aspect based analysis would do it for them.

Related Work

Yun et.al [1] use, supervised machine learning algorithms such as Naïve bayes and Multiclass SVM to predict reviewers rating based on the text and this was further compared to the actual ratings. To built the training set, reviews were classified into individual sentences as review as a whole had various sentiments in different sentences.

The effectiveness of these algorithms were compared based on evaluation metrics such as precision and recall. Sentiment dictionary was also used to use explore feature selection algorithm where Bing Liu Opinion Lexicon was used. After comparisons of these algorithms, Perceptron algorithm had the highest precision and recall while the Multi-class SVM had the least precision and recall. Naïve bayes had better performance but also had the issue of high variance.

Mukherjee et. al [2] performed feature specific sentiment analysis to extract expressions of opinion describing a target feature and classify it as positive or negative. They did feature extraction in both presence and absence of domain knowledge. To identify the associations between the opinion expressions in a review, relation extraction was performed. They depicted the features and corresponding opinions in the form of a graph where dependency parsing was used to capture the relationship between the features and their associated opinions. Clustering was done on the graph to retrieve only those opinion expressions that are most closely related to the target feature (user specified feature).

Sahnni et.al[3] performed sentiment analysis on Twitter data to extract opinions and speed up the sentiment classification using Distant Supervision method and other machine learning algorithms. This model helped to lower computation time and increase the accuracy with the baseline model. This was achieved using subjectivity threshold to filter the training data , incorporating complex processing stage, using Effective Word Score (EFWS) heuristic model for the sentiment classification . Naïve Bayes, Support Vector Machines, Maximum Entropy Model was used as classifiers to extract the sentiment-based opinions from the training data.

Dataset Collection and Dataset Description

We collected relevant data from "https://www.yelp.com/dataset_challenge" which provided access to a dataset in .json file format which was further processed.
The Yelp Dataset consists of 4 million data entries, out of which it consists of 267 Taco Bell franchises and 5838 unique reviews about Taco Bell.
The Yelp dataset consisted of many .json files namely, business.json, checkin.json, reviews.json, tip.json, user.json. Out of these files, we selected the business.json and reviews.json file for our analyses. These .json files consisted of the attributes as mentioned below.

- Input dataset1:

business.json

Steps	Attribute	Function
1	Business id	A unique id for every business
2	Business name	Provides the name of the business
3	Neighbourhood	Hood name
4	Address	Full address
5	City	City
6	State	State
7	Postal code	Postal code
8	Latitude	Latitude
9	Longitude	Longitude
10	Stars	Star rating
11	Review count	Number of reviews
12	Is open	Closed/Open
13	Attributes	List of Business facilities
14	Categories	List of Business categories
15	Hours	Business hours
16	Type	Business type

Dataset sample:

{
"business_id":"0DI8Dt2PJp07XkVvIElIcQ",
"name":"Innovative Vapors",
"neighborhood":"",
"address":"227 E Baseline Rd, Ste J2",
"city":"Tempe",
"state":"AZ",
"postal_code":"85283",
"latitude":33.3782141,
"longitude":111.936102,
"stars":4.5,
"review_count":17,
"is_open":0,
"attributes":["BikeParking: True","BusinessAcceptsBitcoin:
False","BusinessAcceptsCreditCards: True","BusinessParking: {'garage':
False, 'street': False, 'validated': False, 'lot': True, 'valet':
False}","DogsAllowed: False","RestaurantsPriceRange2:
2","WheelchairAccessible: True"],
"categories":["Tobacco Shops","Nightlife","Vape Shops","Shopping"],
"hours":["Monday 11:0-21:0","Tuesday 11:0-21:0","Wednesday
11:0-21:0","Thursday 11:0-21:0","Friday 11:0-22:0","Saturday
10:0-22:0","Sunday 11:0-18:0"],
"type":"business"
}

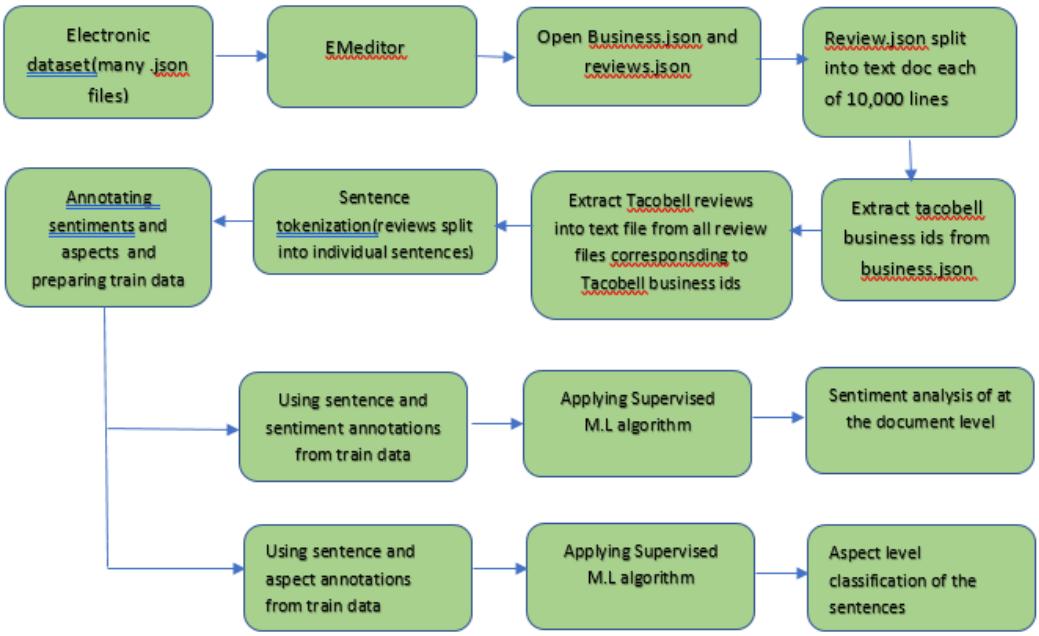
- Input dataset2:
  review.json

Steps	Attribute	Function
1	Review id	A unique id for every review
2	User id	A unique id for every user
3	Business id	A unique id for every business
4	Stars	Star rating
5	Date	Date formatted like YYYY-MM-DD
6	Text	Review text
7	Useful	Number of useful votes received
8	Funny	Number of funny votes received
9	Cool	Number of cool votes received
10	Type	Review

Dataset sample:
{
"review_id":"NxL8SIC5yqOdnlXCg18IBg",
"user_id":"KpkOkG6RIf4Ra25Lhhxf1A",
"business_id":"2aFiy99vNLklCx3T_tGS9A",
"stars":5,
"date":"2011-10-10",
"text":"If you enjoy service by someone who is as competent as he is personable, I would recommend Corey Kaplan highly. The time he has spent here has been very productive and working with him educational and enjoyable. I hope not to need him again (though this is highly unlikely) but knowing he is there if I do is very nice. By the way, Iḿ not from El Centro, CA. but Scottsdale, AZ.",
"useful":0,
"funny":0,
"cool":0,
"type":"review
}

System Design

We used the dataset provided for the Yelp data challenge at the Yelp challenge website "https://www.yelp.com/dataset_challenge". Although the dataset was available, handling such large data containing millions of reviews was challenging. The Yelp reviews were in .json file format from which required files were selected and just the Tacobell reviews were extracted.



Different Python codes were used for some of the above steps:
From the reviews.json file, the reviews corresponding to the Tacobell business ids were selected and extracted into a reviews.txt file and reviews.xls file using python code.

- Code for splitting the review.json files into text files each of 10,000 lines

```
count=0
line1=''
with open("C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media
analytics\\Project\\yelp_academic_dataset_review.json",encoding='utf8') as rad:
    for line in rad:
        count +=1
        line1 +=line
        if (count%10000==0):
            target=open('C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media
analytics\\Project\\review files\\files\\review%s.txt'%(count),'w',encoding='utf8')
            target.write(line1)
            line1=''
            target.close()
```

From the business.json file, we selected the Business id and the Business name columns corresponding to business name Tacobell.

- Code for finding only Tacobell business ids from business.json file and extracting it into .txt file[4]

```
import re
line2=''
with open('C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media
analytics\\Project\\yelp_academic_dataset_business.json',encoding='utf8')as f:
    for s in f:
        result=re.findall('"name":"Taco Bell"',s)

        if result:
            result_id=re.findall(r'"business_id":"(.*?)","name":"Taco Bell"',s)
            if result_id:
                line2+=str(result_id)+'\n'
target=open('C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media analytics\\Project\\review
files\\result_id.txt','w',encoding='utf8')
target.write(line2)
target.close()
```

We would be using Supervised Machine Learning algorithms for sentiment analysis and ground truth needed to be formulated.

- Code for extracting Tacobell reviews corresponding to Tacobell business ids from various review text documents into a single text document

```
import re
output = ''
for i in range(10000,1870000,10000):
    a = open("C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media analytics\\Project\\review
files\\files\\review%s.txt"%(i),'r',encoding='utf8').readlines()
    b = open("C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media analytics\\Project\\review
files\\result_id.txt",'r').readlines()
    for line1 in b:
        b1 = (line1.replace('[', '').replace(']', '').replace('\'','').replace('\n',''))
        for line2 in a:
            if (re.findall(r'"business_id":"%s"' %(b1),line2)):
                output+=line2
with open("C:\\Users\\JASLEEN\\Desktop\\TacoBell_5.txt","a",encoding='utf8')as op:
   op.write(output)
```

Sentence tokenization was done on the Tacobell reviews text document, where each review was split into sentences. Business ids for the corresponding sentences are also present in sentence tokenized review file.

- Code for performing sentence tokenization to prepare train data where each review is split into individual as individual reviews contain more than one sentiment[3]

```
import nltk
import re
import ast
from nltk.tokenize import sent_tokenize
line1=''
line2=''
result_txt=''
with open('C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media analytics\\Project\\review
files\\final\\TacoBell_5.txt',encoding='utf8')as f:
    for s in f:
        a=ast.literal_eval(s)
       # print(a.keys())
        for key,value in a.items():
            if(key=='business_id'):
                line1+=str(key)+' : '+str((value))+'\t'
                for key,value in a.items():
                    if(key=='text'):
                        line1+=''.join(str(key)+' : '+str(value.replace('. ',
'.\n').replace('!','.\n')))+'\n'
    print(line1)
```

- The contents of the reviews.txt file after sentence tokenization is as below.

business_id : 6nnI3DfHn-DTd6tWnZu7Jg
text : Gross how can anyone serve food from a pig pen.
I would not feed my dogs from this nasty so called restaurant.
With employees acting like it was high school running all over the manager.
This place is a health hazard with the back having food everywhere.
And mike the guy on the register giving food away to his friends making paying customers feel like he was doing us a favor for taking my order.
business_id : 6nnI3DfHn-DTd6tWnZu7Jg
text : Taco Bell is definitely my favorite fast food and out of all the locations that I go to I would have to say that this is my favorite the staff is always friendly and the managers always take their time to say hello
business_id : A6HfWbmTpJRlw_-SxwFUg
text : Eating in is ok.
going through the drive through....WRONG EVERY TIME.
Clean place nice folks....just don't drive through.
LOL

- Code for converting the sentence tokenized text reviews into .xls format

```
data = []
with open("C:\\Users\\JASLEEN\\Documents\\IS Sem2\\Social media analytics\\Project\\review
files\\TacoBell_5_reviewsonly_test.txt") as f:
    for line in f:
        data.append([word for word in line.split("  ") if word])
print(data)
import xlwt
wb = xlwt.Workbook()
sheet = wb.add_sheet("New Sheet")
for row_index in range(len(data)):
    for col_index in range(len(data[row_index])):
        sheet.write(row_index, col_index, data[row_index][col_index])

wb.save("newSheet.xls")
```

Data Analysis:
After the above codes, extract reviews from the Review.json file along with the business ids mapping them against the business ids from the Business.json file, the reviews are sentence tokenized, namely split into individual sentences and later manually annotated as per the sentiment (positive, negative and neutral) and aspect of each sentence. The aspect identified after manual annotation were, Food, Price, Parking_availability, Service, Ambience, Location and Other. Wifi was also a recognized aspect but due to low feature count it was disregarded for from further analysis. The format of the training set is as below.

	A	B	C	D	E	F
1	Business id	sentence	annotation	positive	negative	neutral
2	JTN1qe0UBT3T6zIWYcBtjg	I love taco bell	Other	1	0	0
3	JTN1qe0UBT3T6zIWYcBtjg	Pretty good grub for a pittance.	Price	1	0	0
4	JTN1qe0UBT3T6zIWYcBtjg	But as the years went by the beef filling got thinner and thinner and the lettuce started taking up it's place.	Food	0	1	0
5	JTN1qe0UBT3T6zIWYcBtjg	Kudos' to George, you deserve a raise.	Service	1	0	0
6	JTN1qe0UBT3T6zIWYcBtjg	the dining room was well kept	Ambience	1	0	0
7	JTN1qe0UBT3T6zIWYcBtjg	Parking lot is plentiful	Parking_availability	1	0	0
8	JTN1qe0UBT3T6zIWYcBtjg	the location is in a great spot	Location	1	0	0

Also,identified conflicts of annotations amongst our team members and eliminated any discrepancies in the reviews. There are an approximate of 1557 annotated review sentences. Out of the sentences, 75% of the sentences were used as train data an 25% was used as test data.

Feature Extraction and Pre-Processing

Data was pre-processed and the following pre-processing steps were performed:

1. Explicitly converted the sentences into string using the str() function in python to avoid any discrepancies.
2. The sentences were converted into lower-case
3. Stop words removal was performed

A data frame of these sentences and sentiment along with sentences and aspect annotations were constructed. The sentiment annotations were grouped into one 'Result' column such that, 100 represented a positive sentiment, 010 represents a negative sentence and 001 represents a neutral sentiment.

```
In [63]: del df1['sentence']
```

```
In [64]: df2= df1[['sentence-join','result']]
         df2
```

Out[64]:

	sentence-join	result
0	gross anyone serve food pig pen	010
1	would feed dogs nasty called restaurant	010
2	employees acting like high school running manager	010
3	place health hazard back food everywhere	010
4	amd mike guy register giving food away friends...	010
5	dont get wrong love taco bell	100
6	taco bell much	001
7	hard go anywhere people work obviously dont gi...	001
8	pride even endallbeall existence	001
9	drive quickly past taco bell get better one	001

The data was tokenized into unigrams and bigrams , converted it to a matrix of token counts, and further transformed this matrix to a normalized TF or TF-IDF representation. Data was converted into vector form performing Tf-idf transformation. The vector form of data is as below:

```
In [72]: from sklearn.feature_extraction.text import TfidfTransformer
         tf_vec_all = TfidfTransformer()
         tr_features_all = tf_vec_all.fit_transform(ctr_features_all)
         tr_features_all

Out[72]: <790x5659 sparse matrix of type '<class 'numpy.float64'>'
                with 9798 stored elements in Compressed Sparse Row format>
```

```
In [73]: cte_features_all = vec_all.transform(Data_test)
         te_features_all = tf_vec_all.transform(cte_features_all)
         Data_test

Out[73]: 414                                    food kind cold
         5                          dont get wrong love taco bell
         77                                             kudos
```

Implementation using Model-classifiers

Bernoulli's Naïve Bayes: Bernoulli's NB is a binary classification method and is applied to data that is distributed. It works well for text categorization where the word occurrence vectors is used to train the model and predict the sentiments and aspects for the test data. It is represented as :

$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

Logistic Regression: Logistic Regression model is a binomial, ordinal and a multi-nominal logistic regression. Binary logistic regression will outcome the results as either "true" or "not true", for eg: if for a given review, it speaks or highlights the category of food, then the logistic binary classification will identify it as "Food or not Food". Multi-nominal logistic regression highlights the situation where the outcome is multi-class , that is , belongs to either "Food vs. Ambience vs. Service vs. Location vs. WiFi vs. Parking_Availaibility vs. Others. Our project considers logistic binary classification for the aspects.

Perceptron: Perceptron model is a supervised machine learning algorithm for binary classifiers which maps the input value x to output value f(x). w stands for real-weighted values. Value of x stands for either 'true' or 'not true'. It acts as a linear classifier and can overly fit for linearly separable data.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

Evaluation Measures

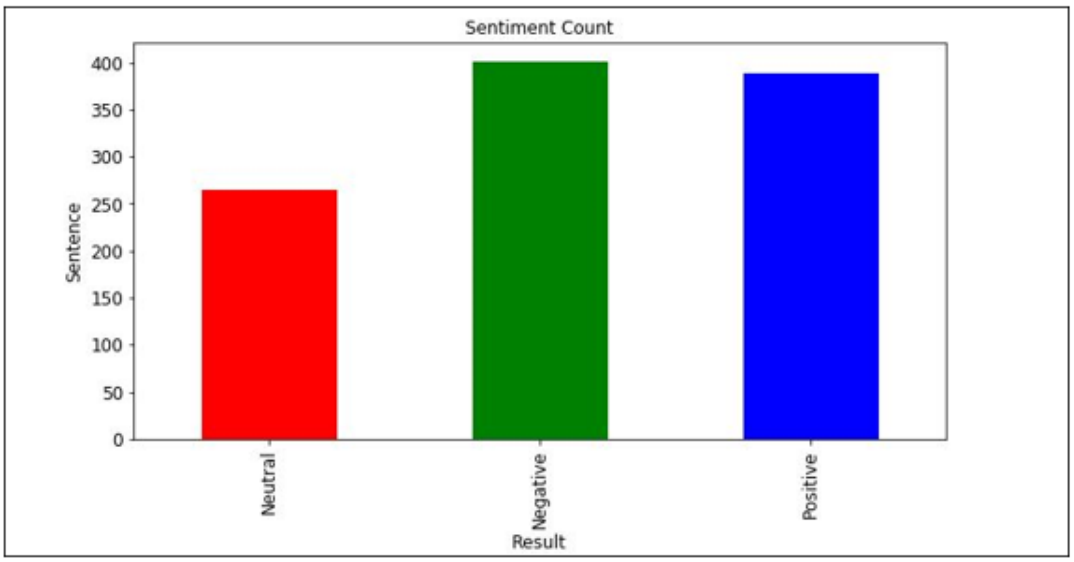
We first identified the total count(frequency) of the sentiments of the entire document which includes the Positive, Negative, Neutral as it is stated as below:

```
In [30]: df2.groupby('result').count()
```

Out[30]:

	sentence-join
result	
001	264
010	401
100	389

A graphical representation of the above result is as below:



An overall document level sentiment analysis was performed with the predictions as below:

```
In [39]: models1 = linear_model.LogisticRegression()
         models1.fit(tr_features_all, Train_labels)
         tfprediction = models1.predict(te_features_all)
         percentile_list = pd.DataFrame({'Prediction': list(tfprediction),'sentence': list(Data_test)})
         percentile_list
```

Out[39]:

	Prediction	sentence
0	010	thought rude especially know item always orde...
1	100	service quick food expect
2	010	got food poisoning particular taco bell
3	100	taco bell least taco bell dang good sweet tea
4	100	shame taco bell
5	100	exactly expect
6	010	else need say head border theyll take care
7	100	thank taco bell paul customer maniac providin...
8	010	adore tyler wish could hear often life need tacos
9	010	guess last couple visits youre right back givi...

Accuracy:

Accuracy is a simplest statistic which measures the ratio of correctly categorized instances to the total number of correct and incorrect categorized instances. It measures the closeness of measured value with the actual value.

To achieve high accuracy, we have the test data accounting to 25% of the 1557 statements collected.

We then trained our model with the training data set and collaborated prediction results of the classifiers: Bernoulli's Naïve Based, Logistic Regression and Perceptron models.

We would also calculate the below measures:

Precision: Precision define the ratio of events that are correct (true positive) to the total number of all correct events and false correct events (True Positive and False Positive)

$$\boldsymbol{Precision = TP/(TP+FP)}$$

Recall: Recall defines the ratio of events that are correct(true positive) to the total number of all correct events and false incorrect events(True Positive and False Negative)

$$\boldsymbol{Recall = TP/(TP+FN)}$$

F-measure: The F-measure (F1 score or F score) is a measure of a test's accuracy and is defined as the weighted harmonic mean of the precision and recall of the test.

$$\boldsymbol{F\text{-}measure = 2 * (Precision * Recall)/(Precision + Recall)}$$

Sentiment Analysis:

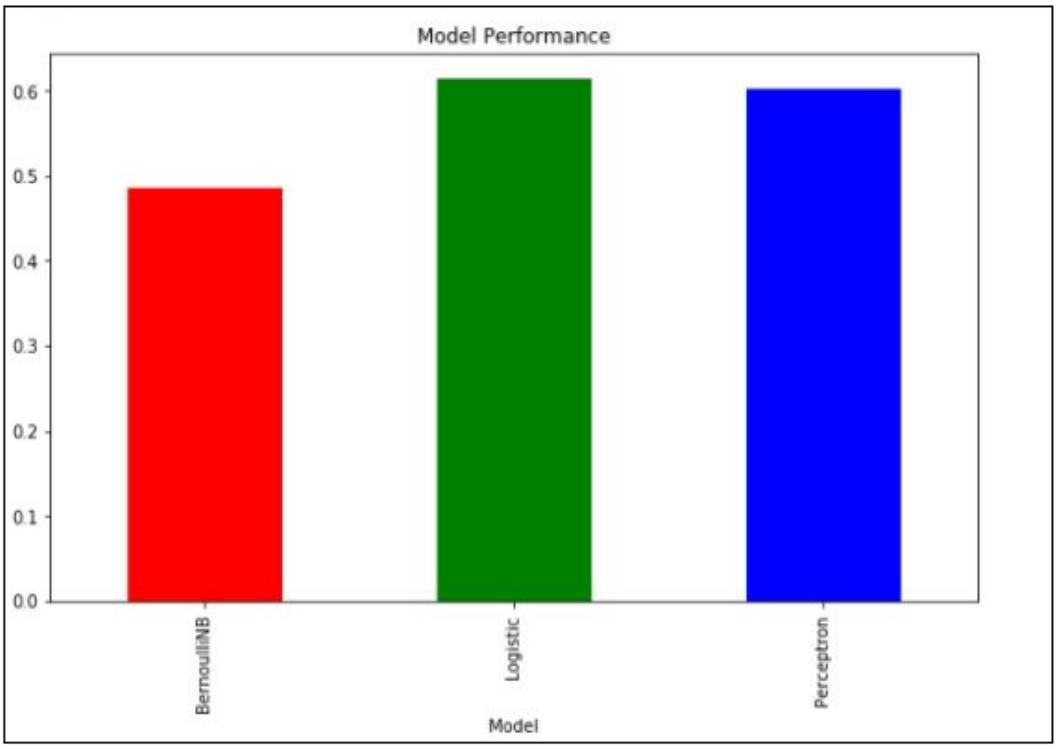
1. **At the document level:** The model prediction(accuracy) of the document level analysis is as below:

```
In [34]: models = {'BernoulliNB':BernoulliNB()
                   ,'Logistic' : linear_model.LogisticRegression(C=1e5),'Perceptron': linear_model.Perceptron(n_iter=1000)}
```

```
In [35]: results_all_bi = pd.DataFrame()

         #stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
         tfprediction = {}
         cprediction = {}
         foldnum = 0
         for name,model in models.items():

                 model.fit(tr_features_all, Train_labels)
                 tfprediction[name] = model.predict(te_features_all)
                 tfaccuracy = metrics.accuracy_score(tfprediction[name],Test_labels)

                 results_all_bi.loc[foldnum,'TF-IDF Accuracy']=tfaccuracy
                 results_all_bi.loc[foldnum,'Model']=name
                 foldnum = foldnum+1
         print (results_all_bi)
```

```
   TF-IDF Accuracy        Model
0         0.484848  BernoulliNB
1         0.613636     Logistic
2         0.602273   Perceptron
```

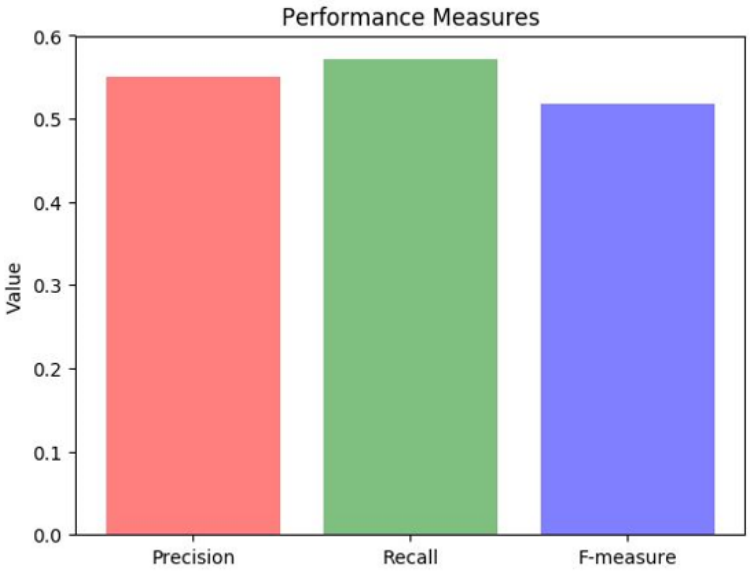
The graph representation of the models is as below:



The measure of Precision, Recall and F-measure is as below of the document level sentiment analysis:

```
In [367]: from sklearn.metrics import *

In [368]: precision=precision_score(Test_labels, tfprediction, average='weighted')
          recall=recall_score(Test_labels, tfprediction, average='weighted')
          f1_score=f1_score(Test_labels, tfprediction, average='weighted')
          print('Precision:',precision)
          print('Recall:',recall)
          print('F1_Score:',f1_score)

          Precision: 0.550715919275
          Recall: 0.57196969697
          F1_Score: 0.518539402634
```

The graph form representation of the above measures are as below:



2. **Sentiment Analysis by binary classification of sentiments:**
   The model prediction(accuracy) of the sentiment analysis at sentiment level is as below:
   Positive Sentiment:

```
#positive
results_all_bi_pos = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_pos = {}
cprediction_pos = {}
foldnum_pos = 0
for name,model in models.items():

        model.fit(tr_features_all_pos, Train_labels_pos)
        tfprediction_pos[name] = model.predict(te_features_all_pos)
        #print(tfprediction[name])
        tfaccuracy_pos = metrics.accuracy_score(tfprediction_pos[name],Test_labels_pos)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_pos.loc[foldnum_pos,'TF-IDF Accuracy']=tfaccuracy_pos
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_pos.loc[foldnum_pos,'Model']=name
        foldnum_pos = foldnum_pos+1
print (results_all_bi_pos)
```

```
   TF-IDF Accuracy        Model
0         0.659091  BernoulliNB
1         0.787879     Logistic
2         0.768939   Perceptron
```

Negative Sentiment:

```
#negative
results_all_bi_neg = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_neg = {}
cprediction_neg = {}
foldnum_neg = 0
for name,model in models.items():

        model.fit(tr_features_all_neg, Train_labels_neg)
        tfprediction_neg[name] = model.predict(te_features_all_neg)
        #print(tfprediction[name])
        tfaccuracy_neg = metrics.accuracy_score(tfprediction_neg[name],Test_labels_neg)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_neg.loc[foldnum_neg,'TF-IDF Accuracy']=tfaccuracy_neg
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_neg.loc[foldnum_neg,'Model']=name
        foldnum_neg = foldnum_neg+1
print (results_all_bi_neg)
```

```
   TF-IDF Accuracy        Model
0         0.625000  BernoulliNB
1         0.685606     Logistic
2         0.693182   Perceptron
```

Neutral Sentiment:

```
#neutral
results_all_bi_neut = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_neut = {}
cprediction_neut = {}
foldnum_neut = 0
for name,model in models.items():

        model.fit(tr_features_all_neut, Train_labels_neut)
        tfprediction_neut[name] = model.predict(te_features_all_neut)
        #print(tfprediction[name])
        tfaccuracy_neut = metrics.accuracy_score(tfprediction_neut[name],Test_labels_neut)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_neut.loc[foldnum_neut,'TF-IDF Accuracy']=tfaccuracy_neut
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_neut.loc[foldnum_neut,'Model']=name
        foldnum_neut = foldnum_neut+1
print (results_all_bi_neut)
```

```
   TF-IDF Accuracy        Model
0         0.750000  BernoulliNB
1         0.715909     Logistic
2         0.708333   Perceptron
```

This performance measures of precision, recall and f-measure is the average of these individual measure that are based on binary classification of sentiments (positive/not_positive, negative/not_negative, neutral,not_neutral)

```
In [122]: print('Precision of Positive Reviews: {}'.format(precision_pos))
          print('Recall of Positive Reviews: {}'.format(recall_pos))
          print('F-measure of Positive Reviews: {}'.format(f1_pos))

          Precision of Positive Reviews: 0.7664474105810515
          Recall of Positive Reviews: 0.7320754716981132
          F-measure of Positive Reviews: 0.6912177549524684

In [123]: print('Precision of Negative Reviews: {}'.format(precision_neg))
          print('Recall of Negative Reviews: {}'.format(recall_neg))
          print('F-measure of Positive Reviews: {}'.format(f1_neg))

          Precision of Negative Reviews: 0.7335786804353996
          Recall of Negative Reviews: 0.7056603773584905
          F-measure of Positive Reviews: 0.6603097517141135

In [124]: print('Precision of Neutral Reviews: {}'.format(precision_neut))
          print('Recall of Neutral Reviews: {}'.format(recall_neut))
          print('F-measure of Positive Reviews: {}'.format(f1_neut))

          Precision of Neutral Reviews: 0.7056603773584905
          Recall of Neutral Reviews: 0.7056603773584905
          F-measure of Positive Reviews: 0.6603097517141135

In [125]: avg_precision = (precision_pos+precision_neg+precision_neut)/3
          print('Average Precision:',avg_precision)

          avg_recall= (recall_pos+recall_neg+recall_neut)/3
          print('Average Recall:',avg_recall)

          avg_fmeasure = (f1_pos+f1_neg+f1_neut)/3
          print('Average F-measure:',avg_fmeasure)

          Average Precision: 0.735228822792
          Average Recall: 0.714465408805
          Average F-measure: 0.67061241946
```

Logistic Model gave a better performance in terms of accuracy as compared to the other two. The average accuracy of binary classification for Logistic Model for all the sentiments is **0.729**.

Aspect-level Analysis:

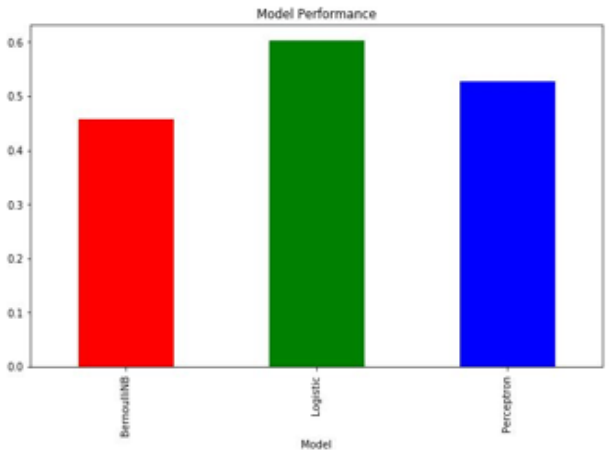
1. **At the document level:**
   An overall aspect level analysis at the document level was performed and below is the model performance (accuracy) and performance measure of precision, recall and f-measure

```
In [66]: results_all_bi = pd.DataFrame()
         tfprediction = {}
         cprediction = {}
         foldnum = 0
         for name,model in models.items():

                 model.fit(tr_features_all, Train_labels)
                 tfprediction[name] = model.predict(te_features_all)
                 tfaccuracy = metrics.accuracy_score(tfprediction[name],Test_labels)

                 results_all_bi.loc[foldnum,'TF-IDF Accuracy']=tfaccuracy
                 results_all_bi.loc[foldnum,'Model']=name
                 foldnum = foldnum+1
         print (results_all_bi)
```

```
   TF-IDF Accuracy        Model
0         0.458333  BernoulliNB
1         0.602273     Logistic
2         0.526515   Perceptron
```

A graphical representation of the model performance is as below:

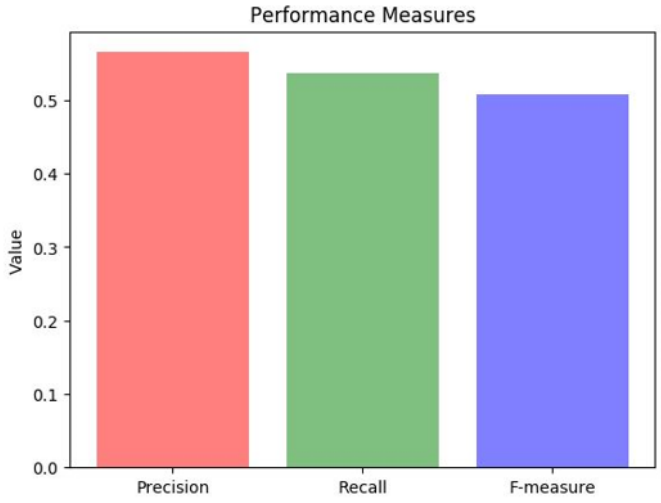
```
In [72]: results_all_bi.plot(kind='bar',color=['r','g','b'],title='Model Performance',figsize=(10,6),x='Model',legend=False)
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x1fc12b7cc50>
```



Find below the average performance measures of precision, recall and f-measure of the overall aspect analysis:

```
In [404]: from sklearn.metrics import *

In [405]: precision=precision_score(Test_labels, tfprediction, average='weighted')
          recall=recall_score(Test_labels, tfprediction, average='weighted')
          f1_score=f1_score(Test_labels, tfprediction, average='weighted')
          print('Precision:',precision)
          print('Recall:',recall)
          print('f1_score',f1_score)

          Precision: 0.565623007778
          Recall: 0.537878787879
          f1_score 0.507302518508
```

Find below the graphical representation of the above measures:



2. **Binary classification of aspects and the average performance predictions:**

In a similar way, the document was divided into train sets of annotated aspects (Food, Ambience, Service, Parking_Availability, Others) and overall aspect level classification was performed for which the model performance(accuracy) of each aspect was determined and an average performance measures of precision, recall and F-measure was calculated.
Individual aspects were annotated as follows, aspect predictions were made and performance(accuracy) of this prediction was measured based on the models.

```
In [458]: serv2_annotation_based=serv2[(serv2.annotation=='Service')|(serv2.annotation=='Not_Service')]
          serv2_annotation_based
```

Out[458]:

	sentence	annotation
0	gross anyone serve food pig pen	Service
1	would feed dogs nasty called restaurant	Not_Service
2	employees acting like high school running manager	Service

Similar train data was prepared for individual aspects from the entire document and predictions of the aspect was made. Models were used to check for accuracy of the predictions.
The data was divided into train and test data and aspect predictions were made on the test data.

```
# partitioning data for ASPECT - SERVICE

X_train_serv2, X_test_serv2, y_train_serv2, y_test_serv2 = train_test_split(serv2_annotation_based.sentence,
                                                                            serv2_annotation_based.annotation,test_size=0.25,
                                                                            random_state=1)
vect = CountVectorizer()
train_dtm_serv2 = vect.fit_transform(X_train_serv2)
test_dtm_serv2 = vect.transform(X_test_serv2)
```

Find below model performance(accuracy) of predictions of the aspect Service:

```
In [100]: #calculating accuracy, precision and recall for service. The next similar comment will calculate for others
          results_all_bi_serv2 = pd.DataFrame()

          #stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
          tfprediction_serv2 = {}
          cprediction_serv2 = {}
          foldnum_serv2 = 0
          for name,model in models.items():

                  model.fit(tr_features_all_serv2, y_train_serv2)
                  tfprediction_serv2[name] = model.predict(te_features_all_serv2)
                  #print(tfprediction[name])
                  tfaccuracy_serv2 = metrics.accuracy_score(tfprediction_serv2[name],y_test_serv2)

                  #model.fit(ctr_features_all,Train_labels)
                  #cprediction[name] = model.predict(cte_features_all)
                  #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

                  results_all_bi_serv2.loc[foldnum_serv2,'TF-IDF Accuracy']=tfaccuracy_serv2
                  #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
                  results_all_bi_serv2.loc[foldnum_serv2,'Model']=name
                  foldnum_serv2 = foldnum_serv2+1

                  #for 'Logistic',model IN models.items():
                      #print (tfaccuracy_serv2)
          print (results_all_bi_serv2)
          #print (tfaccuracy_serv2)
```

	TF-IDF Accuracy	Model
0	0.683019	BernoulliNB
1	0.762264	Logistic
2	0.709434	Perceptron

Similary, other aspects were annotated and divided into train and test data and predictions of the aspect were made after which the model performance was recorded. Find below the list of the model performance(accuracy) of performing binary classification or prediction of each aspect.
Model performance (accuracy) in predicting Aspect – Price

```
#calculating accuracy, precision and recall for PRICE. The next similar comment will calculate for others
results_all_bi_price2 = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_price2 = {}
cprediction_price2 = {}
foldnum_price2 = 0
for name,model in models.items():

        model.fit(tr_features_all_price2, y_train_price2)
        tfprediction_price2[name] = model.predict(te_features_all_price2)
        #print(tfprediction[name])
        tfaccuracy_price2 = metrics.accuracy_score(tfprediction_price2[name],y_test_price2)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_price2.loc[foldnum_price2,'TF-IDF Accuracy']=tfaccuracy_price2
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_price2.loc[foldnum_price2,'Model']=name
        foldnum_price2 = foldnum_price2+1

        #for 'Logistic',model IN in models.items():
            #print (tfaccuracy_serv2)
print (results_all_bi_price2)
#print (tfaccuracy_price2)
```

	TF-IDF Accuracy	Model
0	0.981132	BernoulliNB
1	0.984906	Logistic
2	0.977358	Perceptron

Model performance (accuracy) in predicting Aspect – Food

```
results_all_bi_food2 = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_food2 = {}
cprediction_food2 = {}
foldnum_food2 = 0
for name,model in models.items():

        model.fit(tr_features_all_food2, y_train_food2)
        tfprediction_food2[name] = model.predict(te_features_all_food2)
        #print(tfprediction[name])
        tfaccuracy_food2 = metrics.accuracy_score(tfprediction_food2[name],y_test_food2)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_food2.loc[foldnum_food2,'TF-IDF Accuracy']=tfaccuracy_food2
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_food2.loc[foldnum_food2,'Model']=name
        foldnum_food2 = foldnum_food2+1

        #for 'Logistic',model IN in models.items():
            #print (tfaccuracy_serv2)
print (results_all_bi_food2)
#print (tfaccuracy_food2)
```

	TF-IDF Accuracy	Model
0	0.807547	BernoulliNB
1	0.883019	Logistic
2	0.830189	Perceptron

Model performance (accuracy) in predicting Aspect – Ambience

```
#calculating model performance ambience.
results_all_bi_abm2 = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_abm2 = {}
cprediction_abm2 = {}
foldnum_abm2 = 0
for name,model in models.items():

        model.fit(tr_features_all_abm2, y_train_abm2)
        tfprediction_abm2[name] = model.predict(te_features_all_abm2)
        #print(tfprediction[name])
        tfaccuracy_abm2 = metrics.accuracy_score(tfprediction_abm2[name],y_test_abm2)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_abm2.loc[foldnum_abm2,'TF-IDF Accuracy']=tfaccuracy_abm2
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_abm2.loc[foldnum_abm2,'Model']=name
        foldnum_abm2 = foldnum_abm2+1

        #for 'Logistic',model IN models.items():
            #print (tfaccuracy_abm2)
print (results_all_bi_abm2)
#print (tfaccuracy_abm2)
```

```
   TF-IDF Accuracy        Model
0         0.962264  BernoulliNB
1         0.969811     Logistic
2         0.962264   Perceptron
0.962264150943
```

Model performance (accuracy) in predicting Aspect – Location

```
#predicting model performance for aspect - Location
results_all_bi_loc2 = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_loc2 = {}
cprediction_loc2 = {}
foldnum_loc2 = 0
for name,model in models.items():

        model.fit(tr_features_all_loc2, y_train_loc2)
        tfprediction_loc2[name] = model.predict(te_features_all_loc2)
        #print(tfprediction[name])
        tfaccuracy_loc2 = metrics.accuracy_score(tfprediction_loc2[name],y_test_loc2)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_loc2.loc[foldnum_loc2,'TF-IDF Accuracy']=tfaccuracy_loc2
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_loc2.loc[foldnum_loc2,'Model']=name
        foldnum_loc2 = foldnum_loc2+1

        #for 'Logistic',model IN models.items():
            #print (tfaccuracy_loc2)
print (results_all_bi_loc2)
#print (tfaccuracy_loc2)
```

```
   TF-IDF Accuracy        Model
0         0.932075  BernoulliNB
1         0.950943     Logistic
2         0.935849   Perceptron
```

Model performance (accuracy) in predicting Aspect – Others

```
#measuring model performance of aspect - Others
results_all_bi_other2 = pd.DataFrame()

#stratified = StratifiedKFold(labels, n_folds=5,random_state=20160121, shuffle=True)
tfprediction_other2 = {}
cprediction_other2 = {}
foldnum_other2 = 0
for name,model in models.items():

        model.fit(tr_features_all_other2, y_train_other2)
        tfprediction_other2[name] = model.predict(te_features_all_other2)
        #print(tfprediction[name])
        tfaccuracy_other2 = metrics.accuracy_score(tfprediction_other2[name],y_test_other2)

        #model.fit(ctr_features_all,Train_labels)
        #cprediction[name] = model.predict(cte_features_all)
        #caccuracy = metrics.accuracy_score(cprediction[name],Test_labels)

        results_all_bi_other2.loc[foldnum_other2,'TF-IDF Accuracy']=tfaccuracy_other2
        #results_all_bi.loc[foldnum,'Count Accuracy']=caccuracy
        results_all_bi_other2.loc[foldnum_other2,'Model']=name
        foldnum_other2 = foldnum_other2+1

        #for 'Logistic',model IN models.items():
            #print (tfaccuracy_other2)
print (results_all_bi_other2)
#print (tfaccuracy_other2)
```

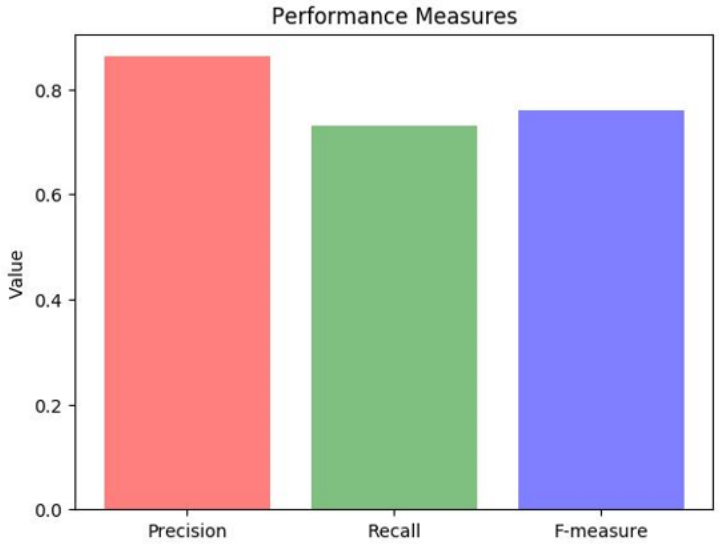
```
   TF-IDF Accuracy        Model
0         0.649057  BernoulliNB
1         0.716981     Logistic
2         0.683019   Perceptron
0.683018867925
```

Logistic Model gave a better performance in terms of accuracy as compared to the other two. The average accuracy of binary classification for Logistic Model for all the

aspects is **0.876**.
The average performance measures of Precision, Recall and F-measure of all the aspects are as below:

```
In [60]: avg_precision_MA = (precision_serv2_MA + precision_price2_MA + precision_food2_MA + precision_other2_MA
                             + precision_abm2_MA + precision_loc2_MA + precision_park2_MA)/7
         print('Average Precision:',avg_precision_MA)

         avg_recall_MA = (recall_serv2_MA + recall_price2_MA + recall_food2_MA + recall_other2_MA
                          + recall_abm2_MA + recall_loc2_MA + recall_park2_MA)/7
         print('Average Recall:',avg_recall_MA)

         avg_fmeasure_MA = (f1_serv2_MA + f1_price2_MA + f1_food2_MA + f1_other2_MA
                            + f1_abm2_MA + f1_loc2_MA + f1_park2_MA)/7
         print('Average F-measure:',avg_fmeasure_MA)

         Average Precision: 0.861996032953
         Average Recall: 0.731201536509
         Average F-measure: 0.759563606129
```

A graph representation of the average of above performance measures is as below:



Conclusion

Yelp is an American multinational corporation that focusses on public crowdsourced reviews. Yelp's reviews and ratings allow the yelpers to make informed decisions about a joint based on location. We speculate that our work performed by dividing the yelp reviews into relevant categories will help the users to make informed decisions based on their personal preferences of categories. On a review level and the feedback given by the Yelpers, and performing the sentiment analysis will help the Yelp organization to overcome their downsides with respect to Ambience or Location or Service or Parking or Food related categories. In this paper, we performed sentiment analysis at the document level and performed sentiment analysis by binary classification of

the sentiment. As per our analysis, the model performances of binary classification gave better performance as compared to multi-class classification. Sentiment classification at document level has the following performance measures, **Precision = 0.55, Recall = 0.57, F1-Score = 0.52**. Aspect classification at the document level has the following performance measures, **Precision = 0.56, Recall=0.53, F1-Score =0.50**. The performance measures identified that the binary classification performed on each of the above categories and sentiments performed better than the multi-class classification. Binary classification at aspect level resulted in the following performance measures, **Precision = 0.861, Recall = 0.731, F1-Score = 0.759**, whereas binary classification at the sentiment level, **Precision = 0.735, Recall = 0.7144, F1-Score = 0.67**.

Future Scope

Analyzing reviews for both at aspect and sentiment level pertaining to specific Taco joints (identified by unique business ids) shall enable Taco-bell joints to improve on the aspects analyzed for improvement. Reviews would be considered as whole rather than splitting into individual statements and sentiment of each individual review would be used to calculate the accuracy, precision, recall and F1-Score. Aspects and sentiments can be combined will help the Yelpers to make appropriate decisions while selecting Taco-Bell joint based on Location, Service, Food, Parking_Availability, and Service. This feedback will also enable the Yelp to improve their business based on the positive, negative, and neutral reviews.

References

1. Yun Xu,Xinhui Wu and Qinxia Wang," Sentiment Analysis of Yelp's Ratings Based on Text Reviews",Stanford University.

2. Subhabrata Mukherjee, Pushpak Bhattacharyya," Feature Specific Sentiment Analysis for Product Reviews", Dept. of Computer Science and Engineering, IIT Bombay.

3. Sahni, T., Chandak, C., Reddy, N., Singh, M. (2017, January 11). Efficient Twitter Sentiment Classification using Subjective Distant Supervision

4. Natural Language Toolkit. (n.d.). Retrieved April 04, 2017, from http://www.nltk.org/

5. 7.2. re — Regular expression operations. (n.d.). Retrieved April 04, 2017, from https://docs.python.org/2/library/re.html