Yelp: Are you a 5-star restaurant?

Kushal Gevaria(kgg5247) Hiteshi Shah(hss7374)

ABSTRACT

Sentiment analysis is an automated mining of user generated opinionated text data such as reviews, comments and feedback. This system classifies text data into their respective sentiments of positive, negative or neutral. Our project will be using this technique to classify text reviews obtained from Yelp dataset (which is available on Yelp[3]) into a star rating from 1 to 5. Since this dataset is very big with data from over 150,000 businesses, we will only be focusing on reviews for Mexican restaurants that are in the database. We want to predict what star rating the writer of the review is inclined to give the restaurant, based on just the text of their review.

For this project, we are using a feature selection model called Bag of Words to extract features from the given text and using those features to classify the reviews to their own star ratings. Using a number of classification algorithms, we are measuring the performances of each classifier by testing them on the test dataset. We are using a 70/30 split to divide our current dataset into training and test sets.

1. OVERVIEW

The dataset provided by Yelp is huge ($\sim$ 5GB), so extracting only the data that is relevant to the project was challenging and time-consuming. Another major challenge we had to overcome were mistakes in grammar and spelling in the reviews as well as the complexity of the English language. We also had to explore various feature selection algorithms since we knew it would have a major influence on the output.

In this internet era, people are keen on virtually sharing all their experiences and voicing their opinions online. Yelp allows users to write a review regarding their experience with a business, as well as give them a rating. We know that the nature of a single review could make or break a business. Using the data that has been aggregated by Yelp over the years, our goal is to learn to predict a star rating based on the review's text alone using existing classification algorithms, because text reviews can be hard for a computer to analyze and understand. Since the dataset already provided star ratings for businesses, we can use that to calculate the accuracy of the algorithms and select the one with the highest accuracy rate. This idea can be extended to many applications that are only text-based and don't provide a rating, such as a review article in an online newspaper for a movie, or YouTube videos where the viewers' comments can be considered to be the text reviews for the video or a simple web page with a recipe and a comment section. We will be using the Python library scikit-learn for splitting the data into training and test sets, classifying the text data, and evaluating the classifier.

2. DATASET

The dataset which we are using for this project has been obtained from the Yelp website[3]. Yelp provides an all-purpose dataset which can be used for learning and to implement many project ideas (Yelp Dataset Challenge). This dataset is available in two formats namely JSON and SQL. They have also provided documentation which helps us to figure out the structure of the dataset in those two formats. We will be using the dataset in the JSON format for this project.

business_id	name	review_id	stars	text
dwQEZBFen	Don Tequila	4RF8dMNBW	4	Enjoyed the bright fun Mexican decor!
dwQEZBFen	Don Tequila	ClgrKJ6dqiM7	4	I've been here at least 5 times now and
dwQEZBFen	Don Tequila	IBCTqmvwvd	5	Terrific service. The place was packed, b
dwQEZBFen	Don Tequila	69kni-xG6qtg	4	Ate here for lunch on a Sunday. Arrived
dwQEZBFen	Don Tequila	rQOasxLFCDI	5	Been dining here since it first opened. W
dwQEZBFen	Don Tequila	eGtRHSII_uN	5	If you have a hankering for Mexican, hea
dwQEZBFen	Don Tequila	O5UfgCBj4os	3	Kind of disappointed with the previous r
dwQEZBFen	Don Tequila	Dq7iR4uVsV	5	I eat here all the time and most of the st
dwQEZBFen	Don Tequila	H4ecwTdoO_	3	Inconsistent service, average food.....I've

Figure 1: Snippet of dataset containing text reviews for Mexican restaurants and their star ratings

Looking at the statistics of the data, there are around 156,000 businesses. Each of those businesses has 1.2 million business attributes. Since this is a lot of data to deal with, for the purposes of our project, we are going to focus only on Mexican restaurants and their reviews to predict the star

rating of a particular restaurant based on the thousands of text reviews it got. After extracting only the data that is relevant to our project from the whole Yelp dataset, we got a much compact dataset that looks like the one shown in the figure 1.

Here, the attribute "review-id" is unique in order to differentiate the reviews from one another, attribute "text" is the textual review for the restaurant with "business-id", and the attribute "star" is the numeric rating for that Mexican restaurant left by the same customer who left the text review.

We noticed that on an average, a Mexican restaurant on Yelp has 16 reviews, and there are 3900 such restaurants. This is more than enough data for training and testing our classification models.

The dataset we were dealing with, consists of approximately 270,000 reviews for Mexican restaurants. So we were desired to know what kind of words would mostly appear in the text reviews given to specifically those Mexican restaurants. This would help us to identify how the text reviews look like corresponding to the star rating they given along with that text review. So, we created a word cloud after compiling all the text reviews into a single file and using a python package "WordCloud" to use that file to come up with a word cloud giving us 2000 most frequently occurring words from the given text reviews [5]. Now, we know that articles and prepositions are used more often in an English sentence which is also called as stop-words. We need to get rid of these stop words in order to access the words which give meaning to the actual sentence. So, we need to perform a preprocessing cleaning step to remove these stop words from all the text reviews. Fortunately, the word cloud package in python provides a method which does this cleaning process. After we get rid of those stop-words, we are left with text reviews with some sensible words giving meaning it and finally, we can create a word cloud with 2000 most frequently occurring meaningful words. Given below is the word cloud created which is patched up with a yelp logo image so that each word that pops up, fit or align across the image boundaries and filling the space in between. This kind of feature is provided by the python "WordCloud" package as shown in the figure 2 [5, 2].

Figure 2: Word Cloud - Mexican Restaurants

Looking at the word cloud created for each review received by Mexican restaurants, we can figure out that most of the words that pop out with bigger font are positive in nature and also there are some words which clearly shows the type of restaurants that we are dealing with which is Mexican in our case. Some words like "Mexican food", "chips salsa", "fish taco", "carne asada" are used more often to describe the type of food customers order going to any Mexican restaurant. Some words like "happy hour", "great place", "love the place", "food good", "really good", "pretty-good" describes the positive nature of most of the review texts that are present in the given dataset. Also after cross-checking, we found out that there are actually more positive reviews than the negative ones. Thus, just looking at the word cloud, we were able to figure out the nature of our text review to perform any further analysis required to perform text classification. These words with their frequency can help to accurately determine if the text is positive(star rating 4 or 5) in nature, neutral(star rating 3) in nature or negative in nature(star rating 1 or 2). Since we know that most of the text reviews in the given dataset are positive in nature, the text classification will result in a maximum number of star rating in the range of 4-5.

Upon further data mining, we came up with graphs (in figures 3 to 7) that show the topmost words that frequented most reviews, given their respective ratings.

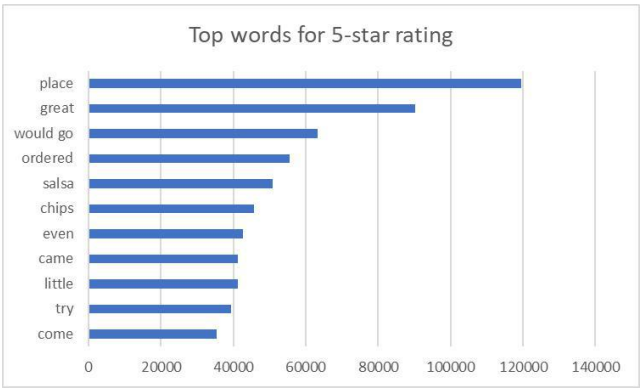

Figure 3: Top words for 5-star ratings

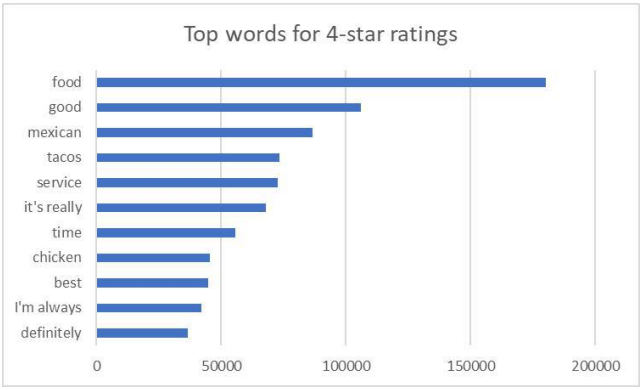

Figure 4: Top words for 4-star ratings

As we can see, surprisingly, this database has a lot more

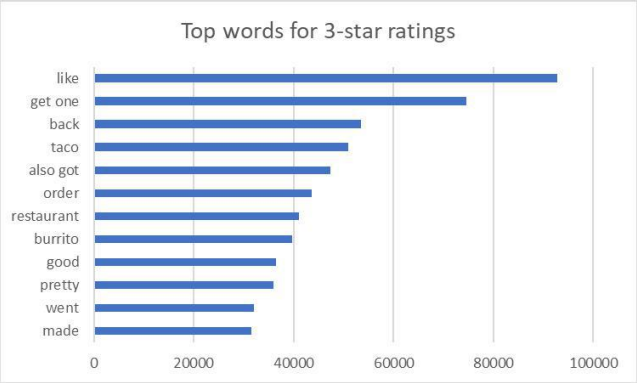

Figure 5: Top words for 3-star ratings

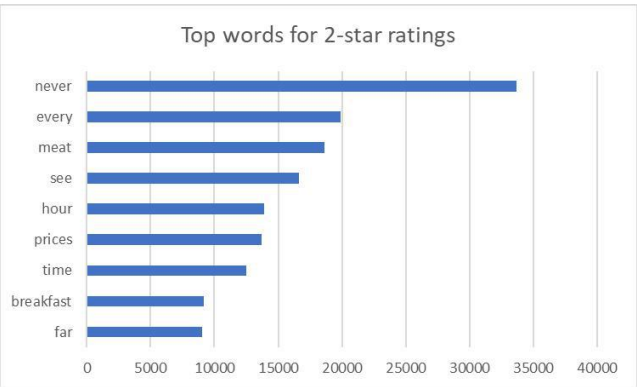

Figure 6: Top words for 2-star ratings

positive reviews (4- and 5-star ratings) than negative (1- and 2-star ratings).

3. PRE-PROCESSING

Sentiment analysis is nothing but the problem of text classification which classifies text sentences to positive or negative class based on the words of the text data and their polarity. One of the biggest problems in dealing with text data is that they are usually available in very high dimensions which may really affect the classifier performance. So there is a need for techniques which can help in reducing those features by removing or eliminating the features which are not relevant and keeping the features which are much important and which help to discriminate the sentence into available class labels which, in our case, are 1, 2, 3, 4, and 5 stars.

For our data pre-processing, we have removed all the punctuations and all the spaces from the review texts. We have also converted all text to lower case for redundancy in subsequent feature selection.

In order to eliminate irrelevant features, we have removed stop words, i.e., words that are frequently used but don't carry much meaning, such as "the", "a", "of", "for", "in" etc.[13]

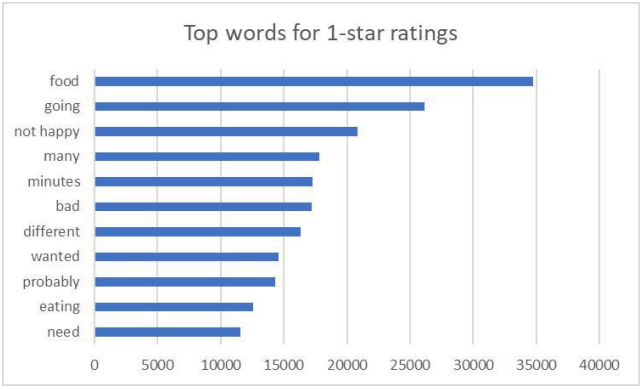

Figure 7: Top words for 1-star ratings

We have also performed stemming or lemmatization on the words. This is where we stripped the words of off their suffixes. For example, the words "looks", "looked", and "looking" will be converted to "look" after stemming. This was a complex process since there were some exceptional cases like "department" v/s "depart". To help with this process we availed the most commonly used stemmer called the Porter Stemmer. While this method was not perfect and did cause some mistakes, it helped tremendously in reducing the number of irrelevant features.

4. MODEL ANALYSIS AND EVALUATION

Since we are using sentiment analysis to extract opinions from the text reviews of a particular restaurant, we used the Bag of Words model to convert this text into features that can be used by supervised learning algorithms for classification.

The Bag of Words model is generally used to represent a given sentence in a list of words. This model will give us most frequent unigrams, or bigrams that are used in reviews for a particular restaurant. Based on these n-grams[1], we can predict the star-rating of the review:

- Unigram Model: In this model, the whole sentence is divided into its individual number of words, and those words are each used as features. For example, the sentence âĂIJit is a good movieâĂİ will output the feature set it, is, a, good, movie.
- Bigram Model: This model is similar to the Unigram model, except, instead of working with single words, it uses a combination of two words to create the feature vector. For example, the sentence "it is not bad movie" will output the feature set it is, is not, not bad, bad movie.
- N-Gram Model: The N-gram model uses a combination of the other models at the same time. For example, if it uses unigrams as well as bigrams, the sentence "it is a movie" will output the feature set it, is, a, movie, it is, is a, a movie.

After extracting the new features by using the above models, we used some classification algorithms like Naive Bayes, Support Vector Machines, and Maximum Entropy to predict the star rating for the restaurants in the test set.

5. IMPLEMENTATION

5.1 Bag of Words Model

As humans, we can just look at a sentence and figure out how positive or negative the sentence is. For example, consider the sample review below for one particular restaurant from the given dataset of Mexican restaurants.

Text Review - "Enjoyed the bright fun Mexican decor! The food was delicious and reasonably priced! And the margaritas were delicious"[3].

In this text review, we can see there are many positive words like "Enjoyed", "fun", "delicious" etc. These words give positive meaning to the review. Therefore, just by extracting some meaningful words from the sentence we can get to know if the sentence or paragraph of review is positive, negative or neutral containing mixed reviews. This is the idea of the Bag of Words model.

Bag of Words is one of the simplest model used in natural language processing as a tool of feature generation for doing sentiment analysis and also to perform text classification. Using this method, we can give a score to the sentence. The range of the scores can be divided into five parts for each of the star-rating classes ,i.e., 1, 2, 3, 4, and 5. Doing so, we generate a feature which can help us to identify the overall numeric rating based on the summation of score values. Thus, it ignores the context or the grammar of the sentence or paragraph which can cause a major issue and erroneous classification of the given text review to its respective class.

What if a word starts with a negation? For example, let us consider the above example itself. If the positive word "fun" or "delicious" starts with a negative word like "no fun" or "not delicious" respectively, then it would completely change the meaning of the sentence and so the review as well. So a unigram model would consider this review as positive and not negative which is the wrong classification of the text review.

To avoid this, we used a unigrams + bigrams model [10]. In this particular model, if a word starts with a negative word like "not", "no" etc., then we do not split them and keep it as a single entity to give a score to. So, for this project, we have tried both - a Bag of Words model with unigrams as well as a Bag of Words model with unigrams + bigrams - and compared their accuracies. The data cleaning that we have performed for this model is removing prepositions and articles like "is", "the", "of" etc. which are used just for grammatical purposes and do not give any contextual meaning to the sentence. These words occur the most but are not considered valuable while doing text classification or sentiment analysis. Our Bag of Words model has also ignored proper nouns like "Andrew", "James", "New York" etc. when extracting the feature from given text review.

5.2 Naive Bayes Classification

The goal of this project is to predict the star-rating of a Mexican restaurant based solely on the text reviews left for the restaurants on Yelp by its customers. In order to do this, we first trained our classification model using the reviews and their classifications in the training dataset and then tried to predict the star-rating for the reviews in the test dataset. The classification model that we'll be focusing on in this section is the Naive Bayes classifier.

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)} \tag{1}$$

where C is a class and x is a set of observations[11].

Here, we are trying to determine if a particular review has 1, 2, 3, 4 or 5 stars. Our set of observations here is one long string in the form of a review. Since the Naive Bayes theorem requires discrete features, the easiest way to generate features is using the Bag of Words model.

Because of this feature extraction, the denominator P(x) will be constant for all the classes. This will affect all the probabilities equally. So we will eliminate the denominator. Finally, we have to calculate the probability of each class and the probabilities of each of the extracted features with respect to their class values [6].

For this, we will count the number of occurrences of each word in each of the reviews that belongs to a particular class. For example, we will count the occurrence of the word "fun" in all the reviews that have a 5-star rating in the training dataset. We will do the same for each of the other numeric ratings that contain the word "fun" in their text reviews. This will allow us to compute the probabilities of each feature given a class.

So, now that we have the word counts, we just have to compute the probabilities and multiply them to get the probability of the class being predicted. Again, let's say that we want to find the probability that a review has a 5-star rating. We found the number of times a word in the review occurred in each of the 5-star ratings and divide it by the total number of 5-star ratings to get its probability. We then multiplied such probabilities of the all the words in the review and also multiplied the probability of a review having a 5-star rating, to get the final probability of the review being a 5-star review.

We did the same for all the other star-ratings and whichever probability is the greatest would be the class that the review is assigned to.

We then trained and tested this classification model using 70% of the data for training and 30% of the data for testing. Getting good results on just the training set could mean that the model suffers from over-fitting and is just picked up random noise. Therefore, it is important to test the model on never-seen-before data to gauge its accuracy correctly.

Consider the example of a restaurant that only has five reviews as mentioned in Table 1.

Table 1: Text Review to Star-rating Map

Text review	Star-rating
"had a lot of fun"	5
"did not have any fun"	1
"food was bland"	2
"service was excellent"	4
"atmosphere was fun"	5

For this set of reviews, we get the probabilities given in Table 2.

Table 2: Text Review to Star-rating Map

Word	Class	Occurrences of Word in Class	Probability of Class
"fun"	1	1/5	1/5
	2	0	1/5
	3	0	1/5
	4	0	0
	5	2/5	2/5

From this we can see that if a review in the testing dataset contains the word fun, it would be classified as having a 5-star rating.

5.3 Maximum Entropy Text Classification

The goal of this project is to predict the star-rating of a Mexican restaurant based solely on the text reviews left for the restaurants on Yelp by its customers. In order to do this, we will first train our classification model using the reviews and their classifications in the training dataset and then try to predict the star-rating for the reviews in the test dataset. The classification model that we'll be focusing on in this section is the Maximum Entropy (or Logistic Regression) classifier.

Unlike the Naive Bayes Classifier, Maximum Entropy does not assume that the features are conditionally independent. It uses gradient descent to calculate the weights of each of the vectors.

For the Maximum Entropy classifier, we need to first figure out the relevant features. This is done using Stop-words Removal and the Bag of Words model.[9].

We also need to calculate the weights of each of these features. These weights are initially zero for all the features, and the values are computed using the gradient descent algorithm. Once we have the values for these weights, we compute the probability for each class and the class with the maximum probability is chosen for the current text review:

1. For each word w and class $c \in C$, we define a joint feature $f(w, c) = N$ where N is the number of times that w occurs in a review in class c.

2. Through iteration, we assign a weight to each joint feature so as to maximize the log-likelihood of the training data.

3. The probability of class c given a review d and weight vectors λ is

$$P(c|d) = \frac{exp \sum_i \lambda_i f_i(c, d)}{\sum_{c' \in C} exp \sum_i \lambda_i f_i(c', d)} \quad (2)$$

Here, λ decides the significance of each feature in the classification.

5.4 Text Classification using Support Vector Machine

Support vector machine is one of the examples of linear classification algorithms where we can classify the dataset into two separate classed using this algorithm[12]. It tries to create a hyperplane that tries to separate the given dataset into two classes with maximum optimality. In our case let's say if we have text reviews which can either be positive in nature or negative in nature, then support vector machine will create a hyperplane that will separate all the positive review texts with the positive label to all the negative review texts with the negative label. This hyperplane will maximize the distance between positive review texts and negative review texts[8].

It is often said that support vector machine works well to classify the dataset into two classes only[8]. But what if we need to classify the text reviews in terms of three classes, i.e., positive, neutral and negative, or in terms of ratings in the range from 1-5. In our case, the dataset is not linearly separable, since we are trying to predict the star ratings based on the text reviews. This can be achieved using support vector machine with the help of the kernel function. Since, for two classes with 2-dimensional data, we need to create a single hyperplane mapped to that 2-dimensional data. But if we need to classify the given dataset in more than two classes, then we can use the kernel function to map the hyperplane created to more than 2-dimensional data which is not linearly separable. This mapping of hyperplane to higher dimensional dataset will help us classify the given dataset to three classes or five classes based on our requirements[8].

Since we need to classify dataset based on 3 classes (positive, neutral, negative) or 5 classes(star ratings from 1-5), usually the dataset is reduced to fit the binary classes which work perfectly well for support vector machine. Doing so includes two approaches which are one v/s all and one v/s one[8]. For the prediction based on three classes, support vector machine creates three hyperplanes which are represented as positive text reviews against others, negative text reviews against others and neutral text reviews against others. Similarly, for the prediction based on five classes, 5 hyperplanes are created using support vector machine algorithm to differentiate between text reviews of star rating from range 1-5.

To perform this support vector implementation to do classification based on text reviews, python provides a method

called "SGDClassifier" under sci-kit learn package under the section of "linear-model". Classification is performed for both cases, i.e., 3 classes, as well as 5 classes and the results, are evaluated and compared with other classification models later in the results section and comparison section respectively.

6. RELATED WORK

The idea and interest to go for this project came after looking at the similar work done by others on the given Yelp dataset. Given below is the summary of the work and analysis done by others on the yelp dataset to perform text classification just based on the reviews for the restaurants.

6.1 Yelp Explorers[7]

This report shows work similar to what we are aiming for. Based on just the text reviews, they were able to classify the star ratings for a particular restaurant. They used sentiment analysis using models like bag of words model and Doc2Vec model and found out best mapping for a vector of words from review to the star rating for a given restaurant.The data for restaurants and reviews were obtained from reviews.json and business.json files and they used the Pandas library in Python to store them for future evaluation. Before starting the sentiment analysis and classification, they visualized the review data by creating a word cloud. A word cloud is used to get the most frequent words used in the reviews. These word counts can help to identify the nature of most of the text reviews. Looking at the word cloud, we can see that most of the words are positive, which means most of the customers have given positive reviews to the restaurants they have been to.

Figure 8: Word Cloud[7]

They also generated a pie chart which represents the distribution of rating. Looking at that pie, they were able to figure out that most of the dataset contained positive reviews since, most of the ratings were in range 4-5. This kind of visualization helped them to identify the dataset their working on before actually performing sentiment analysis and classification on the same.

They also used a scatterplot to show number of words in a review with respect to the star rating it achieved. But that did not make any significant difference. This means that number of words in a particular review did not help in identifying the rating of the review.

To featurize the Yelp dataset for restaurants and their text reviews, they used two featurization algorithms, namely bag of words and Doc2Vec. Bag of words ignores grammar and context of sentence which is most important thing when we consider sentiment analysis. To overcome this, they have used Doc2Vec featurization technique as a two layer neural network. This provided them which better features than the bag of words model.

In the bag of words model, they first used word stemmer and word lemmatizer to reduce the words obtained from reviews to only the important words. They used a dictionary to map 2000 unigrams that they generated from the text reviews with respect to the count of occurrence of those many unigrams. Since unigrams alone cannot help to get any phrase if used in reviews, they also stored most frequent 1000 bigrams and 500 trigrams. After generating unigrams, bigrams and trigrams, they created feature matrix containing feature vector for each restaurant.

In the Doc2Vec approach, they trained the model and split the dataset into three parts namely training, testing and validation set. They used python's gensim package to create a tag for each word in the review given to a particular restaurant. After doing so, feature vector was obtained easily by getting all the words having same tag name.

After doing feature selection, they used three classification algorithms, namely Logistic Regression, SVM, and Random Forest Classifier. Looking at the results obtained for each of those classification algorithms, it was clear that the Random Forest Classifier did not perform well as compared to logistic regression and SVM. They used means square error to measure the performance of those three classification algorithms. As the number of features increases, logistic regression and SVM show improvement with low mean square error which is not the case with random forest. Also, after using 1000 features, those two classification algorithms show steady performance. This proves that using all 3500 features does not help in improving the classification algorithms.

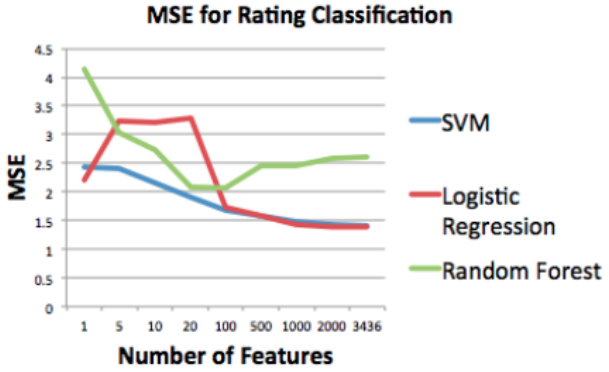

Figure 9: Mean square error with respect to number of features used[7]

They used some hyper-parameters for the three classification algorithms using just 1000 features which are given below:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
        max_depth=None, max_features='auto', max_leaf_nodes=None,
        min_samples_leaf=1, min_samples_split=2,
        min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
        oob_score=False, random_state=None, verbose=0,
        warm_start=False)

LogisticRegression(C=100000.0, class_weight=None, dual=False,
       fit_intercept=True, intercept_scaling=1, max_iter=100,
       multi_class='ovr', penalty='l2', random_state=None,
       solver='liblinear', tol=0.0001, verbose=0)

LinearSVC(C=1e-05, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
```

Figure 10: Hyper-Parameters adjusted for each algorithm[7]

After tuning the hyper-parameters and using just 1000 features, they compared the performance between given three classification algorithms. SVM perform the best 88% accuracy using Doc2Vec feature while SVM with bag of words feature was around 87%. So there was not a great improvement after changing from bag of words model for feature selection to Doc2Vec model.

They used different python packages for their project to use all those classification algorithms, pre-processing required like tokenization, stemming etc. Python package Scikit-learn provided mostly all classification algorithms. They used CountVectorizer for creating n-grams and feature matrix used in bag of words model. They had also built a web application where one can paste the text review for a particular restaurant, and based on the words and context used in the text review it can predict the star rating in the range of 1-5.

6.2 Naive Bayes Classification for Sentiment Analysis of Movie Reviews[4]

This project has used the multinomial Naive Bayes classification algorithm to classify movie reviews according to their overall sentiment (positive/negative). They've used the bag of words representation and train a multinomial Naive Bayes classifier on the data and test the model's performance, using IMDB movie reviews data which contains 2000 reviews, each with a positive or negative sentiment label. They have implemented the project in R.

In the bag-of-words approach, they have represented each word in a review as a feature and each review as a vector of features. They focus only on the number of occurrences of each word i.e., represent each review as a multi-set 'bag' of words.

For preprocessing, they cleaned up the words by eliminating numbers, punctuation, white space, and by converting to lower case. In addition, they've also discarded common stop words such as "his", "our", "hadn't", "couldn't", etc.

They have represented the bag of words tokens with a document term matrix (DTM). The rows of the DTM correspond to reviews in the collection, columns correspond to terms, and its elements are the term frequencies. For example,

Terms Docs	aardman	aaron	aatish	aback	abandon	abandoned
40	0	0	0	0	0	0
41	0	0	0	0	0	0
42	0	0	0	0	0	0
43	0	0	0	0	0	0
44	0	0	0	0	0	0
45	0	0	0	0	0	0
46	0	0	0	0	0	0
47	0	0	0	0	0	0
48	0	0	0	0	0	0
49	0	0	0	0	0	0
50	0	0	0	0	0	0

Figure 11: Document Term Matrix[4]

To reduce the number of features, they ignored words which appear in less than five reviews.

They then applied the multinomial Naive Bayes algorithm to the matrix. In this method, the term frequencies are replaced by Boolean presence/absence features. The logic behind this being that for sentiment classification, word occurrence matters more than word frequency.

Using 75% of the data for training and the remaining 25% for testing, they got an accuracy of 81%.

The main challenges they faced in this project was the preprocessing of data, i.e., removing stop words and stemming. They also felt like they could've got a better accuracy had they used n-grams instead of using just unigrams (one-word occurrences).

7. RESULTS

We have used 70% of the data for training and 30% of the data for testing. Getting good results on just the training set could mean that the model suffers from over-fitting and is just picked up random noise. Therefore, it is important to test the model on never-seen-before data to gauge its accuracy correctly. The results for all three classification models are given in Table 3.

Table 3: Results of all the classifiers using 5 classes

Classifier	Accuracy	Precision	Recall
Naive Bayes	59%	56%	57%
SVM	59%	58%	60%
Maximum Entropy	41%	38%	40%

We felt that these models could be improved by grouping some of the classification labels, since there is quite a bit of discrepancy between the number of positive and negative reviews. So we decided to group together the reviews with ratings 1 and 2 as "negative", ratings 3 as "neutral" and ratings 4 and 5 as "positive". The results of using these new classification labels are given in Table 4.

As we can see, using 3 classes instead of 5, made a massive difference in the classification results. To avoid having to code the whole algorithms every time just to make a few

Table 4: Results of all the classifiers using 3 classes

Classifier	Accuracy	Precision	Recall
Naive Bayes	82%	80%	81%
SVM	85%	82%	84%
Maximum Entropy	64%	63%	61%

adjustments, we used the classifier implementations in the scikit-learn library in Python. Scikit-learn is a library in Python that contains implementations for all the common machine learning algorithms.

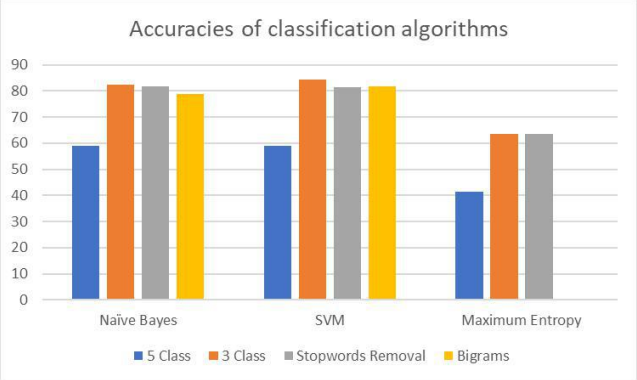

Figure 12: Comparison of all three classifiers

8. PROBLEMS AND CHALLENGES

Most of the challenges involved while going through this project of sentiment analysis was during the preprocessing step. Since we are dealing with a huge chunk of text data, we had to perform a lot of text cleaning process to get the most sensible words out of those sentences that give meaning to the text reviews. For example articles and prepositions like "is", "the", "of", "a" would be considered as nothing while evaluating the sentiments of the text. Even after we remove these unnecessary words from the text reviews, we had to extract the feature from these reviews which is nothing but sensible words that depicts the nature of the sentence. For example words like "great", "awesome", "best" etc. would give positive meaning to the review while words like "worst", "bad", "abysmal" etc. would give negative meaning to the review. This word identification was performed well with the help of the bag of words model. But what if the word starts with a negation? For example, if all those positive words start with a "not" in front, then it would completely change the meaning of the review. So, we also had to evaluate the feature not just based on unigrams, but also based on bigrams. So our bag of words model was created using a unigrams plus bigrams feature extraction procedure.

Now, during the classification process, the doc2vec using logistic regression classification model did not work well with our huge dataset. The accuracy of this model was very low and we were not able to figure out the reason behind it. This may be due to the fact that the reviews are written by the everyday general people and not by the professional writers which will affect this algorithm since doc2vec works on the semantics of the entire sentence as a whole and not just on the words used alone.

Also, the classification algorithms did not work well to classify dataset into 5 classes (star ratings from 1-5) since there is no direct correlation between text reviews with rating 4 and text reviews with rating 5. For example, there are some customers who give 5 as star rating to the restaurant with a text review which does not contain words that justifies the rating given. The similar case happens for the negative ratings as well. It is very hard to identify such cases and classify the dataset into classes like 4 and 5, or 1 and 2 which are very closely related to each other based on the text reviews. Thus, it was convenient to move forward with classification based on just three classes, i.e., positive, neutral and negative which give significant improvement in the prediction.

Even, after classifying based on three classes, there can be a situation where the text review given by a customer was sarcastic in nature. This is clearly an out-lier where for example, a star rating given is 1 or 2 and the text review given by the same customer contains all the positive words giving positive meaning to the review. It is very hard to classify such an out-lier which requires an application of sarcasm detection to identify such cases and eliminate them. These are the problems and challenges faced during this project.

9. REFERENCES

[1] G. S. Hitesh H Parmar, Sanjay Bhanderi. Sentiment mining of movie reviews using random forest with tuned hyperparameters.

[2] Y. Inc. Yelp logo. 2004-2017.

[3] Y. Inc. Yelp open dataset. 2004-2017.

[4] R. Katti. Naive bayes classification for sentiment analysis of movie reviews. April 2016.

[5] A. Mueller. Word cloud python package.

[6] V. Paruchuri. Naive bayes: Predicting movie review sentiment. March 2015.

[7] S. Parul, S. Abhinava, P. Ilakya, and C. Mukund. Yelp explorers. *ACM Trans. Program. Lang. Syst.*, December 2015.

[8] A. Taspinar. Text classification and sentiment analysis - section svm. November 2015.

[9] A. Taspinar. Regression, logistic regression and maximum entropy. March 2016.

[10] A. Taspinar. Sentiment analysis with bag-of-words. January 2016.

[11] A. Taspinar. Sentiment analysis with the naive bayes classifier. February 2016.

[12] Wikipedia. Support vector machine.

[13] X. Zhu. Basic text process. 2010.