Machine Learning Engineer Nanodegree

Capstone Project

Mahesh Narasimhan
Jan 14th, 2018

I. Definition

Project Overview

Yelp, which is an online marketplace for restaurant reviews has collected various data points about restaurants, users, reviews. This data about the usage patterns of restaurant reviews can be utilized to predict sentiment about the reviews. Yelp had posted this on Kaggle as a Business Challenge. Using review data effectively can help organizations increase quality and read sentiment of users thereby increasing sales, user experience, customer retention and customer satisfaction. NLP techniques can help organizations attain useful predictions using these data.

The motivation for pursuing this project is to understand how to work on real world datasets and challenges that companies like Yelp consider to be important and valuable.

Similar work in this field:

1. Opinion Mining and Sentiment Analysis
   http://www.cs.cornell.edu/home/llee/omsa/omsa-published.pdf
2. Evolution on sentiment analysis
   http://www.cs.cornell.edu/home/llee/omsa/omsa.pdf

Problem Statement

Attempt to classify Yelp reviews based on text content. Each observation is review by a particular user. Higher the star better the review

We aim to tackle the problem of sentiment polarity categorization, which is one of the fundamental problems of sentiment analysis.

This type of problem is challenging because you usually can't solve it by looking at individual words. No single word is going to tell you whether the review is positive or negative. And that is the main reason for choosing this problem, I believe that helping to solve this task might bring some enlightenment to other critical NLP tasks.

My goal is to build a sentiment analyser model that predicts whether a user liked a local business or not(output), based on their review on Yelp(input) using NLP techniques and ML algorithms (Multinomial NB).

File description, Datasets and Inputs

https://www.kaggle.com/c/yelp-recsys-2013#description

We will use the Review dataset and attempt to classify them Our data contains 10,000 reviews, with the following information for each one:

Data Format
{
'type': 'review',
'business_id': (encrypted business id),
'user_id': (encrypted user id),
'stars': (star rating),
'text': (review text),
'date': (date, formatted like '2012-03-14', %Y-%m-%d in strptime notation), 'votes': {'useful': (count), 'funny': (count), 'cool': (count)}
}
Summary Statistics:
In the training set:
11,537 businesses
8,282 checkin sets
43,873 users
10000 reviews

Metrics:

We use a confusion matrix since it provides a summary of prediction results on a classification problem. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. The CF matrix provides number of correct and incorrect predictions that are summarized with count values and broken down by each class. This is the key to the confusion matrix.

The evaluation metrics for this model will be the confusion matrix, precision, recall, and error rate. The terms are defined as follows:

- True positives: Entries that are correctly labelled
- False positives: Entries that a wrongly identified with a given label
- False negatives: Entries for a given label that are wrongly identified with

other labels.

It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

Elements along the diagonal represent a correct classification, whereas the off-diagonal represent a misclassification and we use a Classification report to evaluate a model's predictive power, which provides precision, recall, f1 score and support.

Precision is the result of the number of true positives divided by the sum of true positives and false negatives. This can be given by the following equation,
precision = tp / (tp + fp) where tp and fp stand for true positive and false positive respectively

Recall is the result of dividing the true positives by the sum of true positives with false negatives. This can be given as follows,
recall = tp / (tp + fn) where tp and fn stand for true positive and false negative respectively.

From this, the error rate or rate at which the classifier wrongly classifies the samples can be derived, error rate = 1 - (tp / (tp + fn)) = fn / (tp + fn) which is also known as the false negative rate.

The error rate as well as all the other metrics discussed in this section will be calculated using the SciKit-Learn metrics.classification_report and metrics.confusion_matrix methods.

II. Analysis

Data Exploration
Our data contains 10,000 reviews, with the following information for each one:
Since we are trying to predict how a user will rate a business, the only information in a Review object is the user_id and business_id.

1,205 businesses
734 checkin sets
5,105 users
10000 reviews to predict

We split into a training and a test set using train_test_split from Scikit-learn. We will use 30% of the dataset for testing.

Basic insight into the dataset

Data columns (total 10 columns):
business_id 10000 non-null object
date 10000 non-null object

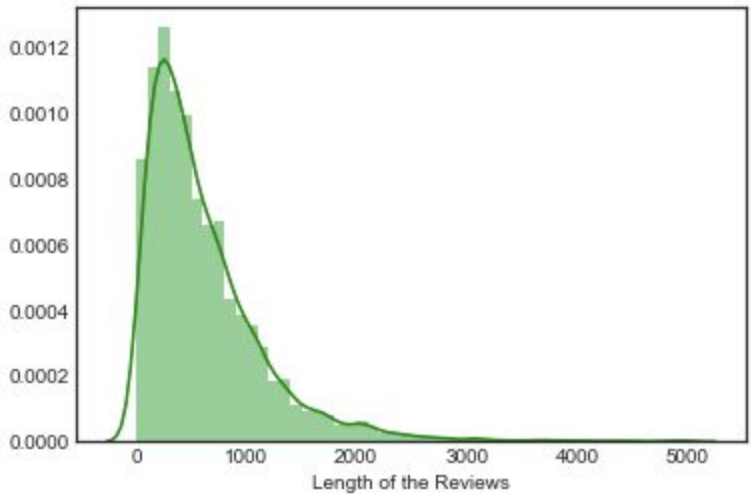
review_id 10000 non-null object
stars 10000 non-null int64
text 10000 non-null object
type 10000 non-null object
user_id 10000 non-null object
cool 10000 non-null int64
useful 10000 non-null int64
funny 10000 non-null int64

business_id	date	review_id	stars	text	type	user_id	cool	useful	funny

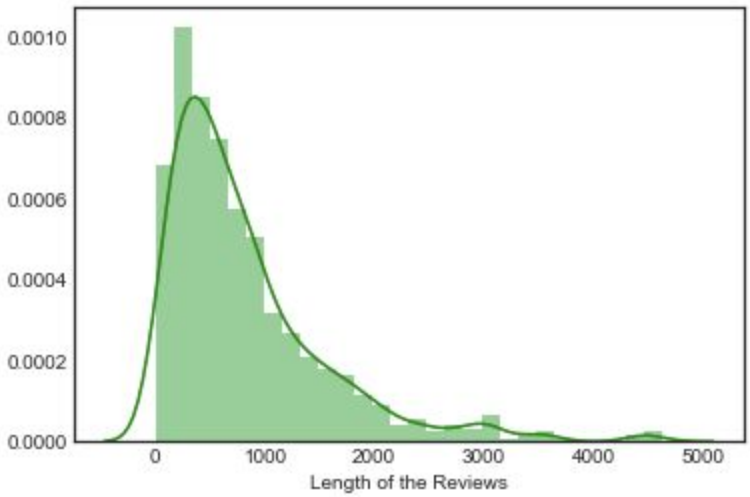
We are interested in predicting only positive and negative and hence let's consider 1 star and 5 start ratings and filter the datasets with these ratings.

I observed the distribution of lengths of positive and negative reviews has the potential to skew our model when training our algorithms. Thankfully our data had an identical number of positive and negative scores with similar distributions of the length of reviews

Distribution of positive scores



Distribution of Negative scores

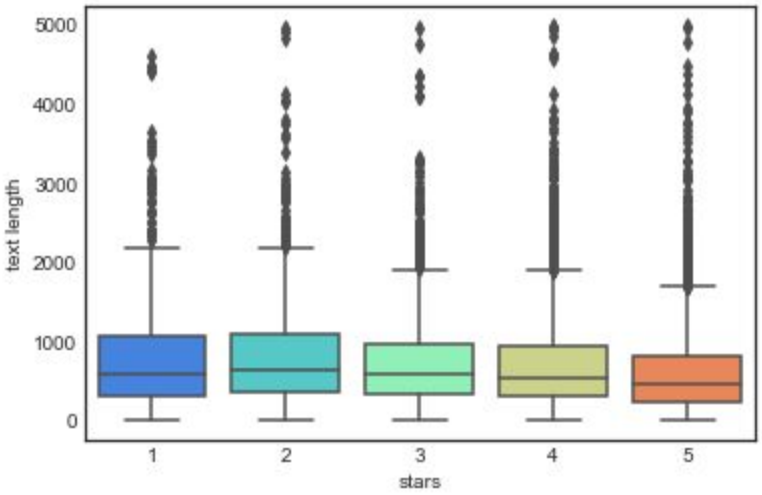


We need to check for outliers. Let's use a box plot to for each star rating. From the plot below, looks like 1-star and 2-star ratings have much longer text, but there are many outliers (which can be seen as points above the boxes).

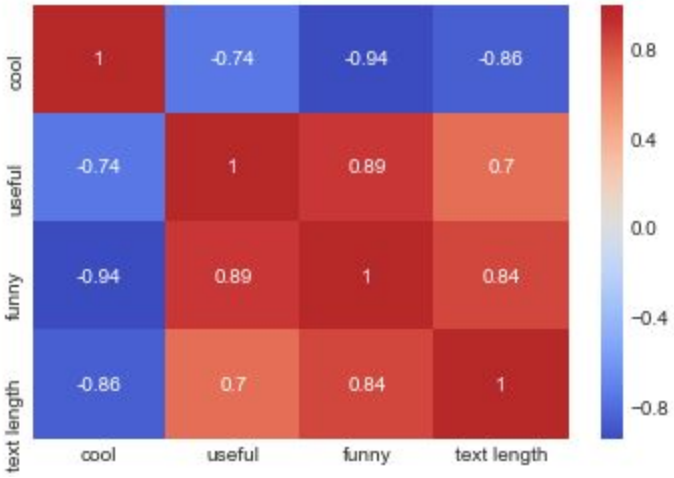
Create a boxplot of text length for each star category.

```
sns.boxplot(x='stars',y='text length',data=yelp,palette='rainbow'
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a24ceed30>
```



Let's group the data by the star rating and use heat map, and see if we can find a correlation between features such as cool, useful, and funny.



From the map, funny is strongly correlated with useful, and useful seems strongly correlated with text length. We can also see a negative correlation between cool and the other three features.

Algorithms and Techniques

We will use Multinomial Naive bayes and Logistic regression as algorithms for our data.

Multinomial Naive bayes:

We use Multinomial Naive Bayes is a specialized version of Naive Bayes that is designed more for text documents. Whereas simple naive Bayes would model a document as the presence and absence of particular words, multinomial naive bayes explicitly models the word counts and adjusts the underlying calculations to deal with in.

Multinomial captures word frequency in texts. Please refer on application of different Baye's theorem Multinomial Bayes

Logistic Regression

Logistic Regression is an algorithm that can be trained as a classifier and uses a linear decision boundary as a means of classifying a set of features.

Being that Logistic Regression is a discriminative as opposed to a generative supervised learning algorithm,it is a great choice to contrast against Naive Bayes. Logistic Regression is derived from Linear Regression and has a very similar cost function. Additionally, Logistic Regression is a great first discriminative classification algorithm to learn and can be used to classify multivariate data as well as fit polynomial terms to find a decision boundary.

Benchmark Model

This is a dataset available in Kaggle and lot of results have been published. Teams have used rule based Sentiment analysis tools and various natural language tool kits. We will use a NB classifier and compare with other models such as Sentiwordnet . Recently, teams have experimented with end-to-end deep learning solutions.

For benchmarking, against a secondary algorithm, Logistic Regression is an algorithm that can be trained as a classifier and uses a linear decision boundary as a means of classifying a set of features.

Logistic Regression is derived from Linear Regression and has a very similar cost function. Additionally, Logistic Regression is a great first classification algorithm to learn and can be used to classify multivariate data as well as fit polynomial terms to find a decision boundary.

III. **Methodology**

Data Preprocessing

We create a dataframe called yelp_class that contains the columns of yelp dataframe but for only the 1 or 5 star reviews.

We can see from .shape that yelp_class only has 4086 reviews, compared to the 10,000 reviews in the original dataset. This is because we aren’t taking into account the reviews rated 2, 3, and 4 stars.

Cleaning the data
We perform this in three steps

1. Remove punctuation
2. Stop words removal
3. Tokenization

Text- Preprocessing:

a. The simplest way to convert a corpus to a vector format is the **bag-of-words** approach, where each unique word in a text will be represented by one number
b. We remove punctuation from the reviews text and return a list of tokens to be passed to next step

Stop words removal:

a. To improve accuracy, I am also looking including, Stopword Removal.

Tokenization:

a. We can use Scikit-learn's CountVectorizer to convert the text collection into a matrix of token counts.
b. At the moment, we have our reviews as lists of tokens. To enable Scikit-learn algorithms to work on our text, we need to convert each review into a vector.Import CountVectorizer and create a CountVectorizer object

Example:

Before punctuation and stop words removal

My wife took me here on my birthday for breakfast and it was excellent. The weather was perfect which made sitting outside overlooking their grounds an absolute pleasure. Our waitress was excellent and our food arrived quickly on the semi-busy Saturday morning. It looked like the place fills up pretty quickly so the earlier you get here the better.\n\nDo yourself a favor and get their Bloody Mary. It was phenomenal and simply the best I\'ve ever had. I\'m pretty sure they only use ingredients from their garden and blend them fresh when you order it. It was amazing.\n\nWhile EVERYTHING on the menu looks excellent, I had the white truffle scrambled eggs vegetable skillet and it was tasty and delicious. It came with 2 pieces of their griddled bread with was amazing and it absolutely made the meal complete. It was the best "toast" I\'ve ever had.\n\nAnyway, I can\'t wait to go back!

After punctuation and stop words removal

['wife', 'took', 'birthday', 'breakfast', 'excellent', 'weather', 'perfect', 'made', 'sitting', 'outside', 'overlooking', 'grounds', 'absolute', 'pleasure', 'waitress', 'excellent', 'food', 'arrived', 'quickly', 'semibusy', 'Saturday', 'morning', 'looked', 'like', 'place', 'fills', 'pretty', 'quickly', 'earlier', 'get', 'better', 'favor', 'get', 'Bloody', 'Mary', 'phenomenal', 'simply', 'best', 'Ive', 'ever', 'Im', 'pretty', 'sure', 'use',

'ingredients', 'garden', 'blend', 'fresh', 'order', 'amazing', 'EVERYTHING', 'menu', 'looks', 'excellent', 'white', 'truffle', 'scrambled', 'eggs', 'vegetable', 'skillet', 'tasty', 'delicious', 'came', '2', 'pieces', 'griddled', 'bread', 'amazing', 'absolutely', 'made', 'meal', 'complete', 'best', 'toast', 'Ive', 'ever', 'Anyway', 'cant', 'wait', 'go', 'back']

There are no punctuations or stopwords, and the remaining words are returned to us as a list of tokens.

Example for vectorization:

Example review text:

Rosie, Dakota, and I LOVE Chaparral Dog Park!!! It's very convenient and surrounded by a lot of paths, a desert xeriscape, baseball fields, ballparks, and a lake with ducks.

The Scottsdale Park and Rec Dept. does a wonderful job of keeping the park clean and shaded. You can find trash cans and poopy-pick up mitts located all over the park and paths.

The fenced in area is huge to let the dogs run, play, and sniff!

Example Tokenization vector:

(0, 2038)	1
(0, 2591)	1
(0, 2699)	1
(0, 2787)	1
(0, 4782)	1
(0, 6122)	2
(0, 6683)	1
(0, 6864)	1
(0, 7159)	1
(0, 9564)	1
(0, 9953)	1
(0, 10044)	1
(0, 11055)	1
(0, 11640)	1
(0, 12185)	1
(0, 13038)	1
(0, 13417)	1
(0, 13637)	1
(0, 14536)	1

```
(0, 14565)	1
(0, 14618)	1
(0, 16493)	1
(0, 17201)	1
(0, 17290)	1
(0, 17455)	1
(0, 17660)	1
(0, 17852)	1
(0, 17928)	1
(0, 18637)	1
(0, 19841)	2
(0, 19917)	2
(0, 20354)	1
(0, 20470)	1
(0, 21961)	1
(0, 22494)	1
(0, 23027)	1
(0, 23987)	1
(0, 24909)	1
(0, 26170)	1
(0, 26284)	1
```

Verifying the features from vectors:

```
print(cv.get_feature_names()[2038])
print(cv.get_feature_names()[19917])
```

```
Chaparral
Paths
```

Process of implementation

We use Pandas, Numpy, Seaborn and nltk as our main libraries for implementation:

Seaborn - We use seaborn for all visualizations. It is a powerful Python visualization library and is the defacto choice for Exploratory Analysis. It makes it very easy to "get to know" your data quickly and efficiently

Pandas & Numpy - Pandas provides high level data manipulation tools built on top of NumPy. NumPy by itself is a fairly low-level tool, and will be very much similar to using MATLAB. pandas on the other hand provides rich time series functionality, data alignment, NA-friendly statistics, groupby, merge and join methods, and lots of other conveniences.

NLTK - NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. We use some of these techniques in our project

Following is the high level implementation details

1. We read the dataset file using pandas data frames,
2. analyse the distribution of data using pandas and seaborn plots
3. Use facetgrid to create a grid of histograms side by side to get the relationship within the columns in the dataset
4. Use seaborn box plots to get the outliers
5. Use heat map to understand correlation within the data
6. Clean the data by removing punctuation, removing stop words using nltk library
7. Use sklearn.feature_extraction.text for tokenization
8. Use Sklearn.naive_bayes for algorithm implementation

There were some complications in using stop words library and generating a new function, however this was made easy with the use of NLTK package and I had come up with def text_process(text) function to remove punctuation and stopwords.

Hyper parameters:

Alpha hyper parameter:

I had tried alpha parameter for Multinomial NB and found that alplha=1.0 or default performs well than any other values to alpha

Accuracy for

Alpha Hyper parameter	accuracy	precision	recall
1.0	92%	0.93	0.98
2.0	87%	0.95	0.36

Tf-idf implementation

I had also used tf-idf implementation - It's a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents.

Tf-idf implementation made things worse as

Reviews	precision	recall	fscore
Negative	0.00	0.00	0.00
Posetive	0.81	1.00	0.90

IV. Results

Model Evaluation and Validation

Multinomial Naive Bayes is a specialised version of Naive Bayes designed more for text documents. Here, we build a Multinomial Naive Bayes model and fit it to our training set (X_train and y_train).

[[157 71]
 [24 974]]

	precision	recall	f1-score	support
1	0.87	0.69	0.77	228
5	0.93	0.98	0.95	998
avg / total	0.92	0.92	0.92	1226

The model predicted positive and negative reviews correctly from the dataset
Eg:

Positive review:

'I have no idea why some people give bad reviews about this place. It goes to show you, you can please everyone. They are probably griping about something that their own fault...there are many people like that.\n\nIn any case, my friend and I arrived at about 5:50 PM this past Sunday. It was pretty crowded, more than I thought for a Sunday evening and thought we would have to wait forever to get a seat but they said we\'ll be seated when the girl comes back from seating someone else. We were seated at 5:52 and the waiter came and got our drink orders. Everyone was very pleasant from the host that seated us to the waiter to the server. The prices were very good as well. We placed our orders once we decided what we wanted at 6:02. We shared the baked spaghetti calzone and the small "Here\'s The Beef" pizza so we can both try them. The calzone was huge and we got the smallest one (personal) and got the small 11" pizza. Both were awesome! My friend liked the pizza better and I liked the calzone better. The calzone does have a sweetish sauce but that\'s how I like

my sauce!\n\nWe had to box part of the pizza to take it home and we were out the door by 6:42. So, everything was great and not like these bad reviewers. That goes to show you that you have to try these things yourself because all these bad reviewers have some serious issues.'

Review prediction -5

Negative review:

'Still quite poor both in service and food. maybe I made a mistake and ordered Sichuan Gong Bao ji ding for what seemed like people from canton district. Unfortunately to get the good service U have to speak Mandarin/Cantonese. I do speak a smattering but try not to use it as I never feel confident about the intonation. \n\nThe dish came out with zichini and bell peppers (what!??) Where is the peanuts the dried fried red peppers and the large pieces of scallion. On pointing this out all I got was " Oh you like peanuts.. ok I will put some on" and she then proceeded to get some peanuts and sprinkle it on the chicken.\n\nWell at that point I was happy that atleast the chicken pieces were present else she would probably end up sprinkling raw chicken pieces on it like the raw peanuts she dumped on top of the food. \n\nWell then I spoke a few chinese words and the scowl turned into a smile and she then became a bit more friendlier. \n\nUnfortunately I do not condone this type of behavior. It is all in poor taste...'

Review prediction - 1

Model Robustness:

Changing the Randomstate does affect the accuracy and scores of our algorithm and we pick the one with high accuracy

Negative Reviews	**random state**	**precision**	**recall**	**f1score**	**accuracy**
	101	0.87	0.69	0.77	0.92
	1	0.8	0.63	0.7	0.9
	no random state	0.83	0.64	0.72	0.91
Positive Reviews	**random state**	**precision**	**recall**	**f1score**	**accuracy**
	101	0.93	0.98	0.95	0.92
	1	0.92	0.97	0.94	0.9
	no random state	0.92	0.97	0.95	0.91

Mean and Variance different random states

When Mean and variance are calculated for different random state, the mean is found to be very similar with minor variances

random state	Mean	Variance
101	4.42	2.1
1	4.39	1.9
no random state	4.39	1.8
1000	4.41	1.7
501	4.42	1.8

Comparison with Benchmark Model(Logistic regression)

The logistic regression model provides an accuracy of 91 and our model yields a slightly better accuracy than the logic regression model with 92% accuracy

Find below the precision, recall and f1-scores of both the models, in which our model scores slightly better than the benchmark model

Model	Review	precision	recall	f1-score
Logistic regression	Negative	0.86	0.68	0.76
	Positive	0.92	0.97	0.94
MultinomialNB	Negative	0.87	0.69	0.77
	Positive	0.93	0.98	0.95

Is our model foolproof?
No, take a look at the below

> **Review text:** Other than the really great happy hour prices, its hit or miss with this place. More often a miss. :(\n\nThe food is less than average, the drinks NOT strong (at least they are inexpensive) , but the service is truly hit or miss.\n\nI'll pass."
>
> **Review prediction -** 5

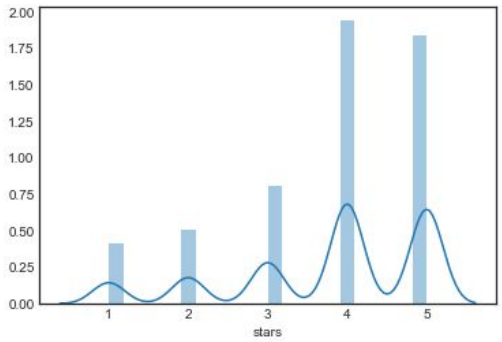
Our model's prediction is incorrect

Overall we have seen the models perform similarly on our yelp data set. Our model though accurate, but is more biased towards positive reviews compared to negative ones.

V. Conclusion

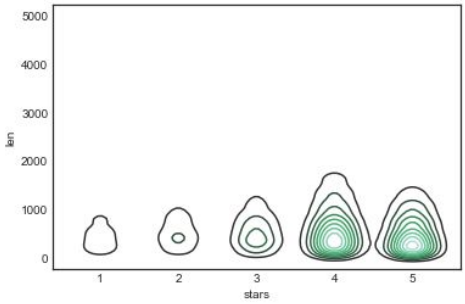
In conclusion, although our model was a little biased towards positive reviews, it was fairly accurate with its predictions, achieving an accuracy of 92% on the test set.

We see variability between text length and rating in spite of the accuracy of our model



We see that there is variability between text length and rating inspite of the accuracy of our results

From density graph, we see that 4 and 5 star ratings are more when compared to other ratings inclining towards bias on the positive side of the rating



Based on the above density of the ratings, we see stars 4 and stars 5 are more than compared to stars 3, stars 2 and stars 1 when compared agaisnt hte length of text

Reflection of the project:

Overall, I found researching about NLP, reading some of the papers about both NPL and sentiment analysis as very interesting . I also had to learn NLP techniques (removing stop words, punctuation) , vectorization and found that as an interesting part of the project

In this project, we proposed a method for predicting sentiment for restaurant reviews from Yelp review dataset. The method was based on a high-accuracy Multinomial Naive Bayes model, deriving vectors and measuring the polarity by removing, punctuation, stop words etc.

On the other hand, similar procedures can be reproduced in other areas like movie reviews and social media posts. Suppose someone would like to find a movie most renowned for its perfect demonstration of ‘love’, by operating sentiment analysis and polarity detection on IMDB or rotten tomato movie reviews, he would definitely get what he desired. That would be our anticipations in the future: gathering opinions from people, extracting information from opinions and generating suggestions from information.

As a next step, we can fine tune this model using Recurrent neural networks as it seems to be a right model for text based predictions .

Also, we can use other available datasets such as IMBD reviews, Amazon reviews and make our program generically predict sentiment irrespective of the datasets.