IE406

MACHINE LEARNING

GROUP 18

“Categorization of hotel reviews into different customer values”

GROUP MEMBERS

NAME	STUDENT ID
Divyesh Rohit	201801086
Ishani Bandyopadhyay	201801102
Janvi Patel	201801178
Tulsi Shah	201801180
Skand Vala	201801210

COURSE INSTRUCTOR: **Prof.** Manjunath Joshi

Overview:

Introduction

Our premise is that a customer, when reserving a hotel room online, should use her time researching different hotels quickly and making the best possible call given her circumstances, and not waste time reading hundreds of different customer reviews. If the system is designed ideally, the task can be accomplished quickly. The system we were trying to make is used by many hotel reservation websites in different ways and comes out to be a very important tool when millions of reviews need to be processed and different sentiments need to be identified. This is a very time consuming manual job and almost no one can afford it. The preliminary work we did will benefit a lot of up and coming websites and people who want to try new things in the *sentiment analysis area*. We will discuss how we implemented an algorithm to rate an individual text review into five different categories based on specific key words which reflect a specific sentiment.

Background

Every member of our team was determined to do some good meaningful work in this project. And we got lucky when everyone agreed readily to work in the sentiment analysis area. We were motivated in this area when we found a research paper based on “Machine Learning-based classification for “Sentimental analysis of IMDb reviews” on the Stanford website. We were planning a small trip and when we ourselves were booking a hotel room, we got this idea. There has been some work done in this area in recent years. People have tried to get accurate results on whether a particular review is positive or negative. The shortcomings we found were as follows: the user is able to get just a general idea, that is the text has some positive or negative sentiments. But if the text had both different perspectives, there is no way the user can get a meaningful output. Our work is different from others in the way that now a user can get accurate positive and

negative sentiment on different areas, and the user can decide which area has the most importance for her. Through this choosing the hotel becomes easier and quicker and smarter.

Problem Statement

Our aim is to classify hotel review *text into* different categories of importance and rate them whether, how positively or negatively has been the user about the hotel for different amenities.

Datasets

As the project we did is unique and nowhere available across the internet, we had to make the dataset on our own, solely based on our conscience, by taking reviews of different types from the internet. After we compiled a big list of reviews consisting of about sixteen hundred reviews, we manually rated each review and classified them into five categories, namely: food, safety and hygiene, location, value for money and hospitality. We decided on these categories after scouring through tens of websites and hundreds of reviews. We felt these are the most important ones, and the categories where we should begin.

We thereafter decided that each category will have either of three values, namely, 0, 1 or 2, which suggests: bad, average and good. So each review will have five values, which we gave input manually to all the sixteen hundred reviews. This was the most tedious task, but the most overwhelmingly important as well because, no matter what code we wrote or what algorithm we used or method we used, the output will be incorrect even if we made a tiny mistake in categorizing the reviews. We checked and double checked and made sure everything is done correctly to the best of our abilities.

Methodology

Before beginning we decided upon that we will begin by using naive bayes as the first method to try and solve our problem. It was done so because naive bayes is effective and simple with reasonable accuracy and it works well with spam filter classification problem, a problem bearing a lot of similarities to our problem. After naive bayes we proceeded with logistic regression, the classic method used for classification problems and then we went ahead with K nearest neighbour algorithm. The reason we used the KNN was because this algorithm assumes similarity between new cases and available cases and classifies them based on similarity. We needed exactly something like this for our problem, as we wanted to classify a new line of text into some categories, based on a training data set, into some values which are decided on the basis of similarity.

So, in the beginning before using any machine learning models or training our model, we proceeded cautiously and observed our training data set. We found that it has a lot of full stops and commas and question marks. This as we proceeded ahead hindered our accuracy. So we removed all the characters which are not in the English alphabet. We even removed numbers from our review texts. The primary reason for doing so is, any sentiment very rarely is affected by numbers and punctuation marks. So, it may confuse our algorithm when it considers the frequency of words(naive bayes) or nearest neighbours in the form of punctuation marks(k nearest neighbours). Thereafter, we converted all the text into a normal standard lowercase, to remove any discrepancies between 'Food' and 'food'. Because machine will understand these two as different words, which we don't want to happen. Then we went ahead with removing all the common words, in machine learning language, *stop words*. The intuition behind doing so was to understand the emotions with more clarity as extra or no stop words in combination with nouns and verbs will give us wrong results. For example: "this food was very good", after removing stop words it becomes "food very good". As we can see, two words were just enough to describe the emotion, and the rest of the words was just extra information, or say extra overhead. The next step in refining our dataset was stemming. This is something we read online, and it's an interesting proposition to use. It suggests that all English words have different forms, such as 'eat' and 'eating'. As both words are same altogether we cannot classify them as different words and use them differently for training the model. For example, 'eating paneer tikka is a must at this hotel' and 'eat paneer tikka must at this hotel' are similar sentences and should be considered the same. Then for reducing the sparsity, we kept a tab on how many words should be considered to predict, and in technical terms, it's called *bag of words*. It chooses a certain number of words, with maximum frequency. The rest is eliminated.

Results and Discussion

Now, let's get to the numbers part. Let's discuss the results for the Naive Bayes model. We got a very good accuracy of more than 70% across all categories. Highest accuracy was found for Hospitality which stood at more than 78% where lowest was found for location which stood at 73%. For food and value for money, it stood around 77%. A lot of predictions were correct and we were very pleased with our results.

Below are some more results we extracted using confusion matrix:

Food

Accuracy : 0.774
Recall : 0.774

Precision : 0.830
F1 Score : 0.774

Safety and Hygiene

Accuracy : 0.7652
Recall : 0.765
Precision : 0.84
F1 Score : 0.765

Location

Accuracy : 0.734
Recall : 0.734
Precision : 0.815
F1 Score : 0.734

Value for money

Accuracy : 0.777
Recall : 0.777
Precision : 0.826
F1 Score : 0.777

Hospitality

Accuracy : 0.783
Recall : 0.78
Precision : 0.85
F1 Score : 0.785

The next model we used was the K nearest neighbour, and results were below average. Below are the numbers:

Food

Accuracy : 0.594
Recall : 0.594
Precision : 0.590

F1 Score : 0.594

Safety and Hygiene

Accuracy : 0.643
Recall : 0.643
Precision : 0.63
F1 Score : 0.643

Location

Accuracy : 0.634
Recall : 0.6341
Precision : 0.616
F1 Score : 0.63

Value for money

Accuracy : 0.548
Recall : 0.5487
Precision : 0.571
F1 Score : 0.548

Hospitality

Accuracy : 0.676
Recall : 0.676
Precision : 0.681
F1 Score : 0.676

KNN worked a little on the worse side because of the following reasons we were able to deduct after some discussions: when our method gives weights to each word from a bag of words depending upon the frequency of words. As the dimensions of the input size increases the similarity feature which KNN uses to classify a new case into one of the older cases degrades. Also, KNN worked particularly slower because of the extra overhead of calculating euclidean distance.

The last model we used was Logistic regression. Below are the results:

Food

Accuracy : 0.8978
Recall : 0.897
Precision : 0.89
F1 Score : 0.897

Safety and Hygiene

Accuracy : 0.899
Recall : 0.89
Precision : 0.89
F1 Score : 0.89

Location

Accuracy : 0.913
Recall : 0.913
Precision : 0.917
F1 Score : 0.91

Value for money

Accuracy : 0.87
Recall : 0.878
Precision : 0.884
F1 Score : 0.878

Hospitality

Accuracy : 0.93
Recall : 0.931
Precision : 0.932
F1 Score : 0.931

This model gave us the best results with a very high accuracy. As logistic regression uses mapping, like $f(x_i) = Y_j$ from input variables x_1, x_2, till x_5 and output will be y_1, y_2, to… y_5. As the model predictions are iteratively evaluated and corrected against the output values, until an acceptable performance is achieved. This gave us better results.

Conclusion

We observed that naive bayes works reasonably well with fever assumptions. For whichever category there is a dip in results, we are assuming that in the testing part of the dataset, the words may not be there in our frequency dictionary, or there won't be any references to neighbours for KNN. As the English alphabet has millions of words, all expressing different emotions with different contexts of the English language and with a reasonably small dataset, the model loses accuracy. The results were very good and if we had gotten our hands on a bigger dataset we would have been able to give better insights and how it can be implemented on a real time system with continuous updates.