Part1:

Firstly we pre processed the data – removed punctuations, stop-words, performed pos tagging and lemmatization.

Then we did basic sentiment analysis using Vader by assigning positive and negative scores for all the words and also displayed few of the highest positive and negative reviews.

Sentiment analysis on positive and negative reviews is done to find out which words have largest effect on predicting the review outcome.

To implement a classifier, we cannot work on raw text. Therefore we perform feature extraction using various models and proceed with text classification.

For every model and classifier, we have displayed confusion matrices and values of performance metrics, log losses with RoC/AOC curves.

1) Bag or words model: number of occurrences of each word in order to learn what words are associated with positive or negative reviews.

For a Unigram model-

a) Multinomial Naïve bayes classifier - uses multinomial distribution on words to predict the tag of a text i.e probability of each tag for a given text.

Time: 950 seconds approx.

Precision: 91.32138

Recall: 93.87768

Accuracy: 92.48%

Log loss: 0.26604

b) Logistic Regression – using SGD classifier

Time: 500.084 seconds

Precision: 87.4157

Recall: 86.73422

Accuracy: 87.13%

Log loss: 0.55187

c) Support vector machine- non-probabilistic binary linear classifier, i.e. it predicts, for each given input, which of two possible classes the input is a member of.

Time: 439.275 seconds

Precision: 89.84497

Recall: 84.8097

Accuracy: 87.62%

For a Bigram model- we went ahead with naïve bayes as it had the highest accuracy among the rest.

Time: 1500 seconds

Precision: 92.6374

Recall: 93.28584

Accuracy: 92.94%

Log loss: 0.327104

2) TF-IDF model: in order to eliminate the highly frequent unnecessary words that start dominating; when IDF is high, the words occurring will be rare in the document.

Time: 950 seconds approx.

Precision: 91.297250

Recall:93.803218

Accuracy:92.44%

Log loss: 0.205144

3) Word2vec model: Uses genism module and trains to find word vectors and run similarity queries. (text is converted into vectors and similarity is found by its belongingness to the same vector space) We display similarities between few words and a list of words that are similar to a given word. Classified using regression,

Precision: 94.564

Recall: 90.778

Accuracy:92.79%

Log loss:0.22367

4) Doc2vec model: almost similar to word2vec approach and also uses the same genism module, but involves tagging text and vectors. Random forest classifier (uses aggregated votes from decision trees to decide the final class)

Accuracy: 90.05%

We can say that word2vec is a better fit for our dataset.

--For all the above classifiers, we have implemented hyper parameter tuning in order to train the model better and obtain better accuracies.

We also display the most informative features i.e words that had a greater impact on accuracy during classification. We notice that words like ‘beautifully’, ’brilliant’, ’outstanding’ are mainly associated with positive emotion and words such as ‘noisy’, ‘smelly’, ‘inconsistent’ are associated with negative emotion.

We observe that the positive reviews mainly refer to the location and the staff while the negative ones refer to the facilities provided. This insight would help the management to look into the areas and tweak their performance to cater to the needs of the customers.

We also note that ‘comfy’ is the most frequently used word to convey a positive emotion and ‘unstable’ is the most frequently used word to convey a negative emotion.

5) We use Universal state encoder module from tensorflow hub to construct a model to predict if a randomly picked review is ‘Good’ or ‘Bad’. This model has the ease of not requiring any pre processing and the sentences are converted into embedded vectors using One hot encoder.

The model has 2 hidden layers with dropout regularization to prevent overfitting and we run it for 10 epochs with 10% used as validation data.

Possible improvements that can be made:

This doesn’t include spell checks for the reviews provided and also it’s not applicable for multilingual datasets.(only English)

Does not account for misleading reviews – can be integrated with sarcasm detection