CS410 Progress Report

Group Name: Team Experts

Group members:

Zhiyan Jiang(zjiang2), Xin Peng(xinp2), Shan Shan(ss163).

Captain: Zhiyan Jiang(zjiang2)

1. The Progress Made

1) Learning and Establish Environment

To begin with, we first learned and briefly analyzed the dataset we picked which is located at https://www.kaggle.com/datasets/andrewmvd/trip-advisor-hotel-reviews. The dataset has over 20000 rows of reviews and ratings with some comments needing to be cleaned and processed. We also searched online and learned about how to perform sentiment analysis. The typical steps are analyzing the data; cleaning and processing the data; building the model with both training and testing.

Then we worked on setting up the environment for coding. To simplify the process of importing libraries and enable code sharing within the group, we decided to use google colab to develop the source code. We set up the account and environment so that each of the team members can all work on it.

2) Data Analysis

First we performed analysis to better understand the data and distribution of different ratings, we drew a histogram to see how many reviews are for each rating category. There are about 1500 rating 1, 1900 rating 2, 2200 rating 3, 6000 rating 4 and 9000 rating 5. Which indicates most people are giving positive reviews.

3) Data Cleaning and Processing

We performed several steps for data cleaning and processing. First we changed all the words to lowercase, this is to ensure the model treats the words equivalently. Then we removed the unnecessary special characters to only keep the numbers and letters. After that we removed stop words which include "the", "a", "an", etc, this is to ensure we leave the most important and meaningful information. We also performed stemming and lemmatization of the words, this is to commonize words into the same format. Based on research online, stemming is to remove the last few characters of the words while lemmatization considers context and changes the words to their base form. We are not certain which function provides better results so for now we decided to keep both. After all the basic cleaning steps, we wrote functions to find the 20 most frequent words in each rating category and then find the common words among them, then we decided to remove these common words from the data for better model training. These words include "hotel", "day", "room", etc. finally, we plot the word cloud for visualization to see what words appear the most for each rating. As it shows, for higher ratings, more frequent words include "great", "nice", "good", while lower rating reviews contain more words such as "bad", "dirty", "old", etc.

4) Logistic regression

As an initial attempt, logistic regression was used to classify the reviewers' ratings. In the preprocessing phase, a subset of 5,000 records from the entire dataset was selected due to the memory limitation of Google Colaboratory. Then a document-term matrix was established using the module *sklearn.feature_extraction.text*. In terms of tokenization, ngram ranged from 1 to 4, max_features was limited to 30,000, and the minimum word frequency was set as 0.5%. The resulting document-term matrix has dimensions of 5,000 x 30,000. All the 5,000 data records were splitted into 90% training set and 10% test set. Eventually a logistic regression was performed using the *LogisticRegression* function from the *sklearn.linear_model*. The parameter "*multi_class*" was left as default ("auto") to automatically handle multiclass classification. The prediction results were presented in the form of the confusion matrix. Within the matrix, entry at index [i,j] denotes the number of reviews with a true rating i classified as rating j. For example, entry at [1,3] refers to the number of reviews with the true rating of 1 but (falsely) classified as rating 3. Thus, only the numbers on the diagonal denotes the corrected classified reviews. The confusion matrix is presented in the table below:

Table 1. Confusion matrix

		Predicted rating					**Accuracy (%)**
		1	***2***	***3***	***4***	***5***	
True rating	***1***	21	8	2	2	4	57
	2	13	14	11	5	4	30
	3	4	7	13	27	11	21
	4	1	4	14	72	61	47
	5	1	2	4	46	149	74

The preliminary results suggest that the trained logistic regression model has the best accuracy for 5-star reviews. Its performance for 3-star reviews is the worst. One possible reason is that the training set for 5-star reviews has the largest size, which helped better train the model. Another reason is that the 3-star reviews are essentially neutral and more susceptible to the errors in the training model. As a summary, the established logistic regression model still has room for improvement.

2. Remaining Tasks

1) Research on performance

After obtaining the preliminary result, it is necessary to investigate the best potential performance. This requires research on available resources published on the Internet.

2) Test more models

A few other potential models are also worth investigating. These models include Naive Bayes, Support Vector Machine (SVM), and Long Short Term Memory (LSTM). The performance of these models are to be investigated.

3. Challenges/issues being faced

Issue 1: We implemented a function to auto correct the spelling of the words, however the function TextBob that we used can only correct simple spelling errors and may also change words incorrectly. In addition, the processing time is pretty long.
Solution: We omitted this step for now due to incorrect correction and long processing time. Also the original data reviews have minimal spelling errors.

Issue 2: We saw initially: wordcloud shows some common words that appear for all different ratings.
Solution: We removed the most common words from the wordcloud.

Issue 3: Stemming word would cut the word into partial form.
Solution: We decided to keep the stemming step for now since it does commonize the words in different formats.

Issue 4: Some spaces observed in data that split one word into two, e.g. "Did n't".
Solution: We decided to remove these words such as "n't" from the comments.

Issue 5: Memory issue due to the size of the model to be too large when max_features=30000.
Solution: We reduced the max_features to be 3000 because we believe that would be a reasonable guess for words vocabulary.