CS410 course project documentation

1) An overview of the function of the code (i.e., what it does and what it can be used for).

The code is a tool of sentiment analysis and can classify emotion on the topic of movie reviews. There are two usages, one is that the user can customize the input and the tool will classify the input into positive and negative emotions. The other usage is to run the classifier model on the test data provided in the GitHub and get the performance of the model.

2) Documentation of how the software is implemented with sufficient detail so that others can have a basic understanding of your code for future extension or any further improvement.

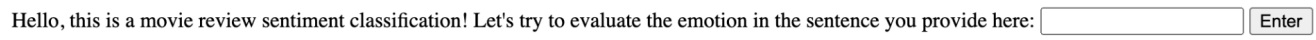
The software is deployed on web by Flask and using the form component provided by wtforms. I have divided the app into MVC model. The input box can be modified in model.py. In controller.py I created function for posting and getting API. The function will deal with processing the input using the model.
The training data is included in the github and I have written the data_loader function in load_data.py. This function will help you to transfer folders of reviews into a list of words list like this:
[['Well', 'this', 'movie', 'actually', 'did'],['have', 'one', 'redeeming', 'quality', 'It', 'made', 'up', 'the', 'funniest', 'season', 'one', 'episode', 'of',]['MST3K', 'I', 'wish', 'Rhino', 'had', 'released', 'this', 'one', 'instead', 'of', 'The', 'Crawling', 'Hand']]
And it will create a numpy array for pos and neg labels like:
The classifier model is built in classifier.py. Following the course week11's contents, I chose a generative model to be the classifier which is the Naive Bayes model. I built unigram and bigram model for classification.
[0,0,0,0,1,1,1]

The unigram model followed the formula in class. The basic idea is to use a dictionary to count the number of pos and neg for a word and

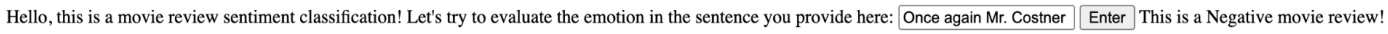
record it. Then in the test data, we just use the dictionary to judge if the word and sentence are positive or negative. The bigram model used the same idea but combine two words together in the dictionary. I also use laplace smoothing method and take log in the model to increase accuracy. Now the unigram model can reach 86% accuracy. And bigram model can beat over 89%.

3) Documentation of the usage of the software including either documentation of usages of APIs or detailed instructions on how to install and run a software, whichever is applicable.

In order to use this web application, you will need to download the files on Github. I will recommend using the anaconda virtual environment to run the app for configuration purposes. Your python version should be 3.9. And for the library used in the app, you can use ‘conda install …’ to install the library. These are the libraries needed: numpy, flask, nltk, wtforms.
There are two modes for the software. One is to run the model on the test data. In this version, simply run ‘python main.py’ in your terminal and the the model will generate the predictions for test data and return the accuracy. This is enabled in main.py.
The second mode is for the web application. In order to open the server of the app you will need to run python controller.py. Then the server is started and you need to open a web browser and type in the website: “http://127.0.0.1:5000/review” and the webpage will look like this:



And then you can type in the review you want to classify in the input box. After clicking Enter the webpage will show you the result as below:



The presentation video is here:https://drive.google.com/file/d/1UNzwjrpwGr11LfZUvkTtpJX0y6omtWtf/view?usp=sharing