Sarkis Chichkoyan

Queen Mary University of London

Student ID: 190425490

ECS763U/ECS763P - Natural Language Processing Coursework 1

Question 1

For this first task, I used a basic parse function that attributes label to the data line 1 and text for the data line 2 and a simple return. Then to implement the pre_process function using nltk libraries, I did tokenisation returning a list of tokens.

Question 2

In this task I first created a global dictionary of features, then I implemented the to feature vector function by creating another feature dictionary to store the features. Then our goal is to iterate through each token in the list of tokens and then printing the dictionary that we locally created. In the the global dictionary of features we store all the features that we will use in our classifier.

Question 3

This section of the code is implementing the cross validation function. The first step for this is to implement, inside the for loop that we are given, training classifier on the fold and evaluating the classifier on the remaining folds which is the testing data.

The next bit is iteration through the testing data to convert the feature vector into a dictionary. Then we use the predict labels function to predict the labels of the testing data and we extract the true labels from the original data.

Now we need different results for our cross validation, in this case I coded for precision, recall, f1-score and accuracy to calculate the performance metrics. Then I also included the average performance of each metrics for each folds and inserted all of the cv results inside a data frame using pandas.

Question 4

In this function, we are going to analyse error on the train-test split. We start by isolating the test samples from the true labels. Then the function predicts labels for the test samples using the classifier and generates a list of tuples showing misclassified samples. The instances of false positives and false negatives are then written to a text file called 'error_analysis.txt'.

Watching at the false positives I noticed examples where observations can be made. In fact we have a concrete example of a false positive here which is very disturbing for the classifier : "Really hope Chelsea starts Baba instead of Ivanovic on Saturday, he's literally aged about 5 years over the summer. » Here the tweet combines positive and negative sentiments. Then in the false negative area I have seen one other exemple that is interesting to understand the problem, "I'll be at the Donald Trump rally next Monday AAC. » Here the presence of the word rally and Donald Trump mislead the classifier to think the sentiment is negative as it is positive.

Question 5

After changing the pre_process code and adding a lot of preprocessing, we see that it is not a very good method to achieve better results. In fact as we can see in the notebook with « over pre processing » I used a lot of different methods and the best accuracy that we get (0.79) is lower than our simple previous code (0.81).

The idea now is to find the best pre-processing by choosing what could be important and what could be removed. I decided to keep with lowercasing, removing the special characters, tokenisation, lemmatisation, stemming, handling the negation words, stop words, expanding contractions, replacing number with placeholders and finally I added an external library called text blob which is helping a lot with sentiment analysis. After doing that we get our best accuracy around 0.83.

	Simple model	**Over Pre-processed**	**Better Pre-processed**
Accuracy	0.81	0.79	0.83

Now the goal is to improve the feature extraction of our better pre-processed data to get a more precise accuracy and classifier. I first tried methods on there own but then tried to combine them to see if we get better results. We observe that N-grams using bigrams and trigrams associated to document length which will include the length of the document as a feature is giving us the best accuracy out of the three methods of feature extraction we tried.

Better Pre-Processed	N grams	TF-IDF	N-grams +document length
Accuracy	0.83	0.83	0.84

Trying to change the cost or the class weight of the SVM didn't change anything for my results.

Finally, we have quite an improvement of our best accuracy compared to the simple model we achieved going from 0.81 to 0.84.