CS671 : Assignment-1

1a) I have used 5 regex in total for replacing all the single quote conversational texts to double quotes(Three regexes for covering all the conversational texts and then two more for removing all the false positives). This series of regex modifies nested single quotes to double quotes too. I replace the single quotes by double quotes using re.sub() function.

First regex - It covers all the conversational texts starting with a lower case alphabet. Eg. - 'to revolutionise society on this
lawn ?'.

After this, I have handled the opening and closing single quotes independent of each other.

Second regex - It covers all the closing quotes and also some of the opening quotes(but that is not a problem in our case as we also need to modify opening quotes).

Third regex - It covers all the remaining opening quotes.

Fourth regex - In the second regex, some of the false positives were also included. E.g. - 'Victoria,' became 'Victoria," and 'artists,' became 'artists,". Using fourth regex, I replace the <">(closing double quote) of such words with single quotes.

Fifth regex - This regex handles cases like "Victoria'; which was converted to this form in the third regex. It converts opening double quote to single quote. E.g.- "Victoria'; was converted back to 'Victoria';

Files are present in the <1a> directory. <1a.py> is the code file and the resultant file is <new_test.txt>. The file <1a.py> reads the file <test.txt> from the root folder and converts it to <new_test.txt> in the <1a> directory. To run the code, use the folllowing command:

< python2 1a.py >

1b) I have used four regexes in total to achieve this task. Iused positive and negative lookback for this and replacec single quotes by double quotes using re.sub() function.

First regex - It marks all the sentences that end with <.> or <?> or <!> ignoring the words like "don't", "Mr." and abbreviations like "G. K. Chesterton".

Second regex - It marks all the sentences that end with <.'> or <?'> or <!'> ignoring the words like "don't", "Mr." and abbreviations like "G. K. Chesterton".

Third regex - Using this regex, I replaced "\n\n" with "</s><s>" except when "\n\n" was preceeded by "--".

Fourth regex - Using this regex, I deleted all "\n\n" which were just after by "--". this was done just to prepare dataset for question 2 nd has no net effect on the original text.

Files are present in the <1b> directory. <1b.py> is the code file and the resultant file is <new_test.txt>. The file <1a.py> reads the file <test.txt> from the root folder and converts it to <new_test.txt> in the <1b> directory. To run the code, use the folllowing command:

< python2 1b.py >

---

2) The code is inside the directory <2>. Initially, I used the sentence terminator from 1b to label the sentences of the text file <fullTest.text>. Labelled sentences were stored in the file <training_data.txt>. I use this as my training data set.

Steps followed:

i. I made a vocablary of all the characters present in the text <fullTest.txt>, i.e., read the file <fullTest.txt> character by character and included all the characters in the vocablary.

ii. Positive data set(all the tokens which are sentence terminators): I used <positive.py> to convert all the tokens(and their context) which are teminator to vector form. For every token that is a terminator too, I take 4 characters to its left and 3 characters to its right(this is my context window). The intuition behind this was that in <1b>, I used rules that decided whether a token is a sentence terminator or not by looking at most 4 characters to the left of token and 3 characters to the right of token. Then every character was converted to a one-hot vector of length equal to the vocab size. In this case, the vocab size was 80. Every vector is of size 80 and context around every token is of the form of a 2D matrix of dimension 7x80 (4+3=7). The positive training data was dumped in the file <positive_data.json>.

iii. Negative data set(all the tokens[.?!] which are not sentence terminators): Followed the same approach as in case of positive data set with just a single difference that here those tokens were considered which were not sentence terminators. The negative training data was dumped in the file <negative_data.json>.

iv. Since the positive data points were much greater in number than the negative ones(data was skewed). So, I took the positive data points equal in number to the negative data points. This removed the skewness. Then I mixed positive and negtive data points and shuffeled them and then trained a **SVM(sklearn.svm.SVC())** on these points. All this was done in the file <train.py>. Trained model is also saved as <model.pkl> after every run.

v. For testing the data, I partitioned the original data into two parts(training data(80%) and test data(20%)).

Accuracy on train data: 98.50%
Accuracy on test data: 98.00%

There is also a file named <transform.py> which takes as input an unlabelled file and outputs a labelled file.

To train the model, run the following commands
:
cd 2/

python2 positive.py

python2 negative.py

python2 train.py

To convert an unlabelled file to a labelled file, run the following command:

python2 transform.py <unlabelled file> <name of the desired labelled file>