**Assignment 4 (Natural Language Processing)**

**Task 1**

**(a)**

**(I)** Vocab size post tokenization: **38911**

**(ii)** Vocab size post tokenization, lowercasing, punctuation and stopword removal: **38557**

**(iii)** Vocab size post all above and stemming: **24696**

**(iv)** Vocab size post all above and lemmatization WITHOUT stemming: **33191 –** rapidminers stemming is much more rigorous

**(b)** TF and TF-IDF output from rapidminer into CSV.

**-** Tried using Python as well to lemmatize since Rapidminer’s lemmatizer was not working correctly. Python script is provided with rapidminer exported list of pre-processed words from all text files.

**(c)** outputted CSV opened in excel then sorted by TF or TF-IDS to get top 10 words

**(d)** TF and TF-IDF generated in rapidminer giving discriminative weight to neg and positive documents. Then again exported to CSV and sorted to get top 10 words.

**Task 2**

Run the file t2.m to run the linear regression model.

output\_lemmatized\_TF-IDF.csv is the export from part1 Task (b) python script.

- Our array size exceeds 8.7 GB. My laptop has 8 GB. However, it worked fine when array size was reduced from around 36000 to 5000.

**Task 3**

**You need to import the LDA toolbox in MATLAB first.**

Run file t3.py in python to lemmatize the .mat file.   
You need to install sciopy.io and nltk libraries for it to work.  
This python script would generate WSDS.csv a list of words against documents and a wordList.csv with list of all words.

To run GibbsSamplerLDA on WSDS.csv and wordList.csv, run the file t3.m in MATLAB.

**(a)** Below is an excerpt from MATLAB’s final output for part A

ans =

'contractannouncement company million brownshirt semisensical'

'program tylermade floodinsurance ralliers sacre'

'developmentally burntcar n't procedurally like'

'blackedout belmarsh footner launchpad luongo'

'thaisgwa intermingled milder youyour yew'

**(b)** Here we simply take a sample of random articles from TIME-Articles.mat by reducing the articles dataset in the python script t3\_b.py. Then we run the same MATLAB script t3.m from part A to generate its topic distribution.

**Task 4**

(a) Using SentiStrength both the negative and positive folders were reviewed and their results generated.

(b) Strength of each sentence in every file was taken. The files were renamed for easy iteration in a MATLAB for loop.

(c) The positive and negative coloumns of all the resultant files were summed. Their difference taken. If difference was negative then it was classified as negative and vice versa. This was compared with the category of file itself (whether it was in the negative folder or the positive folder). A box plot was generated of all our averages for each file.

**Positive Files**

Error reading files ::28

Accuracy ::97.2

False Positives ::0 % approximately

**Negative Files**

Error reading files ::29

Accuracy ::97.1

False Negatives ::0.1 % approximately