CSE556 NLP
Assignment 01

Yash Bhargava 2019289
Harman Singh 2019042

Disclaimer: This assignment has been jointly done by both members with more or less equal contribution. The demo is to be conducted using Google Colab since the dataset is uploaded on Google drive. For conducting the demo on a local machine, relevant parts of the code might need to be changed accordingly.

Q1
A.

a) Following is done for each tweet:

tweet_sans_extra_whitespaces = re.sub(r'^\s*|\s*$', '', tweet)
Extra whitespaces at the start and end of the tweet are stripped.

re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?|\!)\s', tweet_sans_extra_whitespaces)
The tweet is split into sentences based on the logic:
(?<!\w\.\w.) - Negative lookbehind, the sentence should not be split by word_character-dot-word_character-any_character since to take into account URLs and big abbreviations
(?<![A-Z][a-z]\.) - Negative lookbehind, to ensure sentence is not split by titles like Mr., Ms., Dr., etc.
(?<=\.|\?|\!) - Positive lookbehind, to check for period, question mark, exclamation mark
\s - for whitespace

re.findall(r'\w+|[^\w\s]+|[\'\w\-]+', sentence)
'\w+ - checks for one or more word_character
[^\w\s] - not word_character or whitespace to check for special words like user handles and hashtags
[\'\w\-]+ - checks for apostrophe, word_character, hyphen

b) Vowels:
Regex used: /\b[aeiouAEIOU].*?\b/
It matches all the words starting with a vowel. \b refers to the word boundary.
Consonants:
Regex used: /\b[b-df-hj-np-tv-zB-DF-HJ-NP-TV-Z].*?\b/
It matches all the words starting with a '\B@\w+'. \b refers to the word boundary.

c) re.sub(r'[A-Z]', myLowercaseFunction, data)
Regex to convert to lowercase, myLowercaseFuntion is a method which returns the corresponding lowercase character for an uppercase character

Logic for token finding is same from part a

d) Regex used: /\B@\w+/
It matches all the words that start with @ (non-word \B character) followed by a word character of 1 or more length.

e) re.findall(r'(https?|www\.|tinyurl|t\.co).*|.*\.(com|org|edu).*', tweet)
Regex to check for URLs. Finds http, https, www., tinyurl, t.co and domains .com, .org, .edu. Assumption: Since links are visible and clickable in the tweet, we are directly printing the tweets.

f) Regex used: Day
Since each row represents 1 tweet, we are iterating over all the dates and doing an exact match for the days in a week.

B.

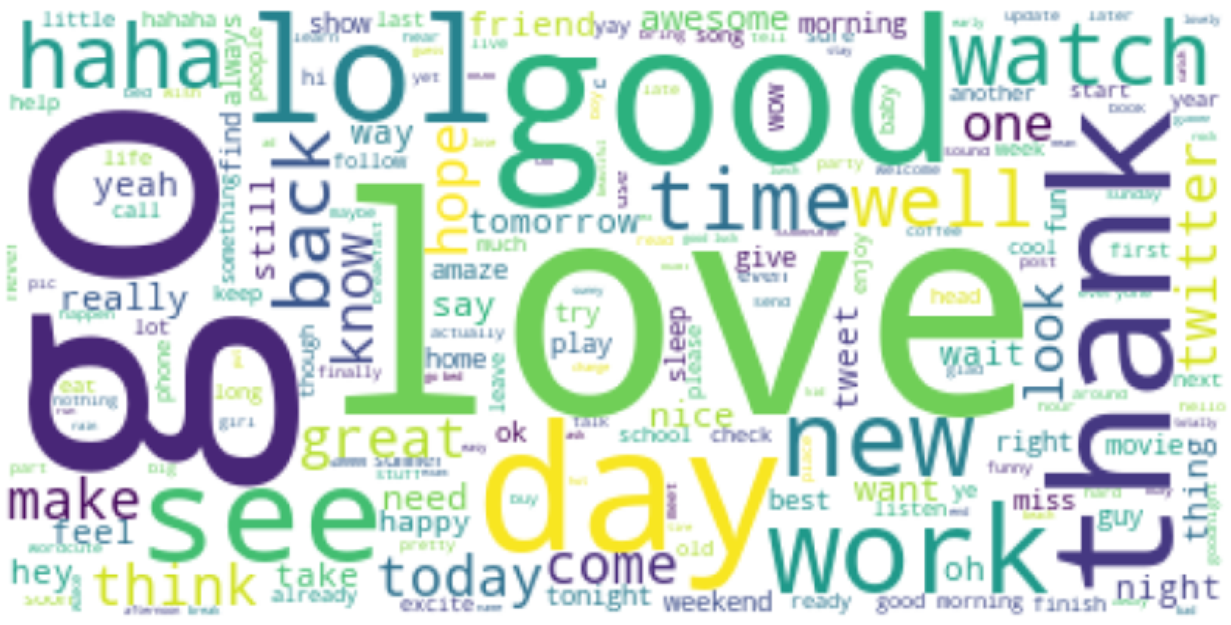
a) Regex used :
For finding total occurrences of the word across all tweets of the class and for finding the total number of sentences in a class containing the word: '\b'+ word + '\b'
Given word: word, and a label for the class
Here \b represents the word boundary. This regex finds all the occurrences of the word in all the tweets for a given class.
Assumption: Finding all the occurrences of the word across all the tweets of a class. Then finding the number of sentences containing the word (sentences in consideration are obtained from 1Aa. Also, reporting the sentences containing the word.
It is assumed that valid dictionary words (definition given by sir in the class and written in the slides) are given as input. No metacharacters should be given. For eg., 'here.' in regex would transform to 'here' followed by any other character.

b) Regex used: /^+ word + \W+?/
The sentence should start with the given word followed by 1 or more occurrences of non-word characters.
Assumption: It is assumed that valid dictionary words (definition given by sir in the class and written in the slides) are given as input. No punctuation marks should be given. For e.g., 'here.' in regex would transform to 'here' followed by any other character.

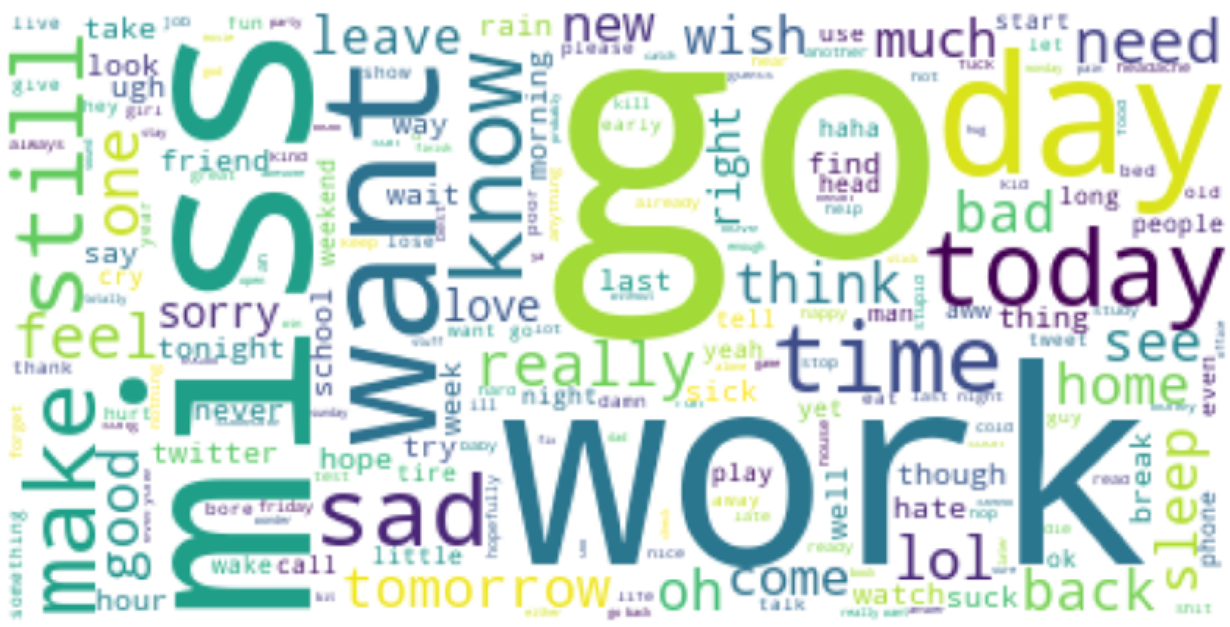
c) Regex used: /\W+ word + [\.\?\!]*$/
The sentence should end with the given word preceded by a non-word character and succeeded by ., ? or ! (optional)
Assumption: It is assumed that valid dictionary words (definition given by sir in the class and written in the slides) are given as input. No punctuation marks should be given. For e.g., 'here.' in regex would transform to 'here' followed by any other character.

Q2
The following preprocessing steps are applied in sequential order on raw data:
1. Expanding contractions and lowercasing - contractions.fix() to expand words like don’t into do not, lowercasing is done using lower()
2. Tokenization - tweet_tokenizer.tokenize() is used to tokenize the corpus
3. Removing extra whitespaces - re.sub(r'^\s*|\s*$', '', x) - extra whitespaces at start and end are stripped
4. Removing user handles and hashtags - re.sub(r'^(@|#).*', '', x) - tokens starting with @ or # are replaced with ‘’
5. Removing URLs and HTML tags -
re.sub(r'^(https?|www\.|tinyurl|t\.co).*|.*\.(com|org|edu).*|^<.*>$', '', x)
6. Removing punctuation - x.translate(str.maketrans('', '', string.punctuation)) - Punctuation is removed using string library
7. Spelling Correction - spell_checker.correction(word) is used to correct spelling using SpellChecker library
8. Removing stopwords - pattern_for_sub = re.compile(r'\b(' + r'|'.join(stopwords_english) + r')\b\s*') - creates a regex expression object using the list of stopwords from nltk library. pattern_for_sub.sub('', x) - replaces stopword with ‘’
9. Lemmatization - lemmatization is done using nltk library based on the PoS tag for the token
10. Removing blank tokens - All the blank tokens from previous steps are removed

Q3
a) wordcloud library is used to generate wordclouds
Wordcloud for positive class



WordCloud for the positive class.



WordCloud for the negative class.

b) Observations -
For negative class - the biggest words in the wordcloud are work, go and miss. The probable reason for their prominence is the stress related to work, the tension and urgency of going somewhere and missing a deadline or milestone - all negative emotions related to the human psyche.
For positive class - the biggest words in the wordcloud are love, go, good, thank. The probable reason for their prominence is the human emotion of love, excitement of meeting someone, doing a good deed or helping someone and thanking people to show our gratitude.

Q4

- Used SentimentIntensityAnalyzer class from nltk.sentiment.vader
- Case 1: Considering the compound score
    - If the compound score generated via VADER sentiment analysis is >=0.05, we consider it to be positive (1)
    - If the compound score generated via VADER sentiment analysis is <=-0.05, we consider it to be negative (0)
    - Else the label returned is (-1)
    - Case a): Considering neutral labels as positive
    - Case b) Considering neutral labels as negative
- Case 2: Not considering compound score: If the positive score is greater than the negative score, we are considering it to belong to the positive class. Else negative.
- Accuracy computation: Computed predictions using above and compared them with the true labels. If the true label for a sample matches with the predicted label, we increase

the count of correctly classified tweets. We divide this number by the total number of samples.