ASSIGNMENT 1

I Regular Expression

- **Methodology**
  - Relevant libraries are imported
  - Path of dataset is imported using pandas
  - Relevant pre-processing is performed using regex on each sub question
  - List, set, and variable declarations is constructed
- **Pre-processing**
  - Split the text into tokens using re.split(r'\s+')
  - Append to list or set
  - Count the number of occurrences
- **Assumptions**
  - For average number of sentences and tokens

    (r'\s+(?=[.!?])|(?<=[.!?])\s+') :: sentence end either with ".!?" . In my regex it will either look head of ".!?" Which is a space so it tokenizes at that point. For the ending delimiters I have considered ".!?".

    ```
    Total number of sentences: 8831
    Total number of sentences in LABEL 1: 4862
    Total number of sentences in LABEL 0: 3969
    Avg number of tokens for Label 1: 0.55
    Average number of tokens for Label 0: 0.44
    ```

    (r"\w+") :: for finding tokens I have considered alphanumeric part (a-z, A-Z, 0-9). Here, it will match one or more-word characters.

    ```
    Total number of tokens: 58915
    Total number of tokens in LABEL 1: 30621
    Total number of tokens in LABEL 0: 28294
    Avg number of tokens for Label 1: 0.52
    Average number of tokens for Label 0: 0.48
    ```

  - Total number of words starting with consonants and vowels

    (r'^[aeiouAEIOU]\w+') :: for vowels it will start with either of "a,e,i,o,u"(and also its capitals) followed by any word character.

    ```
    Words starting with vowels are in LABEL 1: 5047
    ```

Words starting with vowels are in LABEL 0: 4772

(r'[bcdfgjklmnpqstvxzhrwyBCDFGHJKLMNPQSTVXZHRWY]\w+') for consonants starting with the character words exclusing those are vowels and followed by any word character.

Words starting with const are in LABEL 1: 25166

Words starting with const are in LABEL 0: 22810

- Lowercase the text and report the number of unique tokens present before and after lowercase
    1. (r"\w+") found the total number of word characters
    2. (r"[A-Z]") wherever the starting character is capital I have used re.sub method to lowercase it.
    3. set(l2) and then using set to get the unique counts.
    4. For before the lowercasing same proess is followed excluding that lowercase is been performed

Total number of unique tokens after lowercase for label 0: 5620

Total number of unique tokens after lowercase for label 1: 6819

Total number of unique tokens before lowercase for label 0: 6417

Total number of unique tokens before lowercase for label 1: 7852

- Count and list all the usernames

  (r"^@\w+$") :: here the usernames will start from “@” followed by any word character.

UserName count for LABEL 1:1270

UserName count for LABEL 0:786

List of UserName of LABEL 1

['@awaisnaseer', '@Marama', '@gfalcone601',….

List of UserName of LABEL 0

['@sokendrakouture', '@flyingbolt',…

- Count and list all the URLs

  (r"http[s]?://\S+") :: here the URLs will start will http or https followed by a compulsory string and then whatever is there after that till whitespace is found.

  URLS count for LABEL 1: 124

  URLS count for LABEL 0: 58

  List of URLS of LABEL 1

  ['http://blip.fm/~4lfcc', 'http://bit.ly/rwoHR',
  …

  List of URLS of LABEL 0

  ['http://bit.ly/AEbs3', 'http://twitpic.com/3158
  9',…

- Count the number of tweets for each day of week

  In each row, we count the number of tweets and store in x. then update the count for the label, and particular day. @ is used to search number of tweets.regex can be improved, but I think it's satisfactory. In the 2d matrix, row no. represent the label(0 or 1), col no. represent the day(mon to Sunday)

  week=['Mo','Tu','We','Th','Fr','Sa','Su']
  count=[[0,0,0,0,0,0,0],[0,0,0,0,0,0,0]]
  re.findall(fr"@",data['TEXT'][i])

  Tweets for Label 0:[146, 55, 52, 70, 228, 46, 228]

  Tweets for Label 1:[292, 81, 102, 32, 227, 164, 426]

- Total number of occurrences of the given word and sentences containing that word.

  1. For occurrences of sentences for that word using this (r'\s+(?=[.!?])|(?<=[.!?])\s+') split it.
  2. I have used this (fr"\b{word.lower()}\b") to find all that words that contains that word in that sentence.
  3. For occurrences of words for that word using this (r'\W+') split into tokens.

4. I have used this (fr"\b{word.lower()}\b") to find all occurren ces of that word.

```
Total number of occurrences of a word
containing that word: 21
Enter the word: hello
hello
hello…

Total number of occurrences of
sentences given that word: 260
Enter the word: good
@mrstessyman what ever you do have a good day.
good morning!.....
```

- Number of sentences starting with the given word.
    1. Split the text into sentences using (r'\s+(?=[.!?])|(?<=[.!?])\s +')
    2. (fr"^{word.lower()}.*\b") it will match the starting word in t he particular sentence and everything after that till end of a sentence. Here I haven't included ending delimiter like ".!?" so it will also consider those sentences who don't have endi ng delimiter but only a single sentence is there in a row and its ending.
    3. I have lowercased the word so that all the words of different cases becomes lowered.

```
Enter the word: hello
hello twitter
hello everybody.
hello
Number of sentences starting with the custom word
are for LABEL 1: 3
```

- Number of sentences ending with the given word.
    1. Split into sentences
    2. (fr"\b {word.lower()}[.?,!" "' ']*$") this will match the word at the end of sentence when either of the ending delimeter s are found.

```
Enter the word: hello
@corie_michele tell doodie fat guy i said hell
o
@troystith hello!
@sylviegreen69 thanks for saying hello.
@miriamcheah hello
@sweetmexicangal hello!!!!
Number of sentences ending with the custom wor
d are for LABEL 1: 5
```

II Text Pre-processing

- **Methodology**
  - Same as Q1

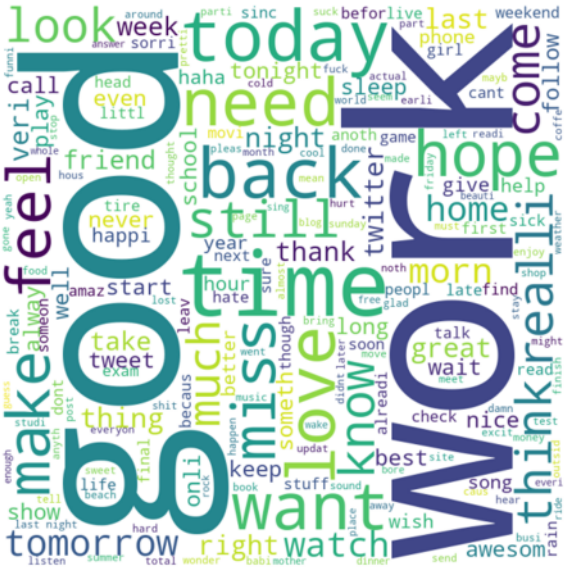
- **Pre-processing (in order of execution)**
  - URL HTML Tags Removal using Regex
  - HTML Tags Removal using Regex
  - Spelling correction
  - Tokenization using NLTK
  - Punctuation Removal using NLTK
  - Stop words Removal using Regex
  - White Spaces using Regex
  - Stemming using NLTK
  - Lemmatization using NLTK

- **Assumptions**
  - (r'http\S+') for URL removal I have considered starting with http only
  - (r'&\w+;') for HTML tag removal word character between "&" and ";" will be removed
  - For stop words I have made my own list of words stop words = {'is','s',' am','or',….} which I am using it.
  - For whitespaces I have used (r'(\w+)') logic means that till this process starts already removal of everything required is done so for white spa ces removal I am getting all the words character.
  - For spelling correction I have used TextBlob library because using NLT K it was not giving correct results to.

III. Visualization

- **Methodology**
  - Same as above
- **Pre-processing(in the order of execution)**
  - Removal of URLs using Regex
  - Removal of HTML Tag using Regex
  - Removal of text with "@, #, numbers using Regex
  - Spelling correction using autocorrect
  - Tokenization using NLTK
  - Lemmatization using NLTK
  - Stemming using NLTK
  - Lowering
  - Removal of Punctuations
  - Removal of Stop words using NLTK
  - Removal of words less than length of 3
  - Removal of white spaces

- **For Label 1 (positive class)**



Here for positive class the majority of the words are **work** and **good.** Then **fee l, make, time, want, back, today, think, need, look** are in majority.

- **For Label 0 (negative class)**

Here for negative class the majority of the words are **work, time** and **good.** T hen the words like **hope, miss, back, need, want** are in majority.

When I compare both the word clouds for positive and negative classes, som e words are majority in both while some are in majority only in one of them. I

t can be visualized at some extent that most words let us know which tweets are written as positive and negative intention.
For example, good and work both are majority in both. So, there are tweets i n which both these words are used but with different emotions and expressio ns. Hence, at times for words like this very minor difference can be seen betw een the word clouds. But still when we enroot deep inside, the difference can be seen, because not everyone writes hate or praised words that freely.

In the conclusion, I can say that this comparison between two word clouds ga ve a brief idea of which words the audiences uses the most.

IV. Rule-Based Sentiment Analysis

- **Methodology**
  - Same as above
- **Pre-processing(in the order of execution)**
  - Removal of URLs using Regex
  - Removal of HTML Tag using Regex
  - Removal of text with “@, #, numbers using Regex
  - Spelling correction using autocorrect
  - Tokenization using NLTK
  - Lemmatization using NLTK
  - Stemming using NLTK
  - Lowering
  - Removal of Punctuations
  - Removal of Stop words using NLTK
  - Removal of words less than length of 3
  - Removal of white spaces

- **Assumptions**
  - For removal of words less than length 2, for increasing the accuracy I did so and to remove unwanted words that were not needed for polar ity score.

    ```
    (r'\b\w{1,3}\b')
    ```

  - I have used autocorrect for spell correction because I felt it was comp aratively faster than what all I have tried and used like spellchecker(it took a lot of time).

    ```
    [' '.join([spell(i) for i in x.split()]) for x in data_new['TEXT']]
    ```

  - For analysing using VADER I have used 0.05 compound as a measure f or Positive, Negative and Neutral polarity scores for every instance of l abel.
  - For accuracy I have used -0.01 as compound as it is giving better accur acy when I compared it.

- For Accuracy I have used a function in which actual positive scores are counted based on compound and same for negative also. Total positive and negative labels I have calculated from the dataset itself.

- **Accuracy for Processed Text**

```
Actual positive count:2105
Actual negative count:573
Total positive label:2287
Total negative label:2000
Positive accuracy = 92.04197638828158% of total 2
287 samples
Negative accuracy = 28.65% of total 2000 samples
Overall Accuracy = 62.46792628878003% of total 42
87 samples
```

- **Accuracy for Raw Text**

```
Actual positive count:2057
Actual negative count:884
Total positive label:2287
Total negative label:2000
Positive accuracy = 89.94315697420201% of total 2
287 samples
Negative accuracy = 44.2% of total 2000 samples
Overall Accuracy = 68.60275250758106% of total 42
87 samples
```

- **Contributions**
  - Jahnvi Kadia MT21123 q1(B), q2, q3, q4
  - Ayush Agarwal MT20501 q1(A)