READ ME

How To Run:

Code file is divided into various code cells. The function of each cell is well commented on the top of it.

Analysis:

Tools Used

1. NLTK
2. Pandas
3. Numpy
4. Num2words
5. RE

Pre-Processing Steps

1. The header of the articles is separated from the body using data.split("\n\n").

2. Have used sent_tokenize to divide the paragraphs into sentences.

3. Further used RegexpTokenizer to pre-process at word level. For a few of the tasks, special symbols like ‘@’ and ‘:’ etc were required. Therefore, I used string.split() instead.

4. Have used Porter’s stemmer to perform stemming of the documents and the input string.

5. Used num2words() to convert the digits to numbers in the given document and input string.

Methodology

Part (1) and (2)

- **Assumptions:**

1. The word count is provided for the text before pre processing.The reason is after applying the pre processing steps like num2words() , the number of words present in the given text document might vary.

2. There are certain words which begin with special symbols hence are not qualified to be called as vowels or consonants.

3. Word count is considered for both header and body of the input article.

4. Although for the sentence count only the body of the articles is considered because the header is given in the <key:value> pair, which does not qualify to be called as a sentence.

Part (3)

Have used regular expressions to find all the email ids present in the given document.

Regular Exp Used : `"[^@]+@[^@]+\.[^@]+"`

I have not made any explicit assumptions about the format of email except:

- There must be exactly one '@' symbol.
- There must be at least one '.' symbol in the email id.

Having considered this , the regular expression is capable of finding all emails ids in a more generic manner.

For example :

```
ryan_cousineau@compdyn.questor.org
<9800.97.uupcb@compdyn.questor.org>
ryan_cousineau@compdyn.questor.org
<cd1i+@andrew.cmu.edu>
ryan.cousineau@compdyn.questor.org
```

Part (4) , (5) and (6)

-Steps :

1) Input string and the given document is stemmed using Porter Stemmer.

2) Num2words is used for both a given string as well as a document.
3) Sentences are fetched from the document using sentence tokenizer.

4) Words are fetched from sentences using the regular expression ''\w+'

5) Have used try except for exception handling.

-Assumption

I have assumed that the word is present in the body of the document. Header is neglected for the same.

Part (7)

Sentences are searched for the '?' symbol at the end in order to qualify it as a question.

Part (8)

Pre-processing steps like stemming and num2words are not performed in order to maintain the correctness of the regular expression for the task.

Reg Exp `: ([01]?[0-9]|2[0-3]):[0-5][0-9]:[0-5][0-9]`

1. The above regex checks for all invalid time like 27 hours, 65 mins etc.

2. In order to detect the min and seconds from any given date, the string must be in the format **"Hour : Min : Sec"**

3. Date can present in both header as well as body of the text document.

Part (9)

Reg Exp : `\b[A-Z][0-9a-zA-Z\.]*[A-Z]\b\.?`

1. The only special symbol which is useful for abbreviation detection is ‘.’ hence other than dot, all special characters are removed from the text.

2. Stemming is intentionally skipped since Upper case characters have used to differentiate between a normal string and an abbreviation.

3. Have used the regexptokenizer(r ’\w’+). It looks for Alphanumeric strings without considering special characters.
4. Therefore in order to save the significance of ‘.’ character, I have replaced ‘.’ → ‘’ . For example U.S.A. → USA. Now regexptokenizer can be used without any problem.

5. In order to qualify an ordinary string as abbreviation, it has to follow following rules:

    - It must begin with an Upper case alphabet.

    - Middle characters can be a combination of uppercase, lower case characters and digits.

    - Last character must be an uppercase character.

    - Characters may or may not be separated by ‘.’ dot character.

    - No special characters are allowed except ‘.’ dot character.

Examples :

```
{'S.L.M.R.', 'KotRB', `ID', 'DoD', 'S.E.E.', 'GMT', 'BBS', 'B.C.',
'AJS', 'KoTL', 'HST', 'CBD', 'SQUID', 'BAT'}
```