HW1: GETTING OUR FEET WET

Part 1:

For this part first I made a function which downloads files from URL and unzips those file and read their contents. I found it very useful since, the code can be utilized whenever we have to download large number of files from websites.

For read the paragraphs, I first find the root of the XML and then iterating over that root to find TEXT tag and extracting all the lines between the /P inside /TEXT tags and then write it in the output file.

I did not find any bug however single sentences are showing in multiple lines in the output result.

OUTPUT file: I have uploaded the file containing first 100 lines with from deserialized.txt file.

first_100_lines_of_raw_file.txt

Part 2: Structuring the data

For this part I made two functions: sentence_tokenization and word_tokenization

Approach: First I read the raw deserialized.txt file line by line. The output was List of strings. Then removed the newline character from the entire file. Since the output was list, I converted it to string and Upper-case format.

Now in my cleaned file, I separated the sentences in the file using sentence tokenizer and wrote the output in a separate file, each sentence per line format and also counted the total sentences:

Total number of sentences in the corpus are 585059

I have used a modified version for removing punctuations from the entire file and it worked well on inverted commas and double triple quotes.

OUTPUT Files containing first 100 lines from sentence tokenizer and word tokenizer.

first_100_sent_tokenization_result.txt

first_100_word_tokenization_result.txt

```python
def sentence_tokenization(input_file,output_result):
    count=0
    if os.path.exists(output_result):
        os.remove(output_result)
    else:
        print("The file does not exist")
    with open(input_file,"r") as output:
        text=output.readlines()

    text_remove_n = [x.replace('\n','') for x in text]
    sentences_final=" ".join(str(x) for x in text_remove_n)
    upper_case=sentences_final.upper()
    sent_tokenized = nltk.sent_tokenize(upper_case)
    for sentence in sent_tokenized:
      count+=1
      with open(output_result,"+a") as result:
        result.write(sentence + "\n")
    print(count)
```

```python
[ ]  import string
     def word_tokenization(input_file,output_result):
         if os.path.exists(output_result):
             os.remove(output_result)
         else:
             print("The file does not exist")
         with open(output_result,"+a") as output_text:
             with open(input_file,"r") as input_text:
                 text=input_text.readlines()
             for line in text:
                 line= line.translate(str.maketrans('','',string.punctuation))
                 word_tokenized = nltk.word_tokenize(line)
                 word_tokens=" ".join(str(x) for x in word_tokenized)
                 output_text.write(word_tokens+'\n')
```

Part 3: Counting and comparing

```
total word_tokens: 16416618
```

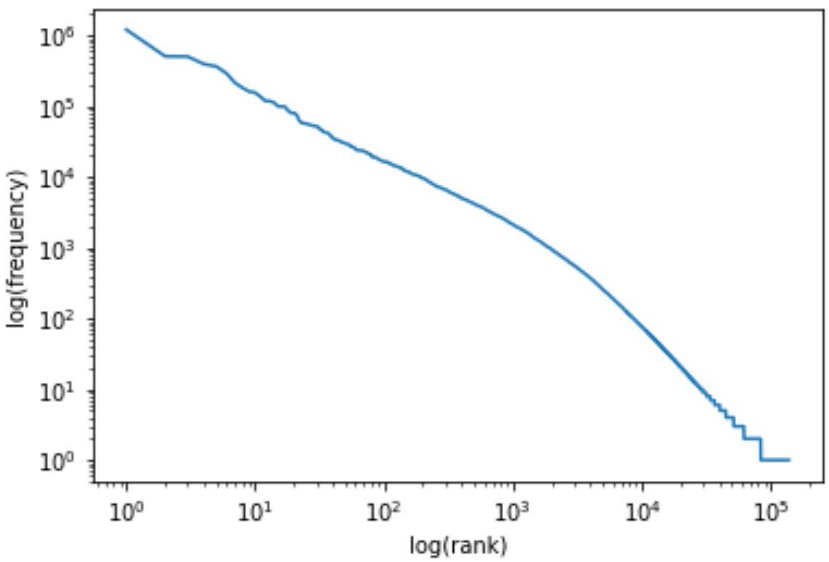
Part 1:

```
Total unique types:   138420
```

Part 2:

```
Total unigram tokens 138420
```

Part 3:



Part 4:

The 30 most frequent words are:

THE 1219252
TO 511901
OF 505729
AND 397237
IN 366589
A 292772
THAT 213959
SAID 182444
FOR 161746
TAIWAN 156616
ON 137044
WILL 119408
WITH 118781
IS 113043
AT 100494
AS 99989
BY 98934
HE 87063
BE 80622
FROM 80583
HAS 76043
WAS 62642
CHINA 58284
AN 57912
PERCENT 56560
ITS 55761
HAVE 54168
NOT 53478
IT 52786
HIS 52227

Part 5/6:
The 30 most frequent words after removing stop words:

```
SAID 182444
TAIWAN 156616
CHINA 58284
PERCENT 56560
ALSO 46602
TAIWANS 46425
CHEN 42420
GOVERNMENT 41490
PRESIDENT 39952
TAIPEI 36700
YEAR 36304
TWO 33265
MAINLAND 32436
NEW 31262
PEOPLE 29547
CHINESE 28906
ACCORDING 28557
ECONOMIC 26985
US 26659
PARTY 24495
BILLION 24212
FIRST 24029
NATIONAL 23910
ONE 23843
FOREIGN 23515
WOULD 22702
YEARS 22394
INTERNATIONAL 21710
OFFICIALS 21655
LOCAL 21156
```

Before removal of stop words, the 30 most frequent word list is more kind of stop words and not meaningful words. There are few words in English "stopwords" collections which are letters rather than words. Also, there are few words which has no meanings in English. In my understanding there are few other words in English which works as stop words like however, but , and so on can be added in to the list. When I removed the stopwords from the corpus I got to see other relevant words which came more frequently and meaningful to the corpus.

Word association metrics

30 highest PMI pairs with threshold 0 are:

```
[16416631.0, 'HANNES', 'FARLEITER']
[16416631.0, 'FREIE', 'DEMOKRATISCHE']
[16416631.0, 'CEP006', '100397']
[16416631.0, 'NICOSIA', 'GORGIE']
[16416631.0, 'GORGIE', 'MURADOV']
[16416631.0, 'CAUSUS', 'BELLI']
```

```
[16416631.0, 'HARDCOVER', 'GILTEDGED']
[16416631.0, 'US1457', 'US1522']
[16416631.0, 'FAYEZ', 'ZAWARNEH']
[16416631.0, 'CEP002', '100797']
[16416631.0, 'NN1', 'NN2']
[16416631.0, 'TULAGA', 'MANUELLA']
[16416631.0, 'LUCILLE', 'ROYBALALLARD']
[16416631.0, 'HALLDOR', 'ASGRIMSSON']
[16416631.0, 'WAHYO', 'DJATMIKO']
[16416631.0, 'FLAVONOID', 'SPONIN']
[16416631.0, 'ZCCZ', 'CEP007']
[16416631.0, 'CEP007', '101097']
[16416631.0, 'FRIEDRICH', 'NAUMANN']
[16416631.0, 'ANDRIS', 'AMERIKS']
[16416631.0, 'GERMANIC', 'MANHOOD']
[16416631.0, 'HIMMLERS', 'NUTTY']
[16416631.0, 'ZAIMAN', 'NURMATIAS']
[16416631.0, 'ESTRADE', 'OYUELA']
[16416631.0, 'TOFILAU', 'ETI']
[16416631.0, 'STEPAN', 'KERKYASHARIAN']
[16416631.0, 'ARY', 'MARDJONO']
[16416631.0, 'MESUT', 'YILMAZ']
[16416631.0, 'SIXCYLINDER', '68LITER']
[16416631.0, 'BACRE', 'WALY']
```

10 highest PMI pairs with threshold 100 are:

```
[153398.88805031445, 'SPONGIFORM', 'ENCEPHALOPATHY']
[151838.8032051282, 'MODUS', 'VIVENDI']
[120848.55581761005, 'BOVINE', 'SPONGIFORM']
[120362.83529719822, 'ALMA', 'MATER']
[109932.796875, 'SRI', 'LANKA']
[86770.30070788108, 'BLACKFACED', 'SPOONBILLS']
[79291.91925647655, 'KUALA', 'LUMPUR']
[76706.4015218939, 'QIAN', 'QICHEN']
[74566.38256410256, 'SAO', 'TOME']
[69702.46683581008, 'AU', 'OPTRONICS']
```

10 highest PMI pairs with threshold 300 are:

```
[41738.34774969683, 'BURKINA', 'FASO']
[38550.93709638842, 'MAD', 'COW']
[34070.27506385696, 'NAUTICAL', 'MILES']
[30122.82386116323, 'POUND', 'STERLING']
[27259.77165861514, 'HON', 'HAI']
[25603.43856823266, 'FALUN', 'GONG']
[22229.481236673775, 'SWEDEN', 'KRONE']
[21665.549543444144, 'DALAI', 'LAMA']
[20088.546430732, 'COSTA', 'RICA']
[17811.42771191289, 'CLOUD', 'GATE']
```

10 highest PMI pairs with threshold 1000 are:

[11844.164700561565, 'ACADEMIA', 'SINICA']
[4951.20469317968, 'STATUS', 'QUO']
[4668.343264510941, 'KEY', 'BAROMETER']
[4316.622711921648, 'SOLIDARITY', 'UNION']
[4094.8842587323657, 'VINCENT', 'SIEW']
[3922.980486932028, 'REFERENCE', 'LEVELS']
[3788.4270090432765, 'REMAINING', 'UNCHANGED']
[3630.409857494104, 'TYPHOON', 'MORAKOT']
[3507.542485549221, 'PANBLUE', 'ALLIANCE']
[3046.2218898305405, 'JAMES', 'SOONG']

PMI for "New York":

[519.3311010715668, 'NEW', 'YORK']

The PMI for "New York" is comparatively very low. In my understanding the conditional probabily is low since the word "NEW" is very common with any other word other then "YORK". "NEW" can pair frequently with other words a lso.
For threshold value 0, the tokens are not very common, the tokens looks like 2 random words. The words in the token with threshold 0 has highest PMI. It may be these words occurs very rare, but what I observe is, as increasing the threshold value, the 2-word tokens shows are most common and comes together like, costa rica, dalai lama and so on.