Data is imported

Make Sure that cik_list.xlsx and Python.py are in same folder

In [1]:

```
import pandas as pd

df = pd.read_excel ('cik_list.xlsx')
```

Complete the url of SECFNAME

In [2]:

```
df['SECFNAME_url']='https://www.sec.gov/Archives/'+df['SECFNAME']
```

Import necessary ibararies

In [3]:

```
import urllib,requests, urllib.request
```

Extract URL text and store it in list_data

In [ ]:

```
list=[]
list_data=[]
for i in df['SECFNAME_url']:
    hdr = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/9(
    req= requests.get(i, headers=hdr)
    content=req.content.decode()
    list.append(req.content.decode())
    list_data.append(content)
    print(req)
```

```
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
<Response [200]>
```

Tokenize text into sentences

In [ ]:

```
import nltk.data
import nltk
nltk.download('punkt')
sentence_list=[]
for i in list_data:
    sentence_list.append(nltk.tokenize.sent_tokenize(i))
```

Remove noise form the text lke non-alapabetical characters and tokenize into words

In [ ]:

```
import re
tokenizedsent_list=[]
for i in sentence_list:
    word_list=[]
    for y in i:
        y=y.strip()
        processed_text = y.lower()
        processed_text = re.sub('[^a-zA-Z]', ' ', processed_text )
        processed_text = re.sub(r'\s+', ' ', processed_text)
        processed_text = re.sub('<[^<]+>', '', processed_text)

        all_words = nltk.sent_tokenize(processed_text)

        word_list+=([nltk.word_tokenize(sent) for sent in all_words])
    tokenizedsent_list.append(word_list)
```

Import Master dictionary and Stop words

Make sure all the filles are in same folder as Python.py

In [ ]:

```
master = pd.read_excel ('LoughranMcDonald_MasterDictionary_2018.xlsx')
shortgeneric=open('StopWords_Generic.txt','r').readlines()
longgeneric=open('StopWords_GenericLong.txt','r').readlines()
currencies=open('StopWords_Currencies.txt','r').readlines()
auditor=open('StopWords_Auditor.txt','r').readlines()
dataandnumbers=open('StopWords_DatesandNumbers.txt','r').readlines()
geographic=open('StopWords_Geographic.txt','r').readlines()
names=open('StopWords_Names.txt','r').readlines()
```

Join all the stopwords to form a stopwords list

In [ ]:

```
stopwords = []

for i in shortgeneric,longgeneric,currencies,auditor,dataandnumbers,geographic,names:
    for y in i:
        y =y.partition('|')[0]
        y=y.strip()
        y=y.lower()
        stopwords.append(y)
```

Creating list of positive,negative and complex wors form Master Dictionary

In [ ]:

```
positive=[]
for i in range(len(master['Word'])):
    if master['Positive'][i]!=0:
        positive.append(master['Word'][i])

negative=[]
for i in range(len(master['Word'])):
    if master['Negative'][i]!=0:
        negative.append(master['Word'][i])

complex_words=[]
for i in range(len(master['Word'])):
    if master['Syllables'][i]>2:
        complex_words.append(master['Word'][i].lower())
```

Importing constraining_dictionary.xlsx and uncertainty_dictionary.xlsx' as lists

Make Sure Both the files are in same folder as Python.py

In [ ]:

```
constraining=pd.read_excel ('constraining_dictionary.xlsx')['Word'].to_list()
uncertainly=pd.read_excel ('uncertainty_dictionary.xlsx')['Word'].to_list()
```

Since positive,negative,constaraining and uncertainty are mutually excluxive, a combined dictionary sentiment_dic is made

In [ ]:

```
sentiment_dict={}
s=-1
for i in negative,positive,uncertainly,constraining:
    for element in i:
        sentiment_dict[element.lower()]=s
    s+=2
```

A counter funtion is created that return all the words in a list with their count

In [ ]:

```
def counter(slist):
    dict={}
    for y in slist:
        if y not in dict.keys():
            dict[y]=1
        else:
            dict[y]+=1
    return dict
```

All the word in a document are joined with their count to form a bag of words and stores in dictsentence

In [ ]:

```
dictsentence=[]
s=0
for i in range(len(tokenizedsent_list)):
    dummy=[]
    for y in range(len(tokenizedsent_list[i])):
        dummy+=(tokenizedsent_list[i][y])
    dictsentence.append(counter(dummy))
```

Stopwords are removed and the processed dictionaries are stored in procesed_sent

In [ ]:

```
procesed_sent=[]
s=0
for i in dictsentence:
    dummy=i.copy()
    for y in i.keys():
        if y in stopwords:
            del dummy[y]
    procesed_sent.append(dummy)
```

Each word in the list of dictionary is checked in our sentiment_dict and complex_words list. The count of positive,negative,constraint and uncertaint words for a document are increased accordingly

In [ ]:

```
score_sentiment=[]
s=0
for i in procesed_sent:
    positive_count=0
    negative_count=0
    complex_score=0
    constraining_count=0
    uncertainty_count=0
    for y in i.keys():
        if y in sentiment_dict.keys():
            if(sentiment_dict[y]==-1):
                negative_count+=i[y]
            elif(sentiment_dict[y]==1):
                positive_count+=i[y]
            elif(sentiment_dict[y]==3):
                uncertainty_count+=i[y]
            elif(sentiment_dict[y]==5):
                constraining_count+=i[y]
        if y in complex_words:
            complex_score+=i[y]
    s+=1
    score_sentiment.append([positive_count,negative_count,constraining_count,uncertainty_count,complex_score])
    print(s,[positive_count,negative_count,constraining_count,uncertainty_count,complex_score])
```

The total count of words and sentecnces for each document are calculated

In [ ]:

```
total_data=[]
for i in range(len(procesed_sent)):
    count=0
    for y in procesed_sent[i].keys():
        count+=procesed_sent[i][y]
    total_data.append([count,len(tokenizedsent_list[i])])
```

The list of respective output columns are made

In [ ]:

```
positive_score=[]
negative_score=[]
polarity_score=[]
average_sentence_length=[]
percentage_of_complex_words=[]
fog_index=[]
complex_word_count=[]
word_count=[]
uncertainty_score=[]
constraining_score=[]
positive_word_proportion=[]
negative_word_proportion=[]
uncertainty_word_proportion=[]
constraining_word_proportion=[]
constraining_words_whole_report=[]
count_cwwr=0
for cik in range(len(procesed_sent)):
    cik_p=score_sentiment[cik][0]
    cik_n=score_sentiment[cik][1]
    cik_c=score_sentiment[cik][2]
    cik_u=score_sentiment[cik][3]
    cik_cwc=score_sentiment[cik][4]
    cik_tw=total_data[cik][0]
    cik_ts=total_data[cik][1]
    positive_score.append(cik_p)
    negative_score.append(cik_n)
    constraining_score.append(cik_c)
    count_cwwr+=cik_c
    uncertainty_score.append(cik_u)
    complex_word_count.append(cik_cwc)
    word_count.append(cik_tw)
    average_sentence_length.append(cik_tw/cik_ts)
    percentage_of_complex_words.append(cik_cwc/cik_tw)
    fog_index.append(0.4*(cik_tw/cik_ts+cik_cwc/cik_tw))
    positive_word_proportion.append(cik_p/cik_tw)
    negative_word_proportion.append(cik_n/cik_tw)
    constraining_word_proportion.append(cik_c/cik_tw)
    uncertainty_word_proportion.append(cik_u/cik_tw)
    polarity_score.append((cik_p-cik_n)/(cik_p+cik_n+0.000001))
    if(cik==(len(procesed_sent)-1)):
        constraining_words_whole_report.append(count_cwwr)
    else:
        constraining_words_whole_report.append('')
```

The list are exported as Output_Score.xlsx

In [ ]:

```
data = {'CIK':df['CIK'].to_list(),
        'CONAME':df['CONAME'].to_list(),
        'FYRMO':df['FYRMO'].to_list(),
        'FDATE':df['FDATE'].to_list(),
        'FORM':df['FORM'].to_list(),
        'SECFNAME':df['SECFNAME'].to_list(),
        'positive_score':positive_score,
        'negative_score':negative_score,
        'polarity_score':polarity_score,
        'average_sentence_length':average_sentence_length,
        'percentage_of_complex_words':percentage_of_complex_words,
        'fog_index':fog_index,
        'complex_word_count':complex_word_count,
        'word_count':word_count,
        'uncertainty_score':uncertainty_score,
        'constraining_score':constraining_score,
        'positive_word_proportion':positive_word_proportion,
        'negative_word_proportion':negative_word_proportion,
        'uncertainty_word_proportion':uncertainty_word_proportion,
        'constraining_word_proportion':constraining_word_proportion,
        'constraining_words_whole_report':constraining_words_whole_report}



file= pd.DataFrame(data, columns = ['CIK','CONAME','FYRMO','FDATE','FORM','SECFNAME','positive_score','negative
                                     'polarity_score','average_sentence_length','percentage_of_complex_words','f
                                    'word_count','uncertainty_score','constraining_score','positive_word_proport
                                    'uncertainty_word_proportion','constraining_word_proportion','constraining_w

file.to_excel ('Output_Score.xlsx', index = False )
```

In [ ]:

```
polarity_score
```

In [ ]:

In [ ]: