In [1]:
```
import pandas as pd
import requests
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
from bs4 import BeautifulSoup
import re
import nltk
```

In [2]:
```
nltk.download('punkt')
nltk.download("stopwords")
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\91706\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\91706\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[2]:
```
True
```

In [8]:
```
df = pd.read_excel("cik_list.xlsx")
df.head()
```

Out[8]:

	CIK	CONAME	FYRMO	FDATE	FORM	SECFNAME
0	3662	SUNBEAM CORP/FL/	199803	1998-03-06	10-K405	edgar/data/3662/0000950170-98-000413.txt
1	3662	SUNBEAM CORP/FL/	199805	1998-05-15	10-Q	edgar/data/3662/0000950170-98-001001.txt
2	3662	SUNBEAM CORP/FL/	199808	1998-08-13	NT 10-Q	edgar/data/3662/0000950172-98-000783.txt
3	3662	SUNBEAM CORP/FL/	199811	1998-11-12	10-K/A	edgar/data/3662/0000950170-98-002145.txt
4	3662	SUNBEAM CORP/FL/	199811	1998-11-16	NT 10-Q	edgar/data/3662/0000950172-98-001203.txt

In [9]:
```
y = 'https://www.sec.gov/Archives/'
links = [y+x for x in df['SECFNAME']]
```

In [10]:
```
reports = []
for url in links:
    r = requests.get(url)
    data = r.text
    soup = BeautifulSoup(data, "html.parser")
    reports.append(soup.get_text())

print(f'Total {len(reports)} reports saved')
```

```
Total 152 reports saved
```

In [12]:
```
with open("StopWords_Generic.txt",'r') as f:
    stop_words = f.read()
```

```
stop_words = stop_words.split('\n')
print(f'Total number of Stop Words are {len(stop_words)}')
```

```
Total number of Stop Words are 121
```

In [13]:

```
master_dic = pd.read_excel("LoughranMcDonald_MasterDictionary_2018.xlsx")
master_dic.head()
```

Out[13]:

	Word	Sequence Number	Word Count	Word Proportion	Average Proportion	Std Dev	Doc Count	Negative	Positive	Uncer
0	AARDVARK	1	277	1.480368e-08	1.239377e-08	3.564730e-06	84	0	0	
1	AARDVARKS	2	3	1.603287e-10	9.725110e-12	9.863549e-09	1	0	0	
2	ABACI	3	8	4.275431e-10	1.386497e-10	6.225591e-08	7	0	0	
3	ABACK	4	12	6.413147e-10	3.159061e-10	9.383557e-08	12	0	0	
4	ABACUS	5	7250	3.874610e-07	3.681624e-07	3.366553e-05	914	0	0	

In [14]:

```
positive_dictionary = [x for x in master_dic[master_dic['Positive'] != 0]['Word']]

negative_dictionary = [x for x in master_dic[master_dic['Negative'] != 0]['Word']]

print(f"Total positve words in dictionary are {len(positive_dictionary)}")
print(f"Total negative words in dictionary are {len(negative_dictionary)}")
```

```
Total positve words in dictionary are 354
Total negative words in dictionary are 2355
```

In [16]:

```
uncertainity = pd.read_excel("uncertainty_dictionary.xlsx")
uncertainity_words = list(uncertainity['Word'])
constraining = pd.read_excel("constraining_dictionary.xlsx")
constraining_words = list(constraining['Word'])
```

In [17]:

```
def tokenize(text):
    text = re.sub(r'[^A-Za-z]',' ',text.upper())
    tokenized_words = word_tokenize(text)
    return tokenized_words

def remove_stopwords(words, stop_words):
    return [x for x in words if x not in stop_words]

def countfunc(store, words):
    score = 0
    for x in words:
        if(x in store):
            score = score+1
    return score
```

```
def sentiment(score):
    if(score < -0.5):
        return 'Most Negative'
    elif(score >= -0.5 and score < 0):
        return 'Negative'
    elif(score == 0):
        return 'Neutral'
    elif(score > 0 and score < 0.5):
        return 'Positive'
    else:
        return 'Very Positive'


def polarity(positive_score, negative_score):
    return (positive_score - negative_score)/((positive_score + negative_score)+ 0.0000

def subjectivity(positive_score, negative_score, num_words):
    return (positive_score+negative_score)/(num_words+ 0.000001)

def syllable_morethan2(word):
    if(len(word) > 2 and (word[-2:] == 'es' or word[-2:] == 'ed')):
        return False

    count =0
    vowels = ['a','e','i','o','u']
    for i in word:
        if(i.lower() in vowels):
            count = count +1

    if(count > 2):
        return True
    else:
        return False

def fog_index_cal(average_sentence_length, percentage_complexwords):
    return 0.4*(average_sentence_length + percentage_complexwords)
```

In [18]:

```
sections = ["Management's Discussion and Analysis",
            "Quantitative and Qualitative Disclosures about Market Risk\n",
            "Risk Factors\n"]
caps = [x.upper() for x in sections]

caps.extend(sections)
```

In [19]:

```
col = ['mda','qqdmr','rf']
var = ['positive_score',
      'negative_score',
      'polarity_score',
      'average_sentence_length',
      'percentage_of_complex_words',
      'fog_index',
      'complex_word_count',
      'word_count',
      'uncertainity_score',
      'constraining_score',
      'positive_word_proportion',
```

```
        'negative_word_proportion',
        'uncertainity_word_proportion',
        'constraining_word_proportion',
        'constraining_words_whole_report']


for c in col:
    for v in var[:-1]:
        df[c+'_'+v] = 0.0

df[var[-1]] = 0.0
```

In [20]:

```
df.head()
```

Out[20]:

	CIK	CONAME	FYRMO	FDATE	FORM	SECFNAME	mda_positive_score	mda_nega
0	3662	SUNBEAM CORP/FL/	199803	1998-03-06	10-K405	edgar/data/3662/0000950170-98-000413.txt	0.0	
1	3662	SUNBEAM CORP/FL/	199805	1998-05-15	10-Q	edgar/data/3662/0000950170-98-001001.txt	0.0	
2	3662	SUNBEAM CORP/FL/	199808	1998-08-13	NT 10-Q	edgar/data/3662/0000950172-98-000783.txt	0.0	
3	3662	SUNBEAM CORP/FL/	199811	1998-11-12	10-K/A	edgar/data/3662/0000950170-98-002145.txt	0.0	
4	3662	SUNBEAM CORP/FL/	199811	1998-11-16	NT 10-Q	edgar/data/3662/0000950172-98-001203.txt	0.0	

5 rows × 49 columns

In [21]:

```
section_map = {i:j for i,j in zip(sections, col)}
s_map = {i.upper():j for i,j in zip(sections, col)}

section_map.update(s_map)
```

In [22]:

```
for i in range(len(reports)):
    text = re.sub('Item','ITEM',reports[i])
    for j in caps:
        x = re.search('ITEM\s+[\d]\(*[A-Za-z]*\)*.*\s+\-*\s*'+j, text)

        if x:
            start,end = x.span()
            content = (text[start:]).split('ITEM')[1]
            if ('...' not in content) and ('. . .' not in content) and len(content) > 2
                tokenized_words = tokenize(content)
                #print(f'Total tokenized words are {len(tokenized_words)}')
                words = remove_stopwords(tokenized_words, stop_words)
                num_words = len(words)
                #print(f'Total words after removing stop words are {len(words)}')
                positive_score = countfunc(positive_dictionary, words)
                negative_score = countfunc(negative_dictionary, words)
                #print(f'Total positive score is {positive_score}')
```

```
                #print(f'Total negative score is {negative_score}')
                polarity_score = polarity(positive_score, negative_score)
                #print(polarity_score)
                subjectivity_score = subjectivity(positive_score, negative_score, num_w
                #print(subjectivity_score)
                #print(sentiment(polarity_score))

                sentences = sent_tokenize(content)
                num_sentences = len(sentences)
                average_sentence_length = num_words/num_sentences
                #print(average_sentence_length)


                num_complexword =0
                uncertainity_score = 0
                constraining_score = 0

                for word in words:
                    if(syllable_morethan2(word)):
                        num_complexword = num_complexword+1

                    if(word in uncertainity_words):
                        uncertainity_score = uncertainity_score+1

                    if(word in constraining_words):
                        constraining_score = constraining_score+1

                #print(num_complexword)
                #print(uncertainity_score)
                #print(constraining_score)


                percentage_complexwords = num_complexword/num_words
                #print(percentage_complexwords)
                fog_index = fog_index_cal(average_sentence_length, percentage_complexwo
                #print(fog_index)

                positive_word_proportion = positive_score/num_words
                negative_word_proportion = negative_score/num_words
                uncertainity_word_proportion = uncertainity_score/num_words
                constraining_word_proportion = constraining_score/num_words

                #print(positive_word_proportion)
                #print(negative_word_proportion)
                #print(uncertainity_word_proportion)
                #print(constraining_word_proportion)
                df.at[i,section_map[j]+'_positive_score'] = positive_score
                df.at[i,section_map[j]+'_negative_score'] = negative_score
                df.at[i,section_map[j]+'_polarity_score'] = polarity_score
                df.at[i,section_map[j]+'_average_sentence_length'] = average_sentence_l
                df.at[i,section_map[j]+'_percentage_of_complex_words'] = percentage_com
                df.at[i,section_map[j]+'_fog_index'] = fog_index
                df.at[i,section_map[j]+'_complex_word_count'] = num_complexword
                df.at[i,section_map[j]+'_word_count'] = num_words
                df.at[i,section_map[j]+'_uncertainity_score'] = uncertainity_score
                df.at[i,section_map[j]+'_constraining_score'] = constraining_score
                df.at[i,section_map[j]+'_positive_word_proportion'] = positive_word_pro
                df.at[i,section_map[j]+'_negative_word_proportion'] = negative_word_pro
                df.at[i,section_map[j]+'_uncertainity_word_proportion'] = uncertainity_
                df.at[i,section_map[j]+'_constraining_word_proportion'] = constraining_
```

```
    constraining_words_whole_report = 0
    tokenized_report_words = tokenize(reports[i])
    report_words = remove_stopwords(tokenized_report_words, stop_words)
    for word in report_words:
        if word in constraining_words:
            constraining_words_whole_report = 1+ constraining_words_whole_report
    #print(constraining_words_whole_report)
    df.at[i,'constraining_words_whole_report'] = constraining_words_whole_report
```

In [23]:
```
df.head()
```

Out[23]:

	CIK	CONAME	FYRMO	FDATE	FORM	SECFNAME	mda_positive_score	mda_nega
0	3662	SUNBEAM CORP/FL/	199803	1998-03-06	10-K405	edgar/data/3662/0000950170-98-000413.txt	0.0	
1	3662	SUNBEAM CORP/FL/	199805	1998-05-15	10-Q	edgar/data/3662/0000950170-98-001001.txt	0.0	
2	3662	SUNBEAM CORP/FL/	199808	1998-08-13	NT 10-Q	edgar/data/3662/0000950172-98-000783.txt	0.0	
3	3662	SUNBEAM CORP/FL/	199811	1998-11-12	10-K/A	edgar/data/3662/0000950170-98-002145.txt	0.0	
4	3662	SUNBEAM CORP/FL/	199811	1998-11-16	NT 10-Q	edgar/data/3662/0000950172-98-001203.txt	0.0	

5 rows × 49 columns

In [24]:
```
df.to_excel("Output Data Structure.xlsx")
```

In [25]:
```
df
```

Out[25]:

	CIK	CONAME	FYRMO	FDATE	FORM	SECFNAME	mda_positive_score	mda_
0	3662	SUNBEAM CORP/FL/	199803	1998-03-06	10-K405	edgar/data/3662/0000950170-98-000413.txt	0.0	
1	3662	SUNBEAM CORP/FL/	199805	1998-05-15	10-Q	edgar/data/3662/0000950170-98-001001.txt	0.0	
2	3662	SUNBEAM CORP/FL/	199808	1998-08-13	NT 10-Q	edgar/data/3662/0000950172-98-000783.txt	0.0	
3	3662	SUNBEAM CORP/FL/	199811	1998-11-12	10-K/A	edgar/data/3662/0000950170-98-002145.txt	0.0	
4	3662	SUNBEAM CORP/FL/	199811	1998-11-16	NT 10-Q	edgar/data/3662/0000950172-98-001203.txt	0.0	

	CIK	CONAME	FYRMO	FDATE	FORM	SECFNAME	mda_positive_score	mda_
...	...	...	...	...	...	...	...	
147	12239	SPHERIX INC	200704	2007-04-02	10-K	edgar/data/12239/0001104659-07-024804.txt	0.0	
148	12239	SPHERIX INC	200705	2007-05-16	NT 10-Q	edgar/data/12239/0001104659-07-040463.txt	0.0	
149	12239	SPHERIX INC	200705	2007-05-18	10-Q	edgar/data/12239/0001104659-07-041441.txt	0.0	
150	12239	SPHERIX INC	200705	2007-05-23	10-K/A	edgar/data/12239/0001104659-07-042333.txt	0.0	
151	12239	SPHERIX INC	200708	2007-08-14	10-Q	edgar/data/12239/0001104659-07-062470.txt	0.0	

152 rows × 49 columns

In [ ]: