In [ ]:
```
import pandas as pd
import numpy as np
import glob
import re
from collections import Counter
import datetime
```

Load Index files

In [ ]:
```
path = 'C:/Users/Rocku/Downloads/SeekingAlpha_EarningCalls/full_indexed_utf8.csv'
df = pd.read_csv(path)
df_ex = df.loc[df['PositionIndex2'] == 1]
df_ex.reset_index(drop=True, inplace=True)
```

In [ ]:
```
path2 = 'C:/Users/Rocku/Downloads/SeekingAlpha_EarningCalls/index_file2.csv'
df_full_index = pd.read_csv(path2)
```

In [ ]:
```
df_ex_example = df_ex[300000:]
total = df_ex_example.count()[0]
```

In [ ]:

```
def count_words(indi_df, qna_index):
    regex1 = re.compile('[.,?!]')

    transcript_dict= {}

    indi_list = []
    for i, transcript in indi_df.iterrows():
        indi_list.append(str(transcript['Transcripts']))

    indi = '. '.join(indi_list)
    indi = indi.replace('..', '.').replace('\n', '.')
    splitted_transcript = regex1.split(indi.strip())[:-1]

    wordcount = Counter(indi.replace(',',' ').replace('.',' ')
                        .replace('!',' ').replace('?',' ').lower().split())

    word_a = wordcount['a'] + wordcount['an']
    word_can = wordcount['can']
    word_could = wordcount['could']
    word_may = wordcount['may']
    word_might = wordcount['might']
    word_must = wordcount['must']
    word_shall = wordcount['shall']
    word_will = wordcount['will']
    word_would = wordcount['would']

    if len(splitted_transcript) != 0:
        transcript_dict['N_WPerS_%s'%qna_index] = round(len(wordcount)/
                                                  len(splitted_transcript), 3)
    else:
        transcript_dict['N_WPerS_%s'%qna_index] = 0

    transcript_dict['N_A_%s'%qna_index] = wordcount['a'] + wordcount['an']
    transcript_dict['N_The_%s'%qna_index] = wordcount['the']
    transcript_dict['N_Can_%s'%qna_index] = wordcount['can']
    transcript_dict['N_Could_%s'%qna_index] = wordcount['could']
    transcript_dict['N_May_%s'%qna_index] = wordcount['may']
    transcript_dict['N_Might_%s'%qna_index] = wordcount['might']
    transcript_dict['N_Must_%s'%qna_index] = wordcount['must']
    transcript_dict['N_Shall_%s'%qna_index] = wordcount['shall']
    transcript_dict['N_Will_%s'%qna_index] = wordcount['will']
    transcript_dict['N_Would_%s'%qna_index] = wordcount['would']

    transcript_dict['Transcript_%s'%qna_index] = indi
    transcript_dict['N_Words_%s'%qna_index] = len(wordcount)
    transcript_dict['N_Sent_%s'%qna_index] = len(splitted_transcript)
    transcript_dict['N_Par_%s'%qna_index] = indi_df.count()[0]

    del indi_list, indi_df, indi, wordcount, splitted_transcript

    return transcript_dict
```

In [ ]:
```
def check_talker(filename, talkers):
    data = df_full_index.loc[df_full_index['FileName'] == filename]
    data = data['Name'].tolist()

    not_in_lists = []
    for talker in talkers:
        talker = talker.strip()
        if ((talker.startswith('Quest')) or (talker.startswith('Operat')):
            pass
        else:
            if (len(talker.split()) < 4) & (talker not in data):
                not_in_lists.append(talker)

    del data, talkers

    return not_in_lists
```

Main

In [ ]:

```
columns = ['URL', 'Title', 'UploadDate', 'CompanyName', 'Exchange', 'Ticker',
           'FileIndex', 'NameIndex','PositionIndex1',
           'Subtitle', 'CallDate', 'Executive',
           'N_Sent_All', 'N_Sent_BeforeQ','N_Sent_AfterQ',
           'N_Par_All', 'N_Par_BeforeQ','N_Par_AfterQ',
           'N_WPerS_All', 'N_WPerS_BeforeQ','N_WPerS_AfterQ',
           'N_Can_All', 'N_Can_BeforeQ','N_Can_AfterQ',
           'N_Could_All', 'N_Could_BeforeQ','N_Could_AfterQ',
           'N_May_All', 'N_May_BeforeQ', 'N_May_AfterQ',
           'N_Might_All', 'N_Might_BeforeQ','N_Might_AfterQ',
           'N_Must_All', 'N_Must_BeforeQ','N_Must_AfterQ',
           'N_Shall_All', 'N_Shall_BeforeQ','N_Shall_AfterQ',
           'N_Will_All', 'N_Will_BeforeQ','N_Will_AfterQ',
           'N_A_All', 'N_A_BeforeQ','N_A_AfterQ',
           'N_The_All', 'N_The_BeforeQ','N_The_AfterQ',
           'N_Words_All', 'N_Words_BeforeQ','N_Words_AfterQ'
          ]
df_output = pd.DataFrame(columns=columns)

columns_not_in_lits = ['FileIndex','FileName', 'NotInList']

df_not_in_list_table = pd.DataFrame(columns=columns_not_in_lits)

failed = set()
next_time = set()
successed = set()

total = df_ex.count()[0]

prev_path = None
# for i, row in df_ex.iterrows():
for i, row in df_ex_example.iterrows():
    if (i%100 == 1):
        print(i, '/', total, row['FileName'], 'proceeding:', datetime.datetime.now())
    filename = row['FileName']

    try:
        file_path = 'C:/Users/Rocku/Downloads/SeekingAlpha_EarningCalls/raw_xlsx/' +
filename
        if (prev_path is not None) and (prev_path == file_path):
```

```
            df_temp = df_temp
        else:
            df_temp = pd.read_excel(file_path, header=None, index_col=0, encoding='ut
f-8')
            prev_path = file_path

        # Get Transcript text
        df_trans = df_temp.loc['Transcript']
        df_trans = df_trans.reset_index(drop=True)
        df_trans.rename(columns={1:'Name', 2:'Transcripts'}, inplace=True)

        # Find talkers in transcript who are not in the Executives and Analysts list
        df_trans.loc[:,'Name'] = df_trans.loc[:,'Name'].str.split(r'\s\W+\s').str[0]
        talkers = list(set(df_trans['Name'].tolist()))
        not_in_list = check_talker(filename=filename, talkers = talkers)

        if (len(not_in_list) == 0):
            # Find QnA Index to divide into two call sessions
            try:
                try:
                    qna_index = df_trans.loc[df_trans['Name'] == 'Question-and-Answer
 Session'].index[0]
                except:
                    qna_index = df_trans.loc[df_trans['Name'] == 'Question and Answer
 Session'].index[0]

                df_trans_before_qna = df_trans.loc[:qna_index-1]
                df_trans_after_qna = df_trans.loc[qna_index+1:]

            except:
                df_trans_before_qna = df_trans
                df_trans_after_qna = df_trans.loc[:0]


            indi_after = df_trans_after_qna.loc[df_trans_after_qna['Name'] ==  row['N
ame']]
            indi_before = df_trans_before_qna.loc[df_trans_before_qna['Name'] ==
row['Name']]

            # Word count
            transcript_dict_Before = count_words(indi_before, 'BeforeQ')
            transcript_dict_After = count_words(indi_after, 'AfterQ')

            try:
                Title = df_temp.loc['Title'][1]
            except:
                Title = 'None'

            try:
                Subtitle = df_temp.loc['Subtitle'][1]
                if Subtitle == 'Executives':
                    Subtitle = 'None'
            except:
                Subtitle = 'None'

            row_dict = {}

            row_dict['CompanyName'] = row['CompanyName']
            row_dict['Exchange'] = row['Exchange']
            row_dict['Ticker'] = row['Ticker']
            row_dict['FileIndex'] = row['FileIndex']
            row_dict['UploadDate'] = row['UploadDate']
            row_dict['NameIndex'] = row['NameIndex']
```

```
            row_dict['PositionIndex1'] = row['PositionIndex1']
            row_dict['CallDate'] = row['CallDate']
            row_dict['Executive'] = str(row['Name']) + '-' + str(row['OriginalPositio
n'])
            row_dict['URL'] = 'https://seekingalpha.com/article/' + row['FileName'].r
eplace('_', '.').split('.')[1]
            row_dict['Title']  = Title
            row_dict['Subtitle'] = Subtitle

            row_dict['N_Sent_BeforeQ'] = transcript_dict_Before['N_Sent_BeforeQ']
            row_dict['N_Sent_AfterQ'] = transcript_dict_After['N_Sent_AfterQ']
            row_dict['N_Sent_All'] = row_dict['N_Sent_BeforeQ'] + row_dict['N_Sent_Af
terQ']

            row_dict['N_WPerS_BeforeQ'] = transcript_dict_Before['N_WPerS_BeforeQ']
            row_dict['N_WPerS_AfterQ'] = transcript_dict_After['N_WPerS_AfterQ']
            if (row_dict['N_Sent_All'] == 0):
                row_dict['N_WPerS_All'] = 0
            else:
                row_dict['N_WPerS_All'] =
round((row_dict['N_WPerS_BeforeQ']*row_dict['N_Sent_BeforeQ']+row_dict['N_WPerS_After
Q']*row_dict['N_Sent_AfterQ'])/(row_dict['N_Sent_All']), 3)

            row_dict['N_Can_BeforeQ'] = transcript_dict_Before['N_Can_BeforeQ']
            row_dict['N_Can_AfterQ'] = transcript_dict_After['N_Can_AfterQ']
            row_dict['N_Can_All'] = row_dict['N_Can_BeforeQ'] + row_dict['N_Can_After
Q']

            row_dict['N_Could_BeforeQ'] = transcript_dict_Before['N_Could_BeforeQ']
            row_dict['N_Could_AfterQ'] = transcript_dict_After['N_Could_AfterQ']
            row_dict['N_Could_All'] = row_dict['N_Could_BeforeQ'] + row_dict['N_Could
_AfterQ']

            row_dict['N_May_BeforeQ'] = transcript_dict_Before['N_May_BeforeQ']
            row_dict['N_May_AfterQ'] = transcript_dict_After['N_May_AfterQ']
            row_dict['N_May_All'] = row_dict['N_May_BeforeQ'] + row_dict['N_May_After
Q']

            row_dict['N_Might_BeforeQ'] = transcript_dict_Before['N_Might_BeforeQ']
            row_dict['N_Might_AfterQ'] = transcript_dict_After['N_Might_AfterQ']
            row_dict['N_Might_All'] = row_dict['N_Might_BeforeQ'] + row_dict['N_Might
_AfterQ']

            row_dict['N_Must_BeforeQ'] = transcript_dict_Before['N_Must_BeforeQ']
            row_dict['N_Must_AfterQ'] = transcript_dict_After['N_Must_AfterQ']
            row_dict['N_Must_All'] = row_dict['N_Must_BeforeQ'] + row_dict['N_Must_Af
terQ']

            row_dict['N_Shall_BeforeQ'] = transcript_dict_Before['N_Shall_BeforeQ']
            row_dict['N_Shall_AfterQ'] = transcript_dict_After['N_Shall_AfterQ']
            row_dict['N_Shall_All'] = row_dict['N_Shall_BeforeQ'] + row_dict['N_Shall
_AfterQ']

            row_dict['N_Will_BeforeQ'] = transcript_dict_Before['N_Will_BeforeQ']
            row_dict['N_Will_AfterQ'] = transcript_dict_After['N_Will_AfterQ']
            row_dict['N_Will_All'] = row_dict['N_Will_BeforeQ'] + row_dict['N_Will_Af
terQ']

            row_dict['N_A_BeforeQ'] = transcript_dict_Before['N_A_BeforeQ']
            row_dict['N_A_AfterQ'] = transcript_dict_After['N_A_AfterQ']
            row_dict['N_A_All'] = row_dict['N_A_BeforeQ'] + row_dict['N_A_AfterQ']

            row_dict['N_The_BeforeQ'] = transcript_dict_Before['N_The_BeforeQ']
```

```
            row_dict['N_The_AfterQ'] = transcript_dict_After['N_The_AfterQ']
            row_dict['N_The_All'] = row_dict['N_The_BeforeQ'] + row_dict['N_The_After
Q']

            row_dict['N_Words_BeforeQ'] = transcript_dict_Before['N_Words_BeforeQ']
            row_dict['N_Words_AfterQ'] = transcript_dict_After['N_Words_AfterQ']
            row_dict['N_Words_All'] = row_dict['N_Words_BeforeQ'] + row_dict['N_Words
_AfterQ']

            row_dict['N_Par_BeforeQ'] = transcript_dict_Before['N_Par_BeforeQ']
            row_dict['N_Par_AfterQ'] = transcript_dict_After['N_Par_AfterQ']
            row_dict['N_Par_All'] = row_dict['N_Par_BeforeQ'] + row_dict['N_Par_After
Q']



            df_new = pd.DataFrame([row_dict], columns=columns)
            df_output = df_output.append(df_new)

            del df_new, indi_after, indi_before, transcript_dict_Before, transcript_d
ict_After
            successed.add(filename)

        else:
            not_in_lists_dict = {}

            if (row['FileIndex'] not in df_not_in_list_table['FileIndex'].tolist()):
                for name in not_in_list:
                    not_in = {}
                    not_in_lists_dict['FileName'] = row['FileName']
                    not_in_lists_dict['FileIndex'] = row['FileIndex']
                    not_in_lists_dict['NotInList'] = name

                    df_not_in_lists = pd.DataFrame([not_in_lists_dict], columns=colum
ns_not_in_lits)
                    df_not_in_list_table = df_not_in_list_table.append(df_not_in_list
s)
                    del df_not_in_lists
                del not_in_lists_dict

            next_time.add(filename)

    except:
        failed.add(filename)
```

In [ ]:
```
df_output.to_csv('word_count_by_index_4.csv', header='column_names', index=False, enc
oding='utf-8')
```

In [ ]:
```
df_not_in_list_table.to_csv('not_in_lists_4.csv', header='column_names', index=False,
 encoding='utf-8')
```