crawling 10K data from web

In [ ]:
```
# -*- coding: utf-8 -*-



#This code is used for downloading 10K report, not 10Q
import os, csv, urllib2, time, re
from bs4 import BeautifulSoup
import pandas as pd
import numpy as np

##### Section 1: Extract the    URLs    from each       firms' search  results returned by     Edgar
os.chdir('H:\python\project')
companyListFile = "CompanyTickerList.csv"  # a csv file with the list of company ticker symbols and names
IndexLinksFile = "IndexLinks10K.csv"  # a csv file (output of the current script) with the list of index links for each firm

def getIndexLink(companyListFile, FormType):
    csvFile = open(companyListFile, "r")
    csvReader = csv.reader(csvFile, delimiter=",")
    csvData = list(csvReader)

    Full_List = []
    for rowData in csvData[1:]:
        tickerCode = rowData[0]

        urlLink = "https://www.sec.gov/cgi-bin/browse-edgar?action=getcompany&CIK=" + tickerCode + "&type=" + FormType + "&dateb=&owner=exclude&count=100"
        pageRequest = urllib2.Request(urlLink)
        pageOpen = urllib2.urlopen(pageRequest)
        pageRead = pageOpen.read()

        soup = BeautifulSoup(pageRead, "html.parser")

        # Check if there is a table to extract / code exists in edgar database
        try:
            table = soup.find("table", {"class": "tableFile2"})
        except:
            print "No tables found or no matching ticker symbol for ticker symbol for" + tickerCode
            return -1

        docIndex = 1
        for row in table.findAll("tr"):
            cells = row.findAll("td")
            if len(cells) == 5:
                if cells[0].text.strip() == FormType:
                    link = cells[1].find("a", {"id": "documentsbutton"})
                    docLink = "https://www.sec.gov" + link['href']
                    description = cells[2].text.encode(
                        'utf8').strip()
                    filingDate = cells[3].text.encode('utf8').strip()
                    newfilingDate = filingDate.replace("-", "_")  ### <=== Change date format from 2012-1-1 to 2012_1_1 so it can be used as part of 10-K file names
                    docIndex = docIndex + 1
                    rows = [tickerCode, docIndex, docLink, description, filingDate, newfilingDate]
                    Full_List.append(rows)

    headers = ['Ticker', 'DocIndex', 'IndexLink', 'Description', 'FilingDate', 'NewFilingDate']
```

```
    data_links = pd.DataFrame(Full_List, columns=headers)
    data_links.to_csv(IndexLinksFile, index=False)

FormType = "10-K" ### <=== Change to other type if needed. For now, we extract "10-K" report

csvOutput = open(IndexLinksFile, "a+b")
csvOutput.truncate()

getIndexLink(companyListFile, FormType)

print "done!"

#### Section2: Extracts the URLs for each firm's 10-K reports

Form10qListFile = "List10K.csv"  # a csv file (output of the current script) with the list of 10-K links for each firm

def get10qLink(IndexLinksFile, FormType):
    csvFile = open(IndexLinksFile, "r")
    csvReader = csv.reader(csvFile, delimiter=",")
    csvData = list(csvReader)

    Full_List = []
    i = 1
    for rowData in csvData[1:]:
        Ticker = rowData[0]
        DocIndex = rowData[1]
        DocLink = rowData[2]
        Description = rowData[3]
        FileDate = rowData[4]
        NewFileDate = rowData[5]

        pageRequest = urllib2.Request(DocLink)
        pageOpen = urllib2.urlopen(pageRequest)
        pageRead = pageOpen.read()

        soup = BeautifulSoup(pageRead, "html.parser")

        # Check if there is a table to extract / code exists in edgar database
        try:
            table = soup.find("table", {"summary": "Document Format Files"})
        except:
            print "No tables found for link " + DocLink

        for row in table.findAll("tr"):
            cells = row.findAll("td")
            if len(cells) == 5:
                if cells[3].text.strip() == FormType:
                    link = cells[2].find("a")
                    FormLink= "https://www.sec.gov" + link['href']
                    FormName = link.text.encode('utf8').strip()
        rows = [Ticker, DocIndex, DocLink, Description, FileDate, NewFileDate, FormLink, FormName]
        Full_List.append(rows)

        nbDocPause = 10  ### <=== Type your number of documents to download in one batch
        nbSecPause = 1  ### <=== Type your pausing time in seconds between each batch
        if i % nbDocPause == 0:
            print i
            print "Pause for " + str(nbSecPause) + " second .... "
            time.sleep(float(nbSecPause))
        i = i + 1
```

```
    headers = ['Ticker', 'DocIndex', 'IndexLink', 'Description', 'FilingDate', 'NewFilingDate', 'Form10KLink', 'Form10KName']
    data_links = pd.DataFrame(Full_List, columns=headers)
    data_links.to_csv(Form10qListFile, index=False)

FormType = "10-K" ### <=== Change to other type if needed. For now, we extract "10-K" report

csvOutput = open(Form10qListFile, "a+b")
csvOutput.truncate()

get10qLink(IndexLinksFile, FormType)
print "done!"

##### Section 3: Downloads the  10-K    reports as      HTML    files
SubPath = "./10-K report/HTML/"  # <===The subfolder with the 10-K files in HTML format
logFile = "DownloadLog10K.csv"  # a csv file (output of the current script) with the download history of 10-K forms

def dowmload10q(Form10qListFile, FormYears):
    csvFile = open(Form10qListFile, "r")
    csvReader = csv.reader(csvFile, delimiter=",")
    csvData = list(csvReader)

    Full_List = []
    i = 1
    for rowData in csvData[1:]:
        Ticker = rowData[0]
        DocIndex = rowData[1]
        IndexLink = rowData[2]
        Description = rowData[3]
        FilingDate = rowData[4]
        NewFilingDate = rowData[5]
        FormLink = rowData[6]
        FormName = rowData[7]

        for year in FormYears:
            if year in FilingDate:
                pageRequest = urllib2.Request(FormLink)
                pageOpen = urllib2.urlopen(pageRequest)
                pageRead = pageOpen.read()

                if ".htm" in FormName:
                    try:
                        htmlname = Ticker + "_" + DocIndex + "_" + NewFilingDate + ".htm"
                        htmlpath = SubPath + htmlname
                        htmlfile = open(htmlpath, 'wb')
                        htmlfile.write(pageRead)
                        htmlfile.close()
                        rows = [Ticker, DocIndex, IndexLink, Description, FilingDate, NewFilingDate, FormLink, FormName, htmlname, ""]
                    except:
                        rows = [Ticker, DocIndex, IndexLink, Description, FilingDate, NewFilingDate, FormLink, FormName, "not downloaded"]
                elif ".txt" in FormName:
                    try:
                        textname = Ticker + "_" + DocIndex + "_" + NewFilingDate + ".txt"
                        textpath = SubPath + textname
                        textfile = open(textpath, 'wb')
                        textfile.write(pageRead)
                        textfile.close()
                        rows = [Ticker, DocIndex, IndexLink, Description, FilingDate, NewFilingDate, FormLink, FormName, textname, ""]
                    except:
                        rows = [Ticker, DocIndex, IndexLink, Description, FilingDate, NewFilingDate, FormLink, FormName, "not downloaded"]
                else:
                    rows = [Ticker, DocIndex, IndexLink, Description, FilingDate, NewFilingDate, FormLink, FormName, "", "No form"]
```

```
                Full_List.append(rows)

        nbDocPause = 10  ### <=== Type your number of documents to download in one batch
        nbSecPause = 1  ### <=== Type your pausing time in seconds between each batch
        if i % nbDocPause == 0:
            print i
            print "Pause for " + str(nbSecPause) + " second .... "
            time.sleep(float(nbSecPause))
        i = i + 1
    headers = ['Ticker', 'DocIndex', 'IndexLink', 'Description', 'FilingDate', 'NewFilingDate', 'Form10KLink', 'Form10KName', "FileName", "Note"]
    data_links = pd.DataFrame(Full_List, columns=headers)
    data_links.to_csv(logFile, index=False)

if not os.path.isdir(SubPath):
    os.makedirs(SubPath)

FormYears = ['2006','2007','2008','2009','2010','2011', '2012', '2013', '2014', '2015', '2016']  ### <=== Type the years of documents you wish to download here, now we only
                                        ### download files for all these years listed.

csvOutput = open(logFile, "a+b")
csvOutput.truncate()

dowmload10q(Form10qListFile, FormYears)
print "done!"
```

Sentiment Analysis

In [7]:
```
#小组分工
from __future__ import division
from string import punctuation
import csv
import urllib
import os

os.chdir('C:/Users/bjd/git/Finance/python spring/project/textfile')#改成你们自己的路径
#请打开你选的股票的一个年报，ctrl+f启用搜索，迅速定位到risk factors，将risk factors
#这个部分整个复制下来，保存到一个新的txt文档中，并将之命名。记住这个txt文档，应该在
#上面那行的路径里面创建
urllib.urlretrieve('http://www.unc.edu/~ncaren/haphazard/positive.txt', 'positive.txt')
urllib.urlretrieve('http://www.unc.edu/~ncaren/haphazard/negative.txt', 'negative.txt')


#把下面这个text的内容改成你们的txt文件的名字
text='ko2016','ko2015','ko2014','ko2013','ko2012','ko2011','ko2010','ko2009','ko2008','ko2007','ko2006'
for i in range(len(text)):
    msgs=open(text[i]+'.txt').read().lower()
    pos = open("positive.txt").read()
    positive_words = pos.split('\n')
    positive_number = 0

    neg = open('negative.txt').read()
    negative_words = neg.split('\n')
    negative_number = 0

    for p in punctuation:

        words = msgs.replace(p, '') #把标点符号去掉，换成空格，此时words是一个string，
                                    #里面的元素是一个个字母
        words = list(words.split(' '))#以空格为间隔，将words变成list，这样里面的元素
                                    #就是一个个单词了
        word_count = len(words)
    for word in words:
        if word in positive_words:
            positive_number += 1
        elif word in negative_words:
            negative_number += 1
    #print word_count,positive_number, negative_number, text[i]
### Uncomment if you need percentage
    positive_pencentage = positive_number / word_count
    negative_percentage = negative_number / word_count
    print positive_pencentage, negative_percentage, text[i]


#将这次运行得到的数值，保存进一个csv文件中。这里直接参照老师上传的10—K1A来（把老师
#那个文件里的数据全部删掉，只保留第一行，然后填进去，filename可以只填对应年份
```

```
0.0330214047542 0.0254920942239 ko2016
0.0311302681992 0.0249042145594 ko2015
0.0309186150505 0.0241471828287 ko2014
0.0295445219532 0.024620434961 ko2013
0.0310969961625 0.0227603546381 ko2012
0.029887029887 0.023452023452 ko2011
0.0262192885955 0.0243857719105 ko2010
0.0262192885955 0.0243857719105 ko2009
0.0255748979153 0.020202020202 ko2008
0.0260481270156 0.023319275614 ko2007
0.0682401231401 0.0218060543869 ko2006
```

CAR Analysis

In [7]:
```
import pandas_datareader.data as pdr
import pandas as pd
import numpy as np
import datetime, os, csv
import statsmodels.api as sm
```

In [8]:
```
os.chdir( "C:/Users/bjd/git/Finance/python spring/project")



input_path10K1A = "10-K1A.csv"
csvFile10K1A = open(input_path10K1A, "r")
csvReader10K1A = csv.reader(csvFile10K1A, delimiter=",")
csvData10K1A = list(csvReader10K1A)
```

In [9]:
```
input_path = "CompanyTickerList.csv"
csvFile = open(input_path, "r")
csvReader = csv.reader(csvFile, delimiter=",")
csvData = list(csvReader)
```

In [10]:
```
start = datetime.datetime(2006, 1, 1) #<==change start time here
end = datetime.datetime(2017, 1, 1) #<==change end time here

# Read-in data for SPY
spy = pdr.DataReader("SPY", 'yahoo', start, end)
spy_adj = spy['Adj Close']
spy_ret = np.log(spy_adj).diff().dropna()

## Read-in data for stocks
stocks = []
for rowData in csvData[1:]:
    ticker = rowData[0]
    stocks.append(ticker)
price = pdr.DataReader(stocks, 'yahoo', start, end)
key_word = 'Adj Close'
cleanData = price.ix[key_word]
adj_close = pd.DataFrame(cleanData)
ret = np.log(adj_close).diff().dropna()
rows = ret.shape[0]

headers = sorted(stocks)
```

In [ ]:
```

```

In [15]:
```
for i in range(7): # <===N=7 securities in this portfolio. Change this to fit the size of your portfolio
    stock = ret.ix[:, i]
    stock_name = headers[i]
    spy_ret = sm.add_constant(spy_ret)
    model = sm.OLS(stock, spy_ret).fit()
    ret[stock_name] = model.resid #<===dataframe of residual
```

In [20]:
```
ret.head()
```

Out[20]:

	AMZN	BA	FE	GS	KO	VZ	WMT
Date							
2006-01-04	-0.012877	0.005512	0.006117	-0.020527	-0.004858	0.025304	-0.000646
2006-01-05	0.007055	-0.012595	0.001129	-0.001152	0.004320	0.010769	-0.014112
2006-01-06	-0.005261	-0.022377	0.003820	0.002354	0.004534	-0.002091	-0.000332
2006-01-09	-0.020164	-0.011049	0.003220	0.008428	0.003147	0.002092	-0.005158
2006-01-10	-0.032642	0.003686	-0.002679	0.011196	0.000177	0.003174	0.002657

In [16]:
```
car = []
for rowData in csvData1OK1A[1:]:
    ticker = rowData[0]
    filingdate = rowData[2]
    positive_pct = rowData[5]
    negative_pct = rowData[7]

    try:
        tic_index = headers.index(ticker)
    except ValueError:
        print "ticker not in list."
    residual = ret.ix[:, tic_index]
    date_index = residual.index.get_loc(filingdate)
    x = 3 #<==analayze 3 days before and 3 days after the report date.
    date_start = date_index - x
    for i in range(1, 2+2*x):
        row = [sum(residual[date_start: date_start + i]), positive_pct, negative_pct]
        car.append(row)

df = pd.DataFrame(car, columns=['CAR', 'positive_pct', 'negative_pct'])

Y = df['CAR']
X = df[['positive_pct', 'negative_pct']]
X = sm.add_constant(X)
reg = sm.OLS(Y, X.astype(float)).fit()
print reg.summary()
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                    CAR   R-squared:                       0.009
Model:                            OLS   Adj. R-squared:                  0.006
Method:                 Least Squares   F-statistic:                     2.568
Date:                Wed, 03 May 2017   Prob (F-statistic):             0.0777
Time:                        16:24:50   Log-Likelihood:                 1006.1
No. Observations:                 539   AIC:                            -2006.
Df Residuals:                     536   BIC:                            -1993.
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [95.0% Conf. Int.]
------------------------------------------------------------------------------
const         -0.0253      0.011     -2.360      0.019        -0.046    -0.004
positive_pct   0.3724      0.238      1.561      0.119        -0.096     0.841
negative_pct   0.5230      0.366      1.430      0.153        -0.196     1.242
==============================================================================
Omnibus:                      288.933   Durbin-Watson:                   0.556
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             3553.679
Skew:                           2.065   Prob(JB):                         0.00
Kurtosis:                      14.882   Cond. No.                         228.
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: