**EDGAR Project – Sentiment Analysis on Financial Statements**

In the US, the Securities and Exchange Commission (SEC) requires public companies to file 10-K report annually. The 10-K form gives detailed report on the financial position of the company and contains a summary of relevant topics such as business activities, risk factors and legal proceedings.

You work as a data engineer / data scientist at a fund. Your employer wants to determine whether negative sentiment 10-K filings can be used to predict short term movements in share prices. You will create a package called *edgar* which will contain all code needed for this task.

Specifically, for each 10-k filing, you want to count the number of words classified as ‘Negative’, ‘Positive’, ‘Uncertain’, etc, and determine whether this has an impact on short-term price movements.

For word classifications, we will use the Loughran-McDonald dictionary for word classifications designed specifically for financial documents. In this dictionary, words such as ‘adversity’, ‘controversial’ and ‘failing’ are negative, whereas words such as ‘achieving’, ‘beneficial’, and ‘succeed’ are positive.

The sentiments classifications in the dictionary include negative, positive, uncertainty, superfluous among others. The dictionary can be found here:  
<https://sraf.nd.edu/textual-analysis/resources/#Master%20Dictionary>

**Navigating the Website**

**Step 1. Search for a Company**

The SEC EDGAR company filings webpage can be found here:  
<https://www.sec.gov/edgar/searchedgar/companysearch.html>

To search for company filings, type in the company’s ticker symbol and click search. For example, to search for Apple’s filings, search for AAPL. This will take you to <https://www.sec.gov/cgi-bin/browse-edgar?CIK=320193>.

**Step 2. Filtering Search Results**

Step 1 will take you to the search results. To search for 10-K filings, type in “10-K” in the Filing Type filter box and click “Search”.

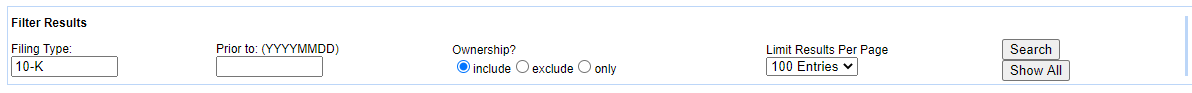


Figure . Filtering Search Results

This will then give you a table of 10-K filings for the company. To see the company filings for a given year, click the “Documents” button for the desired year. This will take you to the report’s artifact page.



Figure . Example of Search Results

**Step 3. Navigating the Artifacts Page**

When on the Artifacts page, we can click the 10-K artifact to get the 10-K report in html format.

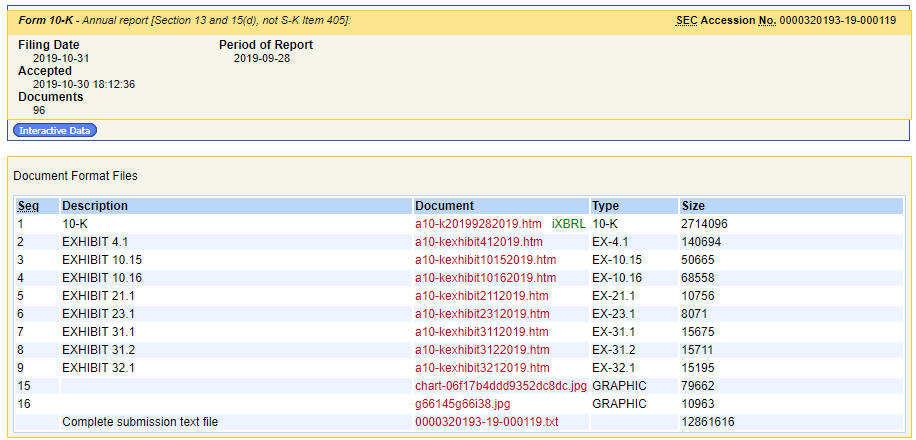


Figure . Example Artifact Page

**Step 4. Getting Clean HTML**

After clicking on the link in Step 3, the html rendered uses inline javascript to populate contents. To load the 10-K report with all data populated inline, click “Open as HTML” in the menu on the top left.

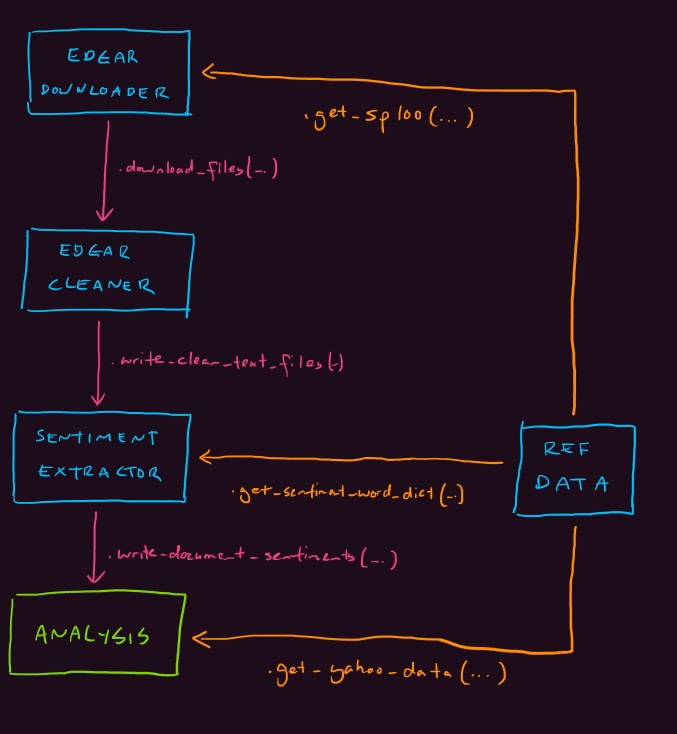


Congratulations - you now know how to navigate to a 10-K report for a given company and year!

**Pipeline Overview**

The high level components are as follows:

|  |  |
| --- | --- |
| **Component** | **Description** |
| EDGAR Downloader | Downloads raw 10-k filings from the SEC EDGAR website. |
| EDGAR Cleaner | Given raw 10-k html files, perform text cleaning. |
| Sentiment Extractor | Given a corpus of clean 10-k documents, count the number of negative sentiment words for the document. Repeat for words classified as positive, uncertain, litigious, etc. |
| Ref Data | Module used for retrieving data such as tickers in the S&P100, sentiment word lists and stock market data. |

****

**Part 1. Data Ingestion**

**Task Description**

Create a module called **edgar\_downloader**. The module should contain the functions:

* **write\_page(url, file\_path)**  
  Takes in the URL and writes the html file to the path specified.
* **download\_files\_10k(ticker, dest\_folder)**   
  Downloads all the html 10-k files for the given ticker into the destination folder. The files downloaded should the named using the following convention:  
  <ticker>\_10-k\_<filing\_date>.html

After you have unit tested your code, use your module to download filings for stocks in the S&P100.

**Running your code**

A user should be able to import your module and call the download\_files\_10k function to download files. For example, running the following will download all of Apple’s filings into my C:/10k\_filings\_raw directory:

import edgar\_downloader  
edgar\_downloader.download\_files\_10k(‘AAPL’, ‘C:/10k\_filings\_raw’)

Running the code above should download reports to the destination folder. Example below.

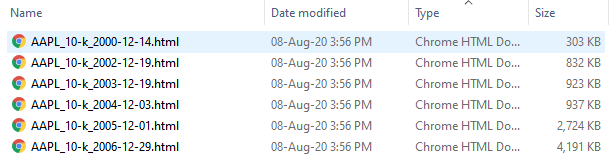


Figure . Example of Downloaded Files in Folder

**Part 2. Data Preparation / Cleaning**

**Task Description**

Create a module called **edgar\_cleaner**. The module should contain the following functions:

* **clean\_html\_text(html\_text)**  
  Takes in a html text string and removes tags and special characters. Returns result as a string.
* **write\_clean\_html\_text\_files(input\_folder, dest\_folder)**  
  Takes all the html 10-k files in the input folder, cleans them using the clean\_html\_text function and writes them into the destination folder as text files. Files downloaded should follow the naming convention: <ticker>\_10-k\_<filing\_date>.txt.

After unit testing your code, use your module to convert the html filings into clean text.

**Running your code**

Running the code below will read the previously written html 10-k filings and save them as text files with tags and special characters removed.

import edgar\_cleaner  
edgar\_downloader.download\_files\_10k(‘C:/10k\_filings\_raw’, ‘C:/10k\_filings\_clean’)

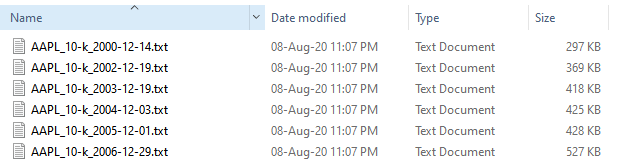


Figure . Example of Cleaned Files in Folder

**Part 3A. Reference Data: S&P100 Data**

**Task Description**

Create a module called **ref\_data**. The module should contain the following functions:

* **get\_sp100()**  
  Returns a list of all tickers in the S&P100. Note that this can be a snapshot of the current constituents of the S&P100 and does not need to be updated live.

**Running your code**

Running the code below will give a list of all tickers in the S&P100.

import ref\_data as edgar\_data  
tickers\_sp100 = edgar\_data. get\_sp100()  
print(tickers\_sp100)

**Part 3B. Reference Data: Yahoo Finance**

**Task Description**

Update the **ref\_data** module. The module should contain the following added function:

* **get\_yahoo\_data(start\_date, end\_date, tickers)**  
  Downloads yahoo finance data and consolidates all data into one table. In addition, returns should be calculated for 1, 2, 3, 5 and 10 business day time horizons. For example, the 1-day return is the return made if the stock was bought today and sold the next day. Returns a dataframe.

An example of the output and the required columns is shown below:

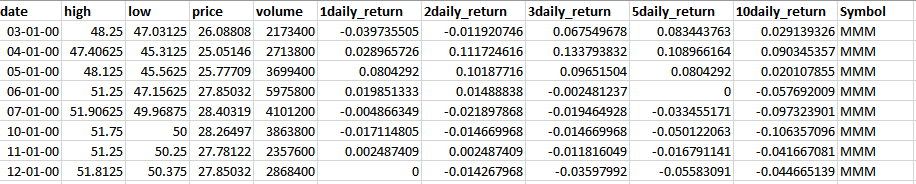


Figure . Snippet of the Yahoo Finance Output File

Use your module to create the yahoo finance reference data for stocks in the S&P100.

**Running your code**

Running the code below will write the yahoo finance reference data to file.

import ref\_data as edgar\_data  
df\_returns = edgar\_data. get\_yahoo\_data('2000-01-01', '2020-08-01', tickers, 'daily')  
df\_returns.to\_csv(‘C:/stock\_returns\_daily.csv, index=False)

**Part 3C. Reference Data: Loughran-McDonald Sentiment Words**

**Task Description**

Update the **ref\_data** module. The module should contain the following added function:

* **get\_sentiment\_word\_dict()**  
  Returns a dictionary containing the LM sentiment words. The keys for the dictionary are the sentiments, and the values will be a list of words associated with that particular sentiment. For example, get\_sentiment\_word\_dict()[‘Negative’] will return a list of words associated with negative sentiment.

**Running your code**

Running the code below will return the sentiment word dictionary.

import ref\_data as edgar\_data  
sentiment\_dict = edgar\_data. get\_sentiment\_word\_dict()  
print(sentiment\_dict)

**Part 4. Sentiment Word Counts**

**Task Description**

Create a module called **edgar\_sentiment\_wordcount**. The module should contain the following function:

* **write\_document\_sentiments(input\_folder, output\_file)**  
  Takes all the clean text 10-k files in the input folder, counts the number of words in the document belonging to a particular sentiment and outputs the results to the output file.

An example of the output and the required columns is shown below:

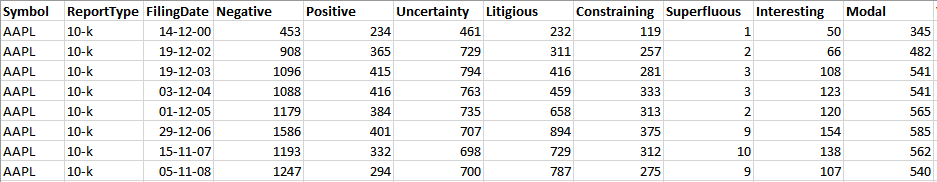


Figure . Snippet of the Sentiment Wordcount Output File

After unit testing your code, use your module to produce a sentiment word count summary for your clean 10-k filings.

**Running your code**

Running the code below will read the previously written clean text 10-k filings and write a table summarizing the sentiment word count for each document.

import edgar\_sentiment\_wordcount as edgar\_sentiment  
edgar\_sentiment.write\_document\_sentiments(‘C:/10k\_filings\_clean’, ‘C:/sentiment\_factors.csv)

**Part 4B. Sentiment Word Counts**

**Task Description**

Insert your results from part 4 into a SQL database so that it can easily be queried by others.

**Part 5. Sentiment Analysis**

**Task Summary**

Join the yahoo financials data (part 3B) with the sentiment word counts data (part 4) to determine whether there is a link between negative word counts and stock price returns.

This part has left intentionally open. Try plotting out the relationship between negative word counts and volume or returns. You can try to think of ways to improve your analysis and adjust previous steps where necessary. Be creative!