**Walkthrough of the code:**

Following is the flow of the code:

**Part 1: The Web Scraping code**

This code reads the ticker.txt file containing all companies in the EDGAR database that is provided by SEC for developers (https://www.sec.gov/about/webmaster-faq.htm#developers), then scrapes the ‘risk factor’ section of the filings made by these companies in the past 3 years in chronological order. However, this step might take around 11 hrs to scrape all the files from the EDGAR database.

The user can also provide a list of company ticker they want to analyze, for example, **‘aapl’** or a list of tickers [appl,amzn..] and the years [2017,2018]. The scraper will then scrape only for these tickers and years. (This will take less than a minute)

The final output is kept in a folder named ‘risk\_factors’ (the code creates this folder in case there is none) in form of text files for each combination of company and filing. The naming format of these text files is: "ticker\_filingDate\_10K/10Q"

Refer to the Web\_scraper\_README.ipynb for ReadMe file and Risk\_Factor\_Scraper.ipynb for full code.The outputs will look as follows:

Folder name:

Files in the folder (for aapl as input):

Individual File example:

Now the files are scraped from the EDGAR database and present in the ‘risk\_factors’ directory.

**Part 2: Code for Analysis of reports**

To analyze these reports, user can use any of the following three techniques:

- Similarity Detection - Sentiment Analysis - Plagiarism Detection

**Technique1: Similarity Detection**

This technique utilizes Natural Language Processing (NLP) to find the similarity in the reports. It takes the output of the web scraper (in the risk\_factors folder) as input and finds the similarity between the documents taking two at a time. These reports are arranged in chronological order and comparisons are made.

To find the similarity, the code uses the Spacy in-built English dictionary: 'en\_core\_web\_lg' and finally outputs a cosine similarity matrix that indicates the similarity between each of the sentences in the two documents. The values of cosine matrix vary from 0 to 1, with 0 being the least similar and 1 being most similar. The code gives three levels of changes:

Small change: Threshold 0.8 ~ 0.99 Moderate change: Threshold 0.5 - 0.8 Big Change: Threshold less than 0.5

These thresholds can be changed as per the requirements. Refer to the ‘Similarity, Similarity Visualization, Sentiment Visualization ReadMe.txt’ for ReadMe file and ‘SimilarityDetection\_SimilarityAndSentiment Visualization.ipynb’ for full code.The outputs will look as follows along with a cosine matrix.

-

**Technique2: Sentiment Analysis**

This technique utilizes Natural Language Processing (NLP) and the financial sentiment dictionary from University of Notre Dame 'LoughranMcDonald\_MasterDictionary\_2018.csv' to find the sentiment in the given reports. The sentiment is divided into 6 categories:

-Negative -Positive -Uncertainty -Litigious -Constraining -Interesting

It takes the output of the web scraper (in the risk\_factors folder) as input and plots the sentiment values for each of categories for the reports. Refer to the ‘SentimentCodeReadMe.txt’ for ReadMe file and ‘Sentiment analysis Apr 2020’ for full code. The outputs will look as follows

**Technique3: Plagiarism Detection**

This technique utilizes longest string comparison to find the similarity in the reports. The code compares documents word by word to detect plagiarism. The length of the common string can be changed, currently it is min 2 words and maximum 25.

It takes the output of the web scraper (in the risk\_factors folder) as input and outputs HTML files as visualization. It also prints the common expression found, its frequency of occurrence and the percentage of similarity between documents.

HTML files that show comparisons between two consecutive documents at a time. The HTML files are linked and can be moved 'Next' or 'Previous' to go to next time frame or previous one respectively

Refer to the ‘ReadMe\_Plagiarism\_Code.txt’ for ReadMe file and ‘PlagiarismCode’ for full code. The outputs will look as follows

Console Output:

HTML Output:

**Part 3: Visualization of Similarity and Sentiment Code**

This code produces HTML outputs for the Similarity and Sentiment techniques of the project. For these, the code takes input as the cosine matrix produced by the similarity code and the Notre Dame dictionary.

Refer to the ‘Similarity, Similarity Visualization, Sentiment Visualization ReadMe.txt’ for ReadMe file and ‘SimilarityDetection\_SimilarityAndSentiment Visualization.ipynb’ for full code. The outputs will look as follows

Similarity Detection Output:

Sentiment Analysis Output: