Replication of Loughran and MacDonald Analysis Report

Group 7 (Anastasia Chu, Shiyu Tang, Yuxuan Yang, Evelyn Li)

Github Link:
https://github.com/xl453/HW1_Replication-of-Loughran-and-MacDonald-Analysis
We also uploaded all Jupyter Notebooks containing our codes and our report to the Canvas–Group7–File Directory.

Step1: Data Downloading

For this data downloading part, we load and save the 10-K and 10-Q reports of S&P 500 firms from year 2011 to year 2021. The specific steps are listed below:

1) First we query the CIK data from WRDS CRSP. The CIK file would be used to match the provided S&P 500 companies list and to look up the corresponding 10-K and 10-Q reports. The CIK file should be further cleaned and processed. For this part, we delete 18 rows with null CIK value and drop the duplicates of company names, CIK and ticker combination.
2) Next, for the provided S&P 500 firms dataset, we assign a label indicating the year and quarter for every entry. Since S&P 500 rebalances its membership every quarter on the third Friday of the quarter end, it's reasonable for us to group all members from quarter start to quarter end. After that, we merge the CIK file and the S&P 500 firms dataset based on column LPERMNO to assign CIK for every S&P 500 company in every quarter from January 2011 to December 2021.
3) Finally, We create a dictionary of lists which contain all S&P 500 companies in a certain quarter of a certain year. Then we could use this dictionary to query the 10-K and 10-Q reports. Specifically, for the 10-K report, since a 10-K report is filed annually, every company will only have one 10-K file per year. Moreover, because the 10-K report in a certain year will be released in the next year, we are actually using the S&P 500 membership in 2011 to 2021 to find 10-K files from 2012 to 2022. For simplicity, we only use the Q4 list of companies in the S&P 500 per year to look up the corresponding 10-K reports.
4) The 10-Q report is filed quarterly, every company has three 10-Q files per year, with the fourth quarter converging with the annual filing. We use the list of companies in each quarter to download the corresponding 10-Q reports.
5) We download all available 10-K and 10-Q files of S&P 500 companies month-by-month from library secedgar. SEC-Edgar implements a basic crawler for downloading filings from the SEC Edgar database. We start with a certain month, and obtain all tickers of

companies which belong to S&P 500 in that month, and then we check whether a company would have available 10-K or 10-Q during that month. If it does, we save that file returned; if not, we skip and jump to the next ticker. Here is the reference to the documentation of this library: https://sec-edgar.github.io/sec-edgar/

Step2: Parse, Clean and Count Words

The goal of this step is to generate a list of words for each 10-K and 10-Q document. We need to clean, tokenize and lemmatize all words in the file. After importing the raw HTML files, we go through the following steps:

1) Remove all elements with <TABLE> tag in the xml/html documents.
2) Remove the <a …> tag that defines a hyperlink, which is used to link from one page to another.
3) Remove html/xml tags to convert files into usual strings.
4) Convert all letters to lowercase to clean up the document text.
5) Eliminate all punctuation and number characters.
6) Tokenize text and remove all non-English words.
7) Lemmatize all words in the file.
8) Remove stop words that don't provide significant value, such as "and", "the", and "a".
9) Remove single letter words with only one character.
10) Remove words with digits.

For the 10-K and 10-Q files over 2016 to 2021, we iteratively read each of them and populate into a defined function (which streamlines the parsing, cleaning and process of making word lists) to generate the word list. We require that each generated word list contains at least 500 words. The final output is a dictionary (for which each key is a ticker) of dictionaries (key is the filing date, value is the corresponding word list of the file with this ticker and this filing date). Considering the limited memory of RAM on Google Colab, we choose to run all files in stages and store the result dictionaries into a JSON file for future use.

Step3: Excess Return Calculation

Next, we need to calculate the 4-day holding-period excess returns relative to the filing date (if we denote the filing date as 0, the holding period should be days 0 through 3) for all the 10-K and 10-Q reports occurring in our dataset. Each 10-Q or 10-K report will have one excessive return value. For each report, this excessive return is calculated by the 4-day buy-and-hold return of the reported company's stock minus the 4-day buy-and-hold return of the CRSP value-weighted market index.

In our project, The daily returns of individual stocks from January 2021 to December 2021 are downloaded from the table crsp_a_stock.dsf from the WRDS dataset. We query 11-year daily

returns for all the tickers corresponding to at least one 10-Q or 10-K report all at once as our stock daily return dataset. Similarly, the daily returns of CRSP value-weighted market index are queried from the crsp_a_stock.wrds_dailyindexret_query table of the WRDS dataset. The 11-year index daily returns are downloaded all at once and saved as our index return dataset.
We take three main steps to obtain the 4-day buy-and-hold excess returns.

1) We have two datasets, one is stock daily returns, and one is index daily return. There are columns date, ticker and daily return. In this step, we will not filter out those days which match the 10-K or 10-Q filing dates (we will do it later). So, in this step, we will calculate the 4-day buy-and-hold returns for all tickers at all dates from January 2011 to December 2021.
2) For each pair of stock and date in the stock daily return dataset, we compute the 4-day buy-and-hold returns with the formula below:
$$StockReturn_{buy-and-hold} = (1 + StockReturn_{day1})(1 + StockReturn_{day2})$$
$$(1 + StockReturn_{day3})(1 + StockReturn_{day4})\text{-}1$$
3) For each date in the index return dataset, we compute the 4-day buy-and-hold returns with the formula
$$IndexReturn_{buy-and-hold} = (1 + IndexReturn_{day1})(1 + IndexReturn_{day2})$$
$$(1 + IndexReturn_{day3})(1 + IndexReturn_{day4})\text{-}1$$
4) The 4-day buy-and-hold returns of individual stocks and 4-day buy-and-hold returns of index are merged together by date. We subtract the index's buy-and hold returns from the individual stock's buy-and-hold returns to obtain the excess returns. The file of excess returns is saved as the excess return dataset.

Step 4: Negative Word Proportion Calculation Using Bag-Of-Words Method

In Step 4, we adopt the "Bag-of-Words Method" (i.e. "Simple Proportion Method") to calculate the negative word proportion. The way we define "simple proportion" is according to what the authors mention in the paper. Suppose we have $document_j$ *and* $word_{i,j}$, which represent a certain document we have and a word that exists in $document_j$. In $document_j$, we have M words. We define a indicator function:

$$I_{i,j} = \begin{cases} 1, if\ word_{i,j}\ is\ a\ negative\ word\ in\ the\ dictionary \\ 0, otherwise \end{cases}$$

Then our simple proportion for a document can be calculated by:

$$negprop_{i,j} = \frac{\Sigma_{i=0}^{M} I_{i,j}\ word_{i,j}}{M}$$

The detailed approach is as follows:

1) We download and read in the Harvard IV Negative Word List Dictionary and the Fin-Neg Dictionary produced by the author of the paper, and clean and prepare them for future use
    - For Harvard IV Negative Word List Dictionary, we turn the .txt file into a list of words and convert all words to lowercase. The length of the Harvard dictionary is 4188.
    - For the Fin-Neg Dictionary, we remove all unnecessary columns (only keep columns Word and Negative), select all words with non-zero entries, check the year when a certain word is added to or removed from the dictionary, turn all relevant negative words into a list, and convert all words to lowercase.
2) Read in all the JSON files which contain the dictionaries of the word list of 10-K and 10Q for different tickers and filing dates from 2011-2021.
3) For both Harvard IV Negative Word List Dictionary and Fin-Neg Dictionary, we count the total occurrences of each negative word in each 10-K or 10-Q file. This procedure can be completed by the sklearn.feature_extraction.text.CountVectorizer module for Python. The final output is two matrices, one for Harvard IV Negative Word List Dictionary and one for Fin-Neg Dictionary. The number of rows of each matrix is the total number of 10-K and 10-Q files and the number of columns is the total number of negative words.
4) We sum up the matrix elements in each row to obtain the total counts of negative words in the corresponding file. The total negative word occurrences in each file are then divided by the total word number of the file to get the negative word proportion. The negative word proportion values for all 10-K and 10-Q reports are saved in a dictionary (with key as ticker) of dictionaries (with key as filing date) and thus also saved in a JSON file for future use.

Step 5: Negative Word Proportion Calculation Using TF-IDF Method

In Step 5, we adopt the "TF-IDF Method" to calculate the negative word proportion. The way we define "negative word proportion" here is according to what the authors mention in the paper. Suppose we have $document_j$ *and* $word_{i,j}$, which represent a certain document we have and a word that exists in $document_j$. In $document_j$, we have M words. Also, we have the TF-IDF $weight_{i,j}$ calculated by the below formula.

$$w_{i,j} = \begin{cases} TF_{i,j} * IDF_{i,j}, \text{ if } word_{i,j} \text{ is a negative word in the dictionary} \\ 0, \text{ otherwise} \end{cases}$$

where

$$TF_{i,j} = \frac{count\ of\ word_{i,j}}{number\ of\ words\ in\ document\ j}$$

$$IDF_{i,j} = log\left(\frac{1 + number\ of\ documents}{1 + document\ frequency\ of\ word_{i,j}}\right)$$

Then our TF-IDF proportion for a document can be calculated by:

$$negprop_{i,j} = \frac{\Sigma_{i=0}^{M} w_{i,j}\ word_{i,j}}{M}$$

The detailed approach is as follows:

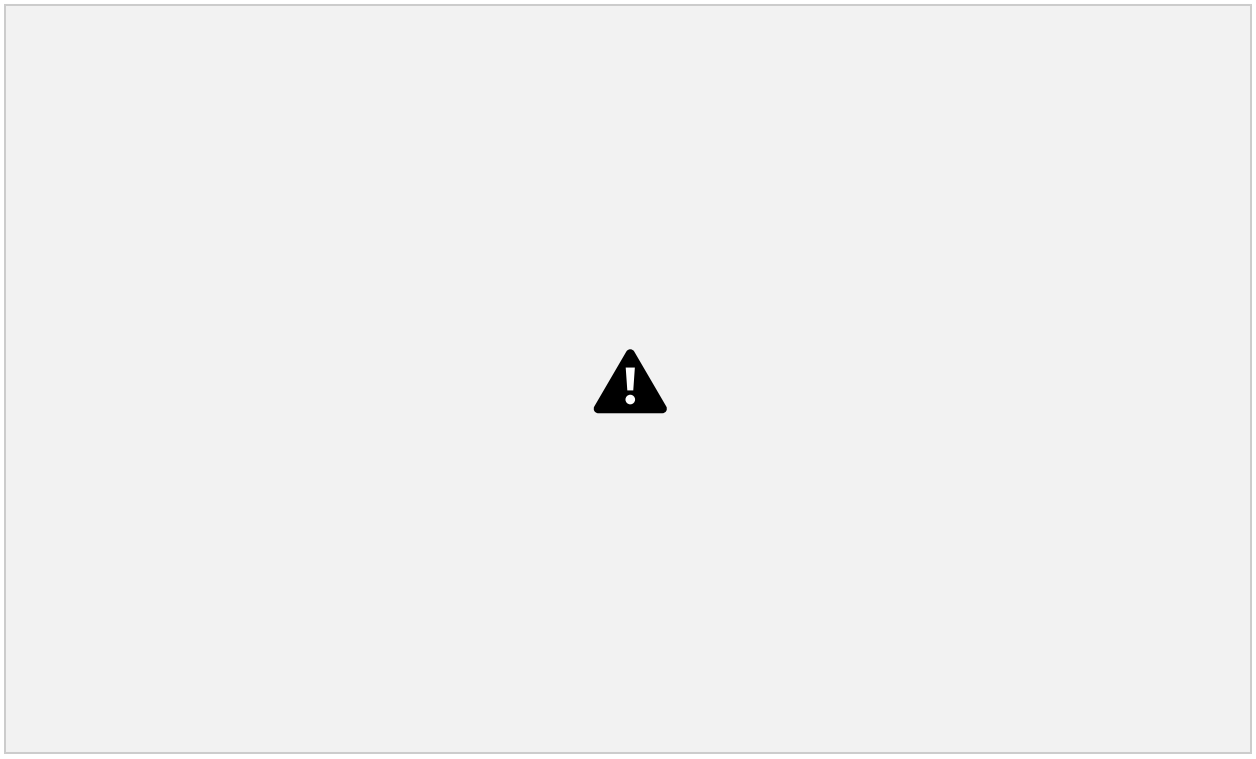
1) We download and read in the Harvard IV Negative Word List Dictionary and the Fin-Neg Dictionary produced by the author of the paper, clean and prepare them.
    - For Harvard IV Negative Word List Dictionary, we turn it into a list of words and convert all words to lowercase. The length of the Harvard dictionary is 4188.
    - For the Fin-Neg Dictionary, we remove all unnecessary columns (only keep columns Word and Negative), select all words with non-zero entries, turn all relevant negative words into a list, and convert all words to lowercase.
2) Read in all the JSON files which contain the dictionaries of the word list of 10-K and 10Q for different tickers and filing dates from 2011-2021.
3) For both Harvard IV Negative Word List Dictionary and Fin-Neg Dictionary, we count the total occurrences of each negative word in each 10-K or 10-Q file. This procedure can be completed by the sklearn.feature_extraction.text.CountVectorizer module for Python. The final output is two matrices, one for the Harvard IV Negative Word List Dictionary and the Fin-Neg Dictionary. The number of rows of each matrix is the total number of 10-K and 10-Q files and the number of columns is the total number of negative words in the two negative word dictionaries respectively.
4) For both Harvard IV Negative Word List Dictionary and Fin-Neg Dictionary, we compute the TF-IDF weight ($w_{i,j}$) of each word in each 10-K or 10-Q file. This procedure can be completed by the sklearn.feature_extraction.text.TfidfVectorizer module in Python. We would get two matrices as the final output. The number of rows of each matrix is the total number of 10-K and 10-Q files and the number of columns is the total number of negative words in the two negative word dictionaries respectively. These matrices provide us the $w_{i,j}$ values corresponding to the negative word occurrences obtained from 3).
5) We multiply the $w_{i,j}$ values from 4) and the corresponding single negative word occurrences from 3) to get two matrices of weighted negative word occurrences for Negative Word List Dictionary and Fin-Neg Dictionary respectively.
6) For both of the matrices of weighted negative word occurrences, we sum up the matrix elements in each row to obtain the weighted total counts of negative words in each 10-K or 10-Q file. The weighted total counts of negative words in each file are then divided by the total word number of the file to get the negative word proportion under TF-IDF method. The negative word proportion values for all 10-K and 10-Q reports are saved as dataframe / .csv file as our TF-IDF negative word proportion dataset.

Step 6. Figure Plot

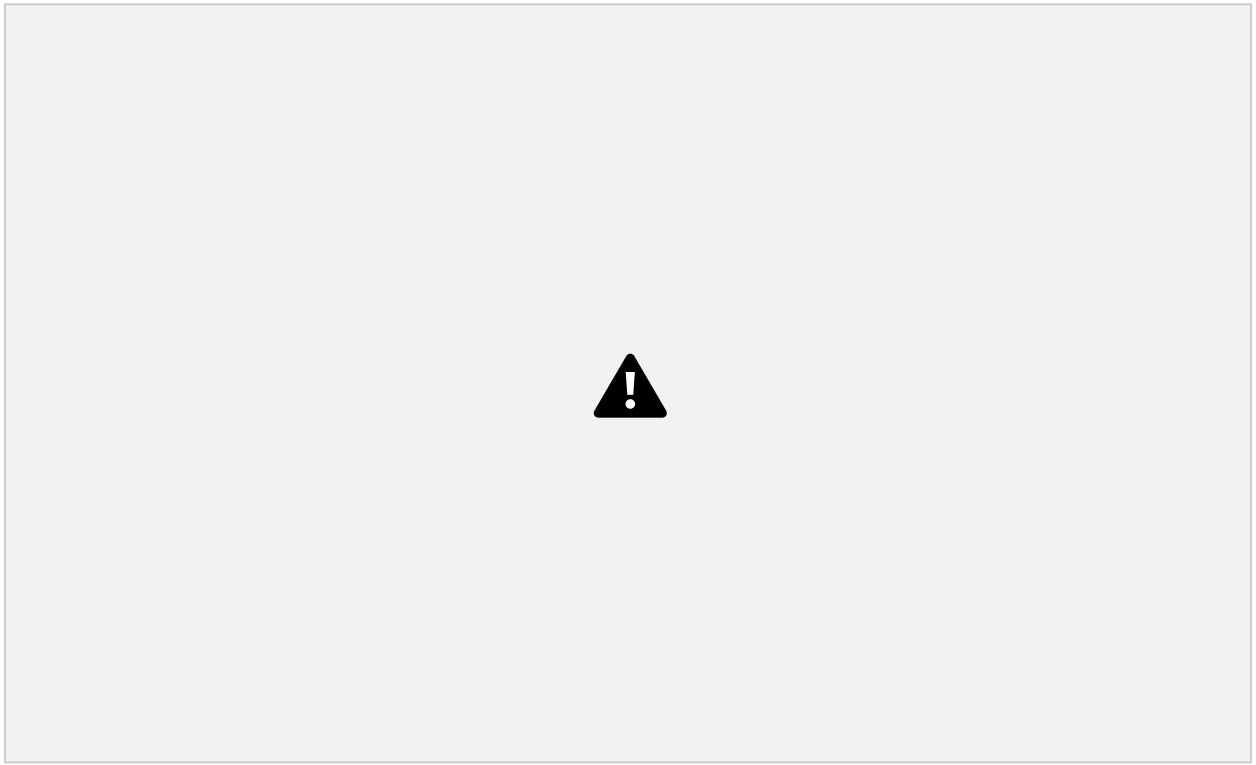
The final step is to plot the medium filing period excess returns by quintiles according to H4N-Inf and Fin-Neg word lists. The detailed approach is as follows:

1) Read in JSON/csv files with negative word proportions calculated for each document from both Step 4 (using Bag-Of-Word Method) and Step 5 (using TF-IDF Method)
2) Load in 4-day Buy-and-Hold Stock Excess Return Data from Step 4
3) We merge the 4-day buy-and-hold excess return dataset, simple negative word proportion dataset and TF-IDF negative word proportion dataset together by tickers and filing dates and using inner merge. In this way, only those returns with tickers and filing dates of 10-K and 10-Q documents will remain in this dataframe. Make the result as a dataframe.
4) For each negative word proportion calculation method, we now have dataframes for both the H4N-Inf and Fin-Neg word lists containing all information, including ticker, filing date, negative word proportion, and 4-day buy-and-hold excess stock return.
5) For each of the word lists (H4N-Inf and Fin-Neg), we split all negative word proportions into 5 portfolios according to the sorted level of negative word proportions.
6) Use groupby to calculate the median filing period excess returns for these 5 portfolios.
7) Replicate figure 1 using our data is shown below:

(1) For negative word proportion calculated by Bag-Of-Word Method:



(2) For negative word proportion calculated by TF-IDF Method:



Result Analysis: Comparing and Contrasting

First, we can see from our final two plots that

1) For the two lines with the Fin-Neg file, the excessive return is more negatively correlated with the negative word proportion calculated by the TF-IDF method. However, for the two lines with the H4N-Inf file, the excessive return is more negatively correlated with the negative word proportion calculated by the Bag-of-Word method.
2) The TF-IDF method more clearly differentiate the lines obtained with H4N-Inf file and Fin-Neg file respectively.

Since the word list in the H4N-Inf file is developed according to psychology and sociology while Fin-Neg word list is created based on negative financial implications, the line derived from the Fin-Neg word list should give a more obvious negatively correlated relationship between the excessive return and the negative word proportion quintile. From this perspective, the TF-IDF method provides us with more reasonable negative word proportion values.

Next, we compare our plots with the original paper's plot. We can see from the paper that the line with Fin-Neg word list shows a roughly linear negative correlation between the excessive returns and the negative word proportion quintile, while the line with H4N-Inf word list seems to show a relationship without a very clear pattern between the excessive return and the negative word proportion quintile. The possible reasons for the difference between the paper's result and out result might be due to the following reasons:

1) The paper was published in 2011 and the author uses the data during 1994-2008, while we use the data during 2011-2021. Therefore,
    - The author uses an earlier version of the H4N-Inf and Fin-Neg list.
    - We use a different sample of files than the author did because the time frame is different.
    - The recent 10-K and 10-Q reports might be written in a more neutral and objective tone, so it is harder to detect the relationship between the excessive return and negative word proportion with a similar method.
    - An increasing number of investors/traders have noticed the correlation between the excessive return and the negative word proportion ever since the paper has been published and thus the excessive return has been diluted.
2) The author uses the 10-K reports from all companies listed on NYSE, Amex and NASDAQ. These companies are then filtered by certain restrictions mentioned in the paper, for example, the companies should have at least 60 trading days in the year and the report should have more than 2000 words. However, we use both the 10-K and 10-Q

reports from the companies listed on S&P 500. The different choices of the report sample set may lead to the difference in the final plot.

3) The parsing process we use is slightly different from the author's in detail. We adopt several steps which the author does not use, for example, we delete all the hyperlink from the reports, lemmatize the reports, and remove the stop words. Moreover, the author deletes all the SEC headers from the reports, but we could not accurately reproduce this step for some files due to the report format or other potential reasons.