Sentiment Analysis with 10-K Filings

Ryan

April 12th, 2021

1 Introduction

For this project, we performed sentiment analysis on 10-k filings using the various methods we learned about in this course. We decided on this project because we both have an interest in the finance world. Sentiment analysis on financial filings is something that people have been researching in the past few years. All techniques used in this analysis were topics covered in this course.

In the first part of this project, we created our data-set of 10-k filings and organized them by company and year. In the second part, we pre-processed our documents using standard NLP techniques to make them easy to process. We continue to analyze various sentiments over the years per company by calculating Jaccard and Cosine similarities. Finally, we created our own method to evaluate and measure the occurrences of sentiment words in a document and used this value to compare with the company's stock value.

2 Data Collection

2.1 SEC EDGAR API

To compose a corpus of 10-K filings, we needed a publicly available source to provides these documents. The US Securities Exchange Commission offers access to their Electronic Data Gathering, Analysis, and Retrieval system [1] in the form and an HTTP API. This API will be referred to as the SEC EDGAR API. This service offers many different types of financial filings such as 10-K filings and 10-Q filings for all publicly traded companies.

From previous experience with data analysis projects, we naively assumed that we would need a large data set for this sentiment analysis. Our initial target was to collect between 5,000 and 10,000 documents from at least 1000 different publicly traded companies for a minimum history of five years. Later on, we realized that we didn't need that many filings, but we did require documents from a longer historical period than five years. The target for our second data set included filings for the past twenty years for twelve companies.

2.2 Document URLs

The SEC EDGAR API offers a search endpoint that receives an internal UUID for a company, a date range and type of document to query. The search endpoint returns a list of filings containing the filing date and URLs to the HTML and raw text versions of the documents. Using this search endpoint, we initially queried plain text URLs for 1228 different companies for a five year range, resulting in a total of 6835 plain text URLs. We recorded the company symbols, filing dates and URLs in a CSV file to scrape later. The second set of document URLs contained 181 URLs for twelve companies over the past twenty years.

2.3 Document Scraping

With the CSV files containing the document URLs, we began scraping the plain text files containing the HTML source code for the 10-K filings. Using the BeautifulSoup Python3 package and an HTML parser we retrieved the important pieces of text from the HTML content. After setting the text to lower case, we wrote the cleaned documents for each company to a directory labeled after their symbol.

3 Data Pre-processing

3.1 10-k Forms

We did some pre-processing before saving the downloaded 10-k forms as mentioned in the previous section, but there was still more to do before we would be able to run our similarity comparisons. We started by reading in each file, replacing all characters that were not a letter, a space, or a period with a space, and then replacing all stretches of whitespace with a single space. After wasting much time re-running it on our 2.9GB of filings, we decided to also save this intermediate set to a text file.

In preparation for the bag-of-words and TFIDFs, we needed to digest the documents down to more granular pieces of text. We then partitioned the documents into individual sentences and further tokenized the sentences into sequences of words.

After tokenization, we compared every word to the NLTK stop-words list. The words that are not in the list of stop-words are then lemmatized using the WordNetLemmatizer, and the document is returned. At this point, we saved the fully-processed documents so that we could simply read them in without

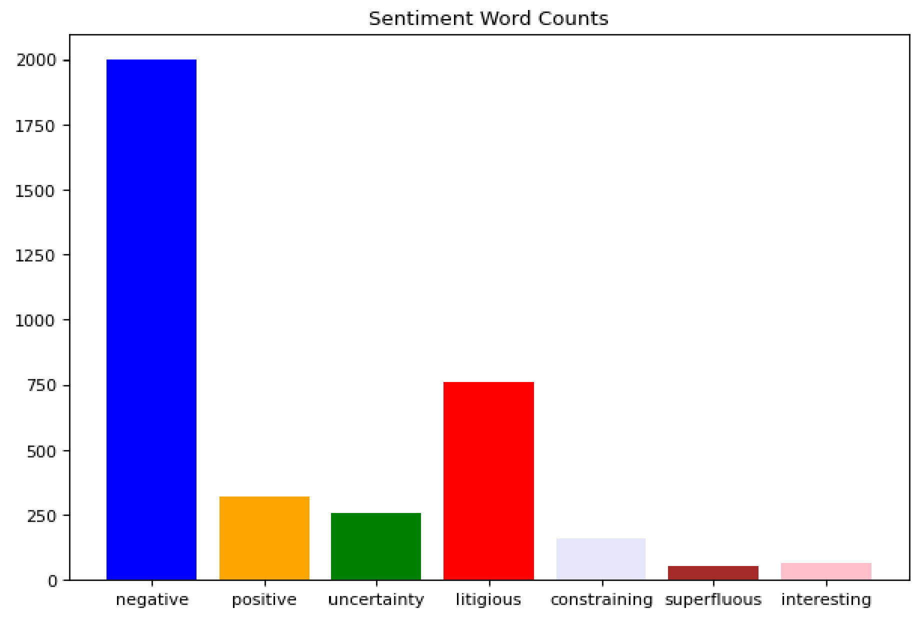


Figure 1: Sentiment Word Counts

re-processing them every time we run the program.

3.2 Loughran-McDonald Sentiment Word List

The Loughran-McDonald sentiment word list was created by Tim Loughran and Bill McDonald at the University of Notre Dame in 2011 [3]. This word list was created specially for financial documents and contains 86,487 unique words, some tagged with an associated sentiment word while others are left untagged. The seven sentiment word categories are: negative, positive, uncertainty, litigious, constraining, superfluous and interesting. Given that this word list was designed for financial sentiment analysis, we decided it would be a good choice for us to use.

The raw word list required some pre-processing before it was usable for our analysis. We lemmatized the words in the word list to better match the lemmatized words in our data. We removed any words from the word list that did not have an associated sentiment word.

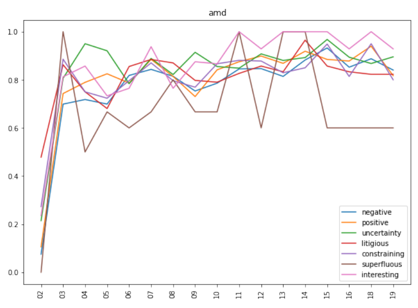
4 Document Similarities

4.1 Jaccard

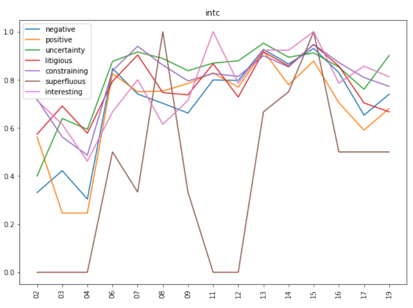
The first similarity index we used is the Jaccard similarity index. This index calculates the ratio between the size of the intersection of the two sets and the size of the union of the sets. In our case, the sets are the bag of words interpreted as a boolean array.

In order to create our Jaccard Similarity scores for consecutive years, we started by creating a sentiment bag of words for each sentiment using sklearn's `CountVectorizer`. For each company we want to evaluate, we fit the vectorizer to all its 10-k forms, and then we transformed each from individually so that we could sort them into a dictionary with the year as key. Each bag of words is also in a dictionary with their respective sentiment as the key.

Once the bag of words are generated for a company, we then calculate the Jaccard score between the years that we have in order to compare the changes in sentiments throughout the years.



(a) AMD Jaccard Similarities



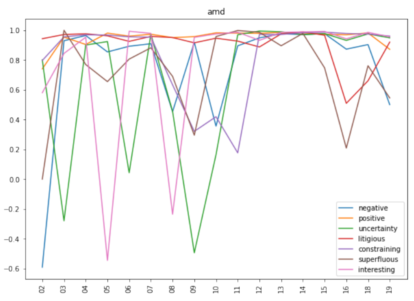
(b) Intel Jaccard Similarities

Figure 2: Intel and AMD Jaccard Similarities

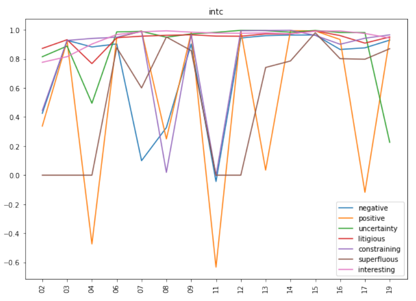
4.2 Cosine

The cosine similarity index is another similarity index used to compare two vectors. The results are from -1 to 1, with -1 being completely different, and 1 being exactly the same.

The process to create the cosine similarities is almost exactly the same as for the Jaccard similarities, except instead of creating a bag of words, we generated TFIDF values using `TFIDFVectorizer`, which we then used as input for the cosine similarity function.



(a) AMD Cosine Similarities



(b) Intel Cosine Similarities

Figure 3: Intel and AMD Cosine Similarities

5 Sentiment Scores and Company Performance

5.1 Sentiment Score Equation

To be able to compare the sentiment words with the price of a company's stock, a method is needed to evaluate the occurrences of sentiment words for a given year. For this evaluation metric, the seven sentiment word categories from the Loughran-McDonald word list are reduced to positive and negative categories; where positive, superfluous and interesting are interpreted as positive, while negative, uncertainty, litigious and constraining are interpreted as negative.

A key-value mapping of years and 0 values is created for a company. Using the calculated sentiment bag of words from the previous section, we take the mean of the bag of words for each of the seven sentiment words. The mean values for positive words are added to the value of the key-value pair for a year while the mean values for negative words are subtracted from value of the key-value pair for a year. Each value in the key-value mapping is piped through the hyperbolic tangent function to squeeze the sentiment score values between -1 and 1. Using the sentiment score function described, values close to 1 represent documents for a year that contain mostly positive sentiment words, while values close to -1 represent documents with mostly negative sentiment words. For the programmatic implementation, see the `sentiment_scores` function in the analysis.py file in our Git Lab repository.

5.2 Sentiment Score Comparisons

Figure 4 shows the sentiment score in comparison with AMD's stock price. The values for years 2002 to 2009 appear to be independent. The sentiment score values and stock price values for years 2010 to 2021 have the same trend but are an inverse of each other. This trend seems to be counter intuitive given that when AMD's stock price is low, the sentiment values are very positive, but

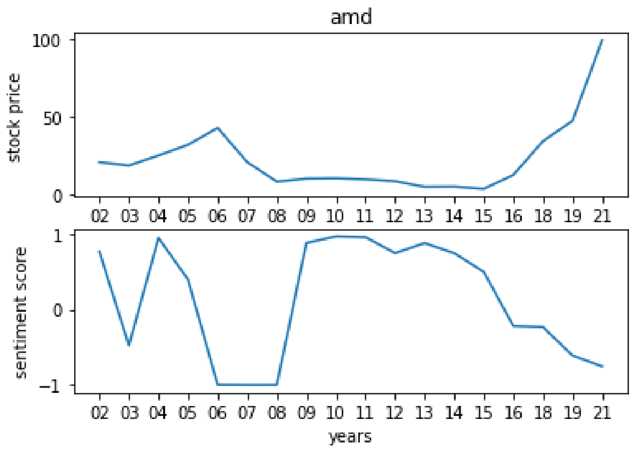


Figure 4: Sentiment Score and Stock Price for AMD

when AMD's stock price starts to grow, the sentiment words start to become very negative.

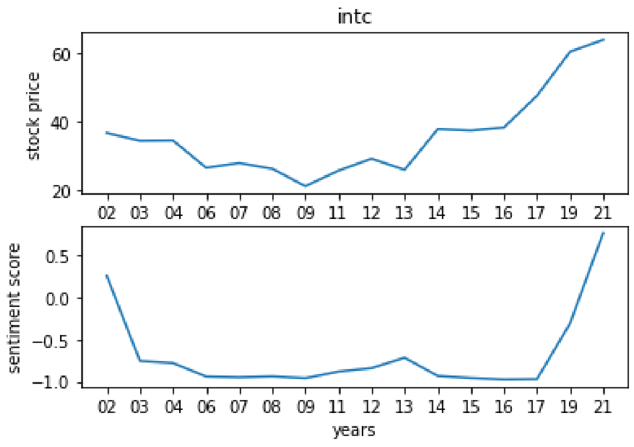


Figure 5: Sentiment Score and Stock Price for Intel

In Figure 5, the sentiment score and stock price values follow a similar trend. From the year 2016 and on, the price of Intel's stock increase while the amount of positive sentiment words in their 10-K filings also increases. The sentiment score and stock price appear to be positively correlated in this example.

Figure 6 shows that the sentiment values and stock prices for Microsoft are not correlated. There is no observable pattern between the values for this example.

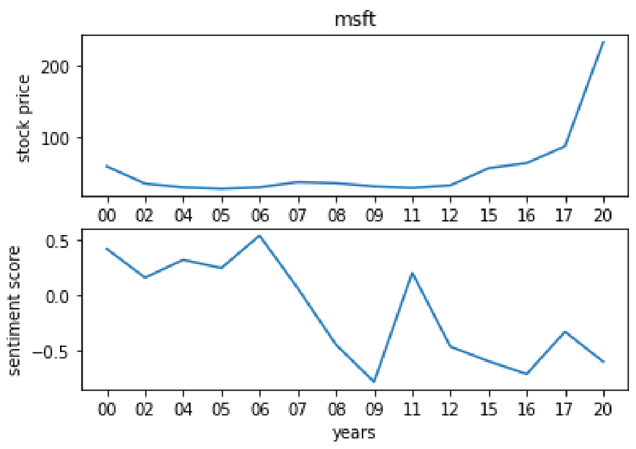


Figure 6: Sentiment Score and Stock Price for Microsoft

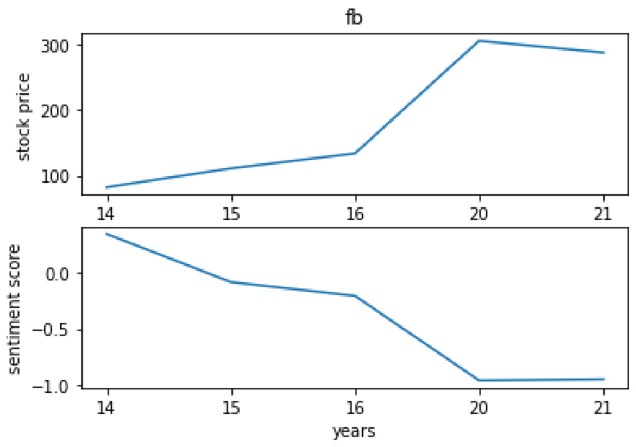


Figure 7: Sentiment Score and Stock Price for Facebook

Both the sentiment values and stock price values in Figure 7 are flipped. Similar to the last half of the year range in Figure 4, when the company's stock price increases, the amount of negative sentiment words increases. This trend in unexpected and seems counter-intuitive.

6 Discussion

Our initial hypothesis was that filings with more positive sentiment than negative sentiment would correlate to an increase in sock price over the years. We were unable to find such a correlation in our analysis. There are three possible explanations for why we did not see the expected results. The first and most probable is that there might not be a correlation between the companies we analyzed and their stock performance. Second, the class imbalance between the sentiment words in the Loughran-McDonald word list is heavily skewed with negative words. This makes the sentiment metrics unevenly distributed as negative. Finally, there could be a bug in our program that invalidates our analysis.

7 Future Work

7.1 Other Sentiment Metrics

In the sentiment value and stock price comparison section, we created our own method for evaluating the sentiment score. In future work we would want to use known method for evaluation the sentiment score. A potential option would be to use the VADER (Valence Aware Dictionary for Sentiment Reasoning) method which assigns a positive, negative and neutral value for each document. In conjunction with our own method, we could compare and contrast the results.

Another potential option would be the TextBlob Python3 package which offers a sentiment analysis function that calculates to polarity and subjectivity of the sentiments. The polarity is the contrast between positive and negative sentiments where the value is between -1 and 1. The subjectivity is a value between 0 and 1, measuring if the content is factual or subjective.

7.2 Machine Learning

An interesting extension of this analysis would be to build a machine learning model that predicts the sentiment category for a given sentence. The simple implementation would predict binary classes, positive or negative. The advanced implementation would predict one of seven sentiment word categories from the Loughran-McDonald word list.

8 Links

1. Git Lab Repo
2. Data Storage

9 References

1. About EDGAR. (2020, January 23). Retrieved April 12, 2021, from https://www.sec.gov/edgar/about

2. SEC EDGAR filings api. (n.d.). Retrieved April 13, 2021, from https://sec-api.io/docs

3. Marketing Communications: Web // University of Notre Dame. (n.d.). Resources // software repository for accounting and Finance // University of Notre Dame. Retrieved April 12, 2021, from https://sraf.nd.edu/textual-analysis/resources/

4. Medium: NLP in the Stock Market. (n.d.). Retrieved April 13, 2021, from https://towardsdatascience.com/nlp-in-the-stock-market-8760d062eb92

5. Udacity. (n.d.). Udacity/artificial-intelligence-for-trading. Retrieved April 13, 2021, from https://github.com/udacity/artificial-intelligence-for-trading /tree/master/project/project_5