# Market Sensor

Using Google News to Predict Market Behaviors

Ziyuan Zhu
Computer Science
Lehigh University
zachzhu2016@gmail.com

## 1. Introduction

This project aimed to implement a reusable pipeline for news & stock data and achieve high accuracy in predicting stock price movements with the results of sentiment analysis on related Google News articles. The baseline used to evaluate the prediction accuracies was random coin flipping.

The most noting achievement of this project primarily encompassed the analyses and implementations on data collection & processing, the final results achieved might be less appealing. The pipeline designed for collecting a customized Google News & Yahoo Finance dataset could be largely reused and adopted in production environment for implementing further analyses.

This paper discusses about the project by aiming to provide answers to the following questions:

1. How to design a pipeline to create a customized news & stock dataset for testing the predictions made by a set of beliefs and price prediction model?
2. What are the difficulties of sentiment analysis on News Articles?
3. How can the model and results be improved?

## 2. Dataset

| Industry | Stock Symbols |
|---|---|
| Semi-conductor | QCOM, ASML, INTC, NVDA |
| Automobile | TM, WKHS, GM, TSLA, FCAU, CVX |
| Oil | BP, XOM, TOT |
| Airline | AAL, LUV, DAL, UAL, JBLU |

**Figure 1: stock selection. All of the stocks selected went public before 01/01/2015.**

There were very few existing datasets for news sentiment analysis and stock prediction due to the copy right issue of sharing news article content in public. The most well-known one was on Kaggle, using Reddit news to predict the price movement of DJIA [5]. However, due to the fact that this dataset only contained one stock information, and that Market Sensor was intended to establish a model that generalizes beyond a single stock, putting together a customized dataset was needed.

The customized dataset used to test Market Sensor was made of 2-year worth of stock and Google News data (from 01/01/2015 to 01/01/2017) related to eighteen different stocks across four industries - airline, automobile, oil, and semiconductor. There were 37527 Google News articles in total for analysis, 32138 of them had text content, while the rest only contained title and source information. Stock information and prices were acquired from Yahoo Finance through an open-source library [3]. The Google News articles were crawled and downloaded separately through another open-source library [2] which did not include article texts, so another open-source library [4] was used to acquire article texts. This dataset was believed to be a good representation given that it contained multiple stocks ranging from different industries as well as a good and even number of news articles associated with each. Moreover, the articles published during a particular date range were only crawled if there was stock data available during the same date range, thus avoiding crawling articles that would never be factored. For example, if a stock began to sell shares on 01/01/2016, all the articles that were related to it and published before 2016 were skipped.

Nonetheless, putting together a customized dataset from scratch has its own challenges. The first obstacle was associated with crawling, Google News detects and bans crawlers by inspecting HTTP header fields and the frequency of requests from a certain IP address. [2] To offload this burden, a paid-service was used [6], otherwise, it would have taken weeks or even months to get 2-year worth of Google News articles. The next roadblock was getting the articles. To prevent this process from being accidentally terminated by exceptions or internet instability, a caching module was implemented to checkpoint and save articles as the downloading progressed. This module continuously saved crawled articles and their content to disk as pickle blobs, which could be read back into the program at any time. In the case of a failure or interruption, the module allowed the downloading process to continue from the last downloaded date. (Note that this was applied to all the computationally expensive steps in Market Sensor.) Running natural language processing algorithms on all of the article texts to derive their sentiment values was another time-consuming step, but with the help of the caching module and Spark, it only had to be done once and was able to run a few degrees faster, easing the

testing of different sentiment analysis methods. As a result, this module had saved tremendous amount of time on repetitive tasks for downloading and preprocessing.

After downloading, both article and stock data needed to be aligned to a single timeline so that the sentiment values of articles can be attributed to the right stock prices. This timeline object was persisted as a Pandas DataFrame and it would be used for analysis.

## 3. Pipeline

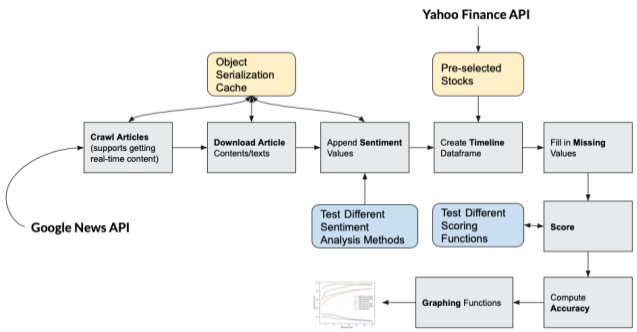


**Figure 2: Pipeline diagram.**

Most preprocessing steps annotated in figure 2 had already been explained in the section two. More details are given to the second half of the pipeline. The pipeline essentially:

1. Gathers data from both Google News API and Yahoo Finance API. (cached)
2. Downloads article texts. (cached)
3. Adds sentiment value to downloaded articles by using the average of the sentiments of the sentences in article texts, sentences with less than 5 tokens / words were considered invalid and were skipped because sampling had shown that articles tend to have irrelevant data (e.g., phone number). (cached)
4. Fits the augmented article data into a DataFrame indexed by article URL's. Stock data already sits in a DataFrame indexed by dates.
5. Creates a timeline DataFrame based on the stock data, whose rows are indexed by dates, and append the URL's of the articles published in a new column.
6. Computes price change percentage using stock's opening and closing price.
7. Iterates through the timeline DataFrame and replaces empty sentiment values with sentiment values from the most recent day of the previous three days if that day had a non-empty sentiment value.
8. Applies a scoring function and derives an overall score of an article based on their sentiment values.
9. Computes accuracies by comparing price change percentage with article scores. Predictions have three values, negative (sell), hold (no action), and positive (buy), and each of which's accuracy is computed.
10. Lastly, graphs accuracies and covariances with stock groups and different window length (the number of days of Google News data, in the past, used to predict the stock price of the current day).

## 4. Sentiment Analysis

Stanford Stanza NLP library was used to conduct sentiment analysis on the crawled articles. Stanza's tokenize processor was used to parse article texts into sentences and its sentiment processor was used to attach a sentiment value (0 = negative, 1 = neutral, 2 = positive). [7] To improve readability, these sentiment values are normalized to the scale of 100 by multiplying 50, where each value equates to 0, 50, and 100 respectively. Both of the title and text sentiment values were calculated by taking the average of all of their parsed sentences. For example, an article text with three sentences with sentiment values: [100, 50, 100, 50] has text sentiment value 75, same applies for titles. The advantage of using a pre-trained sentiment evaluation model was that the predictions could be made in an unsupervised manner solely based on sentiment values but less tailored to a particular dataset. This method was used for its simplicity.

It's worth noting that the articles might not have directly related to a specific stock, they were purely the search results from Google News with a company name and date range being the search query. Therefore, it was possible that a positive-sentiment article about a stock may not be completely related to it, and vice versa. It would also be possible that an article's overall sentiment might have been dominated by a misleading sentiment in one section of an article.

Based on a random sampling of articles experiment to verify the output sentiment values, the average of the sentiment values of all the sentences in an article and title accurately reflected the overall sentiment of an article and the exceptions described in the previous paragraph tended to have less effect.

There were a few ways of filling in the text sentiment value for an article without text. One way was giving a neutral value so that it would only vary the degree of confidence in a prediction without flipping the result, it would not be possible to increase or decrease the average to above or below 50 by adding another 50. Alternatively, under the assumption that a sentiment value may persist in a short-term, the missing value could also be replaced with the text sentiment value of the article from the most recent day of the past three days if that day had a non-empty sentiment value.

## 5. Scoring (out of 100)

In the scope of this project, the prediction of a stock on a particular day was made solely based on the sentiment score derived in section 4, the average sentiment value of article title and that of article text. Two predictions were made separately based on article title and text so that the effectiveness of each could be further studied. However, the price movement on a day may be explained not only by the news sentiment on that particular day but also that of the previous days. Hence, a set of news sentiment scores from different days in the past were factored into the making of predictions.

To select a set of news sentiments in a date range with a certain window length, either a **simple average** or **moving average** could be used. For example, if the model were to predict the price of a stock on 10/15/2015 and the moving-average method with **window length** of 7-day were to be used, the model would use the sentiment values of the articles from 10/05 - 10/11, while a simple-average with the same window length would use 10/09 - 10/15. The results of each along with varying window length were compared in section 6.

The predictions were made according to the following scoring rule:

**Sell**: if score < 45

**Hold**: if 45 < score < 55

**Buy**: if score > 55

## 6. Experiments and evaluations

Three experiments were conducted by using different window length and scoring function. The accuracies were calculated according to the following metric:

**Correct prediction**:

if 45 < score < 55 and absolute price change < 0.25%

**Correct prediction**:

if score > 55 and price change > 0.25%

**Correct prediction**:

if score < 45 and price change < -0.25%

**False predictions**:

None of the conditions above was met

**Accuracy**:

Correct predictions for class A / all predictions for class A

### 6.1. Experiment 1

The first experiment used the moving average method with window length being the control variable.

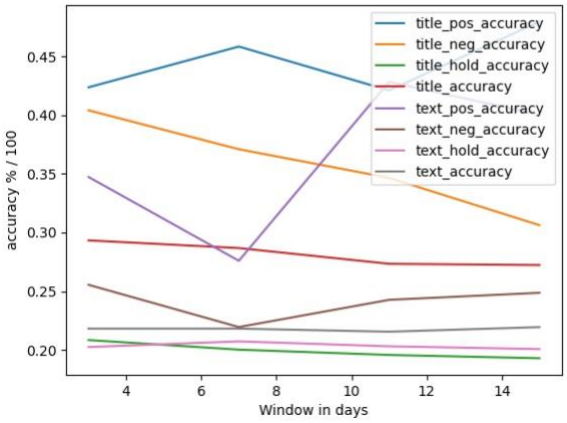


**Figure 3.1: Semi-conductor industry accuracy plot with moving average of window length 3, 7, 11, 15 days.**

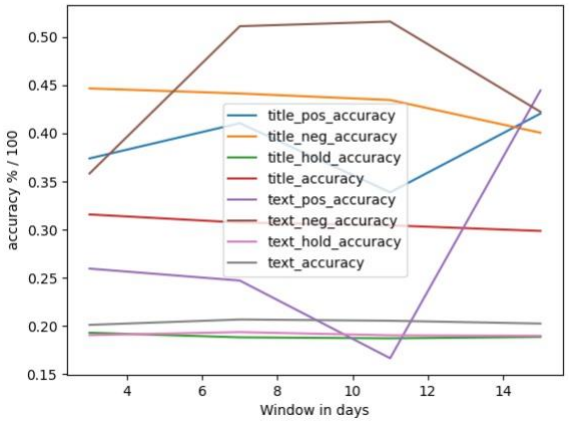


**Figure 3.2: Automobile industry accuracy plot with moving average of window length 3, 7, 11, 15 days.**

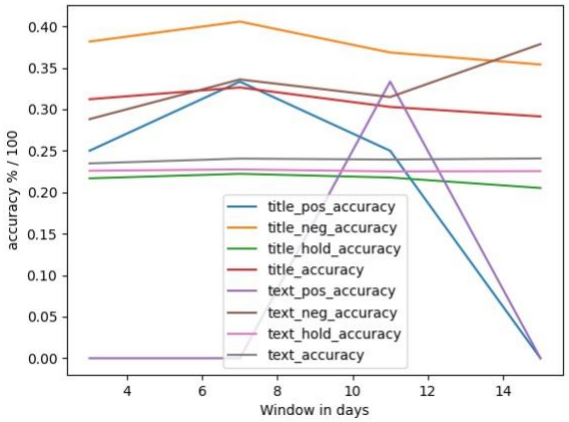


**Figure 3.3: Oil industry accuracy plot with moving average of window length 3, 7, 11, 15 days.**

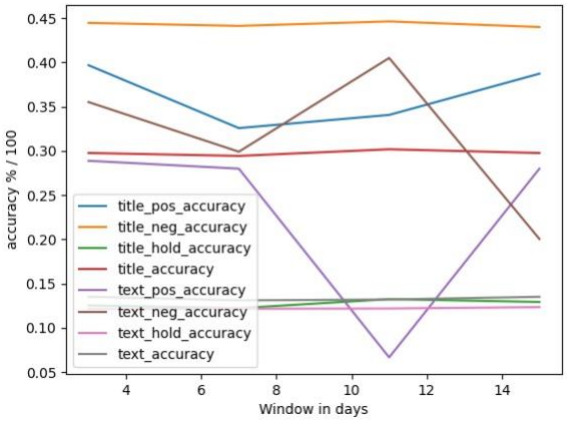


**Figure 3.4: Airline industry accuracy plot with moving average of window length 3, 7, 11, 15 days.**

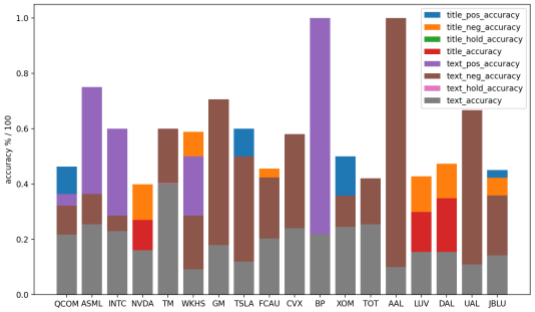


**Figure 3.5: individual stock contribution plot with moving average of window length 11 days. The performance of each industry largely depended on the performance of individual stocks which were further analyzed here. AAL stock achieved extremely high accuracy on predicting 'sell' with negative text sentiment but further analysis showed that this was due to a low number of 'sell' predictions.**

Using random coin flipping to make decisions would have achieved about 10% hold accuracy, 45% buy accuracy, and 45% sell accuracy based on the scoring rule in section 5. The hold accuracy achieved by the moving average method across all the listed industries were close to twice as high as that achieved by random coin flipping, except for the airline industry. This anomaly high accuracy could be explained by step 7 in section 3, where a neutral value was used to fill in empty sentiment values. Another notable outperformance was the accuracy achieved by the sell predictions made from article titles that had negative sentiment, and the sell predictions made from article texts that had negative sentiment in automobile industry with 7-day and 11-day window. The only outperformance achieved based on positive sentiment was the predictions made from article titles that had positive sentiment in oil industry, but the accuracy was not significant higher.

However, the performances in different industries appeared to be inconsistent and no noticeable patterns were observed based on varying window length, but window length of 11 appeared to achieve the best result across the industries.

## 6.2. Experiment 2

The second experiment used the simple average method with window length being the control variable again.

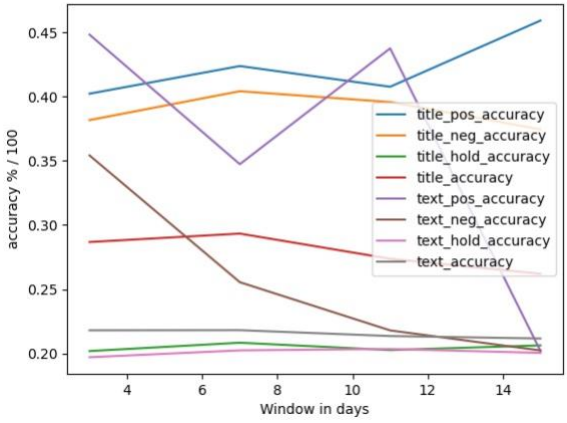


**Figure 4.1: Semi-conductor industry accuracy plot with simple average of window length 3, 7, 11, 15 days.**

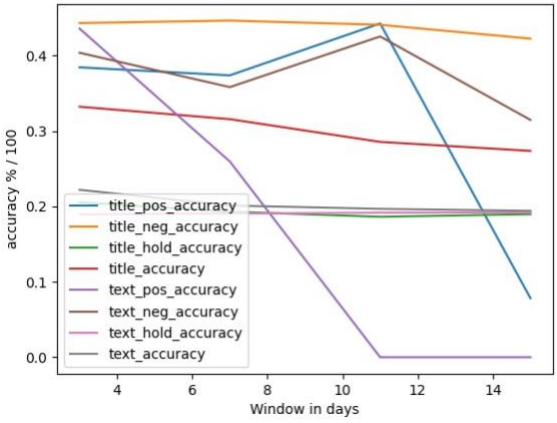


**Figure 4.2: Automobile industry accuracy plot with simple average of window length 3, 7, 11, 15 days.**

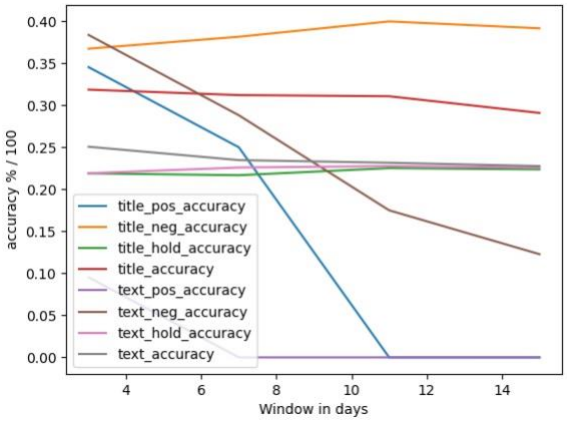


**Figure 4.3: Oil industry accuracy plot with simple average of window length 3, 7, 11, 15 days.**

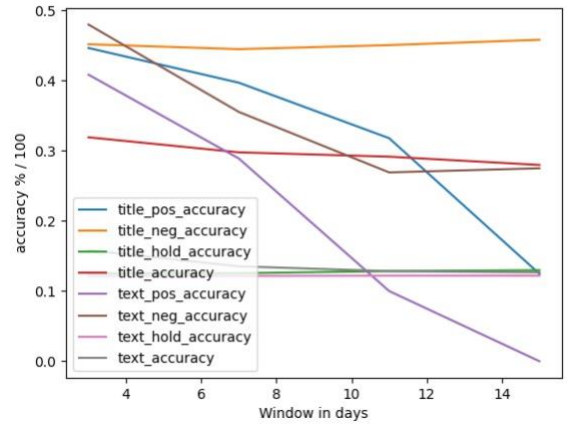


**Figure 4.4: Airline industry accuracy plot with simple average of window length 3, 7, 11, 15 days.**

Figure 4.1 – 4.4 showed that window length tended to have an inverse relationship with prediction accuracy based on positive sentiment in three of the four industries. Predictions based on the sentiment of article title and the accuracy of 'hold' predictions outperformed most other classes of accuracies.

Both experiment 1 and 2 achieved high accuracy on predicting 'hold', however, the most 'hold' decisions were made on the stocks that had many missing article text contents. Section 3 described that the algorithm would first attempt to fill up a missing overall score by using one from a recent day in the past unless none of those days had any overall scores, a neutral score would be used.

## 6.3. Experiment 3

The third experiment measured the covariance for pairs: title sentiment and text sentiment, title sentiment and price change, and text sentiment and price change of each industry with both the simple average and moving average method with varying window length.

| **Relationship** | **Window Length** | **Covariance (degree of freedom = 1) measured by Pandas.Series.cov()** |
|---|---|---|
| Article Text & Title | 3 | 44.68 |
| Article Text & Price Change | 3 | -4.8 |
| Article Title & Price Change | 3 | 2 |
| Article Text & Title | 7 | 16.95 |
| Article Text & Price Change | 7 | -4.1 |
| Article Title & Price Change | 7 | 0.2 |
| Article Text & Title | 11 | 13.71 |
| Article Text & Price Change | 11 | -4.7 |
| Article Title & Price Change | 11 | 2.5 |
| Article Text & Title | 15 | 10.32 |
| Article Text & Price Change | 15 | -1.9 |
| Article Title & Price Change | 15 | -0.2 |

**Figure 5: Covariance Analysis.**

Although some classes of predictions achieved slightly better results than random coin flipping, the covariance between article text / article title and price change did not indicate a strong relationship between them. The covariances based on some relationships and window lengths were even negative, which meant that the article sentiment was in an inverse relationship with price movement. Since both the positive and negative covariances of relationships associated with price were relatively small, the variations may have been caused by a skew data and poor correlation between the two. Nonetheless, the covariance between article title sentiment and price change was the highest when window length was 11-day which further confirmed the observation in experiment 1 with the moving average method.

On the other hand, the covariances between article title sentiment and text sentiment were way above 0, indicating that the articles tended to have similar sentiment for its title and text. However, article-text-and-title covariance started to decrease as window length became greater, this was due to the fact that longer window length accounted for more days' sentiment scores so that the final score derived from them was more stable.

# 7. Learnings

Although the results of the experiments did not achieve significantly higher accuracies than random coin flipping, the process of building this project had been rewarding.

Learnings from building the application: the infrastructure of this application mainly consisted of a module for creating a customized dataset and a scalable pipeline for rerunning different sentiment analysis methods and scoring functions using sentiment values. The

implementations required intensive coding and testing revolved around preprocessing and scoring; mistakes made in early stages of the pipeline became extremely expensive to fix as they and their following stages would have to be rerun and be cached again. One solution was to do a 1-5% of random sampling on the dataset and test the work-in-process algorithm or logic on it first before taking the entire dataset. This helped avoid common program exceptions but sometimes failed to cover all the cases when the number of anomalies was much less than the size of the dataset.

Learnings from making the dataset: it would be much easier to take someone else's dataset from sources like Kaggle but there was no publicly available dataset that perfectly aligned with the goal of this project. Creating a dataset from scratch not only involves the downloading of its contents but also the merging and preprocessing of them, which may be expensive (e.g., human labeling, adding sentiment values). It's also important to design the dataset with efficiency in mind to reduce the cost of processing it in later stages. For example, there were many ways of merging the stock and news dataset but merging all of the article data into the stock dataset indexed by dates would be a nightmare given that each row would then contain unbalanced amount of data and this huge blob of data may slow down the performance and lead to memory issues. It would be much more efficient to use article URL both as a foreign and primary key and store it in the timeline dataset so that other article data stored in another table could be retrieved in O (lg(n)) with binary search. Even when desired datasets are available, further modification may be needed to achieve final goals.

Learnings from the methods studied: the methods used in this project were extremely simple, mostly involved taking averages and designing & testing scoring functions on articles. One tentative conclusion was that it is hard to achieve a high accuracy solely based on a particular approach like sentiment analysis, stock prices are affected by many factors that might not even be able to be captured, where machine learning approaches may be better at extracting the key elements of articles and use them to make predictions.

## 8. Future work

Different ways of determining the final score of an article may be experimented and compared in the future, only simple average and moving average method were explored in this project. In the scope of this project, it would be more important to focus on the deriving the most accurate sentiment value of an article corresponding to a stock rather than one that better at predicting the stock price. In reality, stocks may not be very sensitive to the news sentiment so no matter how accurate the sentiment values are, they may be contributing very little to the price movement of a stock. There may be other latent factors in news articles that could be more easily captured by a machine learning approach rather than a traditional algorithm.

For example, predictions could be made in a supervised manner by constructing a term-frequency and an inverse-document-frequency matrix, converting the top-ranked words using word2cec, and train a classification model like SVM by using the price movement as the ground truth / target value. However, this may deviate from the original goal of using sentiment analysis given that the model would be trained to predict price movement based on the top terms, which does not necessarily correlate to sentiment.

All of the work is open-source and publicly available on GitHub at https://github.com/zachzhu2016/market-sensor.

## 9. References


[1] Patricia S. Abril and Robert Plant, 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan, 2007), 36-44. DOI: https://doi.org/10.1145/1188913.1188915.

[2] Kotartemiy. kotartemiy/pygooglenews. Retrieved December 6, 2020 from https://github.com/kotartemiy/pygooglenews

[3] Ranaroussi. ranaroussi/yfinance. Retrieved December 6, 2020 from https://github.com/ranaroussi/yfinance

[4] Codelucas. codelucas/newspaper. Retrieved December 6, 2020 from http://github.com/codelucas/newspaper

[5] Aaron7sun. 2019. Daily News for Stock Market Prediction. (November 2019). Retrieved December 6, 2020 from https://www.kaggle.com/aaron7sun/stocknews

[6] Anon. News API. Retrieved December 6, 2020 from https://newscatcherapi.com/

[7] Anon. Overview. Retrieved December 6, 2020 from https://stanfordnlp.github.io/stanza/