Stock Market Prediction using News Sentiment with Website Credibility

Chang Jun Qing, Soong Cun Yuan, Vincent Setiawan

Computational Data Science

Abstract

Stock Market Prediction tends to be a common topic when it comes to data science and machine learning. Being able to successfully predict a stock's future price can earn the predictor a large fortune. The aim of our project is to add a twist to the typical Stock Market Prediction with News Sentiment by adding in an additional layer on the credibility of News sources. In this age where Fake News is prevalent, we need to be able to account for news credibility. We achieve this by scoring News Websites with respect to the outlook of the stock that is mentioned in their report. The results seems to indicate that accounting for news credibility provides us with better scores.

Keywords: Natural Language Processing, Stock Market Prediction, News Sentiment, Website Credibility

1. Dataset and Collection

Our project main goal is to use sentiment of news articles on companies to predict the companies future market changes. In order to collect the news sentiment, our group have decided to focus our efforts on stock related news sites as it would likely contain better curated news on the particular stock, compared to news by news sites that is not delegated to stocks. The datasets are collected and captured using the various libraries that are commonly used for data mining, written for and run in Python3 Language. While the other data are attained through the kaggle platform - Huge Stock Market Dataset by Boris Marjanovic.

1.1. Main Python3 Libraries Used

Web Scraping

1. BeautifulSoup
    - BeautifulSoup is a Python library for extracting data from HTML and XML files
2. Selenium
    - Selenium is a portable software-testing framework for web application. This is used in conjunction with beautiful soup which enables a form of automation for web scraping. Seleniums role is use to navigate through the web pages, whereas when the desired page is reached, it will allow BeautifulSoup to extract the HTML/XML files.
3. Requests
    - Requests is a library that enables us to retrieve HTML files from designated websites. This is used for us while scraping data

Data Processing

1. NLTK - VADER
    - NLTK VADER is a sentiment analysis module part of NLTK. This library is used to score the sentiment of our news article
2. Pandas
    - Pandas is a library that provides easy-to-use data structures and data analysis tools for the Python programming language. This is used to help us better manage the extracted data from the automation process for extraction.
3. Keras
    - Keras is a machine learning library that allows quick deployment of neural networks to be applied to datasets. This is used for us to build our model for prediction of the stock data.

1.2. Types of Dataset

- The dataset collected via web scraping (Refer to Figure 1) is a Time-Stamped Data that is Structured, in Labelled-Text format, containing categories as seen in the example below.

- The dataset downloaded from kaggle (Refer to Figure 2) is an Open, Time-Stamped Data, that is Structured containing various forms of prices for a given stock at a given time.

Abc GOOGLE5.csv Author	# GOOGLE5.csv End Date	Abc GOOGLE5.csv Site	# GOOGLE5.csv Start Date	Abc GOOGLE5.csv Text	Abc GOOGLE5.csv Title
Sunny Oh	9,212,017	MarketWat...	9,142,017	Published: Sept 21, 2...	Why is Ireland the U....
Max A. Cherney	9,212,017	MarketWat...	9,142,017	Published: Sept 15, 2...	5 Apple suppliers to ...
Sam Schechner	9,212,017	MarketWat...	9,142,017	Published: Sept 21, 2...	EU gets ready to crac...

Figure 1: Scraped Data Sample

googl.us.txt Date	googl.us.txt Open	googl.us.txt High	googl.us.txt Low	googl.us.txt Close	googl.us.txt Volume	googl.us.txt Open Int
19/8/2004	50.000	52.030	47.980	50.170	44,703,800	0
20/8/2004	50.505	54.540	50.250	54.155	22,857,200	0
23/8/2004	55.375	56.740	54.525	54.700	18,274,400	0

Figure 2: Kaggle Data Sample

1.3. Difficulties & Challenges

Automated Web Scraping

1. HTML Tagging
    - Data Scrapped can be categorized via such format is possible with the use of Python3 Beautiful Soup library. It allows us to scrape specific data based on their HTML tag.
      However challenges do surface as different webpage have their own format of how they tag different portions/category of their article. The process of finding out how each website tag their article, have to be done manually, to correctly extract the right datum for each dataset.
2. CAPTCHA
    - The problem of CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) arises.
      Web journalism website would often purchase services from third party companies like Google for anti-scraping/ automation services on their webpage, preventing us from continuously scraping data, as not before long, the CAPTCHA system catch up to us.
      To prevent this from happening, we implemented User Agent Rotation on our Selenium program, spoofing our user-agent header to reduce incidents of being detected by anti-automation software. We also tried using Virtual Private Networks to change our IP addresses to prevent detection. The new CAPTCHA, reCAPTCHAv3 to be specific, make it hard for our mining process, making it harder to scrape the web without being caught.

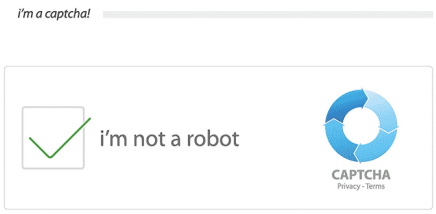


Figure 3: Example of Google's reCAPTCHA

1.4. Collection, combination and Storage of Data

1. Data Collection Storage
    - Collection of Data can be easily managed by running a mySQL client on a host computer/machine.
      To enable better visualisation of categorisation, this can be done using Apache mySQL server. SQL format is chosen as we decided tabular format of datasets make more sense for our uses compared to noSQL formats.
      <Side Note>: Due to the small amount of data we can scrape due to CAPTCHA problems, the storage of data is not an issue and hence was able to just work with the data directly through the use of CSV files and Python dataframe for manipulation.
2. Distributed Job Workflow
    - To enable better efficiency, mining process job can be distributed to multiple machines and then have their mined resources piped through SSH to a central machine hosting the mySQL apache server for collection/ data management. By storing the mined data into this central mySQL server, we can then connect it to Tableau, data visualisation tool.
      <Side Note>: And as previously noted, due to the smaller amount of data, the data mined are directly piped to Tableau instead towards end of project
3. Augmentation of Dataset
    - Datasets of a particular selected company is attained by combining two different sources. One of the dataset comes from Boris Marjanovic Huge Stock Market Dataset obtained through kaggle, and the other is scraped from multiple Stocks news update sites.

2. Data Pre-processing

2.1. Removing noise and garbage data

1. Removing non-related articles
    - Not all the data scrapped of the websites are related to the companies, such as suggested news column in web pages. To remove this, a python script is used to run through each of the articles main body of text to check if articles contains at least one mention of the companies name, else they are highly likely to be unrelated and hence are dropped.
2. Removal of Repeated Articles
    - Repeated Articles can appear, these article may be side articles that are generated as suggestion to the user. This can too be removed by a python script that ensures that no two body of text are the same.

2.2. Reformat data

1. Reformatting Data
    - The dataset from kaggle is processed by arranging the stock prices by week to suit our needs. The dates of the stock prices are also paired with the dates that we were mining.

	Date	Open	Delta	FractionalIncrease	High	Low	Positive
0	2016-11-17	782.50	-27.50	-0.033951	810.060	743.590	False
1	2016-11-25	782.61	0.11	0.000141	793.770	772.650	True
2	2016-12-01	778.55	-4.06	-0.005188	799.740	773.145	False
3	2016-12-08	792.95	14.40	0.018496	792.000	753.360	True
4	2016-12-15	817.36	24.41	0.030784	824.300	787.905	True

Figure 4: Kaggle Data Sample

2.3. Combining data

1. Data Combination
    - After we clean up the data from both the kaggle site as well as the data we scraped ourselves, we combined both data up according to their date. Sentiment score and Site Credibility Scores are also added on to the final dataset. This final dataset contains the features that will be our models input. Each site will have their own credit score tagged as <company>Credit and the sentiment analysis of news from that site is <company>score.

	Open	Delta	ReuterCredit	reuter_score	InvestCredit	invest_score	FoolCredit	fool_score	CnbcCredit	cnbc_score	MarketCredit	market_score
0	778.550	-4.060	0.026139	0.378063	0.023697	0.911567	0.973861	0.996175	0.972168	0.970089	0.976303	0.974950
1	792.950	14.400	0.050411	0.776000	0.068992	0.873125	0.945323	0.995257	0.947928	0.745560	0.947762	0.328733
2	817.360	24.410	0.054512	0.468813	0.062293	0.959300	0.945956	0.988333	0.936020	0.936578	0.950255	0.000000
3	809.100	-8.260	0.041759	0.093980	0.049540	0.982900	0.953223	0.994040	0.958978	0.957580	0.959143	0.000000
4	802.330	-6.770	0.031340	0.634683	0.042534	0.952050	0.961919	0.992733	0.971298	0.985720	0.969111	0.990033

Figure 5: Kaggle Data Sample

3. Problem Scoping, Formulation and Modelling

3.1. Problem Scoping

The aim of the project is to add another dimension into Stock Prediction with News Sentiment[1]. A common issue these days is the existence of Fake News. This started to make people question if a news source is trustable.

3.2. Problem Formulation

Our group aims to tackle this area by adding a website credibility score for each news source that we are taking our news sentiment from. The general idea is the score will be high when the news sites sentiment is in line with the change in that particular stocks price, and similarly, the score will be penalised if the sentiment is opposite the actual change in price.

3.3. Modeling

As we are working with news sources and credibility, we are working on a prediction on change in price every 7 days. This gives us ample buffer room to ensure that an article will be generated within that 7 days. 20% of our data is kept as testing data which will be elaborated in the next section. Of the remaining 80%, 10% of it is used as validation while training the model using Keras.
When scoring the News Articles, there are 3 different scores we are looking at, firstly, we score the stock price using the formula:

$$StockScore = ((NewPrice - OldPrice)/OldPrice)/0.2 \tag{1}$$

$$IfStockScore > 1, = 1 \tag{2}$$

$$IfStockScore < -1, = -1 \tag{3}$$

The stock score is a way we use to manage the change of stock prices where a maximum of 20% increase or decrease would result in the the maximum and minimum score respectively Next, we have the sentiment score where each News Articles sentiment is scored using the NLTK-VADER python library. This score varies from a value of -1 to 1, where the negative scores represent a negative sentiment and positive scores represent a positive one. Both of the above scores are then use to score the credibility of the site then averaged out across the weeks by the following formulas:

$$CredibilityScore = 1 - |SentimentScore - StockScore|/2 \tag{4}$$

$$CumulativeCredibility = (CumulativeCredibility + CredibilityScore)/2 \tag{5}$$

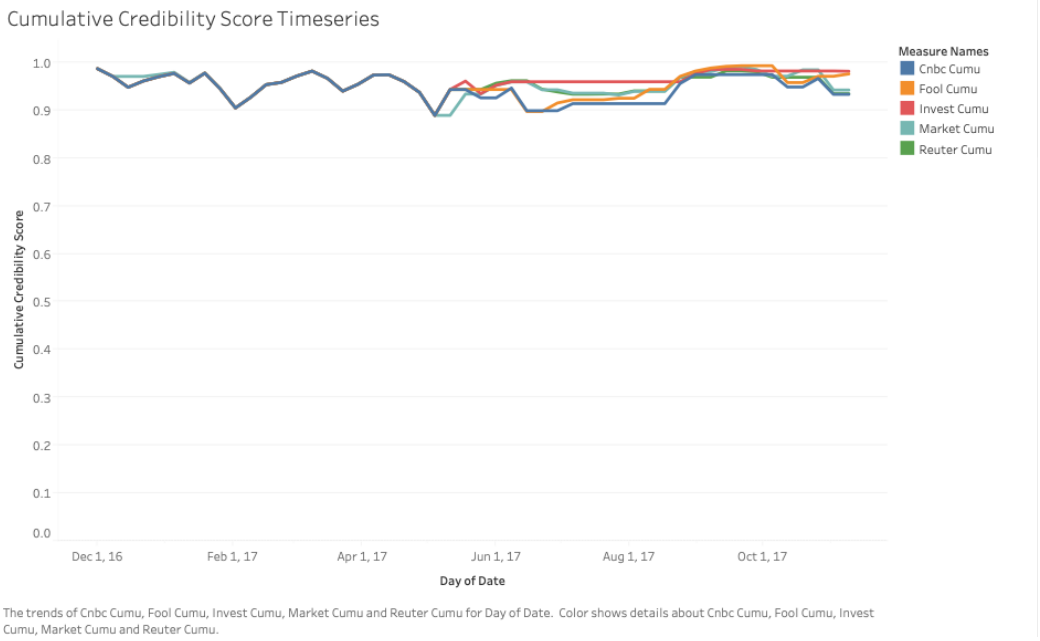
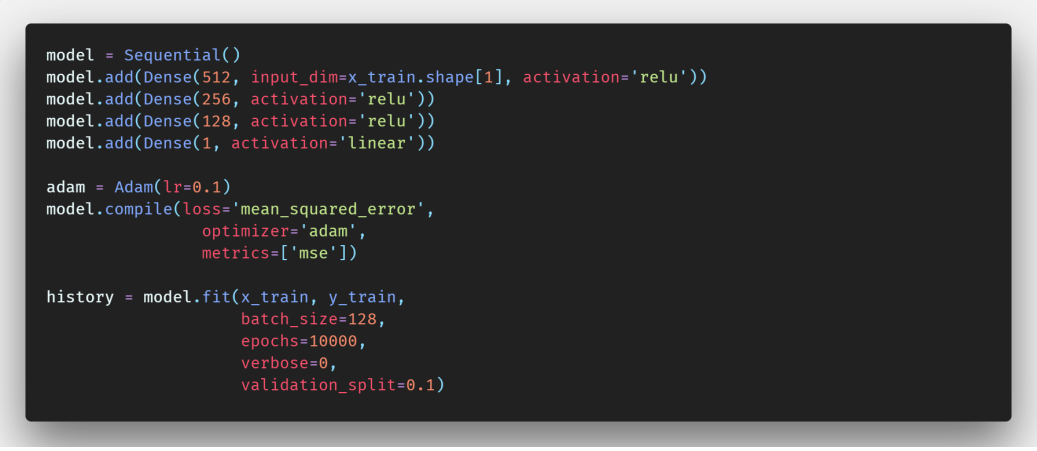


Figure 6: Change in Credibility Score

FIRST MODEL

The sentiment analysis is based off NLTK-VADER. The final data (Refer to Figure 3 the one above) is parsed into a Sequential Neural Network consisting of 3 Dense layers then an output.



```
model = Sequential()
model.add(Dense(512, input_dim=x_train.shape[1], activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='linear'))

adam = Adam(lr=0.1)
model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['mse'])

history = model.fit(x_train, y_train,
                    batch_size=128,
                    epochs=10000,
                    verbose=0,
                    validation_split=0.1)
```

Figure 7: Sample Code

The loss function that is used is mean squared error since we are doing regression and we want to minimise the difference between the predicted change in price and the actual change in price. The output of the model would then be the change in price, delta, of the given stock.

Neural Networks are used as they are often used for pattern prediction. Traditional Time Series methods are only effective when the series is stationary, however, in live trading systems, series are not stationary. Neural network on the other hand does not require any stationarity and will be effective to find the pattern in all these data.[2]

SECOND MODEL

Our second approach is to train the model directly on the fractional increase of the stock price itself instead of learning the sentiment of the article. We are using ELMo, a state of the art character-based word vector which is a learned function of the internal states of a deep bidirectional language model pre-trained on large text corpus.

```
input_text = Input(shape=(1,), dtype=tf.string)
embedding = Lambda(ELMoEmbedding, output_shape=(1024,))(input_text)
dense = Dense(256, activation='relu')(embedding)
pred = Dense(1, activation='linear')(dense)
model = Model(inputs=[input_text], outputs=pred)
model.compile(loss='mse', optimizer='adam', metrics=['mse'])
```

Figure 8: Sample Code

The text is fed into the ELMo[3] embedding which then goes into a fully connected layer with 256 nodes. The final layer is an output layer with one output since we are doing regression on the stock price. The loss function used is mean square error with adam optimizer.
Since this model is character-based, inputting an entire article takes up a lot of gpu memory for tensor allocation. As such, we can only train with small batch size of 4. We only evaluated this model on the Exxon data due to constraint of the training time.

4. Evaluation Methodology

FIRST MODEL

For the first model, our main method of evaluation is using the mean squared error value of the model. Using the remaining 20%, we evaluate our model and obtain a mean squared error of 87.12587992350261. As a comparison, we ran a similar model onto our dataset without the credibility score. That model obtained 128.02448628743488, which indicates that by including credibility score the prediction seems to be more accurate.

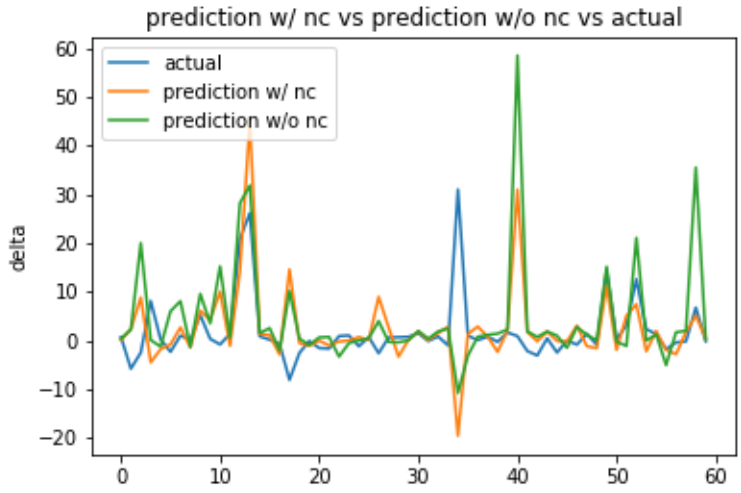


Figure 9: Delta Prediction Comparison

Another method that we used was to see if our model is able to predict the general direction of change. We did this by counting the number of times our model is able to predict a positive or negative change in the price when the actual price has a positive or negative change respectively. Our model with credibility score predicted 37 out of 60 times correctly which is approximately 61.67%. Comparing that to a prediction without using credibility score, that model only predicted 27 out of 60 times correctly, which is approximately 45% of the time.

SECOND MODEL

For ELMo model we evaluated our model with mean squared error value as well and obtain a MSE of 0.6791 for training after 30 epoch. The reason for the large differences in value of MSE for the first and the second model is because for the first model, MSE is evaluated on the actual change in price, whereas for the second model, MSE is evaluated on the fractional change in price which has value of less than 1 most of the time.The testing result is shown below.

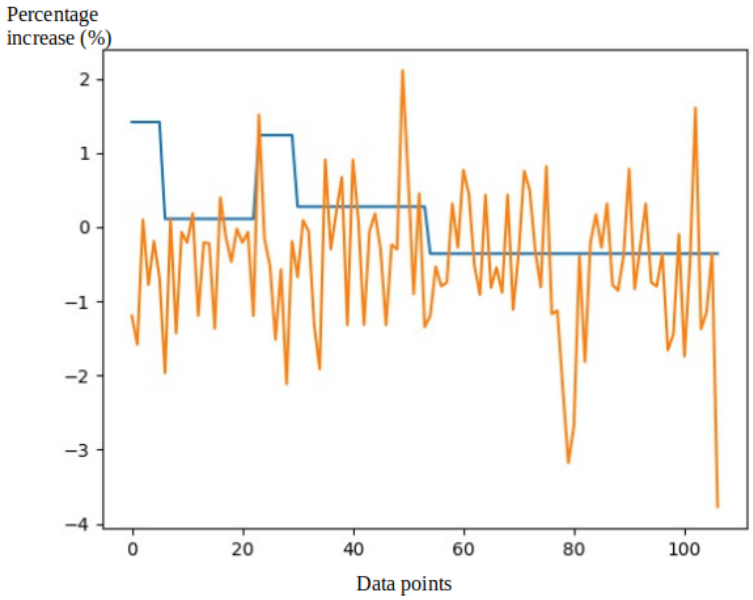


Figure 10: MSE Loss

As we can see, while the model has low training loss, the testing result is not very good. We conclude that this might be due to overfitting due to the limited number of data points available to learn the weights directly on the stock price itself.

5. Results and Discussion

Our result shone a positive light towards the use of credibility scoring to circumvent around fake news and unreliable news sources. The articles sentiment scoring is weighted with the accumulated credibility score of the news source, hence articles from unreliable news sources or fake new sites are heavily penalised to ensure that it will not affect our sentiment stock predictor accuracy. With such positive result, we could further improve it by extending credibility score to individual authors and editors of each news site. This might help further improve our sentiment stock predictor to be more accurate in predicting the next change.

Learning weights directly on the stock price changes instead of using sentiment score from articles may also have potential in the future with more available data.

Main limitations of this study would be the limited data available due to Googles scraping policy. It is difficult to obtain a large corpus of stock-based news specific to a stock without getting constantly blocked by Googles CAPTCHA. Since machine learning approaches benefit from large amount of data, improving on the news corpus will definitely help improve the performances of our approaches.

Another limitation of this study would be that we did not incorporate financial performance and other relevant financial information on the company into the model. Our main aim in this study was to see how accurate can we be with just news and price information on the stock as we wanted to explore, to what extent does positive or negative news affect the price of a stock.

Perhaps we can extend our project to include social media by applying credibility scoring of social media accounts and getting the sentiment of their tweets. Famous bankers/company's CEOs which might indirectly affect the stock prices, hence may help with the accuracy of our stock predictor.

E.g Controversial Elon Musk's tweet criticizing Thai diver, results to Tesla's stock price to drop.

6. Bibliography

[1] Intel, Stock predictions through news sentiment analysis, 2017.

[2] V. Palaniappan, Neural networks to predict the market, 2018.

[3] M. I. M. G. C. C. K. L. L. Z. Matthew E. Peters, Mark Neumann, Deep contextualized word representations, 2018.