# Stock Market Prediction

Salah Alarfaj
UWM CEAS Department of Computer Science
University of Wisconsin
Salarfaj@uwm.edu

## Abstract

The prediction task of the stock market direction is still a challenging task due to its many stimulant. Nevertheless, this study unlike other focused on many stock not just one or an index. Also, it included longer period for the tweets collected than the other papers. Different methods been used in this study that focused on the stock market. Also, a combination of the methods used to show how different stimulant affects the market behavior.

## 1 Introduction

Stock market prices are hard to predict and it's been an active research area for a while. Due to its financial gain it has attracted attentions from the academic and the business side. Building an accurate prediction model is still a challenging problem. It's been known that stock market prices are largely driven by new information and it follows a random walk pattern as stated by the Efficient Market Hypothesis (EMH). Several people have attempted to extract patterns that effects stock market from different stimulant sources. Nevertheless, finding the best time to buy or sell a stock is a difficult task since many factors may influence the stock price.

## 2 Related works

Stock market is still a hot topic due to its natural gain in academic and real-life business. Many researchers tried to address the question with different hypothesis using the random walk theory and the Efficient Market Hypothesis (EMH) [1] it says that the current stock market reflects fully based on all the available informations. So price changes are merely due to new informaiton or news. Becuase news happens randomly in nature so is the stock market prices follows a random walk pattern.

Therefore, they not predictable with more than about 50% accuracy [2]. However, other research specify that the stock market prices do not follow a random walk and can to some level of accuracy be predicted [3] and a degree of 56% accuracy are often reported as satisfying results for stock prediction [4]

There are two distinct trading philosophies for stock market prediction beside EMH and the random walk theories. The fundamental analysis and the technical analysis. The fundamental analysis studies the company books and financial condition, operations, and the macroeconomic indicators to predict the stock. The technical analysis depends on historical data and trends to repeat itself.

Some methods did consider only the historical prices to predict the stock [5] and didn't considered the sentiment on social media. Others did consider sentiment analysis where they used mood mapping to the tweets to predict DJIA price index [6]. Also tried multiple sentiments model then took the majority

vote [7]. While others use sentiments with LDA [9] which is a generative probabilistic model for corpus and JST [10] methods for their prediction [8].

Many research aimed to predict only one stock [7] [11] or DIJA index which showed an accuracy of 73% [6] and the instance in the test set are low such as 14 or 15 instances [12] which conclude insufficient results to the best of my knowledge. Few research however implemented multiple stocks predictions where they used 18 stocks and collect their tweets and sentiments over a period of one year and showed an accuracy of up to 71% for some stocks and an average accuracy of 54% for all 18 stocks [8]. My methods differ from some of the previous works as it's a classification task a not a regression one [7][11].My novel approach as of my knowledge tries to predict many stocks based on their market capitalization including DIJA stocks which lead up to 63 stocks. My twitter data range in date from January 2008 to January 2015 to insure prediction over a long range of time.

# 3 Datasets

Finding the right data is one of the toughest part in machine learning. Four data are collected to work on. The first on is from Twitter by collection the tweets related to the targeted stocks. This was the hardest part as no dataset were available that is satisfying enough to test the hypothesis. The second one is the stock price for each targeted stock. Kaggle provided a dataset that has all the information needed to work on [14]. The third dataset is a news dataset and that last dataset is the Dow industrial index price historical dataset also from Kaggle[15].

**Twitter dataset**

- Twitter API has many tiers
    - Regular: where you only can get data up to 7 days before, and can search with '$' character which is the most important character for stock names search
    - Premium: You can get historic data but you can't use '$' character
    - Enterprise: You can get historic data and you can use '$' character
- It tooks twitter about 2 weeks for them to grant a premium API access key
- Nevertheless, it was useless since '$' character is not supported
- The dataset needed to be collected manually
- It tooks about 2 months to collect the relevant historic tweets form January 2008 to January 2015 with more than 140,000 tweets.
- Only tweets that mentioned targeted stock where collected
- The features collected are:

  Username, Date, Retweets, Favorites, Text, Geo, Mentions, Hashtags, Id, permalink

**Tweets examples**

- "do they have the new 17 mbp at the sf $ aapl store? i should probably avoid..."
- "as hard as I've been on $ MSFT , I like their ad tools which are quite good. But search traffic is still minuscule compared to $ GOOG"

**Stock dataset**

- The stock dataset collected in advance which includes historic data since 2006 and goes back to 2004 for some stocks
- It's separated where each stock has its own file
- This dataset where prepared to fit my method by opening all the targeted files and read the data then sort them by date
- Each based on their IPO might have few hundreds or thousands of records
- Each record represents a full day period

| Date | Open | High | Low | Close | Volume | OpenInt |
|---|---|---|---|---|---|---|
| 11/11/2015 | 18.5 | 25.9 | 18 | 24.5 | 1584600 | 0 |

*Table 1 stock dataset*

**News dataset**

- The news dataset collected in advance which includes historic data from 2008 to 2016
- It has a wide range of news regardless of a specific stock
- It gets the top 25 news for that day
- Each record represents a full day top news

**Dow industrials index**

- The stock dataset collected in advance which includes historic data from 2008 to 2016
- Each record represents a full day period

| Date | Open | High | Low | Close | Volume | Adj |
|---|---|---|---|---|---|---|
| 2016-07-01 | 17924.240234 | 18002.380859 | 17916.910156 | 17949.369141 | 82160000 | 17949.369141 |

*Table 2 DIJA dataset example*

# 3 Prediction

The prediction task is a binary classification task. Because only whether the stock price goes up or down that make one to buy or sell those where targeted.

There exist stock market analytic websites that helps investors make the best decision if a stock is investable or not. This prediction tool can go side by side with those other analytic models. Not all those models targeting the same stocks, where in here I'm targeting 63 stock including DIJA 30 stocks

# 4 Approach

The quality of the dataset is one of the key factor for any prediction task. As I was targeting 66 stocks at first only 63 of them that I could collect useful data for, and only 34 that had 100 and more instance. The main goal of this research is to find whether the stock market for a stock or index will raise or fall. In here I'm dismissing when the stock price stays the same and use closing price to evaluate stock price change.

In my approach I created a new framework to facilitate the process and organize the project. It's based-on layering methodology where each layer is concerned with a main part in making the final model hence the name CPL-D (Collect, Process, and Learn Data) model.

**CPL-D Model Architect**

***Layer 0***

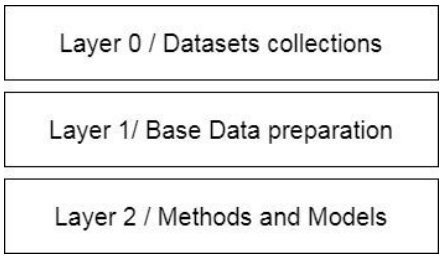


*Figure 1 CPL-D Model Architect*

This layer is our collected dataset that we gathered either from the cloud, database, or gathered from external source. In here API's or collecting tools are place and any tool, configuration, or scripts that works as information gathering.

This is one of the core layers and work as the base for others because the hardest part usually in machine learning is to find the relevant data for the prediction task. When a prediction model is created and chose to be the most accurate model to use. This layer can also work as the base for the prediction model since it will get similar information as the ones used for training.

***Layer 1***

In here we start think of how would we create our base dataset that we will be used to create our methods. We consider our hypothesis and build a generalize dataset.

Data preprocessing takes place in this layer. Where from layer 0 we might have 1,2, or even more dataset collected from different sources to supports our hypothesis and in this layer, we combine them and process them by doing text cleaning for text based tasks or such to fit a base shape dataset.

This isn't necessarily the dataset we load into our prediction model. From this dataset we can create other datasets to test our methods. Think of this dataset produced by this layer as a combined, organized, clean, and general dataset that we can produces subset data, or produce new data from. This layer output is a dataset where it can have all the information needed to derive methods from. It's the base dataset where we can think of it as the features pool.

### *Layer 2*

Here we create the subset dataset to test our hypothesis. The subset dataset is a method that can pick features from layer 1 dataset, create feature from other ones in the dataset, or expand features by using processing algorithms like TF-IDF for text processing.

For each method different features are created to test. You can think of each feature choice as a different method because the results usually change for different feature choice. One method can use TF-IDF where the other one can use wrod2vec binary encoding, others can use different approach like word embedding.

From those methods different training models are used to test the hypothesis. Also for each model an evaluation is made to see how good and accurate our model is. Then we can compare our model's accuracy with each other to find which model and feature choice works the best.

## Model 1

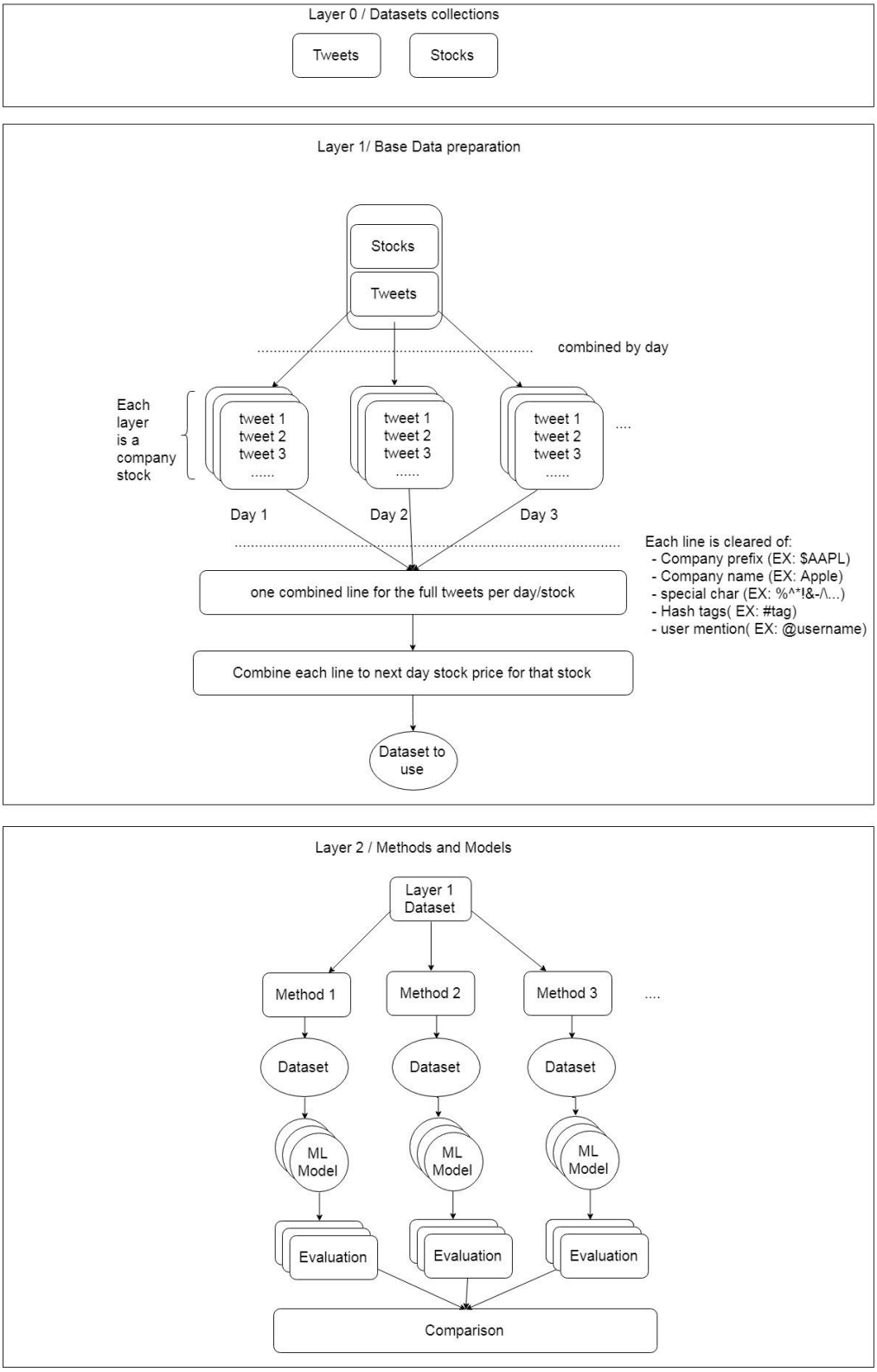


*Figure 2 CPL-D design for Model 1*

### *Layer 0*

The datasets used for this model are tweets collected from twitter, and stock market dataset. The hypothesis I'm testing here is whether today tweets affects next day stock price. For that I chose top 63 stocks based on market capital.

### *Layer 1*

Using datasets from Layer 0 (tweets, and stocks datasets). I first split each tweet by its date. If the tweet timestamp was 19-01-2010 01:20 then I only use the first part 19-01-2010 and group each tweet tweeted in that day together. Afterword will have the data split by date and now we use our targeted stock name list to build a stock dictionary where it will group the tweets for all days that mentioned that stock.

Because our hypothesis is to test whether today tweets affects tomorrow stock price. Each instance in the dataset should be the tweets and the stock status of whether it goes up or down. We first clean the text for each tweet from unwanted names, hashtags, or special characters and so on. Then we combine the tweets for that stock in that day together using space between each tweet.

Next, we need to know the status of that stock for the next day after those tweets so we get the day of the tweets and combined with the next day stock price. This process might lead to some error which I solved in my code. The first error is on that day the tweets collected it might be on holidays or where the stock market is close for that day. So, getting the same day stock price is not in the stock market price dataset. So, I use previous day to the first match on the stock dataset and get the price then set tweets current price as that one. Second error similarly is for the next day price where I searched for the next day stock market price and keep searching until I find a record. Some other stocks might not have previous or next day price in the dataset so those get dropped.

After completing the process, I get a full set of features of all the data collected before with the one I have processed and add those to a base dataset. This dataset contains all the information needed or even helpful to keep the dataset organize like timestamp or even random features like including all the features in the layer0 dataset. The base dataset is not the one we train on. It's mostly the base dataset that build our hypothesis that we then can build our methods on top of it.

### *Layer 2*

#### Method 1

This method is based on sentiment analysis of tweets text for that day. Its contains 9 main features:

- Full tweets for a day (used to create TF-IDF, top 300 word with a count of 55 or more)
- Stock name
- Positive words count
- Negative words count
- Positive words sum
- Negative words sum
- Web Mentions count
- Sentiment count status
- Sentiment sum status
- Stock status {Fall, Raise}

Wordnet is huge dataset that contains many words with their synonyms which are called synsets. Where it distinguishes between nouns, verbs, adjective and adverbs. Wordnet has an extension called SentiWordNet which for each synset it adds 3 measures: 1- Positive score of a word, 2- Negative score, 3- objective score. Where the objective score is calculated as => objScore = 1 – ( PosScore + NegScore).

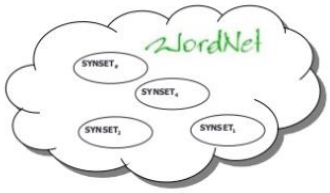


*Figure 3 Wordnet sysnsets cloud*

SentiWordnet uses ranking to rank a word for its positive and negative values. It does that by using $p - normalized\ Kendall\ T\ distance$. [13] It return scores measure for each word. The positive and the negative score. I used those scores to calculate my positive and negative score summation as follows

$$Positive\ words\ sum\ (S_+) = \sum_{i \in T} \sum_{s \in i} posScore_s$$

*Equation 1*

$$Negative\ words\ sum\ (S\_) = \sum_{i \in T} \sum_{s \in i} negScore_s$$

*Equation 2*

Where i is a word in the sentence T and s is each synonym for that word. This sum is counted for all the words in the sentence. Then using the same summation for each word, we compare which one ranked higher than the other and if it's positive rank is higher than the negative one then we classify that word as a positive word and increment the positive words counter by one and vice versa.

$$S_i = \begin{cases} postivie\ if\ S_+ > S_- \\ Negative\ if\ S_+ \leq S_- \end{cases}$$

Where $S_i$ represents a word in the sentence.

The sentiment count status features is a binary classification feature that list whether the positive words count are higher or not it compares the positive and negative count features to pick which one is greater.

$$S_{countStatus} = \begin{cases} Postivie\ if\ S_{positive\ words\ count} > S_{negative\ words\ count} \\ Negative\ if\ S_{positive\ words\ count} \leq S_{negative\ words\ count} \end{cases}$$

The sentiment sum status features is also a binary classification feature that list whether the positive words sum are higher or not it compares the postivie and negative sum features to pick which one is greater.

$$S_{sumStatus} = \begin{cases} Postivie\ if\ S_{positive\ words\ sum} > S_{negative\ words\ sum} \\ Negative\ if\ S_{positive\ words\ sum} \leq S_{negative\ words\ sum} \end{cases}$$

In this method I tested whether sentiment of the whole day tweets affects next day stock market price for that stock.

### Method 2

This method doesn't consider sentiment analysis but uses other features and mapping to build its models. The features for this method are:

- Full tweets for a day (used to create TF-IDF, top 300 word with a count of 55 or more)
- Stock name
- Stock status {Fall, Raise}
- Number of tweets count
- Tweets length sum
- Sell count
- Buy count
- Early to mid-day
- Mid to night
- Night to late night
- Buy sell status

The number of tweets in each line comes from the base dataset (layer1) which is simply how many tweets are in that full tweets line. Because each line represents a day then those can be think of as the number of tweets collected for that day that mentions the target stock.

Sell and buy counts are the number of 'sell' and 'buy' words appear for that day. The interesting features in this method are the timestamp of the tweets. Which are used to see if any correlation arises from when the tweets being tweeted and the stock status.

Because I'm adding all the tweets for that stock in that day in the same line then it made sense to take the sum of the timestamp of the tweets to see whether if more tweets are tweeted in for example early in the day then later.

| NAME | RANGE |
|---|---|
| **EARLY TO MID-DAY** | 3:00am < x <= 12:00pm |
| **MID TO NIGHT** | 12:00pm < x <= 9:00pm |
| **NIGHT TO LATE NIGHT** | 9:00am < x <= 3:00pm |

*Table 3 time map values*

$$\text{Early to midDay} = \sum_{i \in T} \{ i \mid 3{:}00\text{am} < \text{i}_{timestamp} \leq 12{:}00\text{pm} \}$$

$$\text{Mid to Night} = \sum_{i \in T} \{ i \mid 12{:}00\text{pm} < \text{i}_{timestamp} \leq 9{:}00\text{pm} \}$$

$$\text{Night to late night} = \sum_{i \in T} \{ i \mid 9{:}00\text{am} < \text{i}_{timestamp} \leq 3{:}00\text{pm} \}$$

*Equation 3 Time map*

T = tweets

The buy sell status features is a trinary feature that list whether the buy keyword appears more than sell for that day or not. Some days buy and sell count aren't mentioned so for those we have our third attribute or buy and sell words count matched.

$$S_{buySellStatu} = \begin{cases} Buy \ \ if \ S_{buy\ count} > S_{sell\ count} \\ Sell \ if \ S_{buy\ count} < S_{sell\ count} \\ Nothing \ \ if \ S_{buy\ count} = S_{sell\ count} \end{cases}$$

*Equation 4*

## Method 3

In this method I'm testing if combining method 1 and 2 would have a better result. The features used are from both two methods with the TF-IDF of the text for the top 300 words with count of 55 or more.

## Method 4

AS of all the previous methods the full data where used. Where some stocks have only 5 or less instance in them. In this method only, stocks that has 100 instances or more are kept in and dismissing the others. The data been used in the previous methods are

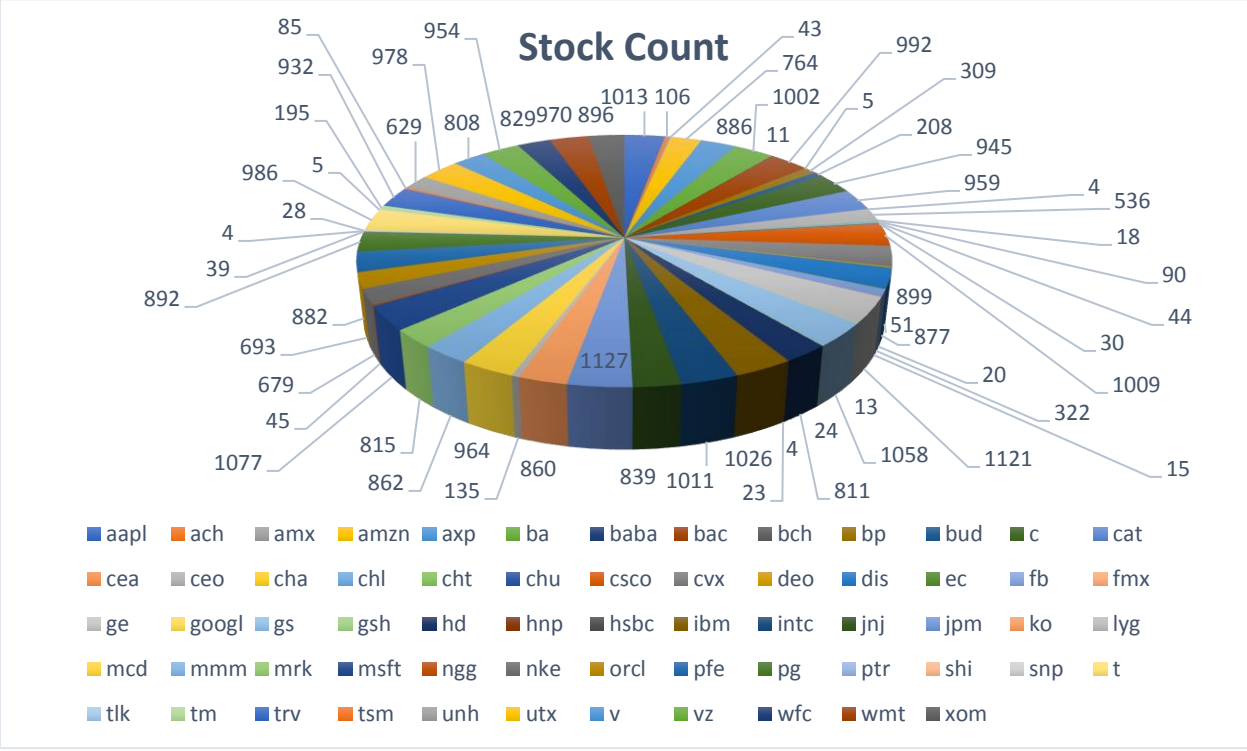


*Figure 4 stock count for the full 63 stocks*

But after reducing the dataset I got

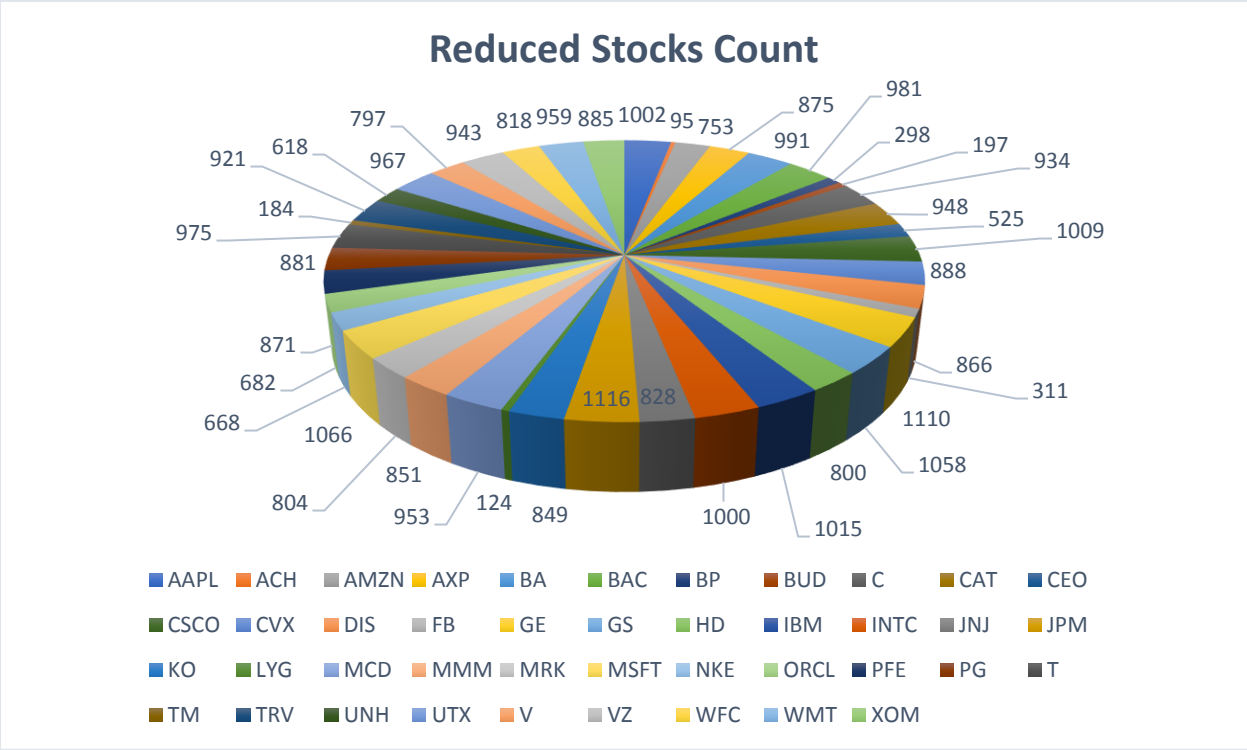


*Figure 5 Stocks count for the reduced 42 stocks*

Which left only 42 stocks. I used methods 1 features which is based on sentiment analysis of text

From the dataset created we can chart sentiment/stock price time series to see the correlation. All the 42 stocks used with a total of 33416 instance where for the previous method it was 34457 range from 2008 to 2011. Charting the full set, we get

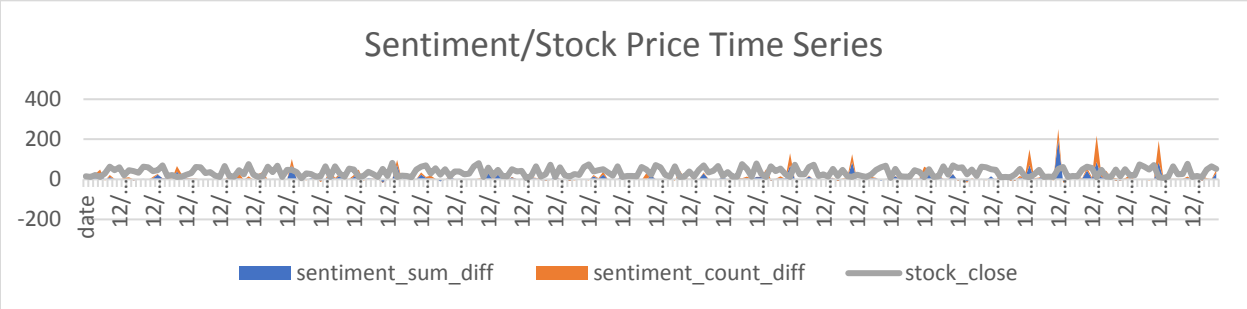


*Figure 6 Time series for the full 42 stocks over 2008 to 2011 period*

Which is cluttered and doesn't show useful info. So, I reduced the chart to only a period of time and picked instance that where in 2008 which result in

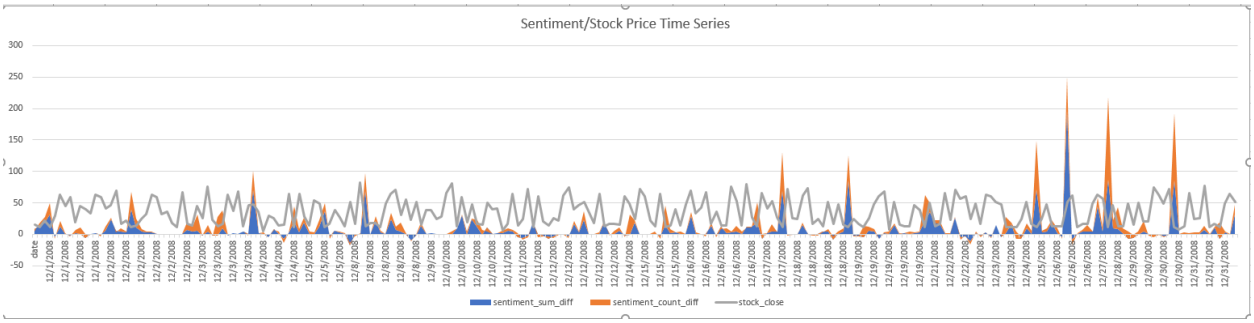


*Figure 7 Time series for the full 42 stocks for only 2008*

Which from the chart we can see that there are some areas where the sentiment sum and count move in the same direction as the stock's price where in other areas they are completely in the wrong direction.

## Method 5

A different approach where taken in this method. Instead of considering sentiments or mapping of some feature. A rule base approach is used to build the features.

First, I considered the most common words that appears for both when the stock rise or fall. I used the top 20,000 words based on the word frequency. I first tooknize the tweets text

$$F(w) \begin{cases} if\ w\ in\ list, [w] = [w] + 1 \\ if\ w\ not\ in\ list, [w] = 1 \end{cases}$$

$$\text{Word Frequency} = \sum_{w \in T} \{\, w \mid F(w) \}$$

*Equation 5 word frequency*

Where T is the full tweet text, and F(w) is a function that keeps track if that word is found before then it's increment it by 1 else it adds that word.

$$\text{positive words} = \sum_{i \in M} \{ w \mid w \in T \,\&\, w \in P(w) \}$$

*Equation 6*

Where M is the number of the top words we are looking for (here I chose 20,000), T is the full text tweets, and P(W) is a function that only let us choose the positive word based on descending order starting from the most frequency word. Using word cloud to draw the positive words.


*Figure 8 positive words for the full range*

For the negative words similar approach where taking

$$\text{Negative words} = \sum_{w \in M} \{w \mid w \in T \,\&\, N(w)\}$$

*Equation 7*

Where here N(W) is a function that only let us choose the negative word based on descending order starting from the most frequency word. Also using word cloud to draw the negative words.

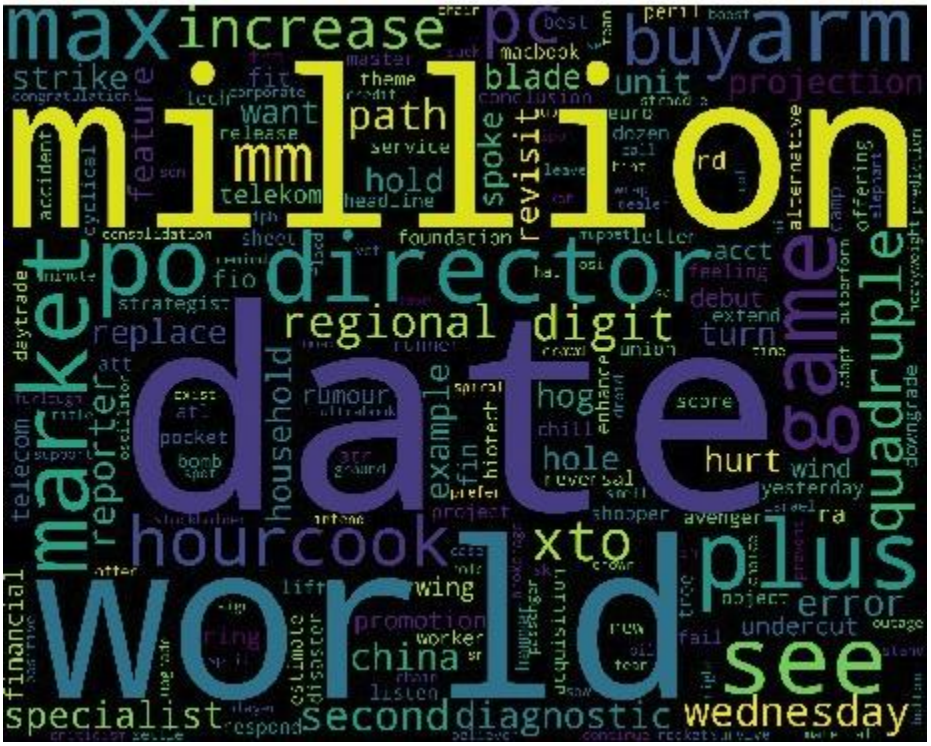


*Figure 9 negative words for the full range*

From those images we can notice that some words appear in both the positive and negative list. Some of them with high frequency as **million**, **date**, and **increase**. However, after taking away all the intersecting words form both lists we have a better indication.

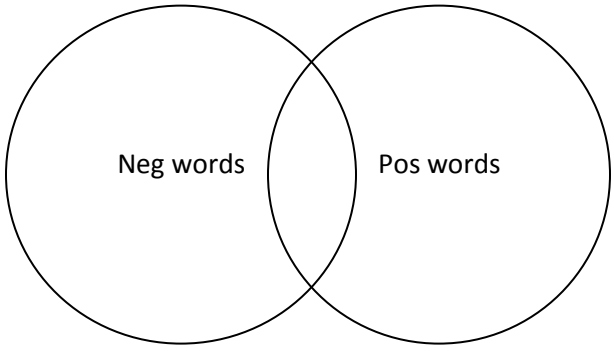


*Figure 10 Intersections of words*

| List | Count |
|---|---|
| **Positive words** | 961639 |
| **Negative words** | 809020 |
| **Intersection words** | 22244 |

*Table 4 words count*

And after taking the intersecting words out the words count dropped to

| List | Count |
|---|---|
| **Positive words** | 23054 |
| **Negative words** | 21474 |

*Table 5 words count after reduced*

Which is more than 22244 words for each. That is because some words are repeating and we are taking all of them out from both lists.

After taking the intersectional word, the positive list now shows

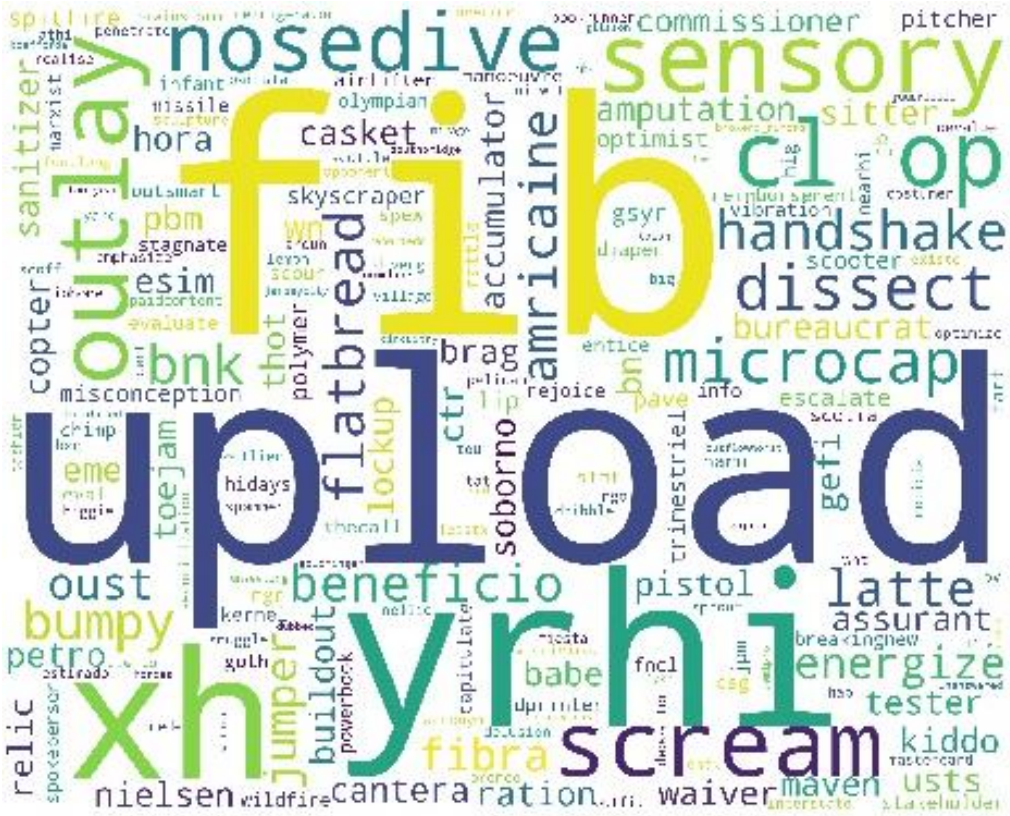


*Figure 11 positive words after reduce*

And the negative list shows

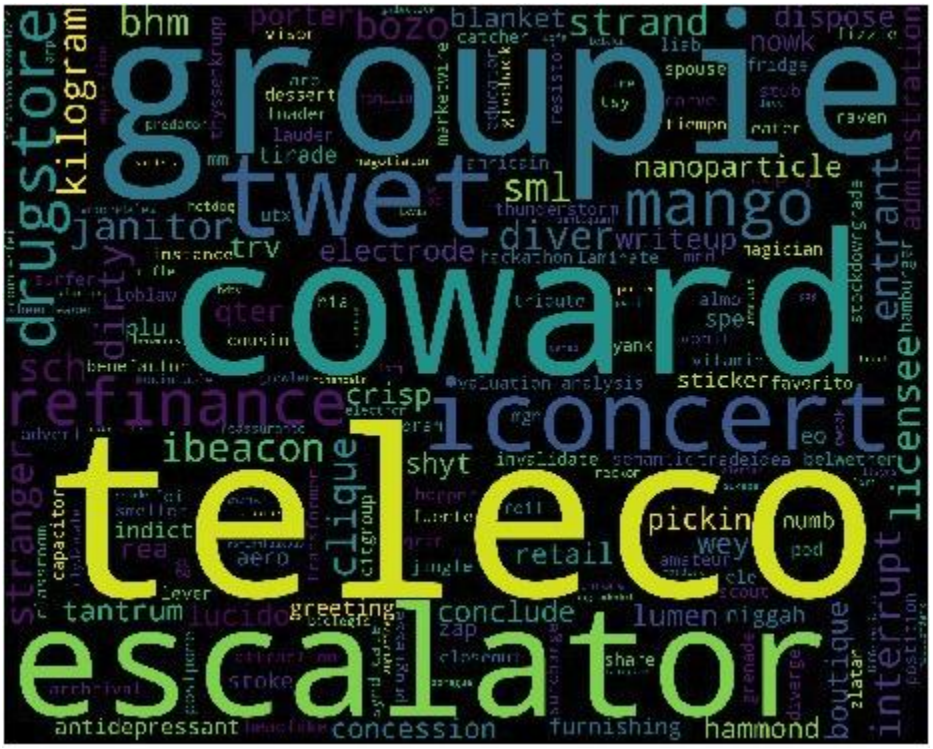


*Figure 12 negative words after reduce*

With that we can see that the words are more distinct for the positive and the negative ones. So now I can manually pick the words that based on my intuition felt it had more soundness to positive and negative meanings as features.

From the word cloud the word chosen are:

- offer
- deduct
- spinonthursdays
- boomfolio
- djkerosene
- rockstarz
- hutzdon
- lasvegas
- 11bn
- transplant
- digi
- compassionateuse
- compassionateusereform
- warrant

Also, I used wrodNet to find the hypernyms of a word and take that as features too. Where I trace each word root and count the frequency of the hypernyms that appears in the word.

```
def get all hypernyms(word):
    w = Word(word)
    reach_root = False
    for syns in w.synsets:
        reach_root = False
        i = 0
        while reach_root == False:
            try:
                if i == 0:
                    add to dict(root dicts, syns.hypernyms()[0], i)
                elif i == 1:
                    add to dict(root dicts, syns.hypernyms()[0].hypernyms()[0], i)
                elif i == 2:
                    add_to_dict(root_dicts, syns.hypernyms()[0].hypernyms()[0].hypernyms()[0], i)
                elif i == 3:
                    add_to_dict(root_dicts, syns.hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0],i)
                elif i == 4:
                    add to dict(root dicts, syns.hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0] ,i)
                elif i == 5:
                    add to dict(root dicts, syns.hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0] ,i)
                elif i == 6:
                   add_to_dict(root_dicts, syns.hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0].hypernyms()[0],i)
                else:
                    reach_root = True
                i = i + 1
            except:
                    reach_root = True
```

*Figure 13 code snippet used to get the hypernyms*

I search for each synsts for that word and for each one it finds the hypernyms and then count the hypernyms it found. Example of a hypernyms tree for the word ‘million’.

{large indefinite quantity, large integer}

=> {indefinite quantity, integer}

=> {number, measure}

=> {entity}

=> {abstraction}

Each word has different level of hypernyms. Some words have only one level where others might have up to 6 or more. In this approach I find the hypernyms up to the 6$^{th}$ level. Taking the hypernyms for the word and its synsets allow us to find a larger range of features to choose from.

Those features then are used with method 4 features. The reason I chose method 4 dataset because it only includes the stocks that has 100 or more instance which is more accurate data since for the previous one if a stock only has 4 instances and all of them are positive then we can always assume that this stock is going to raise which is a false statement.

The hypernyms then are manually pick based on which one has the most count and convey a meaning relates to stock market. The one chosen are

- digit
- agent
- relation
- measure
- electromagnetic_unit
- artificial_language
- direction

- abstraction
- living_thing
- physical_entity
- signal
- matter
- mass_uni
- metric_weight_unit
- property

Also 4 more files are printed out which are the positive and the negative words ordered by the most frequently words and the hypernyms for the positive and the negative words ordered by the hypernyms order then by the number of that hypernyms root appears in the text

### Method 6

Because we are working with textual data. It made sense to use long-short term memory (LSTM) recurrent neural network. Where it will embed the words, and build a word embedding vector and use that to train and predict the model.

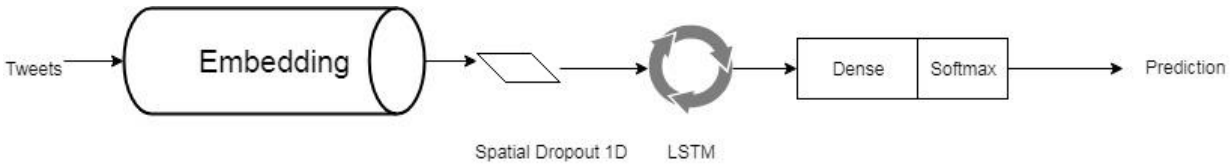


*Figure 14 LSTM Model*

First, we embed the words with an embed diameter of 128, then I used a dropout spatial layer so it will drop the neuron that doesn't make a difference. Then feeding the output to an LSTM layer which has output diameter of 196 from that I build a dense layer with softmax activation function that connects all the neurons together to form the prediction needed. (Keras with tensor flow used to build the model)

**Model 2**

Because the accuracy didn't go over 57% for the best method in model 1 a different approach was taking. As the main goal of this research is to find the best match of stimulant for the stock market, layer 0 dataset where changed and another hypothesis is tested. Which is whether today's news affected next day stock market price.

News usually doesn't mention a specific stock. However, it could result in a direct impact on the stock market economy. Example of such that China manufacture could change their policy or has government regulation or release employees and then some stocks in U.S. might get affected by this news.

With that we can see that some news has a direct affect and that specific stock is mentioned. Other might have a hidden repeal affects to other stocks around the world. Even thou affect might be direct to a specific stock the same news can have hidden repeal affects to others.

With that for this model the approach I took is to measure news effects on Dow Jones Industrial Average. Dow's index takes an average of top 30 companies.

| Dow's 30 companies | | | | |
|---|---|---|---|---|
| MMM 3M | AXP American Express | AAPL Apple | BA Boeing | CAT Caterpillar |
| CVX Chevron | CSCO Cisco | KO Coca-Cola | DIS Disney | DWDP DowDuPont Inc |
| XOM Exxon Mobil | GE General Electric | GS Goldman Sachs | HD Home Depot | IBM IBM |
| INTC Intel | JNJ Johnson & Johnson | JPM JPMorgan Chase | MCD McDonald's | MRK Merck |
| MSFT Microsoft | NKE Nike | PFE Pfizer | PG Procter & Gamble | TRV Travelers Companies Inc |
| UTX United Technologies | UNH UnitedHealth | VZ Verizon | V Visa | WMT Wal-Mart |

*Table 6 DIJA 30 stocks*

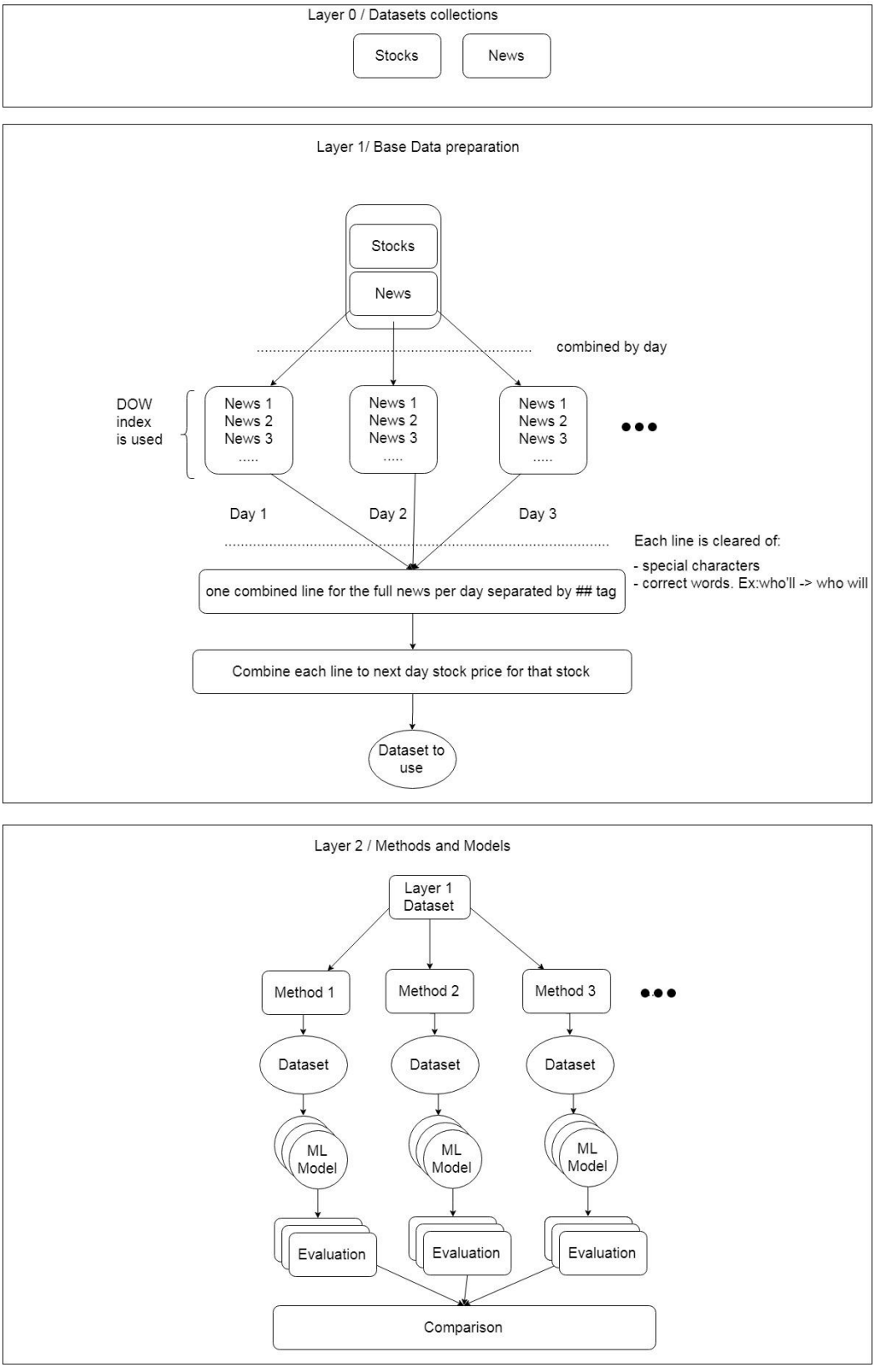


*Figure 15 CPL-D design for model 2*

***Layer 0***

The datasets used for this model are news collected from Kaggle, and stock market dataset for Dow index. The hypothesis I'm testing here is whether today's news affected next day stock market price. For that I chose top 63 stocks based on market capital.

***Layer 1***

Using datasets from Layer 0 (news, and stocks datasets). I first get all dates in the stock price dataset. Then find each headline news that are out for that day. The date combined by only the first part example if the news collected a release at 20-10-2014 02:20 then only 20-10-2014 is used. Therefore, combined all the news for that date regardless of the time.

Then the headlines are combined with the stock price for that day. From those prices a comparison is made to find whether the stock fall or raise and add that too to the headlines and the stock price.

Because the headlines can have many formats. Each headline is cleared for any special characters such as @, #, & … etc then a dictionary is built to fix words like "we'll " and write the full meaningful format "we will". Afterword, I combined each headline for that day together and add " ## " between each. Those to build a full one feature for the news text.

***Layer 2***

### Method 1

This method similarly to model 1 method 1 is based on sentiment analysis of all the news for that day. The features used are:

- Full tweets for a day (used to create TF-IDF, top 300 word with a count of 55 or more)
- Stock name
- Positive words count
- Negative words count
- Positive words sum
- Negative words sum
- Web Mentions count
- Sentiment count status
- Sentiment sum status
- Stock status {Fall, Raise}

SentiWordnet uses ranking to rank a word for its positive and negative values. Using the same equation [eq 1] to find the positive sum of the words and [eq 2] for the negative sum then counting the negative and positive words using [eq 3] to find the features.

In this method I tests whether sentiment of the whole day news affects next day Dow index price.

### Method 2

Similarly, to the approach used for model 1. Because we are working with textual data it made sense to use long-short term memory (LSTM) recurrent neural network. Will take the news and combine them and build a word embedding vector and use that to train and build the model. Will use a model similar to Figure 14 but instead of tweet to feed I'll be feeding the news headlines.

First, we embed the words with an embed diameter of 128, then I used a dropout spatial layer so it will drop the neuron that doesn't make a difference. Then feeding the output to an LSTM layer which has output diameter of 196 from that I build a dense layer with softmax activation function that connects all the neurons together to form the prediction needed. (Keras with tensor flow used to build the model)

# 5 Results

| Model 1 | | | | |
|---|---|---|---|---|
| **Model** | | **Method 1** | | |
| | ***Correctly classified*** | ***Kappa value*** | ***F-Measure*** | ***ROC Area*** |
| **J48** | 55.83% | 0.1067 | 0.557 | 0.566 |
| **IBK, K =3** | 57.56% | 0.142 | 0.574 | 0.601 |
| **Cost Sensitive with IBK, K = 3 and weighs 5= RAISE 4=FALL** | 57.56% | 0.1417 | 0.574 | 0.601 |
| **Bagging (IBK, K=3)** | 57.42% | 0.1409 | 0.573 | 0.608 |
| **Vote (J48, IBK k=3, Cost Sensitive with IBK k=3) with average probability voting** | 57.72% | 0.1442 | 0.575 | 0.611 |
| | | **Method 2** | | |
| **J48** | 55.83% | 0.1064 | 0.556 | 0.567 |
| **IBK, K =3** | 57.91% | 0.1496 | 0.578 | 0.604 |
| **Cost Sensitive with IBK, K = 3 and weighs 5= RAISE 4=FALL** | 57.90% | 0.1493 | 0.578 | 0.604 |
| **Bagging (IBK, K=3)** | 57.61% | 0.1453 | 0.576 | 0.609 |
| **Vote (J48, IBK k=3, Cost Sensitive with IBK k=3) with average probability voting** | 57.73% | 0.1447 | 0.575 | 0.614 |
| | | **Method 3** | | |
| | | | | |
| **J48** | 56.06% | 0.1139 | 0.560 | 0.567 |
| **IBK, K =3** | 58.06% | 0.1534 | 0.580 | 0.607 |
| **Cost Sensitive with IBK, K = 3 and weighs 5= RAISE 4=FALL** | 58.08% | 0.1536 | 0.580 | 0.607 |
| **Bagging (IBK, K=3)** | 57.97% | 0.1523 | 0.579 | 0.612 |
| **Vote (J48, IBK k=3, Cost Sensitive with IBK k=3) with average probability voting** | 57.94% | 0.1491 | 0.578 | 0.616 |
| | | **Method 4** | | |
| **J48** | 55.60% | 0.1032 | 0.555 | 0.565 |
| **IBK, K =3** | 57.99% | 0.1508 | 0.579 | 0.602 |
| **Cost Sensitive with IBK, K = 3 and weighs 5= RAISE 4=FALL** | 57.96% | 0.1499 | 0.578 | 0.602 |
| **Bagging (IBK, K=3)** | 57.72% | 0.1474 | 0.577 | 0.609 |
| **Vote (J48, IBK k=3, Cost Sensitive with IBK k=3) with average probability voting** | 57.79% | 0.1457 | 0.576 | 0.612 |
| | | **Method 5** | | |
| **J48** | 55.54% | 0.1026 | 0.554 | 0.565 |
| **IBK, K =3** | 55.37% | 0.0921 | 0.549 | 0.574 |
| **Cost Sensitive with IBK, K = 3 and weighs 5= RAISE 4=FALL** | 55.33% | 0.0841 | 0.542 | 0.574 |
| **Bagging (IBK, K=3)** | 55.58% | 0.1035 | 0.555 | 0.579 |
| **Vote (J48, IBK k=3, Cost Sensitive with IBK k=3) with average probability voting** | 56.12% | 0.1086 | 0.557 | 0.589 |
| | | **Method 6** | | |
| **LSTM** | 56.54% | 0.1173 | 0.5615 | 0.588 |

For all models 10 cross validation was used.

| Model 2 | | | | |
|---|---|---|---|---|
| **Model** | | **Method 1** | | |
| | ***Correctly classified*** | ***Kappa value*** | ***F-Measure*** | ***ROC Area*** |
| **J48** | 49.17% | -0.0196 | 0.491 | 0.502 |
| **IBK, K =3** | 52.99% | 0.0533 | 0.525 | 0.516 |
| **Cost Sensitive with IBK, K = 3 and weighs 5= RAISE 4=FALL** | 52.99% | 0.0533 | 0.525 | 0.516 |
| **Bagging (IBK, K=3)** | 51.89% | 0.0303 | 0.512 | 0.508 |
| **Vote (J48, IBK k=3, Cost Sensitive with IBK k=3) with average probability voting** | 50.53% | 0.0043 | 0.501 | 0.513 |
| | | **Method 2** | | |
| **LSTM** | 44.66% | -0.1052 | 0.447 | 0.433 |

*Table 7 Model 1 and Model 2 results*

**Predicting each stock alone**

Using Method 4 IBK, with k=3 to find how good the model did for each 42 stocks. Each stock had 10 instances to test the model with.

| Stock name | Correctly Classified | Kappa Value | F-Measure | ROC Area |
|---|---|---|---|---|
| **AAPL** | 63.64% | 0.2667 | 0.636 | 0.683 |
| **ACH** | 54.55% | -0.0377 | 0.563 | 0.625 |
| **AMZN** | 54.55% | -0.0377 | 0.563 | 0.625 |
| **AXP** | 45.46% | 0.1538 | 0.427 | 0.479 |
| **BA** | 63.64% | 0.12 | 0.598 | 0.607 |
| **BAC** | 36.36% | -0.4528 | 0.339 | 0.143 |
| **BP** | 36.36% | -0.3051 | 0.375 | 0.232 |
| **BUD** | 72.73% | 0.459 | 0.727 | 0.783 |
| **C** | 63.64% | 0.2667 | 0.636 | 0.767 |
| **CAT** | 63.64% | 0.2143` | 0.636 | 0.714 |
| **CEO** | 36.36% | -0.0845 | 0.309 | 0.571 |
| **CVX** | 36.36% | -0.1846 | 0.364 | 0.500 |
| **DIS** | 54.55% | 0.0984 | 0.545 | 0.513 |
| **FB** | 54.55% | 0.127 | 0.522 | 0.650 |
| **GE** | 54.55% | 0.0984 | 0.545 | 0.600 |
| **HD** | 45.46% | -0.0645 | 0.464 | 0.429 |
| **IBM** | 45.46% | -0.0313 | 0.396 | 0.717 |
| **INTC** | 81.82% | 0.56 | 0.799 | 0.821 |
| **INJ** | 45.46% | -0.0645 | 0.445 | 0.567 |
| **JPM** | 72.73% | 0.4407 | 0.723 | 0.817 |
| **KO** | 54.55% | 0.1538 | 0.545 | 0.589 |
| **LYG** | 36.36% | -0.3051 | 0.375 | 0.375 |
| **MCD** | 54.55% | 0.0984 | 0.545 | 0.583 |
| **MMM** | 36.36% | -0.2623 | 0.364 | 0.467 |
| **MRK** | 72.73% | 0.3774 | 0.717 | 0.768 |

| | | | | |
|---|---|---|---|---|
| **MSFT** | 90.91% | 0.7442 | 0.916 | 0.889 |
| **NKE** | 45.46% | -0.1 | 0.455 | 0.617 |
| **ORCL** | 72.73% | 0.4923 | 0.727 | 0.839 |
| **PFE** | 63.64% | 0.2414 | 0.656 | 0.688 |
| **PG** | 72.73% | 0.459 | 0.727 | 0.800 |
| **T** | 63.64% | 0.2903 | 0.642 | 0.732 |
| **TRV** | 63.64% | 0.2414 | 0.617 | 0.733 |
| **UNH** | 45.46% | -0.1 | 0.455 | 0.550 |
| **UTX** | 72.73% | 0.3774 | 0.717 | 0.875 |
| **V** | 81.82% | 0.6452 | 0.815 | 0.767 |
| **VZ** | 45.46% | -0.1 | 0.455 | 0.317 |
| **WFC** | 54.54% | 0.0678 | 0.538 | 0.467 |
| **WMT** | 36.36% | 0.0941 | 0.364 | 0.833 |
| **XOM** | 45.46% | 0.0294 | 0.436 | 0.661 |
| **Average** | **56.18%** | **0.125597** | **0.555846** | **0.625462** |

*Table 8 Each stock for the 42-stock result by itself*

# 6 Evaluation

Sentiment analysis method showed good results with the best F-Measure of 0.575 for the vote model. However, method 2 which used time mapping with targeting buy and sell key words showed a small increase with best F-measure of 0.578 using the IBK model.

Combining both methods did showed a better result with an accuracy of 58% and best of F-Measure of 0.580 with both IBK, and the cost sensitive model. Nevertheless, those models had stocks with only 5 days' worth of data, which can be misleading as if all those instances had a raise status than the model will always assume that this stock will always raise which is not true. So, using sentiment analysis also and reducing the dataset to only keep the stocks that has 100 instances or more results in best F-Measure of 0.579 which is slightly less than 0.580 got from method 3 but it's a better result because of the data trained example I used.

Having both keywords and hypernyms method and LSTM didn't show a much better result with best F-Measure of 0.557 and 0.561 accordingly

As method 4 be knowing the data and know that it has better mixing since it targets stocks with 100 instances and more which resulted in targeting 42 stock instead of 63, a 10 instance where taking out randomly from each stock to test the model later on with. Where it showed an accuracy score up to 90% for some stocks and an average accuracy of 56%, average F-Measure of 0.555 with an average AUC of 0.625 those shows that this method does better than random guessing.

# 7 Conclusion

In this work. Many approach been taken to test different stimulant for stock market prediction. Stock market price are affected by many factors that are hard to predict. Some are well predicted with a great accuracy as if the company announced big purchases or increase plane. However, most are not.

Any small detail can shift the investors mind and have them buy or sell the stock. Although sentiment analysis was proposed by previous work my method uses wide range of stocks on longer time period and proposed another method to use the sentiment with adding other relevant features.

Also unlike other previous work, I have tested many other methods using different approaches. Mapping approach where it maps the time of the tweets to three categories (Morning, Mid-day, and night) to test if any correlation arises between the time of the tweets and the stock market price. Another approach was to combine both methods together which showed better results than the bother two stand alone.

Although methods 1 to 3 had tested the full 63 stocks method 4 reduced the targeted stock to only 42 stocks. That was because I only kept the stocks that had 100 tweets collected or more. Based on the data collected this method can be ignored as it will be like method 1. However, it was necessary to reduce the dataset as such since some stocks had only 5-day worth of tweets and if all of them raised then we can say that this stock is always raising which is not a true statement.

Another complete different approach took place in model 2. Usually either studies focus on a specific stock or an index like DIJA. Where in this work a list of stocks been tested and compared with DIJA price by using the same methods used to build the data for model 1. Which helps to shows a comparison between tweets and news as the two-market stimulant.

A limitation in this work is that it doesn't incorporate stock prices that stayed the same as it's only concerned with a binary prediction of whether the stock rise or fall AS far as my knowledge the best accuracy showed for multiple stocks was 54.41% where my method showed an accuracy of 61%.

Also in this work a new framework name CPL-D (Collect, Process, and Learn Data) model where introduced that helps organize and make building and testing machine learning models much efficient and easier. This model helps the researches organize their work from the data collection phase to completely building the machine learning models and can be used to build an ongoing or automated task like the one used in reinforcements model.

# 8 References


- [1] Fama, E. F. (1991). Efficient capital markets: II. The Journal of Finance, 46(5), 1575–1617.Fama, E. F., Fisher, L., Jensen, M. C., & Roll, R. (1969). The adjustment of stock prices to new information. International Economic Review, 10(1), 1–21.
- [2] Walczak, S. (2001). An empirical analysis of data requirements for financial forecasting with neural networks. Journal of Management Information Systems, 17(4), 203–222.
- [3] Qian, B., & Rasheed, K. (2007). Stock market prediction with multiple classifiers. Applied Intelligence, 26(1), 25–33.
- [4] Schumaker, R. P., & Chen, H. (2009a). A quantitative stock prediction system based on financial news. Information Processing & Management, 45(5), 571–583.
- [5] Cervelló-Royo, R., Guijarro, F., & Michniuk, K. (2015). Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data. Expert Systems with Applications, 42(14), 5963–5975.
- [6] Mittal, A. and Goel, A. (2011). *Stock Prediction Using Twitter Sentiment Analysis*. [online] stanford.edu. Available at: http://cs229.stanford.edu/proj2011/GoelMittal-StockMarketPredictionUsingTwitterSentimentAnalysis.pdf [Accessed 9 May 2018].
- [7] Bharthi, S. and Geetha, A. (2017). *Sentiment Analysis for Effective Stock Market Prediction*. [online] researchgate. Available at: https://www.researchgate.net/publication/317214679 [Accessed 9 May 2018].
- [8] Nguyen, T., Shirai, K. and Velcin, J. (2015). Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications*, 42(24), pp.9603-9611.
- [9] Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. Journal of Machine Learning Research, 3, 993–1022.
- [10] Lin, C., & He, Y. (2009). Joint sentiment/topic model for sentiment analysis. In Proceedings of the 18th ACM conference on information and knowledge management (pp. 375–384). ACM.
- [11] Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. Journal of Computational Science, 2(1), 1–8.
- [12] Vu, T. T., Chang, S., Ha, Q. T., & Collier, N. (2012). An experiment in integrating sentiment features for tech stock prediction in Twitter. In 24th international conference on computational linguistics (pp. 23–38).
- [13] Baccianella, Stefano, et al. SENTIWORDNET 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. Istituto Di Scienza e Tecnologie Dell'Informazio, 2010, nmis.isti.cnr.it/sebastiani/Publications/LREC10.pdf.
- [14] Marjanovic, Boris. Huge Stock Market Dataset. Kaggle, 16 Nov. 2017, www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs.
- [15] Aaron7sun. Daily News for Stock Market Prediction. 25 Aug. 2016, www.kaggle.com/aaron7sun/stocknews.