Predicting the Stock Market has always been a huge challenge for both the academics and professionals in the firms. But that's why huge amount of people falls for it, and so do I

Daily Stock Price Prediction Based on News' Headlines Analysis

LFM and MLP

Michael Chen
Mic012@ucsd.edu

INTRODUCTION

Financial Market is extremely risky. Though throughout the years many people had built their economic theories and tried to explain the behaviors in the financial world, few of them can profit from their research and strategies they proposed. Overall, the most popular approaches of predicting the U.S equity market can be narrowed down to technique, fundamental, sentimental, and psychological analysis. While taking all angles analysis is believed to be necessary when trading and investing, this report only focuses sentimental analysis due to the short amount of time I had. However, if I can find features in the headlines of stocks' news that can affect the movements of the market in some degree, this can be significant in the way of helping people and myself better predict the market.

1. DATASET

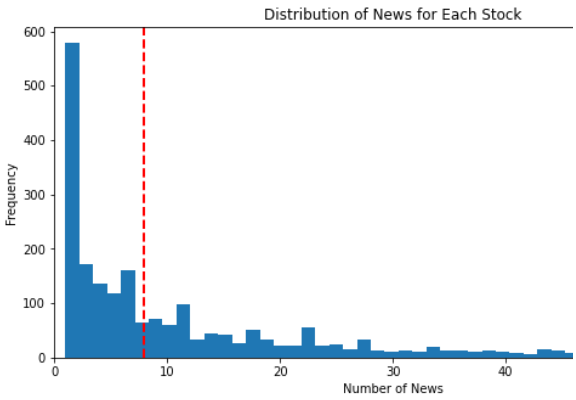
Before going into the detail, we can take a glance at the dataset first. The original dataset I used is collected from Kaggle. The first dataset [1] contains 215447 unique news with their titles, corresponding ticker, and specific dates of news publishment. The second dataset [2] is almost identical to the first one regarding to the columns' name, but it has 843062 unique entries, which is significantly larger. Since both datasets contain repetitive tickers, I concentrated them by tickers and dates. So, the entries now contain title of all news for a ticker on a specific date in a combined string. Then I filtered out dates with fewer than 3 news because too small number of words will likely generate less information and thus have weak effect in predicting the movement of a ticker. Also, fewer news means less volatility and attention on a ticker, making it less affected by public news and sentiment.

So, after those preprocessing, I am left with 75166 entries and 3865 unique stocks. To obtain the next day movement of tickers, I used python package '*pandas_datareader'* to scrape their price from 2009 to 2020 from Yahoo Finance and find the corresponding price change of the next date. For example, suppose I had the ticker 'NIO' and its news on 2020-01-15, I would use 2020-01-16's closing price and compare to 2020-01-15's closing price to determine if the price was going up, flat or down. Moreover, I discounted the price by 0.98 to include commission fee and bid-ask spread that appears in real life trading before appending to the dataset. So, if the stock went up the next day, the label would be 1, which means this trade was profitable, or if the price remained flat or declined, I place 0 instead, indicating a loss from the trade. On the other hand, I used closing price instead of adjusted closing price because I am only considering t1 (one day timeframe) trading opportunity, where the facial value is enough compared to embedding stock splits and dividend's effect on the prices.

After the works above. I then shifted my focus back to the explanatory variables themselves. So, the next step would be few basic text cleaning procedures, some of which include removing punctuations, digits, and numeric numbers, as well as words that contain them, so the text contains purely words with characters. Here I removed punctuations because they deliver minimum amount of information in title. Also, digits and numbers such as Earning Per Share (EPS), microeconomic statistics and other numbers can have unclear meanings since we don't know how they compared to the previous statistics, which is what matters more. For that reason, I would need extra features such as the earning surprise that's outside of textual data, so I decided to remove them for this task. Moreover, I filtered out tickers with fewer than 10 days of news because these companies are small capitals with rare

[1] https://www.kaggle.com/datasets/gennadiyr/us-equities-news-data
[2] https://www.kaggle.com/datasets/miguelaenlle/massive-stock-news-analysis-db-for-nlpbacktests

attention drawn and thus, their ticker price is relatively less predictable by the sentimental analysis. Also, huge amount of news was concentrated in the range 0 to 20 as shown below, so outliers was gotten rid of.



Distribution of days of news for the tickers
Mean: 24; Standard Deviation: 56
Red Line: Median: 8

Therefore, the final dataset I built up has 50977 entries, and based on different featuring and modeling, they will vary and be demonstrated later.

	date	stock	title	return	label
0	2009-07-27	BA	stocks futures flat honeywell aetna spark caut...	0.021628	1
1	2009-07-27	CDNS	verisilicon delivers chip designs time lower c...	0.012550	1
2	2009-07-27	HMC	global markets bulls control asia stocks exten...	-0.009636	0
3	2009-07-27	TGT	update hungary slashes rates signals more cuts...	0.006131	1
4	2009-07-28	CAJ	canon profit tumbles lifts forecast slightly u...	-0.006172	0

First 5 entries

2. PREDICTIVE TASK

As mentioned before, the predictive task I liked to focus was the next day movement of the ticker as a binary result: 1 represents profitable and 0 represents non profitable of buying that stock on that day and holding for one day. So, in terms of evaluation of my models, I chose accuracy over other metrics because all TP, FP, TN, and FN [3] matter in real life trading. The test will be held up on the validation sets, which remains 20 percent of the entire entries consistently among all models.

On the other hand, the features used for prediction are based on the textual news headlines, which would be transformed into numbers before fed to models. There are three ways of feature engineering in my experiments: Bag of Words, TF-IDF [4] and Tokenizing.

For bag of words, there are Unigram and Bigram. I used the top 2000 popular words among all news' headlines for Unigram and 1000 for Bigram.

Then for TF-IDF, which also split into Uni- and Bigram, I believed it would capture those words that uniquely appeared in one category and thus would possess indication of its predicting labels.

Lastly, Tokenizing refers to mapping each unique words to an integer. This is only used for Long short-term memory because LSTM looks for correlation between each word in a sentence, so I had to directly tokenize words into digits to remain the original sentence structure.

For the first two methods, I cast all words into lowercase and their stem ahead to avoid redundant information. Also stop words were removed in Unigram because they barely exist in titles, while not delivering actual contextual meanings by themselves. However, I kept them when using Bigram and tokenizing because some stop words might play a big role in a sentence or a phrase.

Top 200 Unigram Words

[3] TP, TN, FP, FN: True Positive, True Negative, False Positive, False Negative
[4] Term Frequency-Inverse Document Frequency

When it comes to the models, few I had tried included Logistic Regression, Multilayer perceptron (MLP), Latent Factor Model (LFM) and Long short-term memory (LSTM). The logics and evaluation behind these models will be explained next sector, but we should look at the baseline model first.

- Random Guessing, All 1's and All 0's

  One of the main points of building complex machine learning and deep learning models is that hopefully they can learn certain patterns behind the numbers. If models outperformed the innocent pure guessing, they could be tuned and retrained for usage in the future. Thus, I randomly generated 0's and 1's for over 100 times as the baseline model and examined its performance over validation set.

```
Random Guess:  0.4943
All 1:  0.4589
All 0:  0.5411
```

  As we can see, the baseline of All 0's achieved an accuracy score of 0.5411, which is the highest among three. Therefore, I should aim for constructing features and building models that can capture majority of non-profitable days and hopefully a few profitable days so we can learn some correlation between the news' headlines and stocks' price movement.

3. MODELS

- Logistic Regression

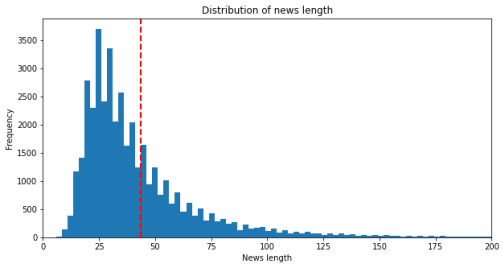
  Logistic regression, one of the most popular Machine Learning classifiers from *sklearn* [5], models the probability of an event taking place. Compared to other models such as Random Forest, SVM [6], and Naïve Bayes, Logistic regression outperformed the others on this set of training and validation data.

  Specifically, there are four ways of featuring and the performance on the validation set are shown below.

	Accuracy
Bag of Words (Unigram)	0.5278
Bag of Words (Bigram)	0.5231
TF-IDF (Unigram)	**0.5340**
TF-IDF (Bigram)	0.5225

  The combination of TF-IDF and Unigram achieves higher accuracy than other featuring. Therefore, I would use this for MLP to dig further into the patterns. Before that, I tried LSTM, which didn't do well since the titles are generally too short for long term memory mechanism of the model. Instead, a user review or poem etc. might be a better place for implementation of recurrent neural networks.

- Long short-term memory (LSTM)
  As demonstrated before, each word was mapped to a unique number. News with more than 5 words and less than 100 words are kept so I removed the outliers from deviating the model.



Distribution of news' lengths
Red Line: Mean: 44
Standard Deviation: 36

[5] scikit-learn
[6] support vector machine

Then I padded feature to fixed length of 120 words since LSTM requires constant shape of data throughout training. So, short news was filled with 0, which represents unseen words, and long news were cut to length of 100, and the feature would have a shape of (38824, 100), where there were 38824 entries with vector size of 100 containing the digits representation of each word. Lastly, I imported the *keras* package, which will be used for MLP later as well, to construct the LSTM model. The full model consists with an embedding layer at the top, the actual LSTM layer with 1024 hidden units and two dense layers with 128 and 64 units at the end. The model stops with early stopping as the validation accuracy wouldn't surpass the TF-IDF Unigram model. Therefore, I turned back to MLP and decided to expand upon that model for better performance.

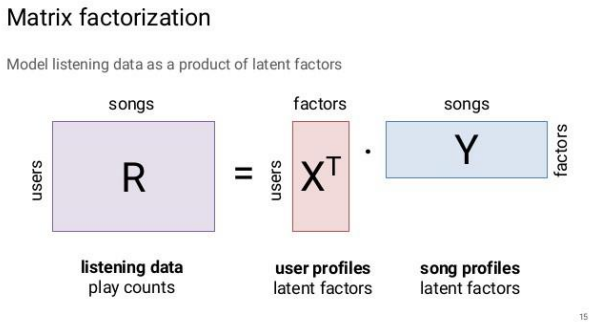
- MLP

MLP, or multilayer perceptron, is a fully connected layer that can find hidden linear relationship between each feature, and then link these correlations from the input to the output. The motive behind using MLP is that there could be more message included in the headlines that a single layer can't catch, just like how we have different opinions on news due to our various brain structure. So, picking up with the previous features, I used TF-IDF to transform words into its processed value, and the training set has shape of (38824, 2000), which means 38824 entries and 2000 numbers that represents the counts of the top 2000 words in given news.

Then, I proceed to constructing the layer with *keras* package. There are total of three layers with hidden units of 512, 128 and 64 each. Each layer follows a dropout layer, which drops 50 percent of the data, to prevent overfitting. The first two layers used *relu* activation and the last layer would transform 64 length vectors to a single probability outcome with sigmoid activation. High probability turns into 1 and low probability turns into 0. The model is trained with 50 epochs and batch size of 32, where early stopping with patience 3 is also implemented so the model would stop training after three rounds where validation accuracy doesn't go up. The default learning rate of the weights is 0.001, loss is *binary_crossentropy* and the optimizer is *rmsprop* [7]. The training stops at the 6th epoch with validation accuracy of 0.5415, which is just a tiny bit better than All 0's baseline model.

- Final Model: MLP + LFM

To further dig into the possible hidden patterns in the news title. I added a Latent Factor model as feature, so the feature size was expanded to 2001 length vectors. The advantage of embedding this additional model is that Latent Factor Model can decompose the given matrix into two matrixes with one dimension represents the hidden feature that we couldn't tell at the first glance.



[8]

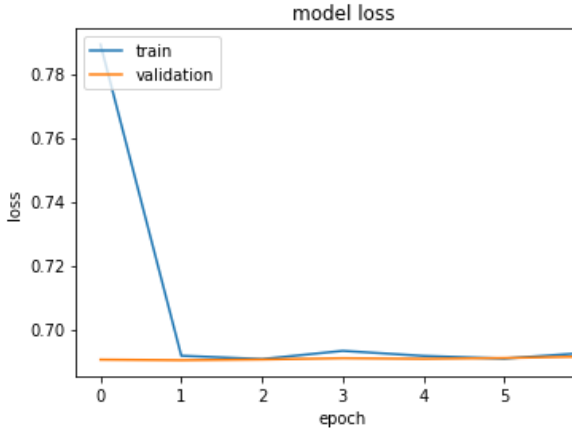
[7] Root Mean Squared Propagation

[8] https://towardsdatascience.com/introduction-to-latent-matrix-factorization-recommender-systems-8dfc63b94875

```
iteration 10, objective = 0.12442562
iteration 20, objective = 0.118881755
iteration 30, objective = 0.102648094
iteration 40, objective = 0.07973423
iteration 50, objective = 0.060773913
iteration 60, objective = 0.048684075
iteration 70, objective = 0.03986988
iteration 80, objective = 0.034281448
iteration 90, objective = 0.03033993
iteration 100, objective = 0.027317949
```

LFM Training

After appending the predicting values of Latent factor models, the MLP slightly improves from the previous results with accuracy score of 0.5460.



Losses of MLP + LFM

Epoch	Validation Accuracy
1	0.5363
2	0.5363
3	**0.5460**
4	0.5328
5	0.5329
6	0.5295

Accuracy on each epoch

4. LITERATURE

In the dataset section, I already described that the datasets are public on Kaggle, and I downloaded them into csv file and read them into *pandas DataFrame* for analysis. As a passion student into trading and majoring in data science. I also studied lots of other datasets in the past. However, most of them are related to technique and financial statistics such as the volume, opening price and closing price of a stock, or three financial statements statistics scrapped from SEC official website. Therefore, there really aren't any similar datasets I have studied as this is my first attempt in doing sentimental analysis in the market. However, a few intelligent professionals had posted their research online and I was surely motivated from their papers.

The first paper "On the importance of text analysis for stock price prediction." by Lee et al [9]. is one of the most popular papers in predicting a stock with textual information and achieving significant improvement. Also, the paper "Using news titles and financial features to predict intraday movements of the DJIA" by Arora et al [10] expands on how deep learning is used to analyze financial news, and how it can be combined with technical indicators to achieve greater predictability of the market.

Some similarities of these existing works with this paper are that textual analysis can have positive effect in predicting stocks movement, but the effect is limited. Also, these predictions can only be conducted on short periods of time as the effect of news would fade away in longer time frame.

On the other hand, this paper is only limited to using textual information to predict the movement, while the other two and most existing works already combined other statistics as they are better reflection of the real financial markets, where the movement can be attributed to fundamental, technical, political and many other factors. Therefore, the result of this paper cannot and does not

[9] https://nlp.stanford.edu/pubs/lrec2014-stock.pdf
[10] https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/custom/15744462.pdf

inform any trading strategy or any market decision and advice, as it's not sufficient for implementation in real life trading and investing if one wants to make a profit.

5. RESULTS AND CONCLUSION

Recalled that one of the baseline models, All 0's, did achieve an accuracy of 0.5411 on the validation set I set up. With all other models I implemented, only the MLP model performs better than the baseline. Combining with the LFM, the model achieves the highest accuracy among all (0.5460). Specific parameters such as hidden size and number of layers, and activation functions are tuned to generate such accuracy. Some intuition behind the parameters might be the for example, the first layer of MLP has hidden units of 512 because the input dimension is 2001, which makes it neither too large nor too small.

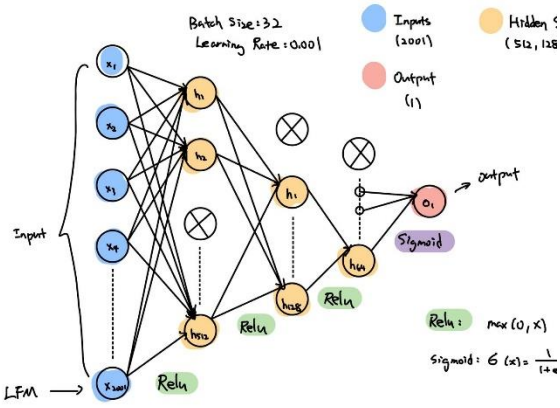


Image Interpretation of the model and its parameters; More details are also included in the final model sector in section 3.

However, 0.5460 compared to 0.5411 shows insignificant improvement as only a few positive labels have been captured. Despite that, TF-IDF played an important role in distinguishing positive and negative news. It outperformed Bag of Words featuring in the same model, which did not work well because most popular words such as 'stock', 'market', 'earn', 'share' etc. are too neutral for prediction. Moreover, the hidden feature of decomposition of LFM also improved models' accuracy from simple MLP. These features are one of the major reasons why the final models worked relatively.

So, for the other models' failure, I had discussed their reasons under their sectors. But if for some broad reasons, I can think of lacking data, too many neutral words in the titles, and the unpredictable characteristic of the market in general.

Despite that, models wouldn't work on every dataset, and that's just a pure fact. Therefore, innovative studies are published every day by intelligent scholars with same passion and fields of studies. This paper might be insignificant in way of predicting the market, but it is sufficient for me to decide to devote myself into studying Natural Language Processing, Machine Learning and Deep learning in the field of financial market. So, this is an end, but also a start to something much greater.