Stock Prediction using Sentiment Analysis

Zeu Sul
Danver Zhao
Sourajit Bhattacharyya

Table of Contents

1. Abstract

Sentiment analysis, widely used in natural language processing (NLP), plays a key role in identifying emotions that are embedded in textual data. This in turn can be effective in providing additional analysis that differs from currently available stock prediction methodologies. Our project attempts to apply semantic analysis to social media posts to predict ups and downs of two stocks; Apple and Tesla. We will focus on comparing the performance of our two different fine tuned models.

2. Data Preprocessing

Steps taken for preprocessing tweets -

- Hashtags removed ie #Exchange → Exchange. This is to help with tokenization.
- Url is replaced with [URL] symbol. Ie https://googe.com → [URL]. Urls do not provide any relevant information besides the fact that the author has an agenda.
- Repeated punctuation is removed. Ie What!!! → What!
- Emojis are replaced with their plaintext version. Ie 🧺 → :basket:

We have collected data from multiple sources:

- Historical stock price movements - yfinance package in python.
- Fine tuning dataset - huggingface zeroshot/twitter-financial-news-sentiment.
- Recent twitter data - twikit package in python, we want to use more recent data because we want to make sure that the data we get doesn't accidentally gets leaked into the fine-tuning dataset, however ever since Elon musk's acquirement of Twitter, the rules around API scrapping becomes strict and all previous available method have stopped working. Using this method we were only able to get 18 tweets per day at a time and around 600 tweets limit a day.
- Historical twitter data over 10 years - stocktwits dataset

3. Fine Tuning

There were few available choices from already pre trained models that predicted stock movements. However, these models did not make use of semantic analysis and instead made use of already existing stock prediction methods, technical analysis (make use of previous datasets of the stock's price movement). Therefore fine tuning already existing models to achieve this projects' goal was seen as the most effective and practical solution.

3.1.1 Model (finBERT)

Prior to fine tuning a model, we must first identify the most advantageous model for predicting the stock. Main focus in choosing a model for fine tuning was mainly on its speciality in financial languages. FinBERT is a pretrained model with its root in BERT models, which specialises in analysing financial vocabularies, and hence reduces the need for extra training in evaluating industry-specific terms and understanding financial concepts.

3.1.2 SETUP

Training this model requires the Transformer library from Pytorch. There are few requirements before attempting to run the notebook shown in **figure 3.1.2.1.**

```
!pip install datasets
!pip install scikit-learn
!pip install transformers
!pip install accelerate -U
```

3.1.2.1 Code snippet of library installation

```
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)

training_args = TrainingArguments(output_dir="test_trainer")

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=small_train_dataset,
    eval_dataset=small_eval_dataset,
    compute_metrics=compute_metrics,
)

trainer.train()
```

3.1.2.2 Code snippet of fine tuning function

3.2.1 Model (BERT)

Getting the BERT classification model and tokenizer with the name - bert-base-uncased

```
# Load the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
```

```
# Load the pre-trained BERT model
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=3)
```

3.2.2 SETUP

After fine tuning, a big challenge is how to map the output of the model to stock price movement. We have tried 3 different ways:

Firstly, we simply summed the values of each class over all texts, then we can get the highest values class. This method turned out to be not effective, as neural based models tend to increase the distance of 2 classes therefore any label beside the highest value one may be unfairly skewed.

Secondly, we tried to count the number of texts that showed that type of emotion. However this method is also flawed as just tweets in general tend to be neutral or positive, this means our model is predicting neutral everytime.

Lastly, we combined the previous two methods to develop averaging total sentiments, but this time we only add the highest value to the total and ignoring the values of the other class, this way the negative values don't impact the total score as much because of class imbalance and at the end we will be left with the average certainty / power of that emotion / class. And it turned out to be a working method when paired with thresholding (if the highest value is x amount higher than the second highest value).

4. Evaluation

4.1 Quantitative Evaluation

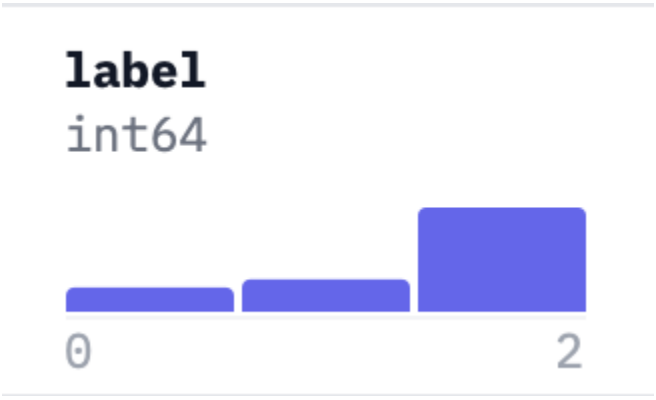
The rule-based model - vaderSentiment achieved a F1 score of 0.38 and Accuracy score of 0.44. We use F1 as it takes into account distribution of data, which in our case is unbalanced.

4.1.1 BERT

After fine tuning BERT using the dataset from hugging face, the accuracy and F1 ran on the testing set is around 0.86 which is around the level of previous papers.

```
Accuracy: 0.8731
Precision: 0.8342
Recall: 0.8337
F1 Score: 0.8339
```

However, when we tested it against a different set of data we found the model to be generating neutral class only. This could be because of class imbalance in the dataset as the distribution of the dataset is exactly like the the output distribution.



(dataset distribution from left to right - 16% Negative, 20% Positive, 64% Neutral)

Another reason could be that the model is classifying correctly but the data given is biased because people usually tend to post about good things rather then bad/negative things. Therefore we need to explore different ways of predicting stock movements and were able to achieve the following F1 and Accuracy score with thresholding which is higher than random guessing.

Threshold	F1	Accuracy
0	0.51	0.51
0.05	0.52	0.53
0.1	0.67	0.7
0.15	0.58	0.6
0.2	0.66	0.66
0.25	0.66	0.66

4.1.2 finBERT

finBERT reflected its strength in the financial domain, as an extended version of its predecessor BERT in forms of three stats: accuracy, precision and f1 score seen in **figure 4.1.2.1.**

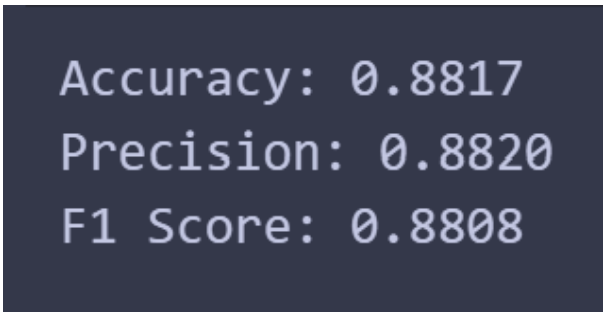


Figure 4.1.2.1 results of the model

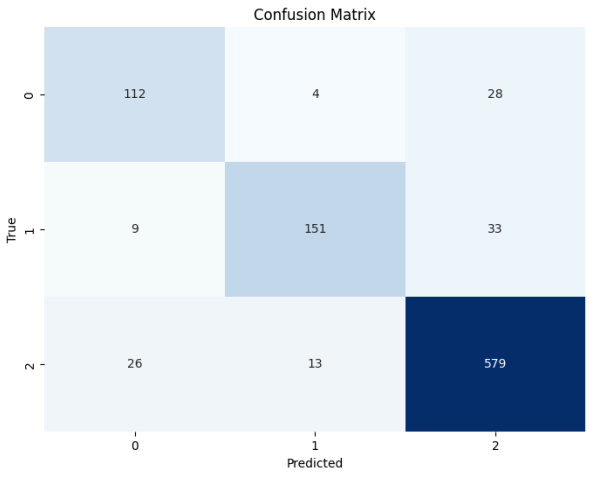


Figure 4.1.2.2 confusion matrix of the model

Figure 4.1.2.2 indicates the model's capability in recognising increasing stock market trends as opposed to the other two changes. However, most of the values are still concentrated in correct prediction.

4.2 Final Stock Prediction

In order to check the efficacy of our sentiment prediction in actually predicting stock market trends, we used a Random Forest (RF) model to predict whether the stock would rise or fall the next day. This was done with Apple tweets and stocks

For the first RF model, we had as features-

- The stock price features ie Open, High, Low, Close, Volume for the current trading day.
- The historical Close prices of the stock of n previous days. This n is our lookback time. The optimal lookback time was determined to be around 10 days.

For the second RF model, we additionally had as features -

- The sum of the sentiment values of all tweets on the current trading day. A negative sentiment reduces the sum, indicating the stock might go down, and vice versa for positive sentiment. The sentiment analysis was done using our better performing finBERT model.

The features were all scaled linearly between 0 and 1, and the target was predicted with both models. The target is a boolean value of whether the stock will rise or fall the coming trading day.

The first RF model without any tweet sentiment data produced an F1 score of 52.5%, whereas the second model with predicted tweet sentiment data produced an F1 score of 53.6%. This indicates that our sentiment predictor does indeed have a positive effect in determining stock price changes.

RF Model 1 - No sentiment data

RF Model 2 - Sentiment data using finBERT

5. Group Scope

PART A	**Minimum: 10 credits**	**10 credits**
NLP (Natural Language Processing) Problem	5 credits per problem	Sentiment Analysis (5)
Text Source/Domain	5 credits per text source/ domains	Social Media Post (5)
PART B	**Minimum: 20 credits**	**20 credits**
Use one existing dataset	10 credits per dataset	huggingface (10) Stocktwits (10)
Create your own labelled dataset	20 credits	
Use an existing lexicon (for evaluation)	10 credits	
PART C	**Minimum: 30 credits**	**40 credits**
Implement a rule-based or statistical model as a baseline	Compulsory	Rule-based model (vaderSentiment)
Use an existing pre trained/fine tuned model	5 credits per model You must compare the performance of models if you are only using fine-tuned models	
Fine-tune a model based on a dataset	20 credits per out-of-the-box fine-tuning method	Fine-tune a finBERT model (20) Fine-tune a BERT model (20)
Extend a finetuning method	30 credits per extended finetuning method	
Integrate a language model with external tools	20 credits	
PART D	**Minimum: 20 credits**	**25 credits**
Quantitative Evaluation	10 credits	F1 and Accuracy (10)

Qualitative Evaluation	5 credits	Explanation of strength and weakness of models (5)
Command line testing	5 credits	
Demo	10 credits	Demo(10)