Sentiment Analysis On Financial News Headlines

Jeremy Perez
Union College
perezj@union.edu

1 Introduction

In this project, I alongside a group of two others, made it our goal to perform sentiment analysis on news headlines pertaining to specific stocks and see if it was possible to make any inferences from this information. With the sentiments found, we wanted to try and predict whether a stock should be bought or sold on that day. We're hoping to learn more about the different ways to apply BERT models and investigate what it looks like to build an application that predicts real-world movements based on our analysis. The task of predicting share prices is ultimately much more complex than the project itself, however, we focus on extracting the non-quantifiable information within the news headlines and seeing what their implications are (if at all) to the real markets.

2 Prior Work

Initially, we will begin by following the research paper by Kalyani, Bharathi, and Jyothi: Stock Trend Prediction Using News Sentiment Analysis. Their results were fairly successful with their best test cases ranging from 88% to 92% accuracy. We'll be building on their work by using a much bigger corpus ranging from 2009 to 2020 and seeing if their model can upscale. On top of this, we'll be branching off the way they calculate sentiment in the headlines by using FinBERT. The paper calculates sentiment in a crude way, using a bag of words method, as well as counting the number of positive words and subtracting the number of negative words in an article. We believe FinBERT can do a better job at calculating the sentiment score.

3 Data & Software

As stated before, our experiment will largely depend on the Kaggle dataset, which provides over 4 million financial news headlines for over 6000 stocks. From this, we'll work alongside the implementation of a FinBERT model to assign a score to the dataset. After this, we'll be building classifiers using sci-kit learn's library to attempt to predict whether a given stock should be bought or sold. We'll be saving our models and any corresponding corpora used on Github, serving as our central repository.

4 BERT Models

BERT (Bidirectional Encoder Representations from Transformers) are language models which were designed to understand the context of words in sentences, however, we're using them in our case for sentiment analysis but overall can be applied to other natural languages processing tasks like language translation and question answering. One key feature of the BERT model is the bidirectionally, which allows it to consider proceeding and following words in a sentence. This is great for sentiment analysis simply because unlike typical language models, which just look at words after a given word, it attempts to use the context from all around the word to best understand how it was used.

5 SVM Classifier

SVM or support vector machines are linear models which can be trained to classify data. The models we use are from Scikit-learn which works by turning data points into vectors and then observing lower dimensional vectors to separate these points and categorize them in classes.

6 Experiment Procedure

First, we will begin by downloading the FinBERT model and tokenizer. We then used scripts to parse the article corpus to extract headlines corresponding the stock we choose to look at and the date they were released. In parallel we collected the opening and closing prices for this stock on those

Model	Accuracy(%)	Precision(%)	Recall(%)	F-Score	Train Headlines	Train:Test
SVM	54.35	89.19	53.88	67.18	1573	.5:.5
SVM	48.97	83.89	47.80	60.90	2517	.8:.2
Naive Bayes	56.67	84.48	56.79	67.92	1573	.5:.5
Naive Bayes	51.42	77.07	50.84	61.26	2517	.8:.2
Random Forest	48.82	48.82	48.82	48.82	1573	.5:.5
Random Forest	47.85	47.85	47.85	47.85	2517	.8:.2

Table 1: SVM and Naive Bayes Classifier's trained on Sentiments from headlines involving NVidia using different ratio partitions between training and testing

corresponding days. From here, we fit our classifier mode with the sentiment score for the headline (X values), as well as the change in stock price (Y values). We then do a standard evaluation on these models, measuring the accuracy, precision, recall, and f-score. We hope to compare different classifiers, such as Naïve Bayes and SVM, as the research paper will be observing these comparisons.

7 Results

Unsurprisingly, our results did not correspond to what we anticipated. Observing Table 1, we can see how results really weren't that successful. Both models F-Scores generally averaged around 60-65% which was far from the 80% achieved by (Kalyani, Bharathi, Jyothi).

Aside from Table 1, other stocks were evaluated as well within the corpus. It seemed like the more headlines a stock had within the corpus the worse it performed. This could be seen between the differences in Train/Test ratio column actually causing worse F-Scores for a bigger training ratio. This must be due to a fault within our code. Running and training these models took a substantial amount of time.

From just these simple results our best models were able to achieve 67.18% but realistically these are inconclusive results simply because the data was likely to spares to evaluate sentiment analysis with stocks

8 Discussion

Despite the given results, our models weren't really making the same predictions from those collected in there paper simply because we're just taking predicted sentiments, individually, and classifying them to the performance of the company which isn't really useful on it's own(at least in our task).

Take this example: *A news article was written casting a company in a bad light. On that same day another casted the same company in a positive light.*

Using this example, it was probably a much better idea to have a model combine the overall sentiments on the day collected and from there compare it to the performance of the company.

Another big fault is the sparseness of the data. Despite having over 4 million articles when individually evaluating each company, the data distributed over time gets really scattered and sparse only making it useful for making the classifications just on individual day which eliminates the possibility of future enhancements using previous dates.

A possible way to overcome this data spareness is by training the classifier on all headlines at our disposal, which pertain to all stocks and then testing on just a specific instance and stock. This in junction with taking into account the other sentiments during the day would surely improve the score.

This comes with a cost which wasn't really discussed much throughout this paper which involves the expensiveness of running and training successful models. For the majority of the testing, we had to allocate a significant amount time towards fine-tuning a way to get results in a timely matter without making it unreasonable! We eventually settled on a system where sentiments are calculated in chunks (mini-partitions of the corpus) instead of attempting to calculated them all at once.

Our program can be further improved also, including much more features which will give the classifier much more context which would allow them to work better.

9 Reflection and Self-assessment

Self-evaluating, I enjoyed being a part of this project! I worked on a little bit of most things and overall, I feel like producing applications isn't all too hard if you just sit down and do a bit of

planning. This project, in my opinion, felt spontaneous and random at first because we could pretty much do whatever we want but eventually, after committing, we had a little MVP(Minimum Viable Product) that we could continue to use and build off of if we wanted to.

If I were to say the things I contributed, I helped out with finalizing the idea, researched ways to implement it, integrated some of the previous implementations, and did a chunk of the documentation. The others in the project did about the same work with one of them writing more of the code and the other ensuring we didn't over complicate the goals and timing of the project. I would say this was a much different project than any of my past projects in CS, simply because we had to work together to accomplish one goal and because of this it felt like we were forced to put our best foot forward. This meant creating and writing maintainable code which is properly understood and well documented. We also made sure to not clutter our repository by ensuring to make as much of our code reusable.

If I were to do this project again, I actually would want to pursue and attempt to answer more questions. I feel like we ended up falling a bit short on the amount of work we all could've handled. The group generally was composed of people with a relatively good understanding of the course material which means we could have left a bit more leeway for a bigger project. Besides this, we handled everything pretty well, we set our goals and just went for it.

10 Conclusion and Future Work

After observing and creating semi-successful results from our models, we can see how these models can surely be enhanced . Considering this was a project for Natural Language Processing, the potential of this project bounded to this topic of processing text so we couldn't look into ways of enhancing this application to make better predictions potentially using more statistical and mathematical approaches. There are many possible approaches to this project, like = implementing a much better neural network instead of the one classifier we had implemented, but probably the best question to investigate following this is

11 Refrences

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Kalyani, J., Bharathi, H. N., and Jyothi, R., "Stock trend prediction using news sentiment analysis", arXiv e-prints, 2016. doi:10.48550/arXiv.1607.01958.