Korea Advanced Institute of Science and Technology

Introduction to Financial Engineering

Stock Price Prediction with Sentiment Analysis

Professor:
Woo Chang Kim

Student:
Federico Berto

Course ID:
IE471

ID number:
20204817

Introduction

The goal of this report is to describe the experimental process and result for predicting stock price with the help of *sentiment analysis* based on Twitter data. In particular, in the first experiment we will predict Tesla, Inc adjusted closing stock price in the last quarter of 2020 [1] using data from the first three quarters.
Let us briefly introduce the models we will use in the code.

Recurrent Neural Networks

Recurrent Neural Networks (RNN) [6] are capable of dealing with sequences of data. As Figure 1 [2] shows, they form an unrolled, sequential undirected graph in which input data are fed sequentially. The vanilla implementation's most notable problem

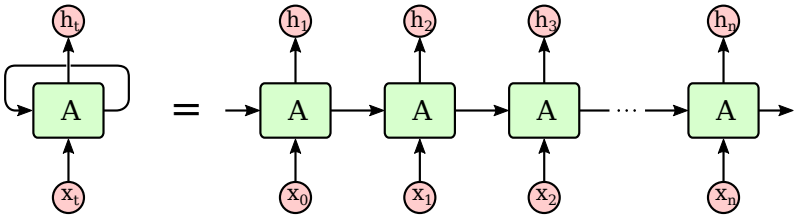

Figure 1: Recurrent Neural Network (RNN) base model

is the so-called *vanishing gradients* problem, in which long time series cannot be dealt with due the parameter updates using gradients; due to the sequential nature of RNNs, the further inputs in the time series are subject to more activation functions, causing updates to be almost independent from them.
For this reason, more advanced versions of Recurrent Neural Networks i.e GRU and LSTM have been created. Nonetheless, the simplicity of RNNs makes them suitable for shorter and less complex time series, since they are faster and also less prone to overfitting due to less parameters.

Gated Recurrent Units

Gated Recurrent Units (GRU) [1] were proposed as an RNN variant in 2014: they are similar to LSTMs without an output gate and has fewer parameters. Figure 2 [3] shows an overview of the architecture. Having fewer parameters than LSTM , GRUs have been shown to perform better on certain dataset and have better generalization capabilities with fewer data.

[1] Source: Yahoo Finance
[2] Source: Github
[3] Source: Wikimedia

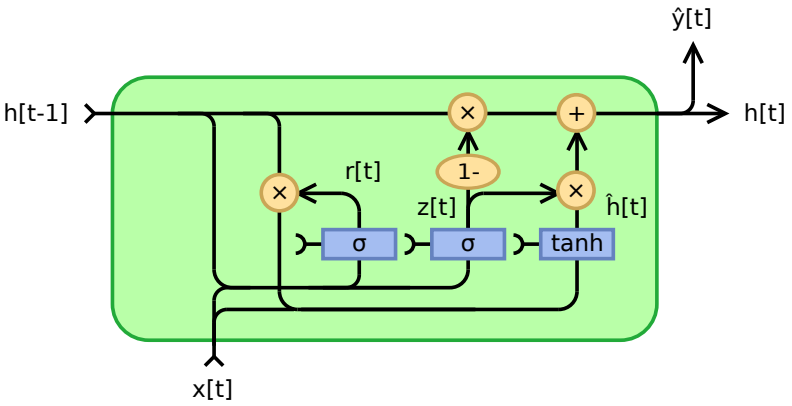

Figure 2: Gated Recurrent Unit (GRU) base model

Long Short-Term Memory

The prediction model is based on the Long Short-Term Memory (LSTM) [3] module in Figure 3[4], which is able to store past information of the data and is thus suitable for time series prediction.

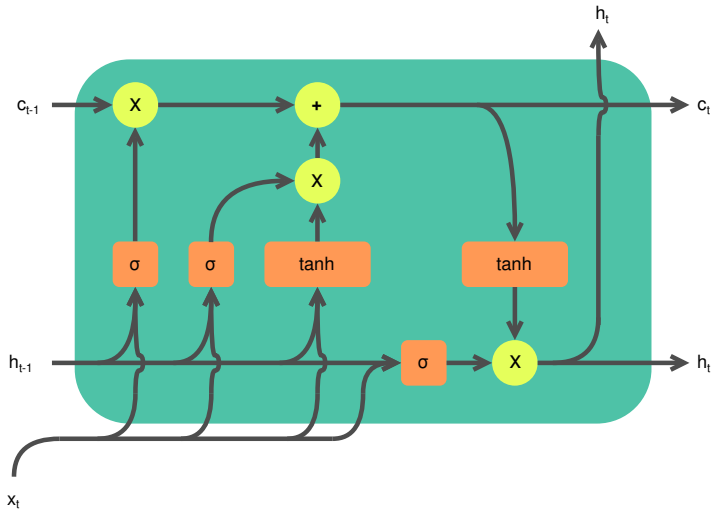

Figure 3: LSTM cell

Further details into the PyTorch [5] implementation can be found in the code, also available on the following Github repository.[5]

[4]Source: Wikimedia
[5]Link: `https://github.com/Juju-botu/financial-engineering-ai`.

Sentiment Analysis with Twitter Data

We use sentiment analysis [4] in order to get better predictions on the original dataset: our goal is to capture public *trends* via Twitter data scraping.

Figure 4: Word cloud of most popular words related to Tesla on Twitter

We use natural language processing with TextBlob [6] to capture these trends and convert them to usable numbers.

Experiments

Predicting Tesla's Stock Price

In this section we provide the required data for the report.

1. Mean Squared Error values for the whole dataset, training data and test data without sentiment analysis:

	MSE
Whole dataset	1312.9493
Training Data	39.4943
Test Data	4995.5659

2. Word cloud image of cleaned tweets: Figure 5.

3. Bar chart of 20 most frequent words of cleaned tweets: Figure 6.

[6] Source: Github

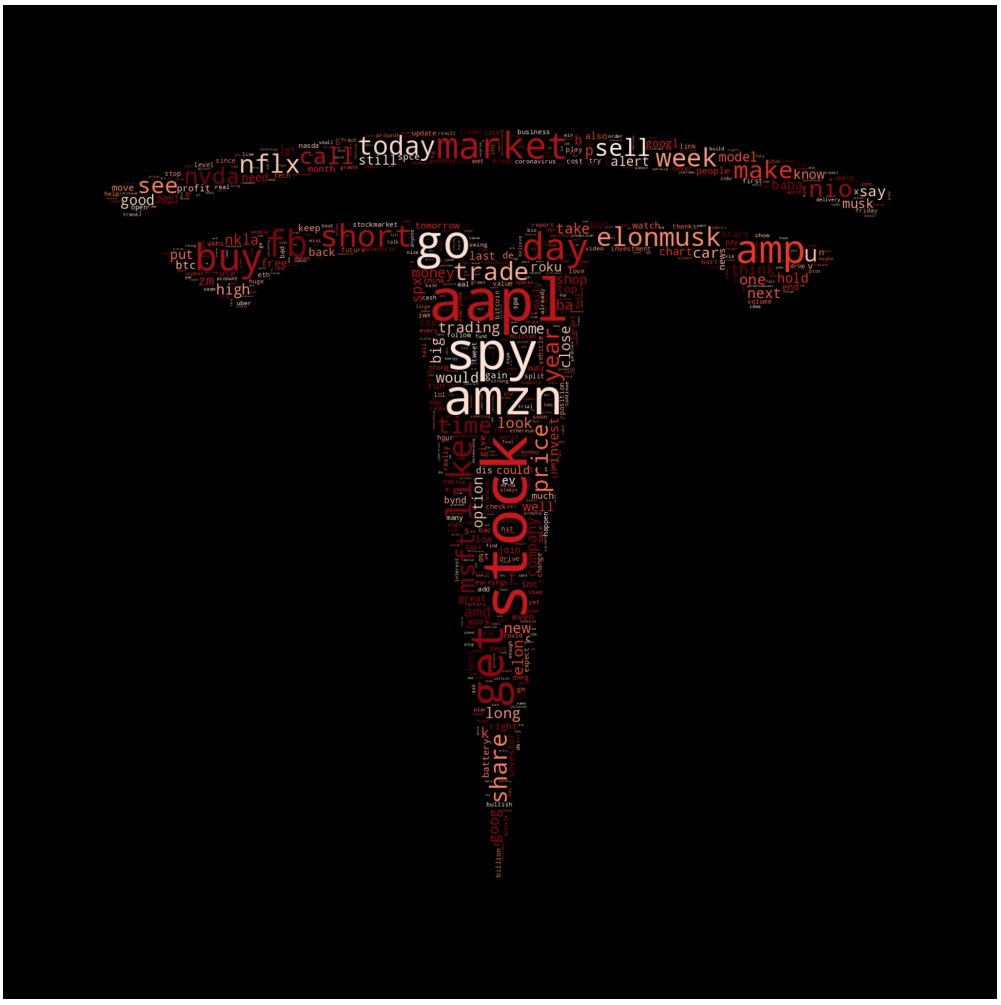

Figure 5: Word cloud of most popular words related to Tesla on Twitter

4. ID and sentiment scores of first five cleaned tweets:

ID	Sentiment Score
1212450794705969152	0.15
1212450579634626560	-0.20
1212450337543602177	0.20
1212450309131227141	0.00
1212449703318753280	0.75

5. Average sentiment score before removing 0 and after, January 1^{st} to 5^{th}:

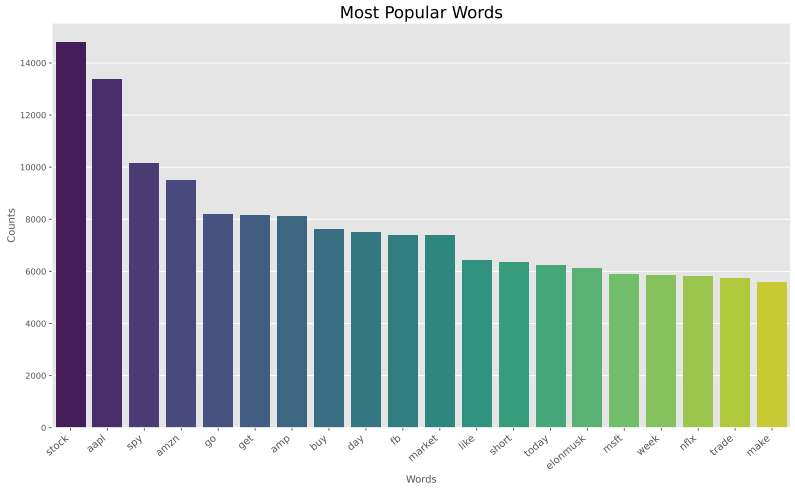

Figure 6: Bar chart most popular words related to Tesla on Twitter

Date	Sentiment	Sentiment Final
2020-01-01	0.099938	0.154961
2020-01-02	0.111398	0.161528
2020-01-03	0.073760	0.128297
2020-01-04	0.083606	0.144796
2020-01-05	0.086658	0.132267

6. Mean Squared Error values for the whole dataset, training data and test data *with* sentiment analysis:

	MSE
Whole dataset	669.5747
Training Data	42.2973
Test Data	2483.8538

7. *Comparison of the results before and after sentiment analysis*: we compare in Figure 7 the results of LSTM with and without sentiment analysis.

	MSE - Without Sentiment	MSE - With Sentiment
Whole dataset	1312.9493	669.5747
Training Data	39.4943	42.2973
Test Data	4995.5659	2483.8538

As we can see, sentiment analysis is able to help with predictions and was able to reduce by half the MSE in the testing region. However, it should be noted that due to the limited size of the dataset, the training results have a high variance; nonetheless,

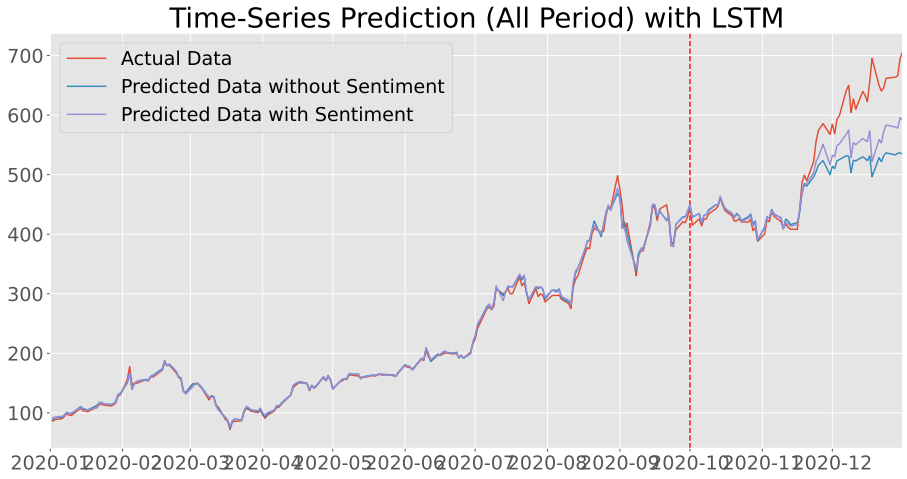

Figure 7: Comparison of LSTM's predictions without and with sentiment analysis

we noticed an improvement on the prediction cababilities of the network even across multiple experiments.

Improving the Predictions

In this section, we show how to improve the algorithm in three ways:

1. Revising the codes with Pytorch Lightning
2. Using a different AI algorithm, namely Recurrent Neural Networks (RNN) and Gated Recurrent Unit (GRU)
3. Improving the MSE error results

Using Pytorch Lightning

We revise the code by using Pytorch Lightning [2]: this PyTorch framework provides a high-level interface by which it is easier to control architectures, results, logging and more. More importantly, it makes the process of moving models to GPU, Tensor Processing Units (TPUs) and even multiple GPUs and TPUs easier while having a negligible overhead. This open-source library is actively maintained by a community highly focused on efficiency and code readability. We refactor the code to be used with Pytorch Lightning.

Results Comparison

We show in Figure 8 a comparison of the algorithms run with the same parameters with Pytorch Lightning. We also report the MSE table for the different approaches:

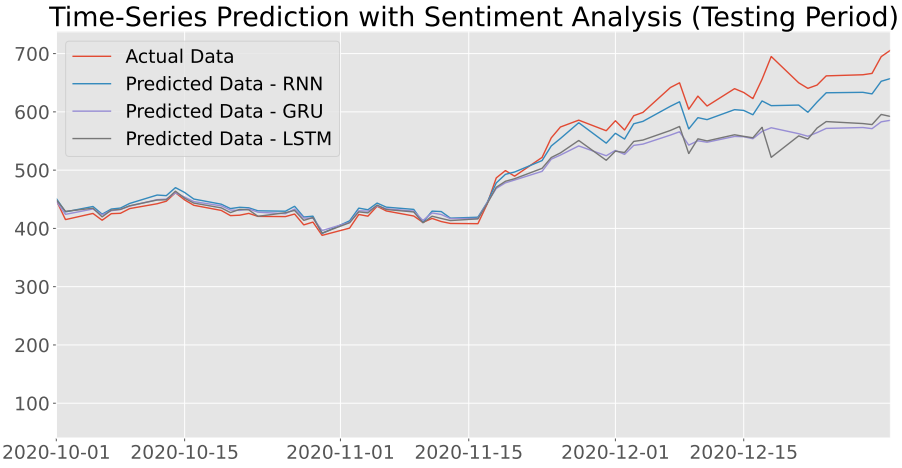

Figure 8: Comparison of different algorithms on Tesla's stock price prediction with sentiment analysis

Mean Squared Error			
	RNN	**GRU**	**LSTM**
Whole Dataset	**156.5683**	657.3547	669.5747
Training Data	29.5203	**26.7311**	42.2973
Test Data	**524.0301**	2481.3120	2483.8538

The results show that, while GRU and LSTM performed similarly on the training data, RNN was able to outperform them on the test set although it has lower complexity. One reason for this behavior could be that, since the test data ranges with previously unseen values, RNN is able to *extrapolate* better and is less prone to overfitting on this particular dataset. Moreover, since we chose the sequence of data as 5, the vanishing gradients problem was not encountered.
This experiment demonstrates that on certain, simple enough datasets less advanced models such as RNN can be useful and even outperform more complicated ones.

Predicting Nvidia's Stock price

As an extension for the above experiments, we also predict Nvidia Stock Prices with the same time span of the previous experiments.

In order to do so, we first built a *data scrapper* based on `twint` [7] and extract data from Twitter.

Figure 9: Word cloud of most popular words related to Nvidia on Twitter

After using TextBlob for the Sentiment Analysis, we feed the data to our model. Figure 10 shows the result graphically.

We also report the MSE table for the different approaches:

Mean Squared Error			
	RNN	**GRU**	**LSTM**
Whole Dataset	39.2606	**27.7972**	28.0963
Training Data	35.3114	29.8236	**28.0955**
Test Data	50.6829	**21.9361**	28.0985

From this experiment, we obtain a similar result of our previous report on stock market prediction without sentiment analysis: GRU seems to perform better than LSTM, possibly due to its ability to deal with overfitting. Given that the predicted data were in the same range of values as the training data, the RNN's advantage in extrapolation was overcome by the more encompassing representation of more

[7] Source: Github

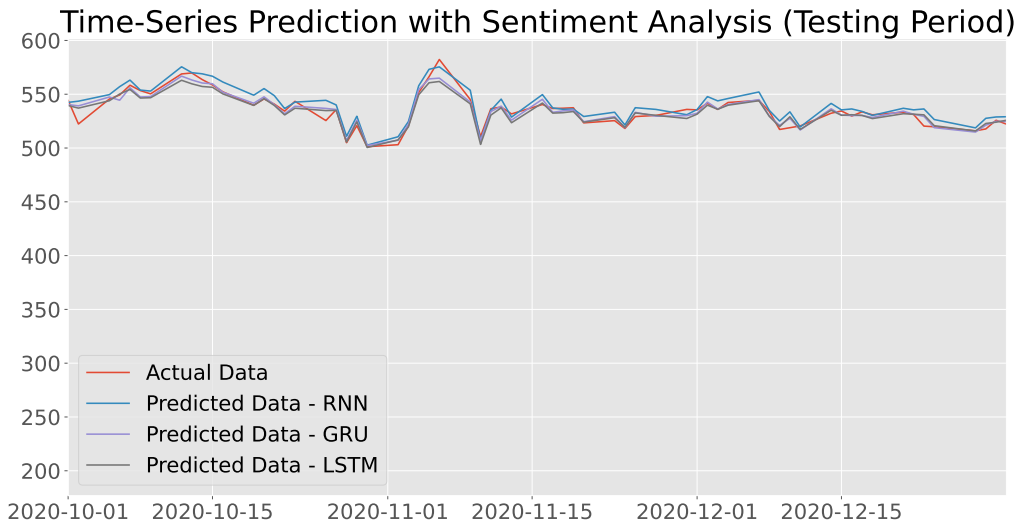

Figure 10: Comparison of different algorithms on Nvidia's stock price prediction with sentiment analysis

advanced models.
We noticed in both experiments that LSTM does not perform equally well as its less complex counterparts on smaller datasets, in which it is more prone to overfitting the training data.

References

[1] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[2] William Falcon et al. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] A. Mittal. Stock prediction using twitter sentiment analysis. 2011.

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai,

and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[6] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.