THE UNIVERSITY OF
MELBOURNE

Stock Price Prediction with Apple Inc. Stock Data and News

COMP90055 Computing Project (25 Credits)

Type of Project: Conventional Research Project

Supervisor: Dr. Miquel Ramirez

Wenyen Wei 949624

Semester 1, 2019

Executive Summary

Deep Learning (DL) is one of the most popular technologies nowadays, especially Recurrent Neural Networks (RNNs). Long-Short Term Memory Model(LSTM) , a specific class of RNNs, has gained its fame in the past few years for its outstanding performance on sequential data such as natural language understanding and generation, or stock market prices prediction. Natural Language Processing (NLP) comes to aid with transforming language data to numeric inputs to the models.

In this project, variations in performance between different RNNs models are explored and compared in order to come up with an optimised model. The result indicates that LSTM networks has outperform Bidirectional LSTM (BLSTM) model in accuracy over our dataset.

Basic NLP methods including stop words removal and term frequency-inverse document frequency measure(TF-IDF) are applied to optimize the news dataset. We have found TF-IDF to outperform simple stop words removal when it comes to stock market prediction accuracy.

Last but not least, we explore the sensitivity of our approach to different regimes of critical LSTM parameters, such as the length of the input sequence to explore the most optimized time model and we observed that the ideal input time sequence length for stock price and financial news is from one to two weeks.

Keywords: BLSTM, LSTM, RNNs, Deep Learning, NLP

Acknowledgement

I would like to thank the project supervisor, Dr. Miquel Ramirez, for helping so much in the process and patiently shape the project with me from the beginning to the end. He is always supportive and responsive to my questions and problems. He also recommended many study resources and even helped with code reviews. Without Miquel's help, I wouldn't be able to work on this challenging and exciting project.

I would also like to thank the contributors of the public stock dataset.

Without the help of them, the project wouldn't be possible.

Table of Contents

List of Figures

1. Introduction

1.1 Aims of the Project

The purpose of the project is to explore the availability of structured and unstructured data on stock markets and to predict the stock price of the selected stock using the data. After considering the aspects of data collection and general performance of the companies, we chose Apple Inc. stock(APPL)[27] as the selected stock for prediction. Furthermore, various combinations of neural networks types and hyperparameters are experimented in order to study the characteristics of the different models as well as to find out the most optimized model to make prediction with the highest accuracy.

1.2 Research Questions

Applications of deep learning to stock market prediction and financial news processing are both very active topics of research, and a very high number of techniques, problems and frameworks have been proposed in the past few years. We choose to scope this research by addressing four questions that span challenges such as the proper processing of language inputs, searching for the best performing input features and quantifying the sensitivity of the learnt model.

1.2.1 How Language Dataset Preprocessing Influence the Outcome

When it comes to natural language datasets, the noise in the dataset can significantly hamper the performance of the model. In order to preprocess the language data in order to achieve higher accuracy, we selected two methods: Stop Words Removal and TF-IDF to experiment and to compare the outcomes of each of them.

Stop words removal can be compared in a straightforward manner to TF-IDF as it simply requires removing common words in English such as articles e.g. “the”, or prepositions like “in” or “of” that, in general don’t have significant effect on the meaning of the input but have high frequencies in the corpus, and thus are more likely to bias the model.

On the other hand, TF-IDF relies on the computation of some basic statistics to calculate the level of importance of a word to a document in a collection. For instance, “Apple” is not a stop word in a regular English stop word dataset, but it has lower TF-IDF in our dataset as we use Apple Inc.’s financial news as input so “Apple” has high frequency over the collection. Since the TF-IDF statistics are calculated from our dataset, the weights obtained are specific to the corpus we consider

in our experiments, and thus we assume that TF-IDF, in general, has better performance improvement over stop words removal.

We are interested to know if the assumption that TF-IDF preprocessing should outperform Stopwords Removal preprocessing is correct.

1.2.2 Does Extra Information Helped or Biased the Performance

The input of our model is a combination of stock prices and news headlines' text. Stock price is a numeric value, e.g."$67", while the headlines text is mapped to a vector of integers representing the frequency of each word in the corpus such as [0, 2, 0, 1, 5, 1,…, 0, 0, 0]. The dimension of our texts set before preprocessing is 2750, which means there are 2750 unique words in the collection (please refer to Section 2.3 for data structure details). The mean lengths of our news headline dataset is between 9 and 10 as shown below in Figure 1.2.2-1.

```
mean         9.653516
std          2.004720
min          1.000000
25%          8.000000
50%         10.000000
75%         11.000000
max         20.000000
dtype: float64
```

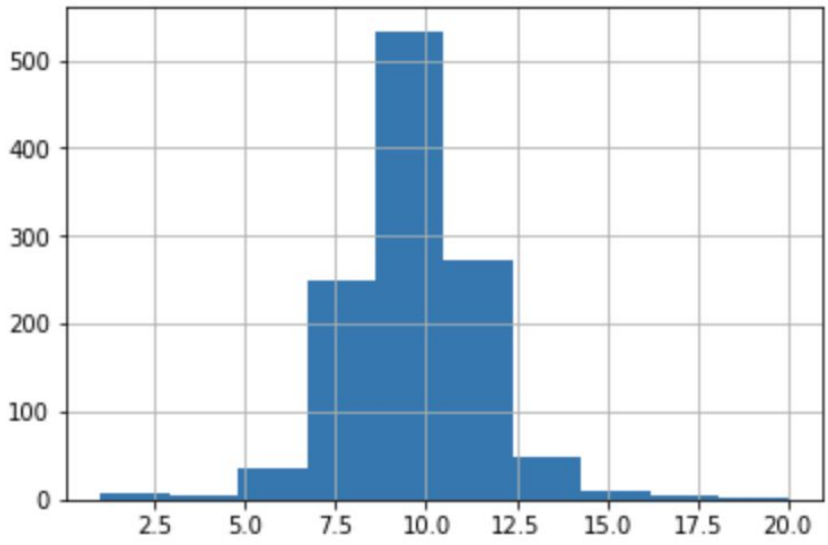

Figure 1.2.2-1: News headline lengths distribution in the dataset

Figure 1.2.2-1 shows an average of just about 99% empty inputs in the text vector, which we think can be considered to be fairly sparse. The question of whether sparsity of the text input vector reduces the performance of the model or it does actually improve the model prediction accuracy is something we explore in the project.

1.2.3 How Time Sequence Length Affect the Result

LSTM is well-known for its powerful performance over sequential data and time series. However, how does the sequence lengths of the data affect the performance? Does longer time sequence perform better than shorter time sequence?

Longer time sequence contains more information from the past, and may be easier for the model to identify structural patterns in the data. Alternatively, it also is possible too that longer time sequence brings larger noise and unnecessary information to the model., increasing the difficulty to identify useful patterns.

We are interested to see the performance variance over difference input time sequence on our dataset.

1.2.4 What is the performance difference between unidirectional LSTM and bidirectional LSTM?

Unidirectional LSTM only preserves information from the past, while bidirectional LSTM preserves information from both forward and backward LSTM cells, that is, from both the past and the future [6].

As bidirectional LSTM sees more information, it should outperform unidirectional LSTM. We would like to explore the performance outcome difference of both models and to experiment the suitability of the models for the project.

2. Dataset Description

2.1 Dataset Sources

There are two kinds of data required in the project: stock price data and financial news data. As there are no currently available dataset with both features, we found a source for the stock price data and we parsed the financial news ourselves.

We retrieved S&P 500 stock data from an open source dataset on Kaggle [1]. The dataset has over 14,000 downloads including several public Kaggle challenges and projects; thus we reasonably assume the dataset is reliable. The time range of the data is from February 2013 to February 2018.

For financial news data, we crawled all the financial news headlines of Apple Inc. on Reuters [2] from February 2013 to February 2018.

2.1.1 News Parsing App Design

A simple `Node.Js` Web App is designed and implemented to parse the news data from Reuters website [2] with using "`cheerio`" [7] as the library for parsing.

The below is part of the algorithm to create all dates strings during February 2013 to February 2018.

```
for (let yy = 2013; yy < 2019; yy++){
    for (let mm = 1; mm < 13; mm++){
        for (let dd = 1; dd < month_length + 1; dd++){
            // parse data of yymmdd
        }
```

The algorithm uses three nested loops. The algorithm seems fine as each month loop contains at most 31 loops, which are the dates in a month. However, the algorithm created heavy load for Reuters website as it sends request to the website 5*12*30 = 1,800 times in just a few seconds. The request from our app to the website was soon rejected by the server.

To overcome the problem, a "set-timeout" mechanism of five seconds between each request was added to the algorithm. After the improvement, the parsing process continued smoothly without being rejected by the website.

Building this parsing app took about ten hours in total from setting up the `Node.Js` app to writing the parsing algorithm, and to finally finish parsing.

2.2 Dataset Attributes

2.2.1 News Data

The news data includes the date and the headline. Below we show a sample of the data obtained from Reuters website after parsing.

01022013: "UPDATE 1-Judge rejects part of Apple App Store suit vs Amazon",
01042013: "US STOCKS-Data helps pace Wall St higher, but Apple drags",
01062013: "Riches in niches: U.S. cops, in-flight movies may be model for Panasonic survival",
01072013: "Samsung sees fourth quarter profit at $8.3 billion on Note sales, components",
01082013: "Apple working on cheaper iPhone: report",
01092013: "TEXT-Fitch: Low-End iPhone May Trim Samsung's Market Lead and Margins",

The data gathered from Reuters is noisy, yet it is not clear that its impact is measurable. One example of noise is “Batch of 59 rare Beatles songs to be released for sale” posted on December 13th, 2013. Although “Apple record” may seem be to one entity under Apple Inc, it is actually a music label founded by “The Beatles”[22] and is thus quite irrelevant to Apple Inc. Another noise example is “Samsung posts record profit; keeps 2013 capex at 2012 level” posted on January 24th, 2013. Samsung is the opponent of Apple, however the news appeared under Apple Inc. financial news category.

Whether or not these news headlines may cause bias to the model is open for discussion and a research question in itself. However, the measurement of irrelevance and relevance of news headlines to the company is not clear. Is the Samsung news really irrelevant to Apple? Can the Samsung news affect Apple stock price? The answers are currently unknown and are considered outside the topic of the project.

2.2.2 Stock Price Data

In our stock data, the attributes include `“open price”, “close price”, “high price”, “low price”, “date”, “volume” and “name”` of the stock as shown in Figure 2.2.2-1.

	date	open	high	low	close	volume	Name
1259	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL
1260	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL
1261	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL
1262	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL
1263	2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL

Figure 2.2.2-1: Stock Price Dataset Attributes

There are 1259 rows of data in the dataset with stock name “APPL”.

Figure 2.2.2-2 is the visualization of the APPL stock price data from February 2013 to February 2018, using the close price of each date.

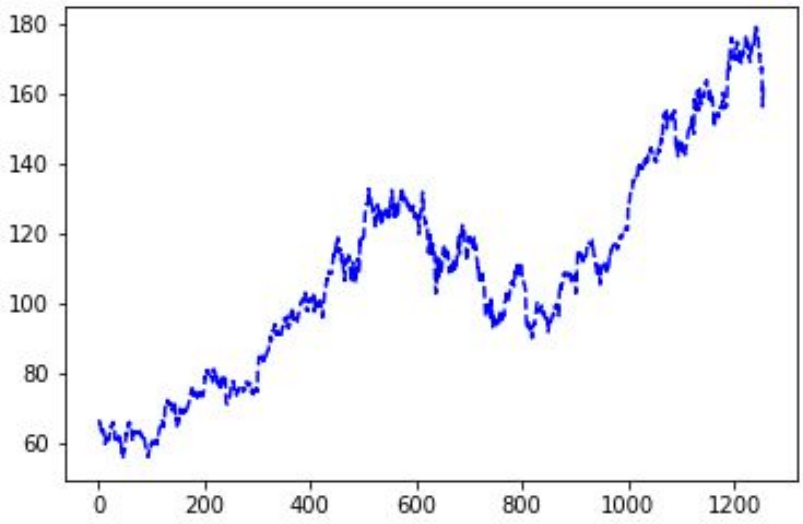

Figure 2.2.2-2: APPL stock price from Feb. 2013 - Feb. 2018(X axis). Y axis is price

2.3 Dataset Integration

We next describe how we integrate the data acquired from the two sources above.

X, the input for one data point, that corresponds to one trading day, looks as follows:

```
X: [array([68.0714, 68.5614, 0, ..., 0, 0, 0], dtype=object)]
```

For simplicity, we firstly use only open price and close price from stock data, which is the first two columns. The stock data is then followed by a vector of bag of words[23] of the headline of the day. The dimension of the data is 2752 in total, containing 2 stock price data and 2750 bag of words vector.

Y, the label, is simply the close price of the next day, as shown below.

```
Y: [66.8428]
```

To summarise, we use “open price”, “close price” and “news headline” of the day to predict the “closing price” for the next day as our base model input.

Further input features optimization will be explored in other sections of the project.

3. Methodology

This section explained the steps and decisions involved in the training process. To begin with, two RNNs models: LSTM and BLSTM are discussed. In the second section, the input structure is discussed in detail, including the hyperparameters and input preprocessing involved. In the third section illustrate the rationale behind the choice of the loss function and optimizer. Last but not least, model evaluation mechanism is explained.

3.1 Recurrent Neural Network Models

3.1.1 LSTM

Long Short-Term Memory (LSTM) [4] is one kind of Recurrent Neural Networks (RNNs) designed to remove vanishing gradients problems of convolutional RNNs by using multiple gates [4] to carry temporal information. LSTM has been found to perform better in longer time sequences in comparison to convolutional RNNs.

As LSTM has a "conveyor belt" of cell states traveling down the training data stream, the gates carefully decide what informations from previous states should be pass on to the next state [5]. The gates in LSTM is consist of a sigmoid neural net layer with weight multiplication [5] with an output number between zero and one. Zero means to not pass any information to the next state while one means pass all the information to the next state.

The first gate is the "forget gate layer", which takes h_{t-1}, the output of the previous cell, and x_t, the input to the current cell to decide the proportion of information to be forgotten. The equation is as follow.

$$f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f)$$

Secondly, LSTM decides the values to be updates, controlled by i_t, a "input gate" sigmoid layer. The equation is similar to the forget gate but with difference weight and bias.

$$i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i)$$

The input gate is then followed by a tanh layer that generates the candidate values $\tilde{C}_t$ to be updated to the states.

$$\tilde{C}_t = \tanh(W_C \times [h_{t-1}, x_t] + b_C)$$

The input gate is then followed by a tanh layer that generates the candidate values $\tilde{C}_t$ to be updated to the states.

Thirdly, to update the cell state from C_{t-1} to a new state C_t. The update is achieved by the following equation using a combination of informations from the previous steps.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

Finally, to generate the desired output h_t, a sigmoid layer combined with a tanh layer that maps the output value to lie between 1 and -1 is used.

$$o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \times \tanh(C_t)$$

To conclude, one LSTM cell first use two sigmoid layers to decide which inputs to forget or to update, and then a non-linear layer of update candidates is used with the gates to generate a new cell state. Finally, a sigmoid layer and a tanh layer are used in the last step to compute the output from the new cell state.

3.1.2 BLSTM

Bidirectional RNNs (BRNNs) is a bidirectional variant of RNNs, trained with not only inputs of time frames up to the preset present frame but instead uses a combination of positive and negative time direction [6]. Two neural net of both forwards and backwards training sequence are connected to the same output layer. BRNNs has shown improvements in sequence learning tasks such as protein structure prediction and speech processing [10].

Bidirectional Long Short-Term Memory (BLSTM) is a combination of BRNNs and LSTM.

3.2 Input Structure

In this Section, we will first elaborate the input shape and how the time sequence is framed. After that, train-test sets split will be briefly mentioned. Finally, the hyperparameters involved in the training process will be introduced.

3.2.1 Input Dimension

The training inputs are made of two elements, the features in a day and the time sequence.

For a single day input, the data is a vector of dimension 2752 that includes the 2 stock price data items, the opening and closing prices, as well as each of the 2750 bag of words components. We need to further process our data to construct a time sequence. For that, we ordered the data by date with a designated time length, following by the end date data as the target label.

To illustrate the above, consider the following example of input when the duration of the time sequence is set to three days:

$$X_i = [Day_i, Day_{i+1}, Day_{i+2}]$$
$$Y_i = [Day_{i+3}]$$

The above is treated as one data point after preprocessing. The data point following it is therefore structured as:

$$X_{i+1} = [Day_{i+1}, Day_{i+2}, Day_{i+3}]$$
$$Y_{i+1} = [Day_{i+4}]$$

3.2.2 Training Set and Test Set

There are a total of 1259 rows in our dataset. We remove rows 500 to 600 and rows 1100 to 1200 from the dataset and set them apart as our test set. The rationale behind splitting the data separately is to prevent testing bias by use on one single time range as test set.

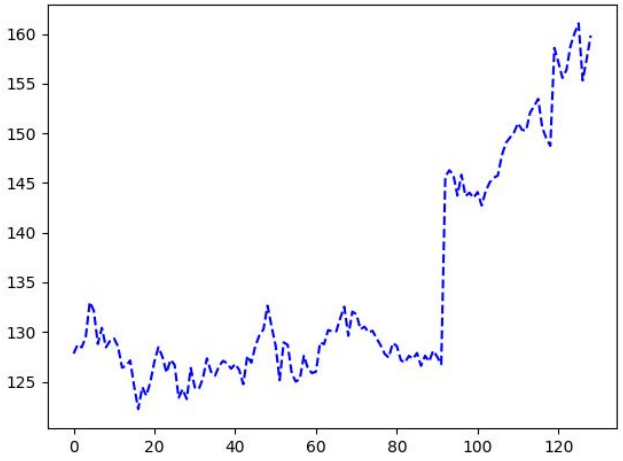

Figure 3.1.2-1 shows the price distribution over the test set

3.2.3 Hyperparameters

The hyperparameters involved in the training includes: sequence size(W), where W is for window, hidden layer size(H), where H is for hidden, epochs(E) and batch size(B).

W is the length of the input time sequence. H is the size of hidden layer in the neural network, default to 64. E is the size of the training rounds. B is the number of data points fed in at a time.

An example for batch size 3 with B stands for batch and D stands for data point would be:

$$B_{x,i} = [D_i, D_{i+1}, D_{i+2}]$$
$$B_{y,i} = [Y_i, Y_{i+1}, Y_{i+2}]$$

3.3 Loss Function and Optimizer

3.3.1 Loss Function

Mean squared error is used as the loss function. The equation for mean squared error is illustrated as follows

$$MSE = 1/n \times \sum_{i=1}^{n} (Y - P)^2$$

Y stands for the label associated with the input, and P is the prediction output.

3.3.2 Optimizer

Root Mean Square Propagation(RMSprop) is selected as the optimizer for the project. RMSprop is an unpublished optimization algorithm introduced by Geoff Hinton in his online course [8]. Despite the fact that it is unpublished, RMSprop has become increasingly popular over the years and it is included in many open source deep learning libraries such as Tensorflow[26].

RMSprop is similar to Gradient Descent with Momentum[25], but with different update parameters to prevent using a too large learning rate as using a too large learning rate can result in the model converging to a suboptimal solution.

3.4 Model Evaluation

3.4.1 Mean Squared Error

In regression tasks like this project, we evaluate the outcomes by the mean squared error of the prediction outputs and the input labels. The lower the mean squared error, the more accurate the model prediction.

3.4.2 Prediction versus Label Graph

We also plot out the prediction graph and label graph to see how the prediction works on difference scenarios such as steep gaps. The following Figure 3.3.1 is one example of the output graph, with the blue line as the input label and the red line as the prediction and the X axis as the data points and Y axis as the prices.

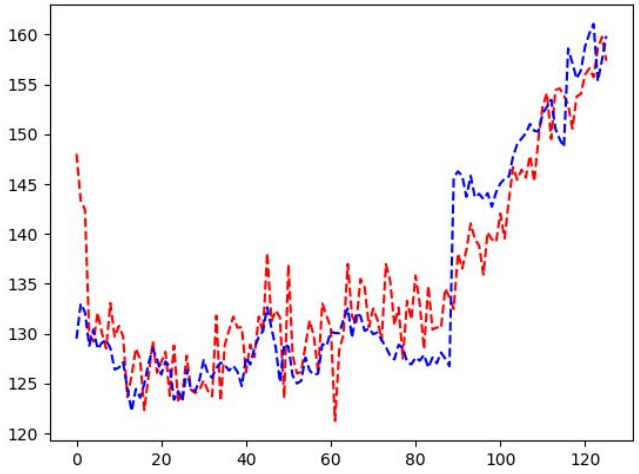

Figure 3.3.1: Example Prediction Output Graph

3.4.3 RNNs as Baseline

Last but not least, we use standard RNNs' prediction performance as baseline, as LSTM and bidirectional LSTM are both assumed to be enhancements over convolutional RNNs.

4. Model Optimization

In this Section, several optimizations are done on different aspects of the project, including the dataset, the input features and the hyperparameters. We explored the methodology on optimizing the performance and examined the output comparison of each scenario to find out the best combination for our final model.

4.1 Dataset Optimization

As there are many ways to process language data into numeric data, we applied two different techniques: stop words removal and TF-IDF on our language dataset to see the performance of each.

4.1.1 Stopwords Removal

To removal common stopwords in the dataset, we use NLTK[9] English `Stopwords` package as the stop words collection. Then we remove the co-occurrence words of our news headlines dataset and the NLTK stopwords collection.

Here is one of the headlines in the dataset, before and after stopwords removal.

```
Riches in niches: U.S. cops, in-flight movies may be model for
Panasonic survival.

Riches niches: U.S. cops, in-flight movies may model Panasonic
survival.
```

As shown in the example, “in” and “for” are removed from the headlines after processing as they are the stopwords selected in the NLTK English Stopwords package.

Stopwords removal can help reduce the dimension of our input data by removing unnecessary words. However, some significant keys may be lost from this process. For instance, phrasal verbs in English such as “turn”, “turn on”, and “turn in” have different meanings but “in” and “on” might be removed as stop words.

The bag of words vector dimension after stopwords removal is 2716. 32 unique stopwords are removed from the dataset.

4.1.2 TF-IDF

TF-IDF is calculated as the product of term frequency(the frequency of a word in a document) and inverse document frequency(the measure of how much of information is provided by a word)[24]. It has a similar effect as stop words removal in that it removes words with higher occurrence over the collection, but it provides a more customized statistics weights to our dataset compare to simple stopwords removal. “TfidfTransformer” package of Scikit-Learn library is applied for TF-IDF to our dataset.

We assume that TF-IDF will perform better than stop words removal as the filtering of irrelevant words is guided by the statistics of our news headlines corpus, rather than assumptions on the meaning of specific words.

The dimension of the input data is the same after TF-IDF, 2750 for bag of words.

4.1.3 Results Comparison

The Mean Squared Error of Stopwords removal is around 38 for test set and 25 for training set, while the Mean Squared Error of TF-IDF is around 40 for test set and 28 for training set.

Despite the result of Mean Squared Error, from the training set output graphs below, we observed the opposite - stopwords removal has higher accuracy variance compare to TF-IDF. The Mean Squared Error number is very likely biased by the first few predictions, as shown in the images below with significant prediction error illustrated in red lines at the beginning.

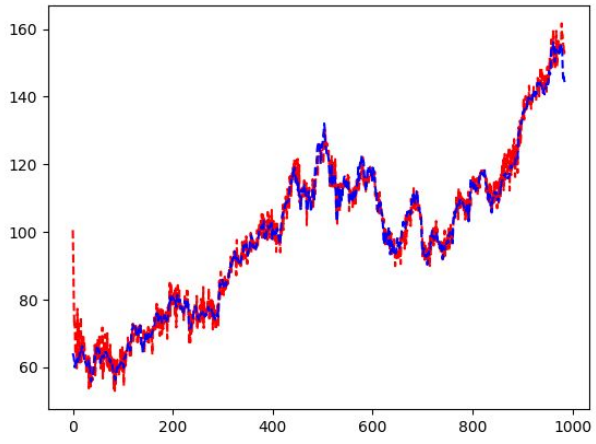

Figure 4.1.3-1: Stopwords removal training prediction

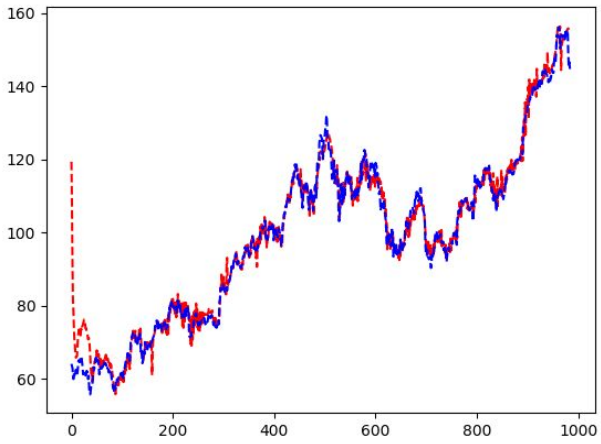

Figure 4.1.3-2: TF-IDF training prediction

The test results are shown below in Figures 4.1.3-3 and 4.1.3-4. The performance of TF-IDF is exceptionally good when compared to stop words removal especially, when there is a sharp change in price as is the case during the time period covered between data points 80 and 100. While we can observe on the left that the prediction didn't follow the steep increase very well, on the right image a significantly steeper edge aligned with the actual stock price can be observed.

Other than that, the error variance is also much smaller in TF-IDF compare to Stopwords removal.

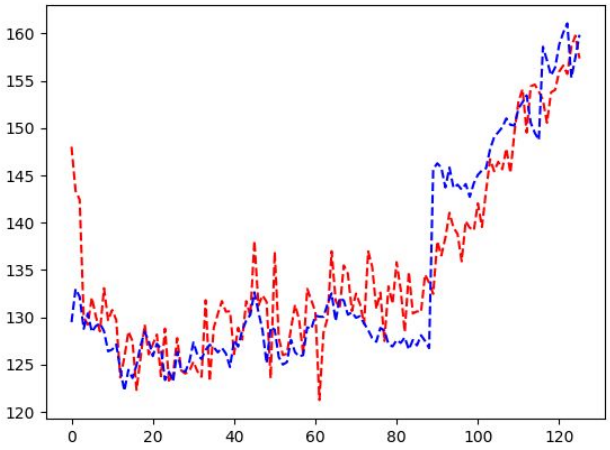

Figure 4.1.3-3: Stopwords removal testing prediction

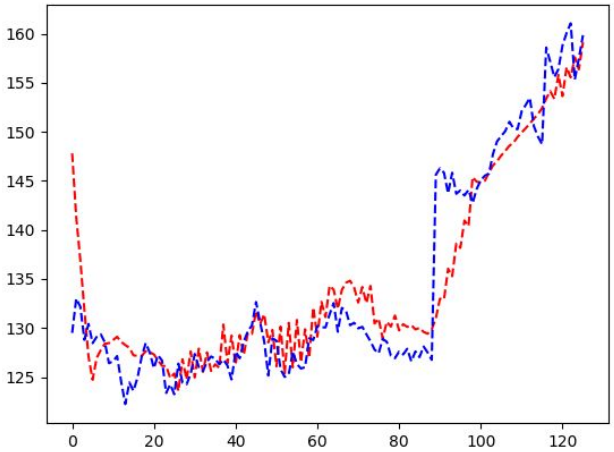

Figure 4.1.3-4: TF-IDF testing prediction

Based on the above observation, we can assume our assumption is valid that TF-IDF outperform stop words removal when it comes to generate robust predictions.

4.2 Input Features Optimization

As discussed in section 1.2.2, the sparsity of the bag of words vector draws the concern that it might bias the model with an average of 99.64% of empty inputs. In this section, we will explore the performance of the model with news headline input and without it.

4.2.1 Necessity of Extra Information

Our working assumption is that extra information should bring more features of the dataset, which is supposed to provide higher accuracy. However, excessive information can slow down the training process and consume extra computing power. Thus, the necessity of extra

information is the topic we would like to explore. Does the extra information in our dataset, which is, the news headline, improve or bias our model?

We use the same hyperparameter settings to test the two scenarios. Namely, we set the batch size to 64, the number of hidden layers to 32, and the time sequence to 10.

4.2.2 Results Comparison

The Mean Squared Error without text is around 50 for training set and 90 for testing set, while the Mean Squared Error with text is around 25 for training set and 38 for testing set.

The graphs below illustrate the significant differences in performance observed between using or leaving out news texts inputs. The error variance is way lower with text inputs both over the training and testing sets.

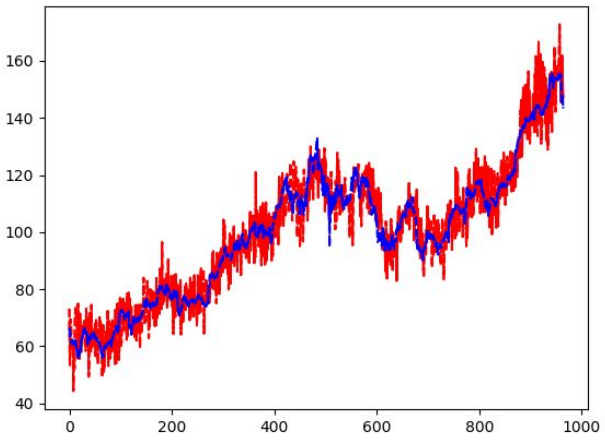

Figure 4.2.2-1: without text training prediction

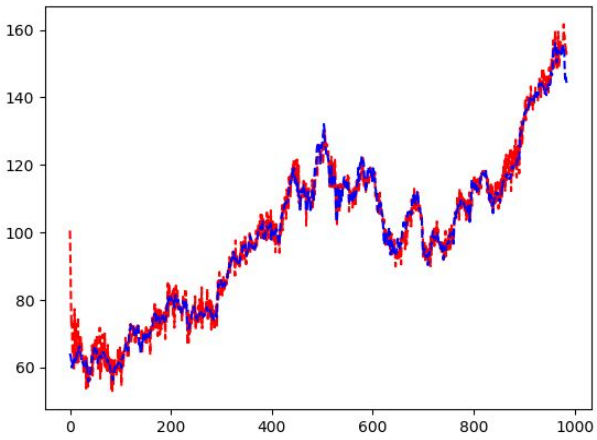

Figure 4.2.2-2: with text training prediction

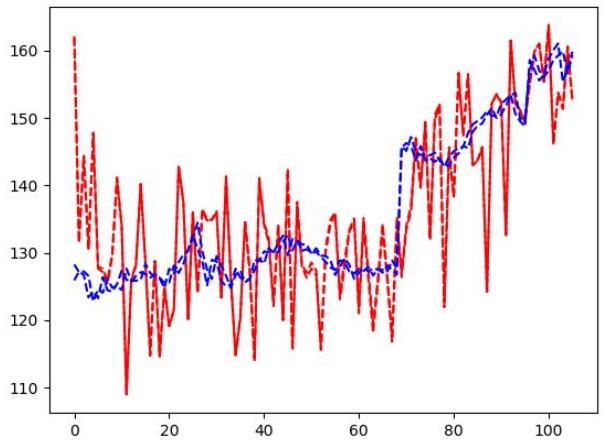

Figure 4.2.2-3: without text test set prediction

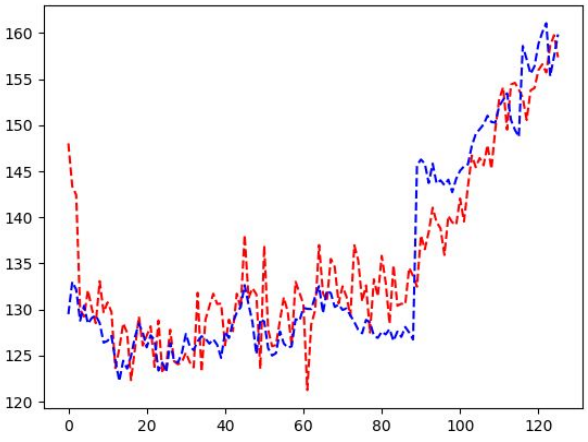

Figure 4.2.2-4: with text test set prediction

We thus conclude that the existence of a sparse bag of words vector doesn't reduce the performance of the model but rather improves dramatically the accuracy of the predictions.

4.3 Hyperparameter Optimization

It is important to configure the hyperparameters for deep learning training as a good set of configurations could greatly save computing resources as shown in Figure 4.3.1-1 that by simply changing the input batch size, the training time difference could range up to about six times.

There are three hyperparameters we are tuning in the project: batch size, hidden layer and learning rate. Please note that all the images in this Section uses epochs as X axis and loss as Y axis.

4.3.1 Batch Size Tuning

Batch size is an important hyperparameter as a too large batch size can consume unnecessary and put a heavy load on memory. On the other hand, a too small batch size can result in the model taking longer time to converge.

Research results confirms that smaller batch size provides higher training stability and generalization performance. In most cases the best results are obtained with batch size = 32 or smaller[11].

We trained the model with the same settings with hidden layer = 128, learning rate =0.001, over 10 epochs of 10520 steps with 4 possible values of batch size: 16, 32, 64, 128.

The below is the loss graphs for each batch size over 10 epochs of 10520 steps in total. The second graph is a detailed look of the first graph.

Batch Size	MSE	Steps	Time
16	**51.0065**	**10.52k**	**00:01:32**
32	50.3884	10.52k	00:02:40
64	62.7034	10.52k	00:04:57
128	75.4140	10.52k	00:08:32

Figure 4.3.1-1 training comparison of different batch sizes

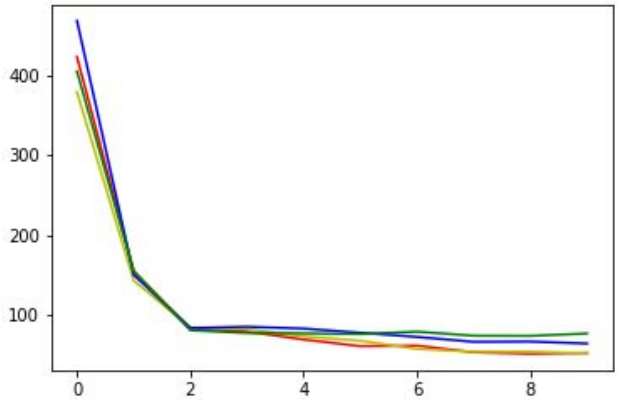

Figure 4.3.1-2 Losses over 10 epochs

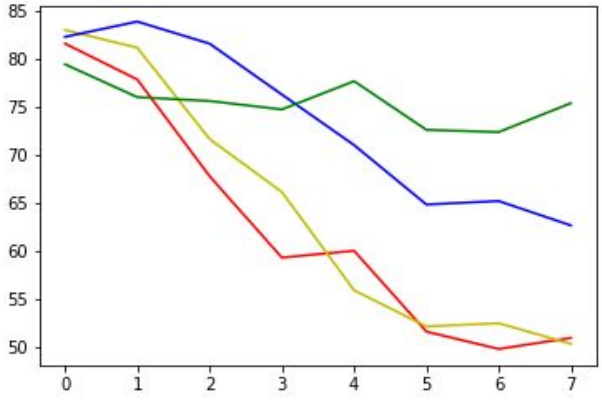

Figure 4.3.1-3 Losses over the last 8 epochs

The results show that batch sizes under 32 performs almost equally well and significantly outperform batch sizes over 32 while batch size 16 achieved almost same loss as batch size 32 within 40% shorter time, which saved a lot of resources in comparison to batch size 32. Batch size of 128 and 64 don't converge as fast during the training steps.

4.3.2 Hidden Layer Tuning

To choose a good hidden layer size to begin with, we experimented hidden layer sizes of 64, 128, 256 over batch size = 16 and learning rate = 0.001 over 10 epochs and find out that hidden layer size of 256 converged the fastest with the lowest mean squared error.

Hidden Layer	MSE	Steps	Time
64	101.4428	10.52k	00:01:12
128	51.0065	10.52k	00:01:32
256	**35.6867**	**10.52k**	**00:02:49**

Figure 4.3.2-1 training comparison of different layer sizes

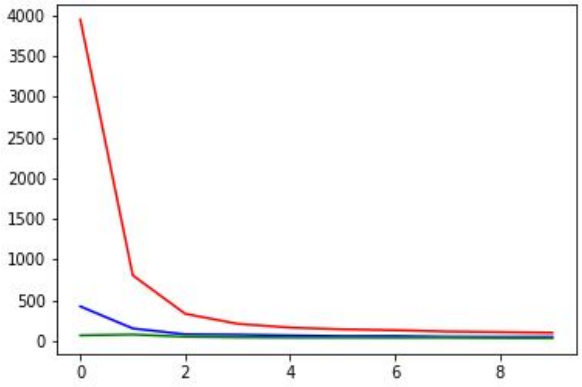

Figure 4.3.2-2 Losses over 10 epochs

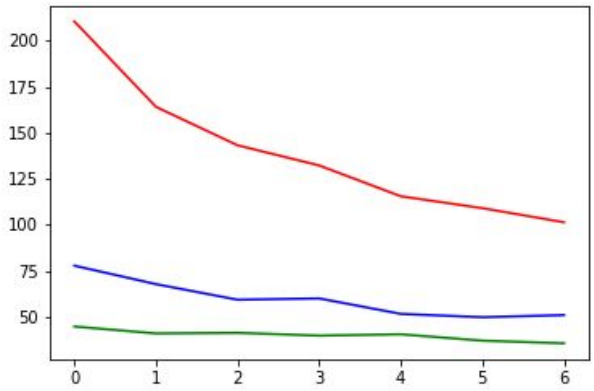

Figure 4.3.2-3 Losses over the last 7 epochs

4.3.3 Learning Rate Tuning

Learning rate controls the speed the model learns as the updated rates are tuned by the multiplication of learning rate and estimated weight error[12].

A large learning rate generally causes the model to learn faster at the cost of ending up at suboptimal weights, while a small learning rate allows the model to arrive at more optimal weights but with extremely long training time or stuck with permanently high training error.[13]

To find out the right learning rate for the training, learning rate of 0.0001, 0.0005, 0.001, 0.005 are experimented with hidden layer size of 256 and batch size of 16 over 10 epochs.

Learning Rate	MSE	Steps	Time
0.0001	83.6342	10.52k	00:02:45
0.0005	**33.3713**	**10.52k**	**00:02:44**
0.001	51.0065	10.52k	00:01:32
0.005	78.9786	10.52k	00:02:06

Figure 4.3.3-1 training comparison of different learning rates

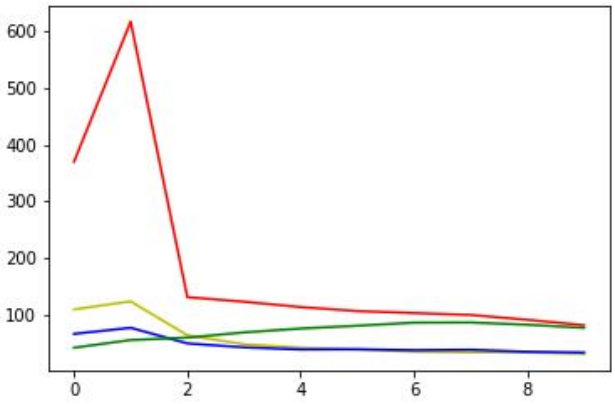

Figure 4.3.3-2 Losses over 10 epochs

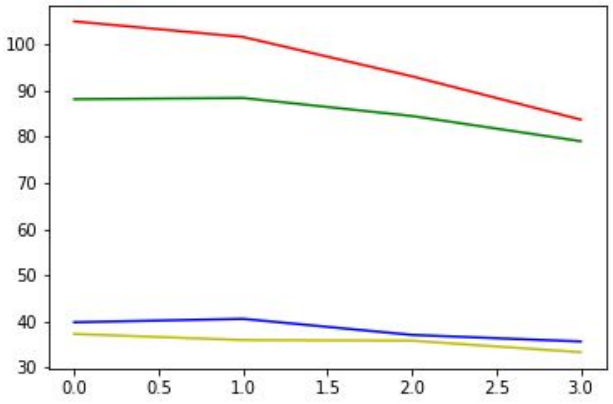

Figure 4.3.3-3 Losses over the last 4 epochs

The result shows that learning rate of 0.0005 and 0.001 converges comparably fast, while 0.0005 achieved a lower loss but longer training time. Learning rate of 0.0001 reached at unreasonably high loss compares to all other learning rates while 0.005 shows an up-going loss curve during the given training steps, which is a sign of overfitting.

4.3.4 Overfitting Resistance

Overfitting is one common problem in the training process. An overfitting model fails to generalize to the optimal global model but instead converges to a suboptimal set of weights.

Regularization assumes that smaller weights help prevent overfitting by eliminating larger weights and thus generating simpler models. L2 regularization, also known as Ridge Regularization, uses the sum of the squares of all input feature weights, causing the outlier input that generates larger weights to have higher loss. That is, L2 Regularization uses a coefficient as penalty term to the loss function to prevent overfitting.

Hence, to prevent overfitting, we added a L2 regularization term to the loss function. We tuned the loss function using L2 regularizer as below with the help of Tensorflow[26] library.

```
beta = 0.01
regularizer = tf.nn.l2_loss(weights)
loss = tf.reduce_mean(loss + beta * regularizer)
```

4.4 The Optimized Model

Based on all the above observations and experiments, the current optimized training model would have a hyperparameter sets of hidden layer = 128, learning rate = 0.0005, batch size = 16 and with TF-IDF processed news headlines input.

5. Result and Analysis

In this section, we firstly discussed the prediction performance of the optimized model. Secondly, different time sequence scenarios are experimented and compared and discussed. Thirdly, we look at the performance difference between LSTM and BLSTM.

5.1 Training & Prediction Accuracies of the Optimized Model

Based on the previous Section, the current optimized training model would have a hyperparameter sets of hidden layer = 128, learning rate = 0.0005, batch size = 16 and with TF-IDF processed news headlines input. The training and prediction accuracies of the optimized model are shown below, with the red line as the predicted value and the blue line as the actual price of the stock.

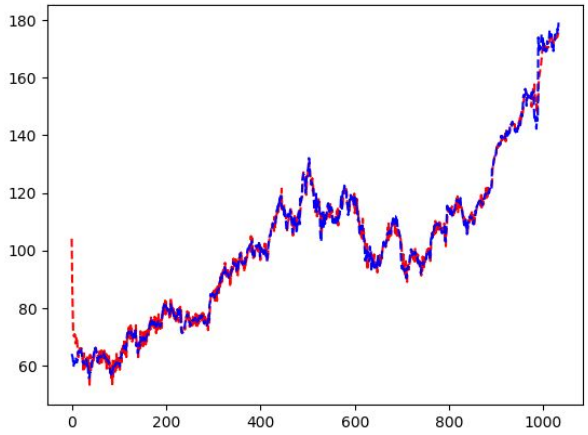

Figure 5.1-1: LSTM training set prediction

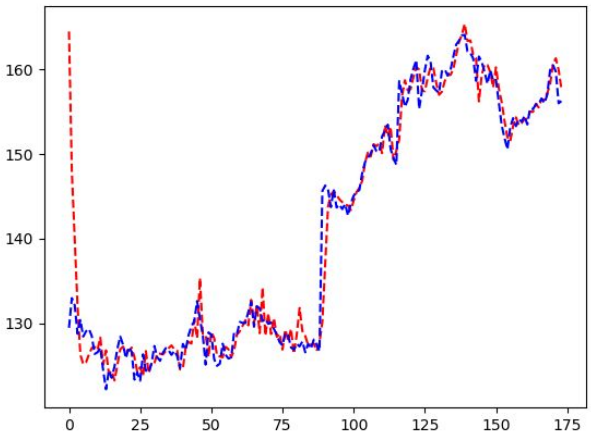

Figure 5.1-2: LSTM test set prediction

The two graphs below report the results of training using basic RNN cell instead of LSTM cell, with the same input and hyperparameter sets. The RNN cell model is used as the baseline to indicate the effectiveness of the LSTM model.

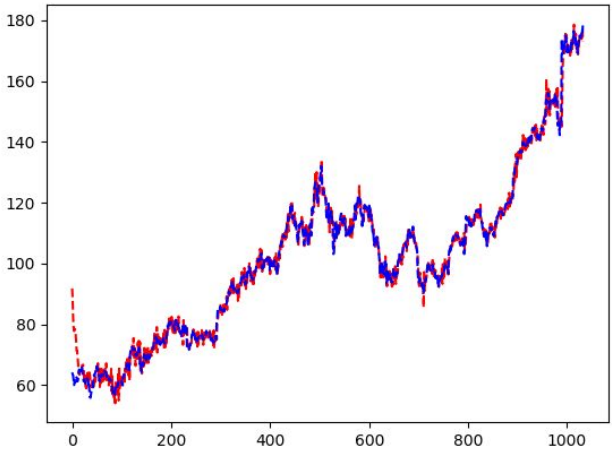

Figure 5.1-3: RNN training set prediction

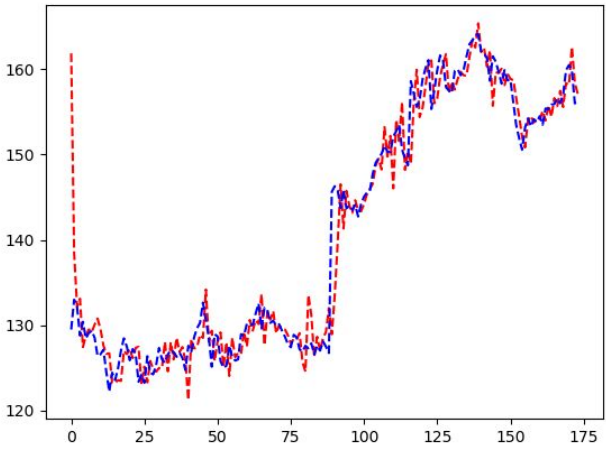

Figure 5.1-4: RNN test set prediction

Comparing LSTM and RNN, there is actually not very significant performance variance in the test set prediction. However, we can still observe a cleaner prediction line in LSTM's prediction compared to RNNs' prediction. To conclude, although not significant, LSTM slightly outperform basic RNNs.

5.2. Different Time Sequence Scenarios

The length of the time sequence could have significant effect on the results as they may contain important information or they carry too much noise to bias the model. Performance comparisons are done in three different time sequence scenarios: 5 days, 10 days and 15 days as for stock markets, each week consists of 5 working days. So it is the comparison between time sequence of 1, 2 and 3 weeks.

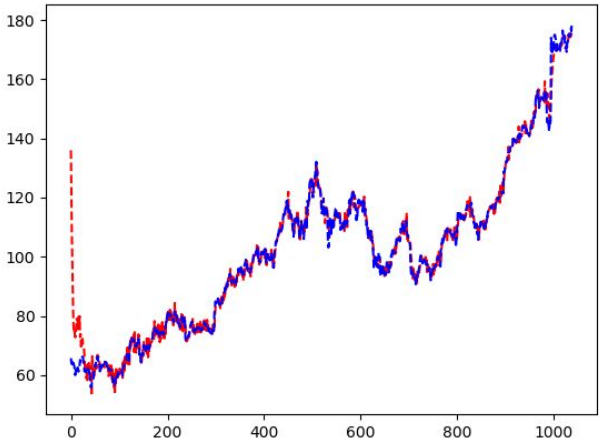

Figure 5.2-1: 5 days training set prediction

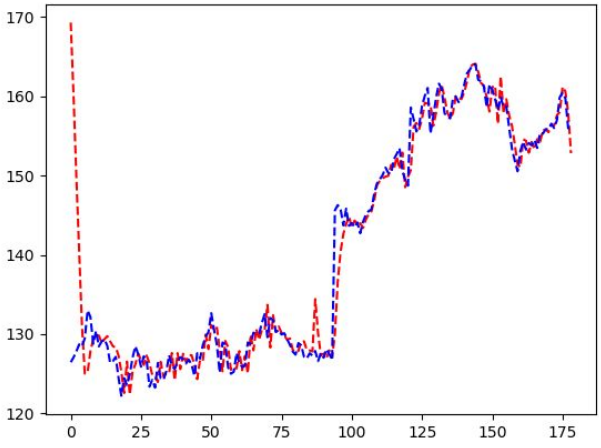

Figure 5.2-2: 5 days test set prediction

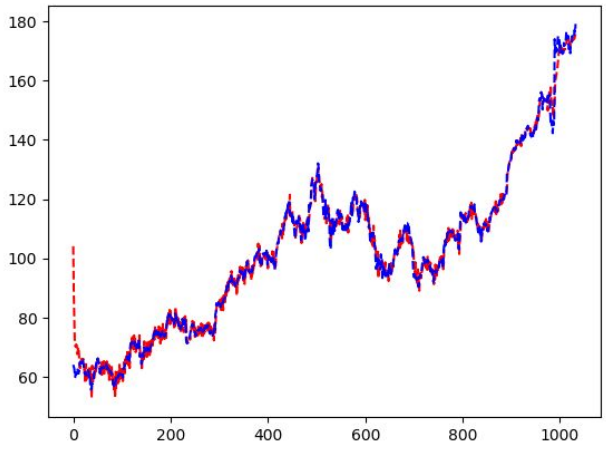

Figure 5.2-3: 10 days training set prediction

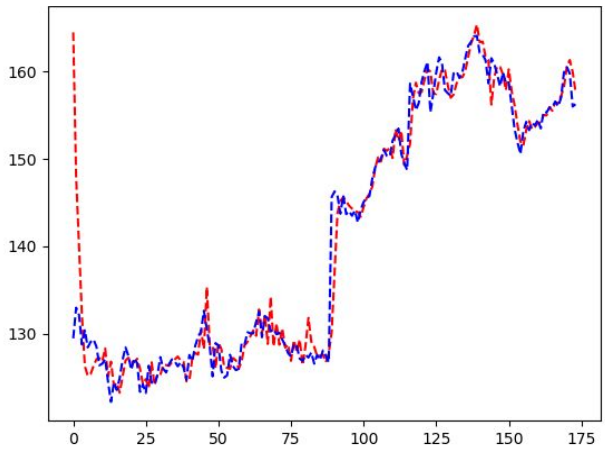

Figure 5.2-4: 10 days test set prediction

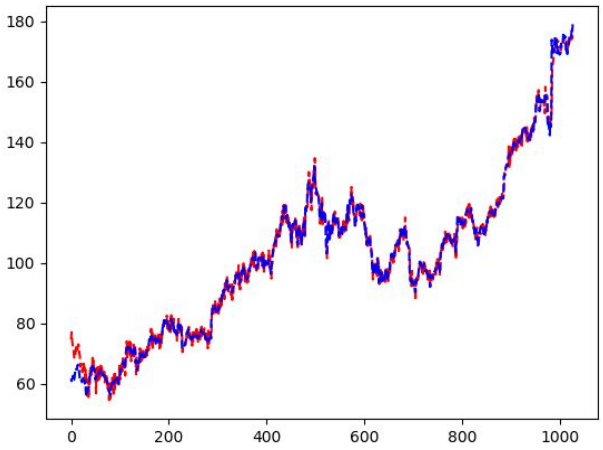

Figure 5.2-5: 15 days training set prediction

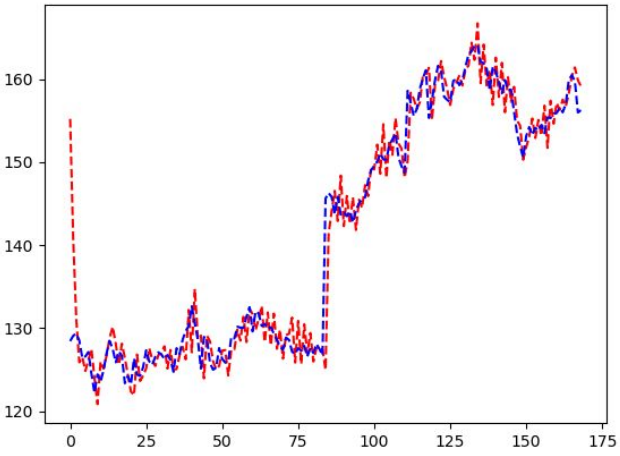

Figure 5.2-6: 15 days test set prediction

We can observed that 5 days and 10 days performed almost equally well with similar clean graphs and just a tiny variance difference, while 15 days appears to be less accurate in prediction compared to 5 days and 10 days.

We can thus refer from the result that the model learns well from 5-10 days time range, and 15 days' data input start to bring noise to the model and reduce the performance.

One thing interesting to note is the high error rate at the beginning of the prediction dataset has significantly reduced as the time range increase. We will continue to explore this issue in a later Section.

5.3 Performance Comparison Between BLSTM and LSTM

As discussed in Section 1.2.4 and Section 3.1.2, BLSTM uses both information from the past and the future to train the model and is proven to show improvements in certain tasks such as protein structure prediction[10]. Here we compare the performance difference between LSTM and BLSTM.

However, an interesting result showed: the performance of BLSTM is not as good as LSTM on the same input with the same hyperparameters setting. To confirm, we experimented with different time sequence settings. The results show that no notable difference in performance has changed even over different time sequence.

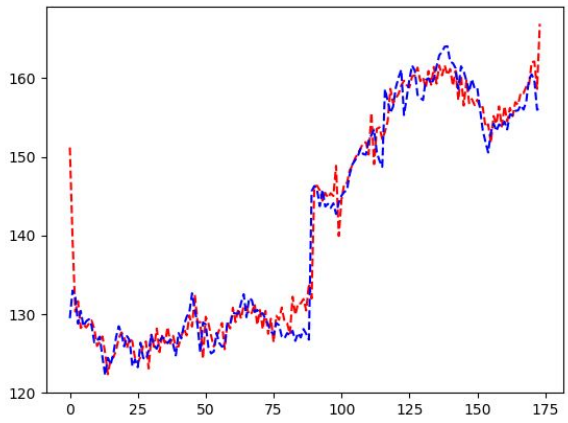

Figure 5.3-5: BLSTM prediction, 10 days

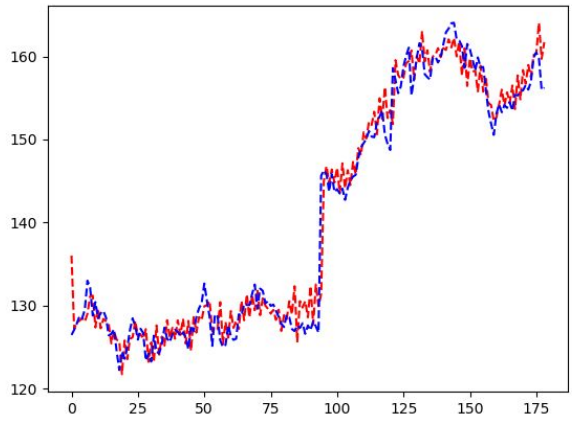

Figure 5.3-6: BLSTM prediction, 5 days

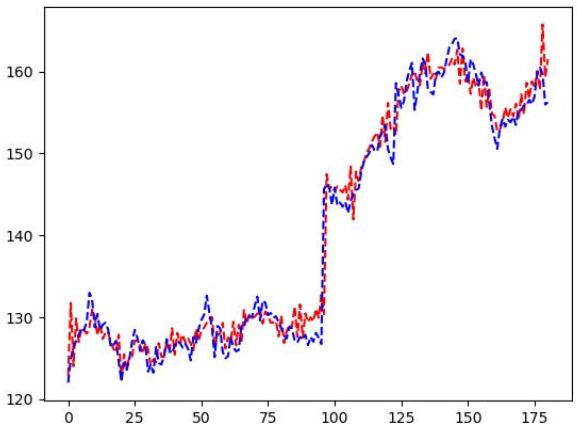

Figure 5.3-7: BLSTM prediction, 3 days

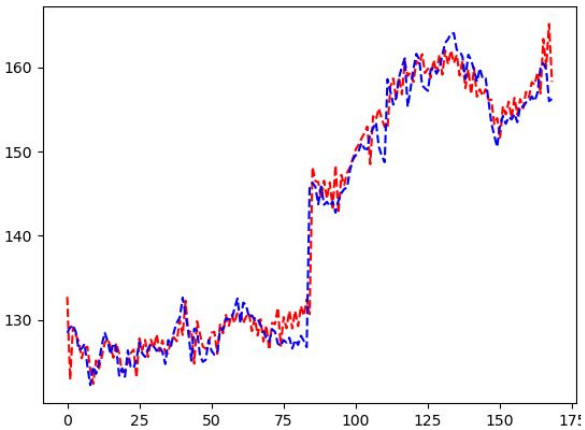

Figure 5.3-8: BLSTM prediction, 15 days

Although BLSTM is proven to improve performance in prediction protein structure and speech processing, it doesn't show improvement in financial news with stock price data. One possible reason could be because stock price depends on the data in the past, as traders and investors used

past information to decide which stock to trade not the future data. Thus the advantage of BLSTM: offering information from the future became irrelevant.

5.4 Prediction Error in Sequence Start

As shown in many graphs in the project, there is almost always a spiking high prediction error at the beginning of the dataset. To find out the reason why it has a high error rate, we look into details of the stock price of the first few batch, to see if there is any irregular pattern.

```
02052015: "Exclusive: Apple's health tech takes early lead among top hospitals",
02062015: "Taiwan stocks fall; Apple suppliers, TransAsia down",
02092015: "Enter the iSwissie",
02102015: "US STOCKS-Wall St rises on Greek deal hopes; Apple hits record",
02112015: "US STOCKS-U.S. stock futures rise on Greek agreement after S&P ends flat",
02122015: "China's Xiaomi to start small in U.S., with earphones and bands",
02132015: "UPDATE 2-Greenlight Capital scales back on Apple shares ahead of rally",
02142015: "Apple studies self-driving car, auto industry source says",
02172015: "HIGHLIGHTS-Top U.S. hedge funds cut Apple stakes ahead of all-time high",
02182015: "Samsung to buy mobile payment service provider LoopPay",
02192015: "BRIEF-Japan Display considering building new plant -Nikkan Kogyo",
02202015: "Apple hiring big brains in car battery space",
02232015: "US STOCKS-Nasdaq ends up 9th session; S&P 500, Dow dip with energy",
02242015: "Visa Europe plans new security that could pave way for Apple Pay",
02252015: "GLOBAL MARKETS-S&P 500 lags global markets as Yellen ends two-day testimony",
02262015: "Apple faces second suit from victorious patent firm",
02272015: "Cook says Apple Watch will replace car keys: Telegraph",
03012015: "Samsung unveils sleek new Galaxy phones to take on Apple",
03022015: "Apple, Google poaching settlement appears headed for approval",
03032015: "UPDATE 2-Apple plans fix next week for newly uncovered Freak security bug",
03042015: "Apple to delay larger iPad production till September: report",
```

Figure 5.4-1 first 20 test set data news

	date	open	close
500	2015-02-04	118.500	119.560
501	2015-02-05	120.020	119.940
502	2015-02-06	120.020	118.930
503	2015-02-09	118.550	119.720
504	2015-02-10	120.170	122.020
505	2015-02-11	122.770	124.880
506	2015-02-12	126.060	126.460
507	2015-02-13	127.280	127.080
508	2015-02-17	127.490	127.830
509	2015-02-18	127.625	128.715
510	2015-02-19	128.480	128.450
511	2015-02-20	128.620	129.495
512	2015-02-23	130.020	133.000
513	2015-02-24	132.940	132.170
514	2015-02-25	131.560	128.790
515	2015-02-26	128.785	130.415
516	2015-02-27	130.000	128.460
517	2015-03-02	129.250	129.090
518	2015-03-03	128.960	129.360
519	2015-03-04	129.100	128.540

Figure 5.4-2 first 20 test set data

Observing from Figures 5.4-1 and.4-2, there is no significant data variance or data error. Thus, we can first eliminate the possibility of noise in the dataset causing the issue. Nonetheless, in Figure 5.4-1, if we count the first 5 dates of the dataset, which is from February 4th to 9th, we can see that there are only 2 news headlines in the time range. That indicates 60% of the data contains empty news, which means the data density is quite sparse.

We thus make an assumption that shorter time range could suffer bias from lower text data density. Looking at the results from section 5.2, we observed the same pattern in time sequence difference that supports our assumption as shorter time range shown higher bias at the start of the dataset in both test and train prediction.

Based on the assumption, we carried out experiment with no text data. By removing the possible news data density bias, we expected to see reductions in the high beginning prediction error.

Nonetheless, no significant improvement was shown and thus rejected the assumption that the high prediction error appears because of low text data density in basic LSTM model.

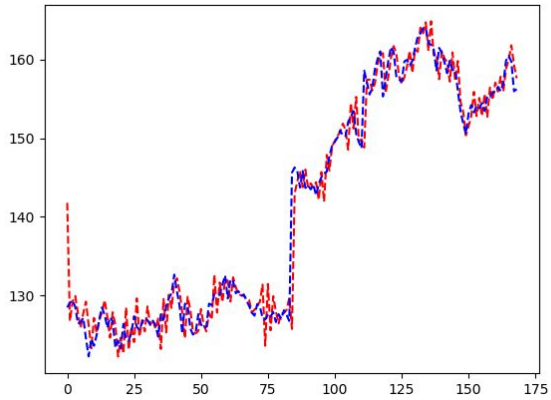

Figure 5.4-3 without text test set data, 15 days

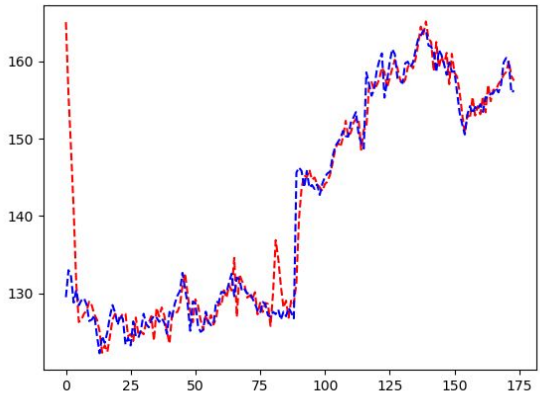

Figure 5.4-4 without text test set data, 10 days

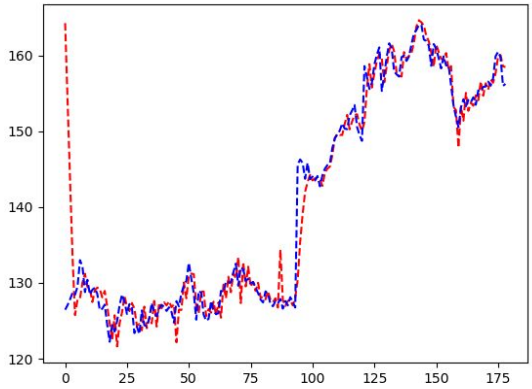

Figure 5.4-5 without text test set data, 5 days

However, similar trend is observed. Longer time range reduced the beginning prediction error. Thus we can conclude that time sequence length is one of the factors that controls starting error, while the real cause of the error is still to be found.

6. Conclusion

In this project, we saw how language dataset preprocessing helps makes the prediction more accurate, and how TF-IDF outperform stopwords removal. We also saw that despite of the sparsity of the text vector, extra information does have positive influence on the prediction accuracy. Thirdly, with the joint effect of text, time sequence length is also one important factor in prediction. We found out that the ideal time range for financial news headline and stock price is between one to two weeks. Lastly, the performance of LSTM provides higher accuracy compare to BLSTM over our dataset. We also observed a common trend indicates that longer time sequence is one factor to reduce the beginning prediction error in LSTM.

6.1 Future Work

There are two possible future work for this project: prediction error reduction in sequence start and attention.

Prediction Error Reduction in Sequence Start

As shown in many graphs in the project, there is almost always a spiking high prediction error at the beginning of the dataset. Although we tried to figure out why, the reason is still unknown. The only trend we observed so far is, the larger the time sequence, the lower the error. However we couldn't find out the real reason to this. Thus this would be something interesting to work on to find out the cause.

Attention

Despite the great performance of LSTM, it is criticized for requiring large amounts of computing powers. Recently, a so-called Attention mechanism has been introduced [14] that consumes less computing power to train models in comparison to LSTM.

One of the early attempt of the project was to see the joint effect of the stock market. However, the large requirements of computing power slowed down progress significantly. Eventually we narrowed down the focus to consider only one stock.

It would be great to try applying attention model to reduce the hardware requirements for training and be able to try more stocks in the training dataset to look at joint effects of the stock market to improve the performance of our model.

6.2 Reflective Essay

The project has been really challenging, as it requires deep learning application while I didn't have experience doing deep learning project. However, I enjoyed the process and used the opportunity to really learn some of the structures and ideas behind LSTM and deep learning.

The project experience enables me to understand more domain knowledge in LSTM and BLSTM. I look forward to keep studying the related knowledge in the future thanks to this project.

Last but not least, special thanks to Miquel, my supervisor for the project, for his patient support and help during the whole project.

References

[1] S&P 500 stock data: Historical stock data for all current S&P 500 companies. Retrieved April 27, 2019, from https://www.kaggle.com/camnugent/sandp500

[2] Apple Inc (AAPL.OQ) News. Reuters.com. Retrieved April 24, 2019, from https://www.reuters.com/finance/stocks/company-news/AAPL.OQ

[3] Algorithmic Trading using Sentiment Analysis on News Articles. Retrieved May 21, 2019, from https://towardsdatascience.com/https-towardsdatascience-com-algorithmic-trading-using-sentiment-analysis-on-news-articles-83db77966704

[4] Hochreiter & Schmidhuber, "LONG SHORT-TERM MEMORY", Neural Computation 9(8):1735-1780, 1997.

[5] Understanding LSTM Networks. Retrieved May 2, 2019, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[6] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks", IEEE Transactions on Signal Processing, vol. 45, no. 11, November 1997

[7] Cheerio: Fast, flexible, and lean implementation of core jQuery designed specifically for the server.
https://github.com/cheeriojs/cheerio#readme

[8] Geoffrey Hinton. Neural Networks for Machine Learning Online Course. Retrieved May 27, 2019, from https://www.coursera.org/learn/neural-networks/home/welcome

[9] Natural Language Toolkit
https://www.nltk.org/

[10] A. Graves and J. Schmidhuber, "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures," Neural Networks, vol. 18, no. 5-6, pp. 602-610, June/July 2005.

[11] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. arXiv preprint arXiv:1804.07612, 2018.

[12] Jason Brownlee, How to Configure the Learning Rate Hyperparameter When Training Deep Learning Neural Networks, Better Deep Learning, January 23, 2019. Retrieved April 30, 2019, from https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

[13] Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016. Retrieved May 29, 2019, from http://www.deeplearningbook.org

[14] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In ICML.

[15] Rizwan Khan. Understanding Root Mean Square Prop or RMSprop Algorithm. Retrieved May 27, 2019, from https://www.datacamp.com/community/news/understanding-root-mean-square-prop-or-rmsprop-algorithm-k9mujc5ywr

[16] Vitaly Bushaev. Understanding RMSprop — faster neural network learning. Retrieved May 23, 2019, from https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a

[17] Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. Hasim Sak, Andrew Senior, Francoise Beaufays.

[18] Dynamic Recurrent Neural Network Tensorflow Example. Retrieved May 6, 2019, from https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/3_NeuralNetworks/dynamic_rnn.py

[19] Simple Reinforcement Learning with Tensorflow Part 0: Q-Learning with Tables and Neural Networks. Retrieved June 2, 2019, from https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-0-q-learning-with-tables-and-neural-networks-d195264329d0

[20] Predict Stock Prices Using RNN: Part 1. Retrieved May 14, 2019, from https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html#normalization

[21] Sequence Modeling: Recurrent and Recursive Nets: Chapter 10. Retrieved April 4, 2019, from https://www.deeplearningbook.org/contents/rnn.html

[22] Apple Records. In Wikipedia. Retrieved June 2, 2019, from https://en.wikipedia.org/wiki/Apple_Records

[23] Zhang Y, Jin R, Zhou ZH. Understanding bag-of-words model: a statistical framework. Int J Mach Learn & Cybe 2010;1(1):43-42.

[24] J. Ramos. Using tf-idf to determine word relevance in document queries. In ICML, 2003

[25] Rohith Gandhi. A Look at Gradient Descent and RMSprop Optimizers. Retrieved June 2, 2019, from https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b

[26] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[27] Apple Inc. Stock Quote. Yahoo Finance. Retrieved June 7, 2019, from https://finance.yahoo.com/quote/aapl/