Using Recurrent Convolutional Neural Network (RCNN) to Predict S&P 500 Movements

Stanford CS224N Custom Project

Xiaoye Yuan
Institute for Computational
and Mathematical Engineering
Stanford University
`chamus@stanford.edu`

Carolyn Kao
Institute for Computational
and Mathematical Engineering
Stanford University
`chkao831@stanford.edu`

Abstract

Financial news releases and time series are important data sources to predict market directions and movements. The existing forecasting methods mostly work with each input independently. With increasing computational capabilities and artificial intelligence techniques nowadays, we could combine the predicting power of both sentiment analysis and quantitative modelling. In this project, we utilize information from financial news headlines and technical indicators to extend the ability of predicting short-term equity market movements, achieving higher accuracies than a single source system, obtaining comparable results to the state-of-the-art recurrent-convolutional models that are designed for such tasks, further coming up with a novel approach in refining the baseline model architecture.

1 Introduction

To be a successful investor, one needs to study market behavior and develop the ability to forecast the market. Such tasks could be challenging since the short-term volatility is affected by various unexpected events in reality. As a result, market sentiments may change rapidly and prices move fast.

Fundamental analysis and technical analysis are two traditional methods to predict financial market movements. Being more long-term focused, the fundamental analysis evaluates a security's intrinsic value by examining related economic and financial factors including the balance sheet, strategic initiatives, micro-economic indicators, and consumer behavior. Alternatively, the technical analysis constructs mathematical and statistical models to forecast the direction of prices through analyses of the historical market data such as price and volume. Despite its capability of handling data with different frequency, the technical analysis can not make use of unstructured data, e.g. financial news. Nowadays, thanks to the increasing computational capabilities and techniques to deal with massive dataset, people have started to combine the historical time series of technical analysis with fundamental financial textual data, applying quantitative modeling as well as machine learning techniques to extend the ability of predicting short-term equity market movement.

Our goal in this work is to successfully implement some prediction models that involve the convolutional architecture, followed by the recurrent neural network, to predict the intraday directional-movements in financial time series from scratch. Using a set of technical indicator (as further illustrated in Table 1) and sequences of news titles published the day before the prediction day as inputs, we aim at utilizing deep learning methods to detect and analyze complex patterns and interactions in the data in order to make the most out of our trading process. Based on the efficient market hypothesis, the market would adjust quickly to absorb new information as the stock prices sufficiently reflect all new available information in such a semi-strong market, making it theoretically impossible for investors to benefit by trading on new information. However, practically speaking, we would like to apply deep learning methods (in particular, NLP techniques) to show that it is possible to outperform the weakly efficient market using hybrid models with the help of the financial news time series and the short temporal effect from the news articles in the financial market.

2 Related Work

To explore the the stock market with sentiment analysis, some authors propose the use of hybrid models by integrating the text mining techniques that extract information relevant to the forecasting process and the financial market indices to perform the prediction. Castillo et al. [1] collect the market indices related to a number of companies and incorporate them with stock-market events for those companies and features extracted from the micro-blogging messages. Additionally, in the paper "Combining News and Technical Indicators in Daily Stock Price Trends Prediction," Zhai et al. [2] present a system that combines the information from both related news releases and technical indicators to enhance the predictability of the daily stock price trends. Their work presents robust systems that achieve higher accuracies and returns than a single source system does. This further lays a solid foundation towards our incorporation of the seven technical indicators (as in Table 1). Specifically, the extracted indicators we use in this work include the Stochastic %K, Stochastic %D, Momentum, Rate of Change, William's %R, A/D Oscillator, and Disparity 5, which are commonly used to describe assets. Furthermore, those papers confirm the positive influence of the use of a hybrid input, leading to the conclusion that both sources of information are relevant.

With this in mind, we dive deeper into more published work that utilizes deep learning models with some representation methods such as event-embedding and word-embedding for financial time series. The authors in [3] present various algorithms, including logistic regression, support vector machine, and LSTM, to extract information from the news headline corpuses and predict intraday movements in the S&P 500. Moreover, Ding et al. [4][5] build models with complex characteristics of words or events with lower-dimensional dense vectors, further showing that the performance of daily prediction is superior than weekly or monthly prediction.

Our project was heavily inspired by the paper "Deep learning for stock market prediction from financial news article" by Vargas et al. [6]. This paper proposes methods in various forms of Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) that show improvements (in terms of accuracies) from previous studies [5]. Particularly, unlike traditional financial natural language processing models which only use news articles as input, this paper proposes a new hybrid model by taking both news articles as well as technical indicators, which represent features generated from historical financial time series, as input data. Regarding the text representation, the model uses sentence embedding instead of word embedding, resulting from the word sparsity in empirical financial news datasets. Their proposed models perform a mean operator over the sequence of word embedding vectors and produce better results compared to the word embedding models. Those news are extracted from the day before the forecasting day, and the resulting models outperform a set of models that uses news from the past day, week and month, reinforcing the hypothesis that the information in the news articles have a short temporal effect in the financial market. Moreover, in terms of the model architecture, with a combination of technical indicators and news titles, a Recurrent Convolutional Neural Network (RCNN) architecture is proposed in the paper. Aiming at forecasting the intraday directional movements of the S&P 500 market index, the RCNN model benefits from both CNN and RNN. This results from the fact that CNN has a relatively superior ability to extract semantic information from texts, yet RNN is better at capturing the contextual information and simulating complex temporal characteristics. Therefore, the model takes advantage of both fundamental analysis and technical analysis.

3 Approach

Overview: According to Vargas et al. [6], we design the recurrent convolutional neural network (RCNN) model to predict intraday directional-movements in financial market. In general, the model will take in financial news data and market indices, feed the preprocessed textual and numeric inputs into CNN and RNN architectures and output directional movement prediction(either increase or decrease) for next day's S&P 500 close price, i.e. the last price at which the stock traded during the regular trading day.

The input layer consists of a technical indicator layer[1] and an embedding layer. For the embedding layers, we implement our models such that we could choose from either word embedding or sentence embedding upon the setup. Word embedding utilizes the pretrained GloVe model [7] to encode any sequence of financial news headlines. Each word in the news title is represented by a 300-dimensional vector. Furthermore, linguistic regularities such as semantics and syntactic regularities will be encoded. On top of word embedding, sentence embedding would additionally average word vectors embedded in a title and take the mean as a sentence vector, hence solving the word sparsity problem. More embedding details would be provided later in Section 4.1.

Feature	Formula	Feature	Formula
Stochastic %K	$\frac{C_t - LL_n}{HH_n - LL_n}$	William's %R	$\frac{HH_n - C_t}{HH_n - LL_n} \times 100$
Stochastic %D	$\frac{\sum_{i=0}^{n-1} K_{t-i}}{n}$	A/D Oscillator	$\frac{H_t - C_{t-1}}{H_t - L_t}$
Momentum	$C_t - C_{t-4}$	Disparity 5	$\frac{C_t}{MA_5}$
Rate of Change	$\frac{C_t}{C_{t-n}}$		

Table 1: Summary of 7 technical indicators, $C_t/H_t/L_t$ are closing/high/low price at time t, MA is n day moving average, HH_n/LL_n are highest high and lowest low in past n days [2]

Following the input layer, we would perform several convolutional operations (such as convolution, max pooling, activation, dropout, etc.) that capture local information through the combinations of previous word or sentence embedding in a specific window and extract senmantic information from text. Sequentially, we would build recurrent layers with the help of Long Short-Term Memory (LSTM). The advantage of introducing the recurrent layer lies in its capabilities of catching the context information and simulating complex temporal characteristics. In addition, LSTM model controls the cell state by forget gate, input gate and output gate, which helps model the long-horizon dependencies and solve the vanishing/exploding gradient problem.

Lastly, we construct the output layer using a fully-connected dense layer. To stabilize the resulting probability distributions of output labels, we then utilize log softmax and negative log likelihood (NLL) in our experiments. The output layer would eventually return a tensor in the form of `[batch_size, 2]`, generating a binary output of `[1,0]` or `[0,1]` according to the specified batch size. A label `[1,0]` represents that the S&P 500 close price will increase in the following day and label `[0,1]` represents that the S&P 500 close value will decrease.

Baseline model: The architecture for our baseline model is presented in Fig. 1.

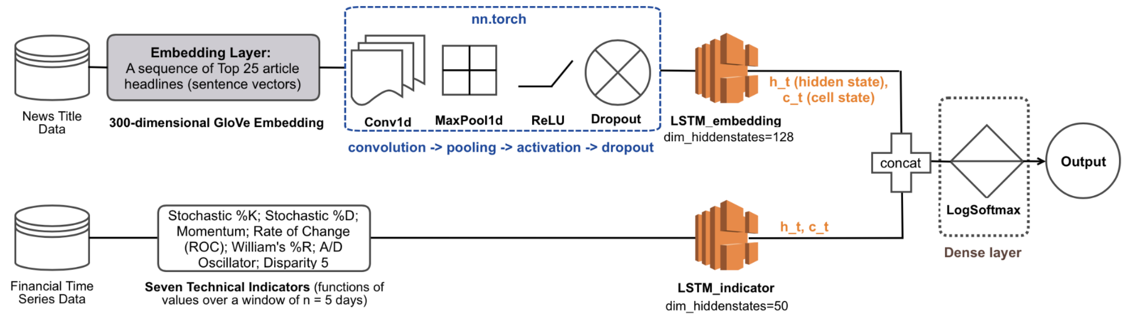

News Title Data
Embedding Layer: A sequence of Top 25 article headlines (sentence vectors)
300-dimensional GloVe Embedding
nn.torch
Conv1d
MaxPool1d
ReLU
Dropout
convolution -> pooling -> activation -> dropout
LSTM_embedding
dim_hiddenstates=128
h_t (hidden state), c_t (cell state)
concat
LogSoftmax
Dense layer
Output
Financial Time Series Data
Stochastic %K; Stochastic %D; Momentum; Rate of Change (ROC); William's %R; A/D Oscillator; Disparity 5
Seven Technical Indicators (functions of values over a window of n = 5 days)
h_t, c_t
LSTM_indicator
dim_hiddenstates=50

Figure 1: RCNN Baseline Model Architecture

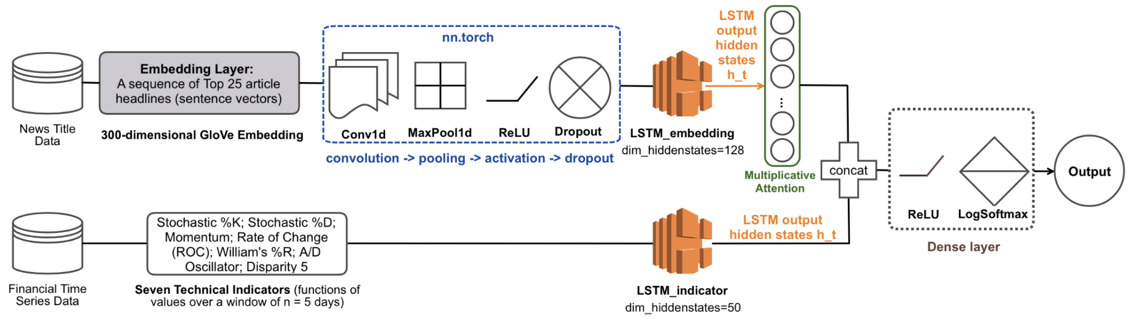
Following the word embedding layer, a 1-D convolution is firstly carried out, in which the size of the convolving kernel is 4 and the number of channels produced by the convolution (i.e. filter units) is 64. Then, we down-sample those data with 1-D maxpooling, with size of the sliding window of 2, stride of 1 and padding of 1. Afterwards, to introduce a non-linearity into the model, we utilize ReLU as

[1]As further specified in Table 1, the technical indicator layer is constructed by features including Stochastic %K, Stochastic %D, Momentum, Rate of Change, William's %R, A/D Oscillator and Disparity5 using S&P 500 data.

our activation function. Finally, in the last stage of the convolutional layer, we apply regularization with dropout probability of 0.5. Subsequently, in terms of the recurrent layers, two LSTMs are used to allow modelling the long-term dependencies in sequence data. In specific, the hidden state dimensions for the embedding LSTM (the upper LSTM in Fig. 1) and the indicator LSTM (the lower one) are 128 and 50. Lastly, we map the terminal hidden states and cell states to a binary probability distribution with softmax activation in the output layer.

While some details of implementation (e.g. number of units for LSTMs, size of delay window, etc.) are pointed out in the research paper [6], we implement all models from scratch due to the lack of source code bases.

RCNN Model with Multiplicative Attention on Title_LSTM: As presented in Fig. 2, by the end of the LSTM for titles, we pass its terminal hidden states (h_t) through an attention vector to help our model be attentive to the days in the window that most help in predicting the next day's movement. The deployed attention vector basically encodes a bilinear form of the query and the values and allows for multiplicative interaction of LSTM hidden states with the windowed-contents. Such an attention vector is designed to have weights sum to one. By selectively attend to these encoder hidden states, we give the attention model an improved sense of interpretability as each weight corresponds to the importance of a certain day in the window on the final binary output. Other than that, the remaining structural designs and implementations mostly follow the baseline architecture.



News Title Data
Embedding Layer: A sequence of Top 25 article headlines (sentence vectors)
300-dimensional GloVe Embedding
nn.torch
Conv1d
MaxPool1d
ReLU
Dropout
convolution -> pooling -> activation -> dropout
LSTM_embedding
dim_hiddenstates=128
LSTM output hidden states h_t
Multiplicative Attention
concat
ReLU
LogSoftmax
Dense layer
Output
Financial Time Series Data
Stochastic %K; Stochastic %D; Momentum; Rate of Change (ROC); William's %R; A/D Oscillator; Disparity 5
Seven Technical Indicators (functions of values over a window of n = 5 days)
LSTM output hidden states h_t
LSTM_indicator
dim_hiddenstates=50

Figure 2: RCNN Model with Attention on Title_LSTM Architecture

4 Experiments

4.1 Data

For historical news headlines, we utilize a Kaggle dataset [8] in which the daily top 25 headlines are collected from Reddit WorldNews Channel. Ranging from `2008-06-08` to `2016-07-01`, all news are ranked by its date and by how hot they are within a day. In total, there are 73537 available entries. For each day, the dataset also includes the label 1 or 0 that respectively indicate the rise or fall in the S&P 500 close value. Upon data collection, we preprocess the news data by tokenizing strings into words, removing non-alphabetic tokens such as punctuation, generating corpus with preferable data structure for usage, etc. In relation to our financial time series, we retrieve the S&P 500 index series through Yahoo Finance over the same period as our news headlines data. This includes the high, open, low, close and volumn information of S&P 500. With those obtained stock prices, we further calculate a set of seven technical indicators as presented in Table 1. Below Figure 3 plots the first 100 days' time series of the corresponding 7 technical indicators, which indicates the existence of obvious time dependence among S&P 500 prices.

In terms of the train, validation and test splitting, in chronological order, we allocate the first 80% of the data for training (around 1600 days), the next 10% for validation (around 200 days), and the remaining for testing.

Upon each experiment, we implement the embedding in two ways, such that we could choose from either word embedding or sentence embedding. For both embeddings, we represent valid English words in headlines utilizing GloVe, an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

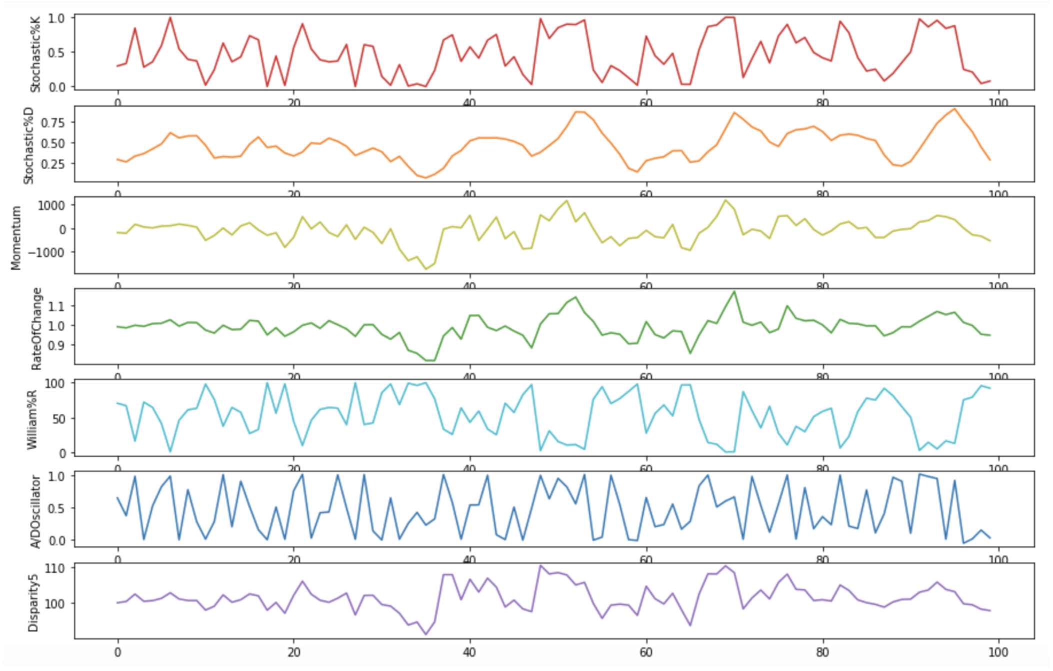


Figure 3: First 100 days' values of technical indicators

For word embedding, we first identify a global maximum length among headlines. Then we tokenize each word and find the 300 dimensional vector representation for it using GloVe. Finally, we construct a matrix representation of dimension (300, maximum length) for each headline. If the length of the headline is less than the global maximum length, we will pad the matrix to the desired size with zeros.

Due to the word sparsity in the data set, sentence embedding may be a better representation of the underlying textual structure. For sentence embedding, we also represent each word using a 300 dimensional vector. Instead of padding to the maximum sentence length, we use the average of all word vectors in a headline to represent the corresponding title.

4.2 Evaluation method

To evaluate the influence of using both news articles and technical indicators on predicting the movements of financial time series, we utilize the multiclass Cross-Entropy Loss objective (1), as specified in Vargas et al. [6], to train the model and evaluate the performance.

$$\mathscr{L} = \frac{1}{m} \sum_{i=1}^{m} y_i \log\left(\widehat{y}_i\right) + \left(1 - y_i\right) \log\left(1 - \widehat{y}_i\right) \tag{1}$$

We then implement the binary classification accuracy using Log Softmax (2) and Negative Log Likelihood Loss (NLL) (3) to obtain the binary classification accuracy on the predicted labels.

$$\operatorname{LogSoftmax}\left(x_i\right) = \log\left(\frac{\exp\left(x_i\right)}{\sum_j \exp\left(x_j\right)}\right) \tag{2}$$

$$\ell(x, y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n}} l_n, & \text{if reduction} = \text{'mean'} \\ \sum_{n=1}^{N} l_n, & \text{if reduction} = \text{'sum'} \end{cases} \tag{3}$$

Reduction specifies the reduction applying to the NLL output. In our experiments, we use the mean reduction, as the loss will not change if we alter the batch size.

4.3 Experimental details

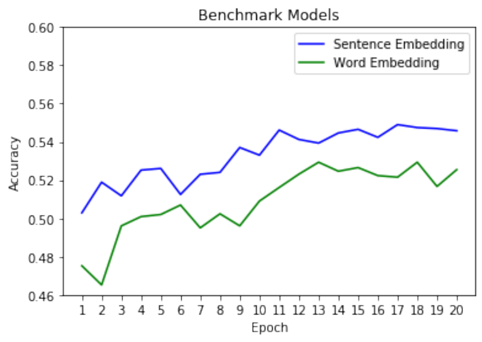
We train our RCNN models with batch size of 32, learning rate of 0.001, number of epochs of 20 and Adam optimizer. Then, we also use batch size of 128 in testing our models. The total running time for each model is less than 20 minutes under such settings. The algorithm we use to train the model is the Stochastic Gradient Descent, using momentum of 0.9.

4.4 Results

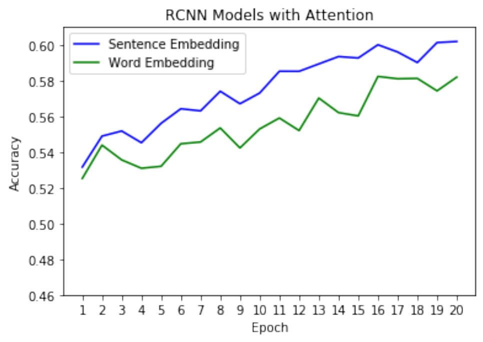
Our training results are listed in Table 2. Our model performances align fairly well with the published scores as attached in Appendix A. As previously noted, as the authors of [6] didn't release their detailed implementation or source codes, we suppose that such differences might result from various factors, including the usage of non-identical datasets, unequal choices of hyperparameters and other micro-structures in mechanisms, disparate methods of preprocessing the raw data, etc. Further investigation between Table 2 (our proposed work) and Table 3 (published work) illustrates that our proposed SI-RCNN-ATTN roughly outperforms or meets the best results among the scores from the published work, with the exception of the EB-CNN (Event-Embedding CNN) by Ding et al. [5]. Detailed analyses on a variety of embedding methods and reasons behind choosing specific models to implement would be later explained in Section 5. With that being said, we admittedly acknowledge the relatively subordinate numerical results in comparison to the published ones while we try our best to finetune and improve the model structures and parameters as time permitted, and further focus on coming up with a novel model structure (with attention), which distinguishes our work from the published work. Further investigation from the curves' shapes in Figure 4 and 5 indicates that there's no sign of overfitting in our experiments, as our models learn the training data quite well and succeeds to generalize the knowledge to test data.

Model	Abbreviation	Training	Test
RCNN-Baseline (sentence, with indicators)	SI-RCNN	55.91%	54.86%
RCNN-Baseline (word, with indicators)	WI-RCNN	53.92%	52.73%
RCNN-Baseline (sentence, without indicators)	S-RCNN	54.72%	53.18%
RCNN-Baseline (word, without indicators)	W-RCNN	51.89%	50.97%
RCNN-Attention (sentence, with indicators)	SI-RCNN-ATTN	60.18%	60.03%.
RCNN-Attention (word, with indicators)	WI-RCNN-ATTN	58.34%	58.09%.
RCNN-Attention (sentence, without indicators)	S-RCNN-ATTN	59.67%	58.29%.
RCNN-Attention (word, without indicators)	W-RCNN-ATTN	58.10%	57.83%.

Table 2: Train and Test Accuracies of S&P 500 Index Prediction for Different Models

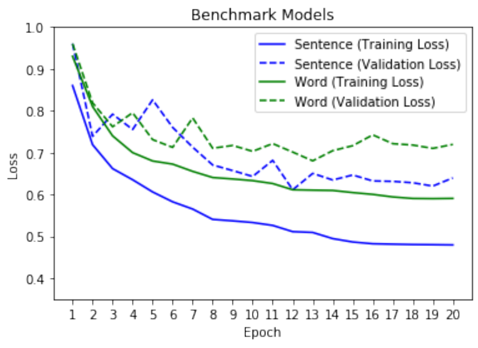


(a) SI-RCNN vs. WI-RCNN

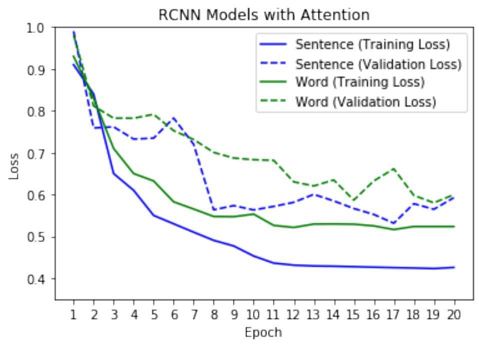


(b) SI-RCNN-ATTN vs. WI-RCNN-ATTN

Figure 4: Model Performance by Best Test Accuracies



(a) SI-RCNN vs. WI-RCNN



(b) SI-RCNN-ATTN vs. WI-RCNN-ATTN

Figure 5: Model Performance by Loss over Epochs

5 Analysis

Embedding is the way to represent textual data mathematically, taking into account the context and other surrounding words that individual word occurs with. In this project we work with 2 different embedding and examine their differences empirically. We find that sentence embedding can always outperform word embedding in terms of accuracy and efficiency. We interpret this result from two aspects, the first is that the length of news title varies a lot. Although padding help construct the embedding for each headline in same dimension, it does introduce additional padding zeros and sparsity structure. Also it's not computationally efficient to operate on headline matrices comparing to headline vectors. The second reason is that news headlines are different from news passages. It captures the most important content in a more precise way. Therefore, averaging each word vectors make sense given the news titles have a limited length. If the headline is too long, for example, over 100 words, it may include too much variation hence the average of word vectors become less meaningful.

The original paper[6] mentioned that the accuracy can be improved by incorporating event embedding, a more powerful method to represent the news headlines. It will extract the type of events happened and different parties involved in the news headlines. However, due to the lack of detailed implementation of how to construct event structure, we choose not to include event-driven embedding in this project. This explains the result that the RCNN model with attention using sentence embedding and technical indicator inputs brings the second best accuracy, only lower than EB-CNN model in table A, providing the room for future improvement.

During our experiments, we find that including seven technical indicators as inputs is necessary in terms of prediction accuracy. Comparing models with only textual inputs and both textual and numerical inputs reveals the influence of the sequence of technical indicators as complementary input is relevant, leading to a better performance. The reason behind is related to the essence of financial market. Although intra day movement is largely driven by market sentiment and investors' confidence, the technical indicators brings information about the fundamental values of asset prices, which prevents market over-reaction or under-reaction in some sense. As a consequence, including them as inputs of the second LSTM model enhanced the ability to produce more accurate estimation of the directional movement in a short time frame.

Speaking of the adoption of the multiplicative attention vector by the end of embedding-LSTM, our network is able to achieve superior performances with the superposed hidden states from different days in the time window considered, specific from today to the past-4 days, aiming to predict the next day. Applying attention gives us best performance in the binary classification. As seen in Figure 6, although starting out evenly across various windows, our trained models start to act attentively in a heavier sense to the the most recent windows as the training process goes on throughout the epochs. This indicates that the movement of the stock price in the financial market normally takes around 1 to 2 days to react to the news articles in forecasting the near-term market indices, suggesting that the weak-form Efficient Market Hypothesis does not hold strictly in practice as the market doesn't seem to perfectly absorb and reflect all new available information on the next day right away.

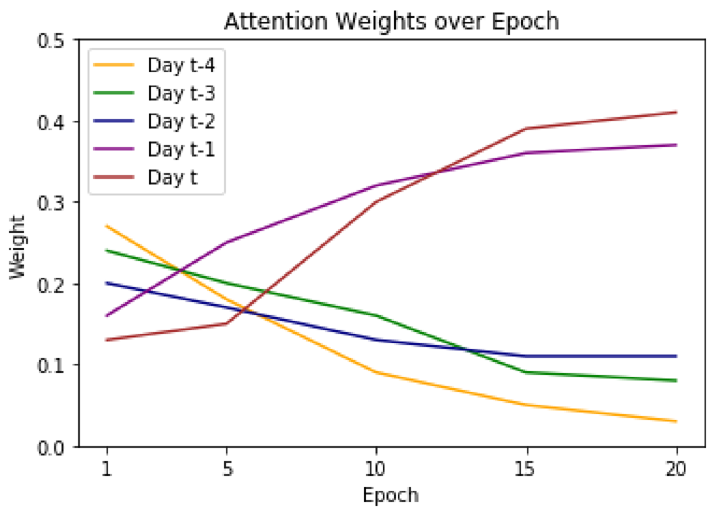


Figure 6: Attention Weight over Epoch for SI-RCNN-ATTN

6 Conclusion

This project presents the architecture of a RCNN model with attention mechanism using a hybrid inputs of both news title and market indices, which estimate the short-time market directional movements in an efficient manner. We examine various models with or without the attention mechanism or technical indicator inputs, and the effect from different embedding methodology for the financial news headlines. Among those variations, we find that technical indicators is positively related to better performance, hence the numerical input is relevant even in this Natural Language Processing setting. In addition, embedding the news headlines by averaging the vector representations for each word in the headline will outperform the model only with word embedding padded to the maximum news headline length. Furthermore, after including the attention mechanism, the model can achieve the highest accuracy among all models we implemented.

In general, financial market is unpredictable, especially under a fairly short time frame. Our best implementation, together with the original paper's implementation, still have limiting ability to achieve an ideal accuracy. It's more reasonable to interpret the prediction as an additional source of information, rather than a main guidance for any potential investment opportunity. For future improvement, we suggest exploring more implementing details about event-driven embedding, which is challenging but has the potential to improve the accuracy of intra day market movement prediction.

References

[1] Eduardo Ruiz, Vagelis Hristidis, Carlos Castillo, Aristides Gionis, and Alejandro Jaimes. Correlating financial time series with micro-blogging activity. pages 513–522, 05 2012.

[2] Yu Zhai, Arthur Hsu, and Saman Halgamuge. Combining news and technical indicators in daily stock price trends prediction. pages 1087–1096, 06 2007.

[3] Marc Velay and Fabrice Daniel. Using NLP on news headlines to predict index trends. *CoRR*, abs/1806.09533, 2018.

[4] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2327–2333. AAAI Press, 2015.

[6] Manuel Vargas, Beatriz Lima, and Alexandre Evsukoff. Deep learning for stock market prediction from financial news articles. pages 60–65, 06 2017.

[7] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[8] J. Sun. Daily news for stock market prediction, version 1., 08 2016. Data Retrieved on Feb 14, 2021, `https://www.kaggle.com/aaron7sun/stocknews`.

A Published Scores by Vargas et al.

Model	Training	Test
W-CNN	58,93%	57,22%
S-CNN	60,71%	60,96%
W-RCNN	59,76%	60,22%
S-RCNN	61,31%	61,49%
WI-RCNN	62,72%	61,29%
BW-SVM	56,42%	56,38%
E-NN	58,94%	58,83%
WB-NN	60,25%	-
WB-CNN	61,73%	60,57%
E-CNN	61,45%	-
EB-NN	62,84%	-
EB-CNN	65,08%	64,21%

Table 3: Baseline Results of S&P 500 Index Prediction [6]