SENN: Stock Ensemble-based Neural Network for Stock Market Prediction using Historical Stock Data and Sentiment Analysis

Louis Owen
Undergraduate Program in Mathematics
Faculty of Mathematics and Natural Sciences,
Institut Teknologi Bandung
Bandung, Indonesia
louisowen6@gmail.com

Finny Oktariani
Combinatorial Mathematics Research Group
Faculty of Mathematics and Natural Sciences,
Institut Teknologi Bandung
Bandung, Indonesia
f.oktariani@math.itb.ac.id

Abstract*—Stock market prediction is one of the most appealing and challenging problems in the realm of data science. In this paper, authors investigate the potential of exploiting sentiment score extracted from microblog text data along with historical stock data to improve the stock market prediction performance. The sentiment score is extracted by using an ensemble-based model which utilize the power of Long Short-Term Memory (LSTM) and Multi-Layer Perceptron (MLP) along with Convolutional Neural Network (CNN). We propose Stock Ensemble-based Neural Network (SENN) model which is trained on the Boeing historical stock data and sentiment score extracted from StockTwits microblog text data in 2019. Furthermore, we propose a novel way to measure the stock market prediction model performance which is the modification of the classic Mean Absolute Percentage Error (MAPE) metric, namely Adjusted MAPE (AMAPE). It has been observed from the experiments that utilizing SENN to integrate sentiment score as additional features could improve the stock market prediction performance up to 25% and also decreasing the margin of error up to 48%. With the training data limitation, the proposed model achieves a superior performance of 0.89% AMAPE. Our codes are available at https://www.github.com/louisowen6/SENN.

***Keywords*—stock market prediction, neural network, SENN**

I. Introduction

There are numerous factors responsible for the price changes in stock market, which makes the stock market prediction is a challenging task. To address this issue, there are 2 well-known methods which have a very different approaches, namely technical and fundamental analysis. Technical analysis exploit the mathematical indicators extracted from the historical stock data as well as the stock price itself to predict the price movement. This analysis is only focusing on the historical market data and ignoring the contribution from the other important factors, including macroeconomics, microeconomics, and political issues, as well as the sentiment of people about a particular company. Those ignored factors are in the realm of fundamental analysis.

In the last two decades, the emergence of machine learning research has grown exponentially. Moreover, the popularity of deep learning has exploded over the other machine learning methods in the last decade [1]. This phenomenon implicates that machine learning, especially deep learning, has widen its application field. Convolutional Neural Network (CNN), Multi-Layer Perceptron (MLP) and Long Short Term Memory (LSTM) are some of the most applicable deep learning methods in various fields. CNN is widely used in the computer vision field [2], as well as in the Natural Language Processing field [3]. MLP has an extremely wide range of applications, e.g., health [4], sport [5], finance [6], and many more. LSTM mainly applied in the sequential data, e.g., text data [7], voice data [8], and video [9].

It is shown that sentiment of people towards a particular company has a strong correlation with the stock price movement [10]. Therefore, extracting sentiment score from the public opinion arguably will enhance the stock market prediction performance. This kind of work has been done by utilizing the power of deep learning on the microblog and headline news text data [11]-[12]. Exploiting both historical stock data and sentiment score as the input to machine learning model also gives prominent result to the stock market prediction performance [13]-[15]. Shastri, Roy, & Mittal [14] treated this task as a regression problem, they proposed a neural network based model to predict the stock market price while the sentiment score is extracted through Naïve Bayes classifier. On the other hand, Picasso, Merello, Ma, Oneto and Cambria [13] proposed a robust predictive model able to forecast the trend of portfolio composed by the twenty most capitalized companies listed in NASDAQ100 index which takes indicators of technical analysis and sentiment of news articles as the input to the classification model. Al-Mansouri [15] explored the use of text mining, clustering, and machine learning models to develop a system that combines technical and sentiment analysis to predict the price movement of the S&P 100 cluster of stocks. Authors consider their works as a starting point for deeper exploration.

In this paper, authors aim to investigate the potential of exploiting sentiment score extracted from microblog text data along with historical stock data to improve the stock market prediction performance. First of all, authors pick out the most volatile stock to be worked with from the DJIA index by using a segmentation and voting methodology, and the output of this phase is Boeing stock. The sentiment score extraction phase is done by using an ensemble-based model which utilize the power of CNN, LSTM and MLP. Based on the extracted sentiment score and historical stock data, LSTM followed by fully-connected layers is used as the prediction head to predict the stock price. Our proposed architecture is called *Stock Ensemble-based Neural Network (SENN)*. The remainder of the paper is organized as follows: Section 2 explains our approaches to solve the stock market prediction task; Section 3 describes the available dataset

978-1-7281-8235-3/20/$31.00 ©2020 IEEE

along with experiment settings and result; Section 4 points out the conclusion and future works.

II. METHODOLOGY

In this paper, we will investigate the effect of exploiting sentiment score which is extracted from microblog text data as an additional feature on the stock market prediction performance. Long Short Term Memory Network (LSTM) followed by fully-connected layers is used as the prediction head. Fig. 1 shows the SENN architecture. The sentiment score is extracted by using an ensemble-based model which utilize the power of LSTM and MLP along with Convolutional Neural Network (CNN). There are four models which built the Stacked Ensemble model, including MLP Feature Driven, MLP Simple Word Embedding, CNN, and LSTM. Those four models have their own unique purpose. The first model aims to extract the information from manually-curated features, while the second one aims to extract the information directly from the vector representation of a particular text. We also want to extract the local and global information from a particular text, which is performed by the third and last model, respectively.

As we view the stock market prediction as the regression problem, we have to define the appropriate metric to measure the model performance. *Mean Absolute Percentage Error* (MAPE) is the well-known metric for regression problem. MAPE measures the model performance by computing the absolute percentage error, as defined in (1). However, this metric does not take into account the "direction" of the prediction results. Same "direction" means that the results of $pc(t + 1)$ - $pc(t)$ for the actual price and predicted price gives the same sign, where *pc* is the stock closing price, and vice-versa. To tackle that problem, we propose a novel way to measure the model performance in the stock market prediction task, namely *Adjusted Mean Absolute Percentage Error* (AMAPE) as defined in (2). AMAPE able to solve the aforementioned problem by multiplying function *c* to MAPE and divide it by two. Intuitively, this means that AMAPE will gives two times more penalty when the predicted price is in the wrong "direction" compared to when the predicted price is in the right "direction".

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| * 100 , \quad (1)$$

where n is the number of samples, y_i is the actual price and $\hat{y}_i$ is the predicted price.

$$AMAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{2 * y_i}\right| * c(y_i, y_{i+1}, \hat{y}_i, \hat{y}_{i+1}) * 100 , \quad (2)$$

where

$$c(y_i, y_{i+1}, \hat{y}_i, \hat{y}_{i+1}) = \begin{cases} 1, if\ sign(y_{i+1} - y_i) = sign(\hat{y}_{i+1} - \hat{y}_i) \\ 2, if\ sign(y_{i+1} - y_i) \neq sign(\hat{y}_{i+1} - \hat{y}_i) \end{cases}, \quad (3)$$

where

$$sign(x) = \begin{cases} 1, if\ x \geq 0 \\ 0, if\ x < 0 \end{cases}, \quad (4)$$

n is the number of samples, y_i is the actual price and $\hat{y}_i$ is the predicted price.

Furthermore, we also trained separate ensemble-based model which aims to extract the sentiment score from microblog text data which will be used in both of the classification and regression approaches. We use *Cosine Similarity* as the metric to assess the performance of that ensemble-based model.

Let $y_r \in \mathbb{R}^n$ denotes the dependent variable vector of the regression problem, where n denotes the number of samples, and let $X \in \mathbb{R}^{n*k}$ denotes the input matrix for the regression model, where k is the number of features. X is the sequence of vectors which is defined as in (5).

$$X = [x(1), x(2), x(3), ..., x(n)], \quad (5)$$

where $x(t) \in \mathbb{R}^k$ is the sample at time t, where $t \in \{1,2,3,..., n\}$.

Thus, given the q past input samples $x(t-q)$, $x(t-q-1)$, $x(t-q-2)$, …, $x(t-1)$, the $y_r(t)$ is defined as in (6), where $t \in \{q+1, q+2, q+3, ..., n\}$ and $q \in \{1,2,..., n-1\}$.

$$y_r(t) = pc(t), \quad (6)$$

where *pc* is the stock closing price. Note that the time-steps hyper-parameter q will be tuned during the hyper-parameter tuning phase.

Al-Mansouri [15] explore the use of clustering to improve the stock market prediction performance, authors also do the stock clustering using hierarchical agglomerative clustering algorithm with complete linkage along with Dynamic Time Warping (DTW) as the distance matrix. In our work, we do stock clustering in order to minimize the scope of study given the limited time and limited API access to extract the microblog text data, rather than to improve the stock market prediction performance. Volatility-based voting methodology is performed after we got the cluster of stocks. The voting is done using 2 volatility metrics, i.e., standard deviation and Adjusted Volatility (AV) which is defined as in (4).

$$AV(S) = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{N-1}\sum_{t=2}^{N}\left|po_k(t) - po_k(t-1)\right|\right), \quad (4)$$

where K is number of entities in cluster S, N is the length of the time-series data, and *po* is defined as the stock opening price. Based on these two metrics, we vote which cluster has the highest volatility. In this paper, we only worked with the chosen cluster because of the aforementioned motivations. Note that we allow one cluster consists of only one entity. The implementation of hierarchical agglomerative clustering with DTW is done by using *dtaidistance* [16] python package.

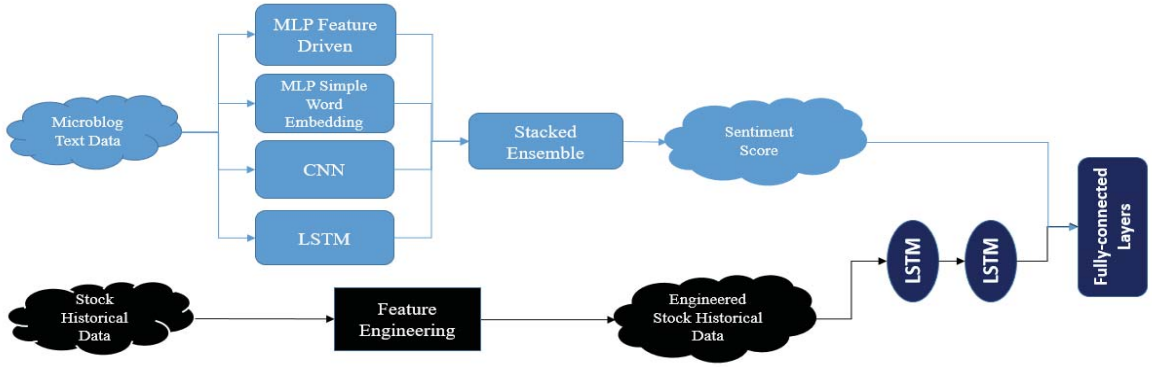

Fig. 1. SENN architecture.

As we have 2 different kind of data sources, namely historical stock data and microblog text data, we have to do different kind of data preparation for both of them along with the union of both of them.

A. Historical Stock Data

The historical stock raw data consists of ***Open***, ***High***, ***Low***, ***Close***, ***Volume***, and ***Date*** features. However, we have to generate more features in order to exploit more information from those raw features, including:

- ***close before n***, defined as the stock closing price on the *n* time-frame before, we choose *n*=1,2,3, …, 9,
- ***time***, defined as the integer-encoded results from the real time,
- ***weekday***, defined as the integer-encoded results from the real day of the week,
- ***true range***, defined as the True Range stock mathematical indicator,
- ***close diff Upper Bollinger***, defined as the differences between stock closing price and the Upper Bollinger Band,
- ***close diff Lower Bollinger***, defined as the differences between stock closing price and the Lower Bollinger Band,
- ***SMA indicator*** defined as the integer-encoded results from the Simple Moving Average stock mathematical indicator, where 1 denotes the condition when SMA_15 (SMA on 15 consecutive time-frame) is bigger than SMA_30, and 0 denotes when the condition is the other way around,
- ***Bollinger indicator***, defined as the integer-encoded results from the Upper Bollinger Band, Middle Bollinger Band and Lower Bollinger Band stock mathematical indicator, where 1 denotes the condition when pc (stock closing price) is bigger than the Upper Bollinger Band, 2 denotes the condition when pc is between Middle Bollinger Band and Upper Bollinger Band, 3 denotes the condition when pc is between Lower Bollinger Band and Middle Bollinger Band, and 4 denotes the condition when pc is smaller than Lower Bollinger Band.

Before exploiting all of those features to the model, it is important to normalize those features values. The normalization is done by transforming the values using the transformation function which is defined in (5). Note that the transformed values will be in the range of zero to one.

$$MinMaxScaling(x_i) = \frac{x_i - MIN(X)}{MAX(X) - MIN(X)}, \quad (5)$$

where X is the desired random variable.

Note that historical stock data is a time-series data. Thus, the data splitting methodology is different from the usual random train-test split or k-fold cross-validation. We have to performed time-series cross-validation in order to exploit the data as the input to the model in the training stage.

B. Microblog Text Data

Data preparation for microblog text data is divided into two, namely data preparation for the MLP Feature Driven model and data preparation for the MLP Simple Word Embedding model. For CNN and LSTM models, the input of the two models is the cleaned text which is done by removing HTML encoding, e.g., “&amp”, “&quot”; removing mention tag; removing RT (retweet); transforming all URL into “_url”; transforming slang, abbreviation, and elongated words into normal words using dictionary [17]; and transforming ordinal words into ordinal number, e.g., “first” is transformed into “1st”.

Data preparation for the MLP Feature Driven model is done by generating new features. Note that microblog text data is categorized as unstructured data, while the MLP model only accepts input in the form of numbers. Therefore, it is necessary to generate new features that are able to capture information from the cleaned text data, so that it can be exploited as the input for the MLP model. There are three types of manually-curated features will be generated from this phase: stock features, linguistic features, and sentiment lexicon features. There are twenty binary features in total for the first feature type and twenty-two features for the second feature type, including four binary features, three integer features, six verb-related *Part-of-Speech* tag features, one *Pointwise Mutual Information* feature, four *Term Frequency – Inverse Document Frequency* (TF-IDF) *n-grams* features, and four *RF n-grams* features. Definition of *RF n-grams* is defined same as the definition from [11]. The implementation of POS tag is done by using *StanfordNLP* [18] python package. For the latter feature type, we exploit the publicly available sentiment lexicon, including *AFINN*

[19], *BingLiu opinion lexicon* [20], *NRC Hashtag Sentiment Lexicon* [21]-[23], *General Inquirer lexicon* [24], and *SentiWordNet* [25]. There are nineteen new features aggregated for this type of feature as shown in Table I.

Data preparation for the MLP Simple Word Embedding is done by transforming the cleaned text into its word-embedding vector representation. Each word in a particular text is transformed into 300-dimensional vector by the help of *Google News Word2Vec* [26]. However, each text has different length and MLP only takes input with constant dimension. Thus, to tackle this problem, we utilize the average-pooling method, resulting a 300-dimensional vector representation for each text.

The usual k-fold cross-validation can be performed since we assumed there is no temporal dependency for the microblog text data. The full data is separated into three parts, namely train, validation, and test data. Each of the MLP Feature Driven Model, MLP Simple Word Embedding, CNN, and LSTM is trained on the train data using 10-folds cross-validation. Those four trained model will be used to predict both validation and test data. Thus, there are four prediction results for each the validation and test data. Then two new tables with five columns are created, each for validation and test data. Those columns consist of four predicted results from the four trained models along with the actual sentiment score. The newly created table from validation data will be exploited as the train data for the Stacked Ensemble model and the newly created table from the test data will be exploited as the validation data.

III. Experiments

The historical stock data is obtained using Yahoo Finance API with the help of *yfinance* [27] python package. Authors choose to scrapped all stocks which is listed in the Dow Jones Industrial Average (DJIA) Index during 2019, with one-hour lag time. On the other hand, the microblog text data which is exploited as input in the model creation stage is obtained using the python script prepared by *SemEval-2017* [28]. There are 1454 rows in the provided data which have been annotated with sentiment score by the financial experts. After the data is obtained, we have to do the stock clustering and voting on all of the thirty stocks listed in DJIA Index.

First, we have to perform Hierarchical Agglomerative Clustering with DTW as a distance matrix on all of those stock historical data. By choosing 661.18 as the distance threshold, we succeed to split those stocks into 11 clusters. *Second*, using the volatility-based voting methodology, we conclude that Boeing is the most volatile stock listed in DJIA Index during 2019. We considered only the most volatile stock segment because: 1) the returns coming from a small price difference are often nullified by the transaction cost, and 2) the large price difference will give more possibility to produce positive returns, even after cost reduction [13]. *Finally*, we collect the people's opinion about Boeing during 2019 which will be exploited as the input to the trained ensemble-based model. We scrapped those text data via the *StockTwits* API [29].

A. Historical Stock Data

First, we exploit only the Boeing historical stock data during 2019, which have been prepared based on the previous section, as the input to the model. We use LSTM followed by fully-connected layer as our model, as shown in Fig.1 (bottom part). We set the time-steps hyper-parameter q= 7. The model consists of 2 LSTM layers with 256 and 128 units followed by *Dropout Layer* (DL), respectively. A two -layered fully-connected network with 128 and 64 nodes in the first and second layer, respectively, is appended to the LSTM layers. The output layer consists of 1 node with a linear activation function. This model is trained with following settings: 350 *epochs*, 25 *batch size*, 0.0005 learning rate with *Adam* optimizer, and MSE loss function. The performance results of the model after being trained with 10-folds cross-validation can be seen in Table VI (left).

TABLE I. Sentiment Lexicon Features

Aggregation Strategy	AFINN	Bing Liu	NRC Hashtag	General Inquirer	Senti-Word-Net
Sum	V		V		V
Max	V		V		V
Min	V		V		V
Pos Ratio	V	V	V	V	V
Neg Ratio	V	V	V	V	V

B. Microblog Text Data

Then, we have to extract the sentiment score from the *StockTwits* microblog text data. There are 61 manually-curated feature generated for the input of **MLP Feature Driven (MLP FD)** from the raw text data. However, after we perform missing-value, constant and correlated feature analysis, there are only 55 remaining features which are ready to be exploited as the input for MLP FD model. The input for **MLP Simple Word Embedding (MLP SWE)** is the 300-dimensional vector representation generated from the text data. Both **CNN** and **LSTM** take the cleaned text, with 50 maximum words in each text, as the input. *Padding* is performed on the text with length less than 50. The architecture details along with hyper-parameters used for MLP FD and MLP SWE can be seen in Table II.

TABLE II. Architecture Details for MLP FD & MLP SWE

Architecture Details & Hyper-parameters	MLP FD	MLP SWE
Hidden Layers	3	
Nodes	50, 30, 15	30, 30, 30
Activation Function	*ReLU* for hidden layers, *Tanh* for output layer	
Dropout Layer	In each hidden layer (0.5)	
L2 Regularization	In each hidden layer (0.01 / 0.004)	
Epochs	250	200
Batch Size	32	
Learning Rate	0.0005	0.0001
Optimizer	*Adam*	
Loss Function	MSE	

The architecture of **CNN** is divided into two, namely the feature-extraction and fully-connected network. The visualization of the feature-extraction network is shown in Fig. 2. *First*, the input of 300-dimensional sentence-embedding vector is passed into the *Gaussian Noise* layer. *Then*, four kind of convolution is slide over 1,2,3, and 4 word(s) at a time. For each type, we employ 25 such convolution, which means there are 100 convolutions in total. Note that the size 1 convolution outputs 300-dimensional vector, size 2 outputs 299-dimensional vector, size 3 outputs 298-dimensional vector, and size 4 outputs 297-dimensional vector. *Third*, the *Tanh* activation layer is performed on the convolution output. *Fourth*, a 1-max pooling is performed on each of the convolution output, which resulting in 100 of univariate vectors. *Finally*, those univariate vectors are concatenated to form a single 100-dimensional feature vector which is exploited as the input to the fully-connected network. The architecture of the second part of this CNN can be seen in Table III. The **LSTM** model consists of *Gaussian Noise* Layer which is performed after the input embedding layer and two LSTM layers with 100 units followed by DL, for both layer. Then, it is followed by the fully-connected network which its architecture can be seen in Table III.

The performance results of the Stacked Ensemble building block model after being trained with 10-folds cross-validation can be seen in Table IV. All of those 4 models performs well where the average cosine similarity values are more than 0.5. For comparison, CNN performs best while MLP Feature Driven is the worst. However, regardless of their individual performance, we are still exploiting all of them since those 4 models has their own unique purpose.

After all of the four models has been trained, we exploit the prediction result from those models as the input to the **Stacked Ensemble** model, as discussed in the previous section. While each of those four models has their own goal, the Stacked Ensemble model aims to find the most appropriate weights which will be given to them, by utilizing the power of MLP. This model's architecture can be seen in Table V. After being trained on the train data, this Stacked Ensemble model gives 0.662 of *Cosine Similarity* when it is

TABLE III. FULLY-CONNECTED NETWORK ARCHITECTURE DETAILS FOR CNN & LSTM

Architecture Details & Hyper-parameters	**CNN**	**LSTM**
Hidden Layers	2	
Nodes	15, 15	50, 10
Activation Function	*Tanh* for all layers	
Dropout Layer	In each hidden layer (0.45 / 0.3)	
L2 Regularization	In each hidden layer (0.008 / 0.05)	
Epochs	75	350
Batch Size	32	
Learning Rate	0.0005	0.0001
Optimizer	*Adam*	
Loss Function	MSE	

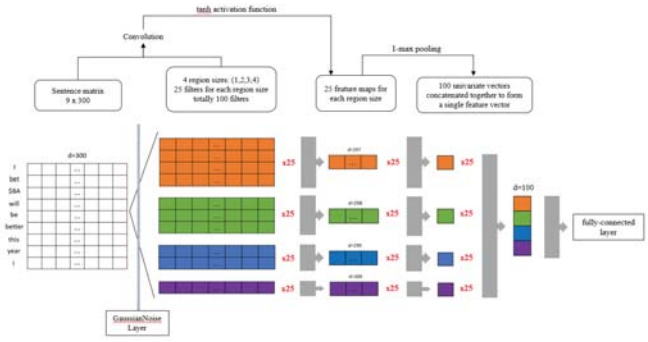

Fig. 2. CNN Feature Extraction Network for Microblog Text Data.

TABLE IV. PERFORMANCE RESULTS OF THE STACKED ENSEMBLE BULDING BLOCK

Model	**Cosine Similarity**	
	Average	**Standard Deviation**
MLP FD	0.534	0.079
MLP SWE	0.570	0.085
CNN	0.574	0.064
LSTM	0.546	0.084

TABLE V. ARCHITECTURE DETAILS FOR STACKED ENSEMBLE

Architecture Details & Hyper-parameters	**Value**
Hidden Layers	1
Nodes	4
Activation Function	*ReLU* for hidden layers, *Tanh* for output layer
Dropout Layer	0.05
L2 Regularization	0.02
Epochs	300
Batch Size	32
Learning Rate	0.00075
Optimizer	*Adam*
Loss Function	MSE

tested on the validation data. The result is proven better than if we only performed standard average ensemble method, which gives only 0.638 *Cosine Similarity* score when it is tester on the same validation data.

C. *Union of Historical Stock and Microblog Text Data*

Finally, as we have mentioned in the beginning of this paper, we aim to investigate the potential of exploiting sentiment score by utilizing our proposed model, SENN. Thus, we have to merge the extracted sentiment score and the historical stock data which will be exploited to the prediction head, as shown in Fig. 1. By using the trained Stacked Ensemble model, we extract the sentiment score from the people's opinion towards Boeing stock during 2019 in *StockTwits* platform. Then, the extracted sentiment score will be merged with the Boeing historical data with one-hour lag-

time during 2019. However, the merging process is not straight-forward, because of the historical data is prepared in a one-hour lag- time format, while the text data scrapped from *StockTwits* is purely based on when people give their opinion. Thus, we have to aggregate the extracted sentiment score based on the time format of the historical data.

In this paper, authors use four aggregation statistics, namely *average*, *maximum*, *minimum*, and *standard deviation*. The aggregation is performed in seven distinct time range. For instance, to aggregate the score at 09:30 of January 3, 2019, the aggregation is performed for all of the extracted sentiment score after the 15:30 of January 2, 2019 until the 09:30 of January 3, 2019. To aggregate the extracted sentiment score at 10:30 of January 3, 2019, then the aggregation is performed after 09:30 until 10:30 of January 3, 2019. This procedure is also applied for the remaining combinations. Note that, it is possible that there is no sentiment score extracted in the given time range. We tackle this problem by imputing the missing value with the aggregated sentiment score in the previous time range.

In the end, there are four new features extracted from the microblog text data and ready to be exploited as the input to the prediction head, including ***std sentiment score***, ***max sentiment score***, ***min sentiment score***, and ***mean sentiment score***. Those features will be exploited as the input to the full-connected layer in the prediction head, as shown in Fig. 1.

We set the time-steps hyper-parameter $q = 7$. The model consists of 2 LSTM layers with 256 and 128 units followed by DL, respectively. A two-layered fully-connected network with 128 and 64 nodes in the first and second layer, respectively, is appended to the LSTM layers and takes four additional inputs from the microblog text data. The output layer consists of 1 node with a linear activation function. This model is trained with following settings: 275 *epochs*, 25 *batch size*, 0.0004 learning rate with *Adam* optimizer, and MSE loss function. The performance results of the model after being trained with 10-folds cross-validation can be seen in Table VI (right).

It is clearly observed from the Table VI that there is 25.08% reduction of average AMAPE followed by a large decrease of the standard deviation, 48.66%. The reduction of average AMAPE shows that SENN succeeded to push down the prediction error which means that exploiting sentiment score as the additional input to the model succeeded to enhance the stock market prediction performance. This result also aligned with [10], which states that sentiment of people towards a particular company has a strong correlation with the stock price movement. The large reduction of standard deviation also shows that SENN is more confident in doing a prediction, compared to the model which exploits only historical stock data as its input.

If we look closer to the results on Table IV, qualitatively, the newly proposed metric AMAPE shows the same behavior as the traditional MAPE with the difference only in their absolute value. This could be due to our proposed model has a good performance with not many data points are predicted in the wrong "direction". On the other side, if the prediction model has a bad performance in "direction" wise, the AMAPE will be closer to the MAPE value. It can be explained by looking at the ratio between MAPE and AMAPE. From (1) and (2), we can conclude that

TABLE VI. PERFORMANCE RESULTS COMPARISON BETWEEN HISTORICAL STOCK DATA AND SENN

Average (Standard Deviation)			
Historical Stock Data		**SENN**	
AMAPE	**MAPE**	**AMAPE**	**MAPE**
1.188 (0.561)	1.594 (0.721)	0.890 (0.288)	1.201 (0.361)

the value of AMAPE divided by MAPE is bounded in the range [0.5, 1], where the infimum will be reach when all of the data points are predicted in the right "direction", and the supremum will be reach in the other way around. The results from Table VI shows that the ratio between AMAPE and MAPE for both historical stock data and SENN model is around 0.75. This means that both of them has a quite good performance in "direction" wise.

IV. CONCLUSIONS

In this paper, we proposed a robust neural network model for stock market prediction problem, namely Stock Ensemble-based Neural Network (SENN). Our proposed model utilizing the power of CNN, LSTM, and MLP by exploiting historical stock and microblog text data in order to predict the stock market movement. We also proposed a novel metric to measure the performance of stock market prediction using the regression-based approach, namely *Adjusted Mean Absolute Percentage Error* (AMAPE). Based on the experiments, SENN outperforms the performance of similar architecture network on the stock market prediction problem.

ACKNOWLEDGMENT

The authors thank Marcus Wono Setya Budhi and Fajar Yuliawan for valuable advices and helpful discussion throughout this work.

REFERENCES

[1] K. Hao, "We analyzed 16,625 papers to figure out where AI is headed next," *MIT Technology Review*, 2020. [Online]. Available: https://www.technologyreview.com/s/612768/we-analyzed-16625-papers-to-figure-out-where-ai-is-headed-next/. [Accessed: 10- Mar- 2020].

[2] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017, doi: 10.1109/tpami.2016.2577031.

[3] H. Kim and Y. Jeong, "Sentiment Classification Using Convolutional Neural Networks," *Applied Sciences*, vol. 9, no. 11, p. 2347, 2019, doi: 10.3390/app9112347.

[4] K. Ravichandran, S. Arulchelvan and K. PeriyaKannan, "Perception of medical awareness of media analyzed by multilayer perceptron," *Journal of Media and Communication Studies*, vol. 10, no. 10, pp. 118-127, 2018, doi: 10.5897/jmcs2018.0627.

[5] K. Huang and K. Chen, "Multilayer Perceptron for Prediction of 2006 World Cup Football Game," *Advances in Artificial Neural Systems*, vol. 2011, pp. 1-8, 2011, doi: 10.1155/2011/374816.

[6] A. Victor Devadoss and T. Antony Alphonnse Ligori, "Forecasting of Stock Prices Using Multi Layer Perceptron," *International Journal of Web Technology*, vol. 002, no. 002, pp. 52-58, 2013, doi: 10.20894/ijwt.104.002.002.006.

[7] A. Angadi, S. Muppidi and S. Keerthi Gorripati, "Deep LSTM for Emoji Twitter Sentiment Exploration via Distributed Embeddings," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 11, pp. 1714-1718, 2019, doi: 10.35940/ijitee.k1521.0981119.

[8] V. Gupta, "Voice Disorder Detection Using Long Short Term Memory (LSTM) Model," *Arxiv.org*, 2018. [Online]. Available: https://arxiv.org/pdf/1812.01779.

[9] X. Jiang, K. Xu and T. Sun, "Action Recognition Scheme based on Skeleton Representation with DS-LSTM Network," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1-1, 2019, doi: 10.1109/tcsvt.2019.2914137.

[10] F. Audrino, F. Sigrist and D. Ballinari, "The impact of sentiment and attention measures on stock market volatility," *International Journal of Forecasting*, vol. 36, no. 2, pp. 334-357, 2020, doi: 10.1016/j.ijforecast.2019.05.010.

[11] M. Jiang, M. Lan and Y. Wu, "ECNU at SemEval-2017 Task 5: An Ensemble of Regression Algorithms with Effective Features for Fine-Grained Sentiment Analysis in Financial Domain," in *Proc. of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 888-893, doi: 10.18653/v1/S17-2152.

[12] D. Ghosal, S. Bhatnagar, M. Shad Akhtar, A. Ekbal and P. Bhattacharyya, "IITP at SemEval-2017 Task 5: An Ensemble of Deep Learning and Feature Based Models for Financial Sentiment Analysis," in *Proc. of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, 2017, pp. 899-903, doi: 10.18653/v1/S17-2154.

[13] A. Picasso, S. Merello, Y. Ma, L. Oneto and E. Cambria, "Technical analysis and sentiment embeddings for market trend prediction", *Expert Systems with Applications*, vol. 135, pp. 60-70, 2019, doi: 10.1016/j.eswa.2019.06.014.

[14] M. Shastri, S. Roy and M. Mittal, "Stock Price Prediction using Artificial Neural Model: An Application of Big Data", *ICST Transactions on Scalable Information Systems*, vol. 0, no. 0, p. 156085, 2018, doi: 10.4108/eai.19-12-2018.156085.

[15] E. Al-Mansouri, "Using Artificial Neural Networks and Sentiment Analysis to Predict Upward Movements in Stock Price," B.S. thesis, Dept. Computer Science, Worcester Polytechnic Inst., Worcester, Massachusetts, 2016.

[16] "dtaidistance", PyPI, 2017. [Online]. Available: https://pypi.org/project/dtaidistance/. [Accessed: 18- Jan- 2020].

[17] GitHub, 2014. [Online]. Available: https://github.com/coastalcph/cs_sst/blob/master/data/res/emnlp_dict.txt. [Accessed: 16- Sep- 2019].

[18] P. Qi, T. Dozat, Y. Zhang and C.D. Manning, "Universal Dependency Parsing from Scratch," in *Proc. of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, 2018, pp. 160-170. Available: https://nlp.stanford.edu/pubs/qi2018universal.pdf.

[19] F.A. Nielsen, "A new ANEW: evaluation of a word list for sentiment analysis in microblogs," in *Proc. of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, 2011, vol. 718, pp. 93-98. [Online]. Available: http://arxiv.org/abs/1103.2903.

[20] M. Hu and B. Liu, "Mining and Summarizing Customer Reviews," in *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004)*, Aug 22-25, 2004, Seattle, Washington, USA

[21] S. Kiritchenko, X. Zhu and S. Mohammad, "Sentiment Analysis of Short Informal Texts," *Journal of Artificial Intelligence Research*, vol. 50, pp. 723-762, 2014, doi: 10.1613/jair.4272.

[22] S. Mohammad, S. Kiritchenko and X. Zhu, "NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets," in *Proc. of the seventh International Workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, USA, 2013.

[23] X. Zhu, S. Kiritchenko and S. Mohammad, "NRC-Canada-2014: Recent Improvements in Sentiment Analysis of Tweets," in *Proc. of the eight international workshop on Semantic Evaluation Exercises (SemEval-2014)*, Dublin, Ireland, 2014.

[24] "Guide to General Inquirer Category Listings", Wjh.harvard.edu, 2000. [Online]. Available: http://www.wjh.harvard.edu/~inquirer/spreadsheet_guide.htm. [Accessed: 01- Oct- 2019].

[25] S. Baccianella, A. Esuli and F. Sebastiani, "SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining," in Proc. of the Seventh International Conference on Language Resources and Evaluation (LREC'10), Valletta, Malta, 2010. Available: http://www.lrec-conf.org/proceedings/lrec2010/pdf/769_Paper.pdf.

[26] "Google Code Archive - Long-term storage for Google Code Project Hosting.", Code.google.com, 2013. [Online]. Available: https://code.google.com/archive/p/word2vec/. [Accessed: 05- Oct- 2019].

[27] "yfinance", PyPI, 2019. [Online]. Available: https://pypi.org/project/yfinance/. [Accessed: 05- Jan- 2020].

[28] "SemEval-2017 Task 5", Alt.qcri.org, 2017. [Online]. Available: http://alt.qcri.org/semeval2017/task5/. [Accessed: 18- Aug- 2019].

[29] "Stocktwits for Developers", Api.stocktwits.com. [Online]. Available: https://api.stocktwits.com/developers/docs. [Accessed: 05- Jan- 2020].