Forecasting the Market with Machine Learning Algorithms: An Application of NMC-BERT-LSTM-DQN-X Algorithm in Quantitative Trading

CHANG LIU, JIE YAN, and FEIYUE GUO, Southwestern University of Finance and Economics
MIN GUO, China Great-Wall Asset Management Co., Ltd

Although machine learning (ML) algorithms have been widely used in forecasting the trend of stock market indices, they failed to consider the following crucial aspects for market forecasting: (1) that investors' emotions and attitudes toward future market trends have material impacts on market trend forecasting (2) the length of past market data should be dynamically adjusted according to the market status and (3) the transition of market statutes should be considered when forecasting market trends. In this study, we proposed an innovative ML method to forecast China's stock market trends by addressing the three issues above. Specifically, sentimental factors (see Appendix [1] for full trans) were first collected to measure investors' emotions and attitudes. Then, a non-stationary Markov chain (NMC) model was used to capture dynamic transitions of market statutes. We choose the state-of-the-art (SOTA) method, namely, Bidirectional Encoder Representations from Transformers (*BERT*), to predict the state of the market at time *t*, and a long short-term memory (*LSTM*) model was used to estimate the varying length of past market data in market trend prediction, where the input of *LSTM* (the state of the market at time *t*) was the output of *BERT* and probabilities for opening and closing of the gates in the *LSTM* model were based on outputs of the *NMC* model. Finally, the optimum parameters of the proposed algorithm were calculated using a reinforced learning-based deep Q-Network. Compared to existing forecasting methods, the proposed algorithm achieves better results with a forecasting accuracy of 61.77%, annualized return of 29.25%, and maximum losses of −8.29%. Furthermore, the proposed model achieved the lowest forecasting error: mean square error (0.095), root mean square error (0.0739), mean absolute error (0.104), and mean absolute percent error (15.1%). As a result, the proposed market forecasting model can help investors obtain more accurate market forecast information.

CCS Concepts: • **Computing methodologies** → **Machine learning algorithms**;

Additional Key Words and Phrases: Machine learning, forecasting, quantitative trading, eXogenous variables, *SOTA*

ACM Reference format:
Chang Liu, Jie Yan, Feiyue Guo, and Min Guo. 2022. Forecasting the Market with Machine Learning Algorithms: An Application of NMC-BERT-LSTM-DQN-X Algorithm in Quantitative Trading. *ACM Trans. Knowl. Discov. Data* 16, 4, Article 62 (January 2022), 22 pages.
https://doi.org/10.1145/3488378

This study is funded by Social Science Foundation in Sichuan (grant no.: SC20TJ022) and by Fundamental Research Funds for the Central Universities (grant no.: 2021YJG050).
Authors' addresses: C. Liu, J. Yan, and F. Guo, Southwestern University of Finance and Economics, 555 Liutai Avenue, Chengdu, Sichuan Province, 611130, China; emails: {changliu0826, 2227902357, 2896079743}@qq.com; M. Guo, China Great-Wall Asset Management Co., Ltd, 12 Gaoshengqiao Road, Chengdu, Sichuan Province, 610041, China; email: eileen1999@163.com.

1556-4681/2022/01-ART62 $15.00
https://doi.org/10.1145/3488378

1 INTRODUCTION

Quantitative trading (see Appendix [3] for full trans) using **artificial intelligence (AI)** and **machine learning (ML)** algorithms has become a common phenomenon in both academic society and investment professionals for the past decade. There are redundant existing studies in this line of research. For instance, Yoo et al. [2005] conducted a comprehensive survey on ML technologies used in market forecasting and reported the latest development, along with their advantages and disadvantages. Guresen, Kayakutlu, and Daim [2011] assessed the effectiveness of **neural networks (*NNs*)** in stock market index prediction, including **multilayer perceptron (*MLP*)**, dynamic **artificial NNs (*ANNs*)**, and hybrid *NNs*. Adhikari and Agrawal [2014] combined an *ANN* with a random walk model for financial time-series forecasting. Ebadati and Mortazavi [2018] improved an *ANN* by introducing a hybrid **genetic algorithm (GA)** into their study to predict stock prices and time series.

However, there are mainly two problems in the current applications of AI algorithms in quantitative trading as summarized by Shynkevich et al. [2017]. On the one hand, using past price information alone would generate less reliable results. Yoo et al. [2005] argued that including exogenous variables in forecasting models would considerably improve algorithms' forecasting accuracy. The exogenous variables should reflect other dimensions of information other than prices. On the other hand, current quantitative trading existing studies on AI applications focused on proposing a single method with optimized parameters to forecast the market trend for all market situations without considering the dynamic evolvement of market scenarios under which parameters may be substantially heterogeneous. Consequently, this problem may not only lead to slow convergence rates since optimum parameters are difficult to estimate for different segmented data but also make out-of-sample predictions unstable.

To be more specific, the current ML-based stock market forecasting methods suffer from the following issues. First, although sentimental factors are proved to be significant in predicting stock market trends [Wu. et al. 2007], they are still under-studied since most of the existing works covered only partial market emotions [Pinto and Filipa 2018] or indexing techniques such as **principal component analysis (PCA)** [Chen et al. 2014]. On the other hand, Hai et al. [2015] proposed an indirect method using sentimental topics for stock market trend prediction, which considered emotional proxy indicators rather than direct emotional measures. Second, although the transition in market status could be modeled by a stochastic process [Wang et al. 2017], seldom studies attempted to integrate comprehensive multi-dimensional market emotions to study the transition in market status. Furthermore, it would be more difficult to model market status transitions for immature stock markets such as the Chinese stock market [Jiang et al. 2009]. Finally, existing forecasting models seldom considered another parameter in modeling, such as the length of past data or the backtesting window. The situation is more complicated by the fact that different market statuses may require different lengths of backtesting windows, which makes the windows dynamic. Current studies only considered static windows [Zuo and Kita 2012a,b; Ticknor 2013; Cervelló-Royo et al. 2015; Patel et al. 2015a,b].

In this study, an application of a **non-stationary Markov Chain (NMC)** model based on **Bidirectional Encoder Representations from Transformers (*BERT*)** and **long short-term memory (*LSTM*)** algorithm for dynamical time frame with **deep Q-Network (*DQN*)** optimization using exogenous variables (*NMC-BERT-LSTM-DQN-X*) is proposed to address the above issues. First, the transition of market status, presented as the probabilities of specific market statuses, is estimated using the NMC-X model with exogenous variables measuring investors' emotions and attitudes toward the future stock market. Second, we use the BERT-transformer algorithm to build a fine-tunable pre-training model based on Transformer and then to predict the state of the market at time $t + 1$. Third, the *LSTM* method is used to estimate the dynamic backtesting time frame (see

Appendix [6] for full trans) using the output from the *NMC-X* model and the *BERT* model to estimate the probabilities of forget gates for the hidden cell state of the last layer. Finally, a **deep reinforced learning (DRL)** algorithm of DQNs is used to optimize the parameters in the *NMC-BERT-LSTM-X* algorithm toward optimum forecasting performance.

The study contributes to the literature in the following ways: first, the transitions of stock market status are modeled via the NMC-X process integrating investors' emotions and attitudes toward future market trends, which could provide more accurate estimation results. Second, the *BERT* model is designed to predict the state of the market at time $t + 1$, which is used as inputs of the *LSTM* algorithm. Third, instead of using static backtesting windows, this study uses the *LSTM* algorithm to assess the dynamic lengths of backtesting windows under different market states, which would produce flexible parameters for market forecasting. Fourth, the *DRL*-based *DQN* algorithm would be used to improve forecasting results, using the parameters of the *NMC-BERT-LSTM-X* algorithm as the action space and forecasting performance as the reward. Finally, the *NMC-BERT-LSTM-X* algorithm not only show better prediction results but also superior evaluation results than the existing algorithms based on the state of the art (*SOTA*) method (*BERT*).

In the remaining part of this article, Section 2 outlines the main theoretical framework. Section 3 presents the model, sample, and methodology. Section 4 explains the results of the study. Section 5 presents the conclusion.

2 RELATED WORKS

2.1 Network-Based Algorithms

Kasgari et al. [2013] discovered the ability of MLP in *ANNs* to predict bankruptcy. Adhikari and Agrawal [2014] combined *ANNs* with a random walk model for financial time-series forecasting. Using ML models (*NNs* and **support vector machine [*SVM*]**), Weng et al. [2017] combined traditional stock time series with technical indicators to propose an effective and intelligent stock market forecasting system with an accuracy rate of 85%. Ebadati and Mortazavi [2018] improved *ANNs* by introducing a hybrid method of GA into their study to predict stock prices and time series. Weng et al. [2018] have attempted high-frequency forecasting using ensemble methods and online data sources based on a knowledge base and an AI expert system. Zhou et al. [2018] presented a new hybrid empirical mode decomposition and factorization machine-based *NN* to predict stock market trends. Pang et al. [2018] proposed a deep *LSTM NN* with an embedded layer and an *LSTM NN* with an automatic encoder to predict the stock market. Matsunaga et al. [2019] used graph *NNs* to forecast the Japanese Nikkei 225 market index. Their solution for dynamical forecasting is to roll the prediction window forward. Ahana et al. [2020] used *NNs* to perform multi-step-ahead prediction of stock closing prices on an offline basis.

2.2 Non-Networks Based Algorithms

Asadi et al. [2012] claimed that using traditional AI algorithms in financial time series prediction is difficult and that integrated learning appears to achieve better performance when compared to traditional AI algorithms. Fortuny et al. [2014] used text-mining techniques to predict stock prices. Elsir et al. [2015] conducted a comparative study on regression, *ANNs*, and SVM technologies used in stock market forecasting. Patel et al. [2015] compared the performance of *ANNs*, SVM, random forest, and naïve Bayes on the prediction of stock price changes and explored the relationship between ML and these four methods. Chou and Nguyen [2018] used a sliding-window metaheuristic to predict the trend of stock price changes in Taiwan. Senapati et al. [2018] used a modified particle swarm optimization algorithm to construct an intelligent stock market price forecasting model. Zhang et al. [2018] proposed a novel approach for forecasting stock prices by combining support vector regression with the firefly algorithm. Kumar et al. [2020] focused on the application of computational intelligent approaches such as *ANNs*, fuzzy logic, GAs, and other evolutionary techniques for stock market prediction. Xie et al. [2021] used an SVM regression model optimized by an intelligent fuzzy algorithm to predict the situation of the securities market, which improves the prediction accuracy of the stock market.

2.3 Sequential Forecasting Algorithms

Maciel et al. [2015] suggested a recursive probabilistic fuzzy modeling approach to identify prediction systems affected by outliers and noisy data. The forecasting accuracy of market index volatility (**Mean Absolute Percentage Error (MAPE)** and **root mean square error (RMSE)** perspectives) was further improved by Kristjanpoller and Michell [2018] using **generalized autoregressive conditional heteroscedasticity (GARCH)** models with a combination of external variables, including investors' sentimental factors. Their results laid a solid ground for predicting emerging stock market conditions based on sentimental factors. Rangapuram et al. [2018] used *DeepState* to predict future time series. Nakagawa et al. [2019] proposed *DeepFactors,* reduced prediction variance, and significantly improved prediction accuracy. Cheng et al. [2018] developed a novel fuzzy time-series model based on rough set rule induction for forecasting stock index. Emotional or sentimental factors were critical in the prediction of market conditions, as reported by Martino et al. [2019]. Cheng et al. [2019] used sentiment information based on social networks' comments to predict stock returns. Yang et al. [2020] used a **deep learning (DL)**-based algorithm to predict online shopping behavior with consumers' sentiment dictionary mining. They claimed that the model with sentiment factors produced effective and superior forecasting results. Griffith et al. [2020] used threshold GARCH to study the impact of investors' sentiments on market returns and conditional volatility. Li et al [2020] analyzed investor sentiment in the stock market based on *BERT* model proved to be a *SOTA* method and they found that investor sentiment in online reviews had important impact on stock yield. Alaparthi and Mishra [2020] studied the relative effectiveness of four different emotion analysis techniques (Sent WordNet; traditional supervised ML model using logistic regression; *LSTM*; and *BERT*) and showed that undisputed superiority of *BERT* model in sentiment analysis for sequential forecasting. Jang et al. [2020] propose to an algorithm based on *BERT* to present an effective combination of the news information sentiment analysis and various macroeconomic indicators in order to predict the US Dow Jones Index. Li et al. [2021] analysed the sentiment of Chinese stock reviews based on *BERT* model and constructed a sentiment analysis model which has higher precision. Salisu et al. [2021] forecast the monthly realized volatility of the US stock market covering the period of February 1885 to September 2019 using a moving average heterogeneous autoregressive model.

2.4 Integrating Investors' Emotions and Attitudes in Financial Forecasting

The introduction of emotional factors improves the accuracy of financial forecasting. Nyman and Ormerod [2014] made a forecast of the stock market with Michigan Consumer Sentiment Index and found the prediction with the sentimental index more accurate. Nguyen, Shirai, and Velcin [2015] selected some emotional indicators from social media for analysis and established a model to predict stock prices considering investors' emotions and attitudes. The model was confirmed to be more accurate than a traditional stock prediction model solely based on historical prices. Xu and Zhou [2018] also verified that the future return of stocks in China's A-share market was positively related to emotional changes, especially in the prediction of smaller portfolios (see Appendix [8] for full trans) of securities. Breaban and Noussair [2018] considered emotional factors and forecasted the stock market for possible price bubbles and market crash possibilities.

In summary, there are three main issues in stock market forecasting: first, although investors' emotions and attitudes have a significant influence on stock market forecasting, they have been only partially integrated into the forecasting; second, the length of past market trend data should vary according to different market statutes; and third, the transition of market statutes should be considered when forecasting stock market trends. As a result, we proposed that investors' emotions and attitudes measured via sentimental factors are first collected to estimate the market status. Then, the dynamic transitions of market states are captured by *NMC* model. The input parameters of *LSTM* model are estimated by the *BERT* method, while the length of backtesting data is dynamically assessed by combining with *LSTM* model. Finally, the *DQN* algorithm would be used to optimize forecasting results, using the parameters of the *NMC-BERT-LSTM-X* algorithm as the action space and forecasting performance as the reward.

3 VARIABLES, MODELS, ARCHITECTURE, AND ALGORITHMS

As demonstrated by Gong and Lin [2018], exogenous variables such as sentimental and illiquid factors (see Appendix [2] for full trans) would provide incremental information on market prediction. Prediction models solely based on past price information could not produce accurate forecasting results either for returns or volatilities of assets, according to Bauwens and Otranto [2016]. As a result, sentimental and illiquidity factors are included as exogenous variables to enhance models' prediction performance. Section 3.1 describes exogenous variables, followed by Section 3.2, introducing the *NMC-LSTM-DQN-X* algorithm.

3.1 Exogenous Variables and Variables Screening

Several commonly used sentimental factors have been introduced by Tsang et al. [2007], Baker and Wurgler [2007], Kaniel et al. [2007], Gregory and Cliff. [2005], and Li and Wu [2010]. In this study, 21 most referred sentimental variables are adopted and reported in Table 1.

On the other hand, liquidity depends on the magnitudes of price movements if a large volume of the positions is to be executed. A groundbreaking definition of illiquidity factors was provided by Amihud [2002], and ever since, several studies have examined various versions of the factors. Among them, special attention should be paid to the works of Amihud [2002], Baker and Stein [2004], Asness et al. [2013], Fang and Peress [2009], Bekaert et al. [2007], Sadka [2006], Brav et al. [2010], Kumar [2010], and He and Krishnamurthy [2013] because most illiquidity variables are correlated according to Bauwens and Otranto [2016]. This study adopts Amihud's [2002] original definition of illiquidity factors with necessary revisions toward a high-frequency algorithm-trading mechanism. The modified definition of an illiquidity factor $ILLIQ_{i,t}$ for stock i on a day t is presented in Equation (1):

$$ILLIQ_t = \frac{1}{N} \sum_{i=1}^{N} \frac{|Rise\ or\ Fall|_{i,t}}{Trading\ Amount_{i,t}} \tag{1}$$

Where N represents the total number of observation periods for the illiquidity factor $ILLIQ_t$ for stock i on day t. In Equation (1), the numerator reflects the accumulation of price fluctuations during the observation period, and the denominator represents the transaction volume. Therefore, Equation (1) shows the price fluctuation of securities per unit volume. Specifically, with the same trading amount, the smaller the *ILLIQ*, the smaller the impact on the price of securities, and the better the liquidity of the security. In contrast, the greater the *ILLIQ*, the worse the liquidity. According to the "risk-benefit" matching principle, securities with poorer liquidity should demand a higher risk compensation. In other words, securities with higher *ILLIQ* values should have higher returns, as proved by Asness et al. [2013]. We attempt to verify this effect in the following sections.

Table 1. Sentimental Variable Pool

Names	Definition
E_1MEPSRevision	Prediction of EPS (1 month-on-month correction)
E_3MEPSRevision	Prediction of EPS (3 month-on-month correction)
E_6MEPSRevision	Prediction of EPS (6 month-on-month correction)
E_AvgEPSRevision	Prediction of EPS (average month-on-month correction)
E_AvgNetProfit12M	Prediction of net profit for 12 months
E_EPSF12Growth	Prediction of EPS for 12 months/EPS in the last year-1
E_EPSF1Std	The standard deviation of predicted EPS in first year
E_EPSFYGrowth	The standard deviation of predicted EPS in the second year/the standard deviation of predicted EPS in first year-1
E_F12EP	Prediction of net profit for 12 months/equity of calculation day/price
E_F12EPSChg	The changes of predicted EPS in 12 months
E_F12EPSChgM	The changes of predicted EPS in 12 months
E_F12EPSStdM	The standard deviation of predicted EPS in 12 months/average
E_F12NetProfitC	The changes of net profit for 12 months
E_F12PE	Price/prediction of net profit for 12 months/equity of calculation day (only net profit is positive)
E_F12ROE	Prediction of net profit for 12 months/recent shareholders' equity (only equity is positive)
E_NetProfitChg	Prediction of net profit for first year (month-on-month correction)
E_NetProfit12MD	Prediction of net profit for 12 months-net income attributed to shareholders in last year
E_NetProfit12MG	The growth of prediction of net profit for 12 months and net income attributed to shareholders in last year
E_NPF1MaxMin	Prediction of net profit for first year (max)/(min)-1

3.2 NMC-LSTM-DQN-X Algorithm

This section would introduce the *NMC-BERT-LSTM-DQN-X* algorithm. In summary, the entire architecture for the *NMC-LSTM-DQN-X* algorithm is as follows: first, the transition of market statutes, presented as the probabilities of specific market statutes, is estimated using the *NMC-X* model (i.e., NWC with exogenous variables) to measure investors' emotions and attitude toward the future stock market. Second, the *BERT*-transformer algorithm would be used to build a fine-tunable pre-training model based on Transformer, which predicts the state of the market at time $t + 1$. Third, the *LSTM* method is used to estimate the dynamic backtesting time frame using the output from the *NMC-X* model and the *BERT* model. Finally, the DRL-based *DQN* is used to optimize the parameters in the *NMC-BERT-LSTM-X* algorithm toward optimum forecasting performance. A detailed architecture of the algorithm is provided in Figure 1.

Assumptions for each sub-module are as follows:

For the NMC sub-module: (1) A homogeneous Markov hypothesis: it is assumed that the market state of unstable Markov chains (i.e., NMC) at any time t depends only on the state at time $t - 1$ of the previous time and has nothing to do with the state at $t - k$, for $k \geq 2$. (2) An observation independence hypothesis: it is assumed that the observation at any time depends only on the state of the Markov chain at that time and has nothing to do with other observations.

For the *BERT* sub-module: (1) randomly select pairs of transition probabilities P_{ij} at time t given by the NMC and replace them with a special hidden states h_a with symbol [MASK] at time t.

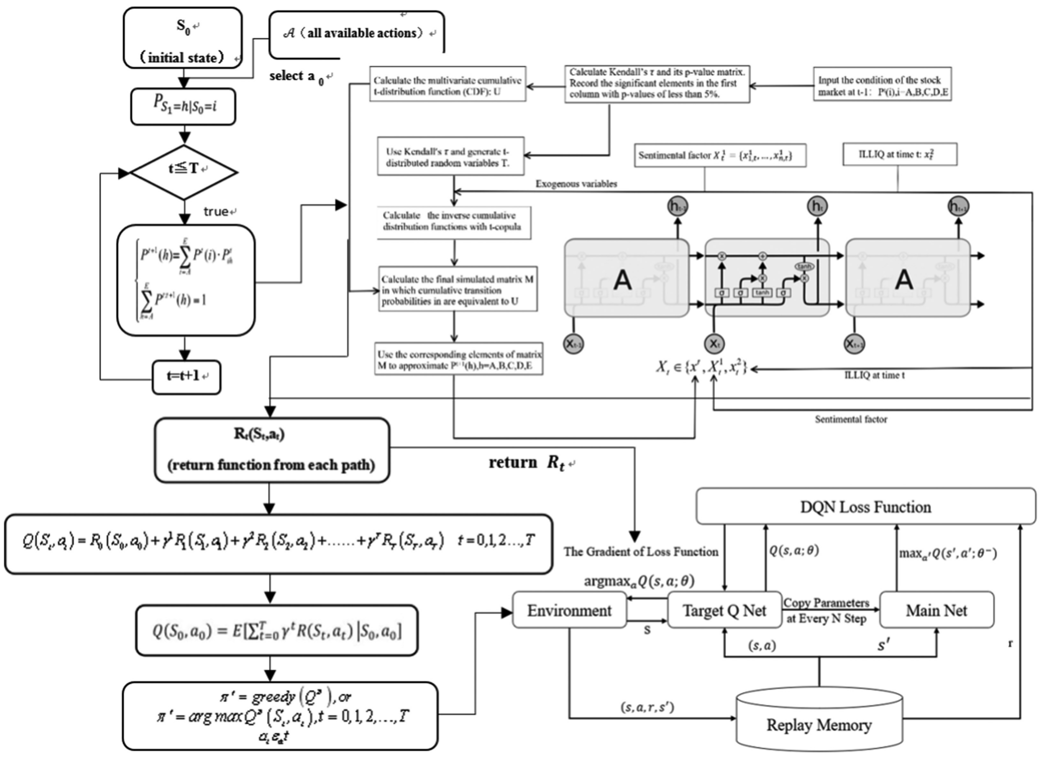
S0 (initial state)
A (all available actions)
select a0
t≤T
true
t=t+1
Calculate the multivariate cumulative t-distribution function (CDF): U
Calculate Kendall's τ and its p-value matrix. Record the significant elements in the first column with p-values of less than 5%.
Input the condition of the stock market at t-1: P(i),i=A,B,C,D,E
Use Kendall's τ and generate t-distributed random variables T.
Sentimental factor
ILLIQ at time t
Exogenous variables
Calculate the inverse cumulative distribution functions with t-copula
Calculate the final simulated matrix M in which cumulative transition probabilities in are equivalent to U
Use the corresponding elements of matrix M to approximate
ILLIQ at time t
Sentimental factor
Rt(St,at) (return function from each path)
return Rt
DQN Loss Function
The Gradient of Loss Function
Environment
Target Q Net
Copy Parameters at Every N Step
Main Net
Replay Memory

Fig. 1. Detailed technical architecture of *NMC-BERT-LSTM-DQN-X.*

(2) Input the *X* variables embedded in *Feed* function and update h_a at time $t + 1$. (3) The outputted h_a would be normalized and be ready to called in *LSTM.*

For the *LSTM* sub-module: (1) In comparison to one **recursive *NN (RNN)*** cell with only one *NN*, one *RNN* cell only needs to store one hidden layer state *h*. *LSTM* can materially reduce the complexity of the general *RNN* in modeling dependence over a long length of backtesting data. (2) The probability of the forget gate opening in the *LSTM* algorithm is an explicit function of the transition probability modeled in *NMCs.*

For the *DQN* sub-module: (1) The *DQN* has only one determinate value function network, and the action space is accountable. (2) The *NN* structure used by the *DQN* can be designed to estimate the action value (i.e., Q-value), which is an efficient improvement of the traditional *Q-Learning* algorithm.

In the following, Section 3.2.1 will introduce the NMC-X model, and then Section 3.2.2 will show the *BERT-X* model. After that, Section 3.2.3 introduces the *LSTM* algorithm that takes the output of *BERT* as input. Finally, Section 3.2.4 introduces the *DQN* algorithm model. In each section, the model, architecture and algorithm will be introduced.

3.2.1 Non-Stationary Markov Decision Process with Exogenous Variables (NMC-X). The Markov chain under a transition process is assumed to be non-stationary, where the states $\prod \in \{A, B, C, D, E\}$ represent the market' situations during the past three consecutive trading days: *A* is when the market is up more than 3%; *B* is when it is up 1%–3%; *C* is when it is −1% to 1%; *D* is when it is down 1%–3%; and *E* is when it is down over 3%. The states' definitions have no overlaps.

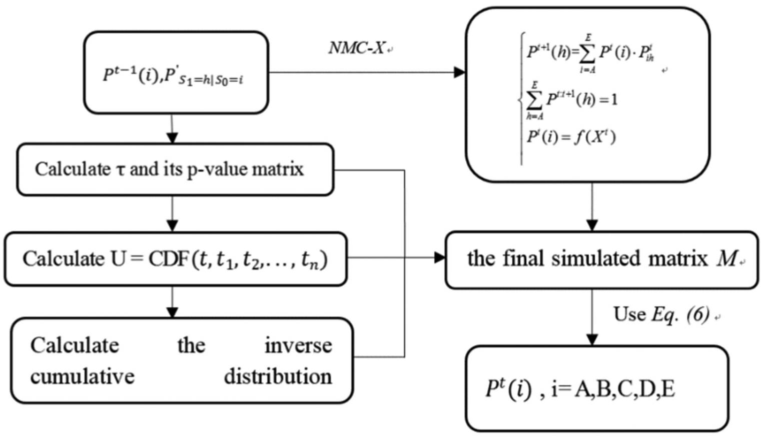
$P^{t-1}(i), P'_{S_1=h|S_0=i}$
NMC-X
$P^{t+1}(h)=\sum_{i=A}^{E} P^t(i)\cdot P^t_{ih}$
$\sum_{h=A}^{E} P^{t:t+1}(h)=1$
$P^t(i)=f(X^t)$
Calculate τ and its p-value matrix
Calculate U = CDF$(t, t_1, t_2, \ldots, t_n)$
the final simulated matrix M
Use Eq. (6)
Calculate the inverse cumulative distribution
$P^t(i)$, i=A,B,C,D,E

Fig. 2. NMC-X architecture.

According to Smith et al. (1996), long-range probability can be calculated as follows: = 1

$$\begin{cases} P^{t+1}(h) = \sum_{i=A}^{E} P^t(i) \cdot P^t_{ih} \\ \sum_{h=A}^{E} P^{t:t+1}(h) = 1 \\ P^t(i) = f(X^t) \end{cases} \tag{2}$$

where P^t_{ih} stands for the transition probability from state i to state h at time t, $i, h = A, B, C, D, E$. Let exogenous variables $X^1_t = \{x^1_{1,t}, \ldots x^1_{n,t}\}$ and x^2_t stand for sentimental factor and *ILLIQ* at time t, respectively. $x^1_{i,t}, i = 1 \ldots, n$ refers to the individual sentimental variables defined in Table 1. Algorithm 1 shows the steps of NMC-X estimation (explanations for the steps are provided between /* */ in the algorithm), whereas Figure 2 presents its architecture.

ALGORITHM 1: Calculation of $P^t(i), i = A, B, C, DE$ with t-copula simulation

```
Input: $P^{t-1}(i), P'_{S_1=h|S_0=i}, r^j_{t-1}\ q^j_{t-1} R^j_{t-1} R^{j'}_{t-1}, j = 1, \ldots, m,$
Output: $P^t(i), i = A, B, C, D, E$
    Step 1: Calculate $\tau$ and its $p$-value matrix. /*Improvethe matrix required for t-copula calculation*/
    Step 2: Use $\tau$ and generate t-distributed random variables $T = (t, t_1, t_2, \ldots, t_n)^T$ with degrees of
      freedom $v$
    Step 3: Calculate the multivariate cumulative t-distribution function
      $(CDF) : U = CDF(t, t_1, t_2, \ldots, t_n)$ /* Prepare to calculate the final matrix $M$ later*/
    Step 4: Calculate the inverse cumulative distribution functions /* Calculate the correlation of
      external variables in two situations*/
    Step 5: Calculate the final simulated matrix, $M$, by mapping the results from steps 3–5 in such a way
      that the joint cumulative transition probabilities in $M$ are equivalent to $U$ in Step 3
    Step 6: Use the corresponding elements of matrix $M$ and Equation (3) to approximate
      $P^t(i), i = A, B, C, D, E$
```

3.2.2 *BERT*.

BERT, which stands for Bidirectional Encoder Representations from Transformers, is originally designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. In this study, we use the BERT-transformer algorithm to build a fine-tunable pre-training model based on Transformer and use

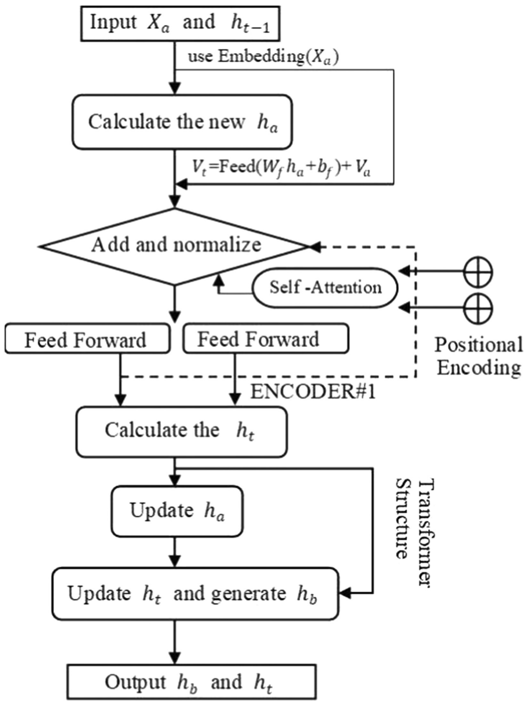

Fig. 3. *BERT* architecture.

its deeper structure to maintain extremely high accuracy of forecasting for the hidden states h_t which represents the hidden state of the market at time *t*, determined jointly by h_{t-1} and X_t. The procedure for *BERT* follows the below steps. First the initial state h_1, and h_a is determined by the residual structure:

$$h_a = h_1 + X_a \tag{3}$$

The normalized h_a is output to the forward propagation network and the output value of a Transformer is obtained again by a single residual structure. In the first level, h_t is calculated as follows:

$$h_t = Feed(W_f h_a + b_f) + h_a \tag{4}$$

where Feed is a linear function. The output value of a Transformer is represented by h_t and all the calculation processes in the Transformer are represented by Trans. That is, for any vector X_a input to the Transformer, denoted by the following equation:

$$h_t = Trans\,(W_t X_a + b_t) \tag{5}$$

Algorithm 2 shows the algorithm for Transformer Structure (explanations for the steps are provided between /* */ in the algorithm), whereas Figure 3 presents its architecture.

The structure of the transformer is shown in the figure.

3.2.3 Long Short-Term Memory with Exogenous Variables (LSTM-X). An ***LSTM* model with exogenous variables (*LSTM-X*)** is a time *RNN* suitable for processing and predicting important events with relatively long intervals and delays in time series. It is also a special type of *RNN* that can solve the gradient vanishing problem in *RNNs* effectively. Let $X_t \in \{x^t, X_t^1, x_t^2\}$ be the entire data for training the model, where x^t represents market data. $X_t^1 = \{x_{1,t}^1, \ldots x_{n,t}^1\}$ refers

ALGORITHM 2: Algorithm for Transformer Structure

```
Input: $X_a, V_{t-1}$
Output: $V_t, V_b$
    Step 1: Calculate the new $h_a$ : /*Update $h_a$ by $X_a$ and $h_{t-1}$*/
            $h_a =$ Positional_Encoding(Input_Embedding ($X_a$));
            $h_a = h_1 + X_a$;
    Step 2: generate parameters Q, K, V:
            $X_a = Q = K = V$;
    Step 3: Update $h_a$:
            $h_a =$ LayerNorm($h_a$+Multi-Head_Attention (Q, K, V));
    Step 4: Calculate the $h_t$ by $h_a$: /*Prepare to output $h_b$ and $h_t$*/ $h_t = Feed(W_f h_a + b_f) + h_a$;
    Step 5: Update $h_t$ and generate $h_b$ : $h_t = Trans(W_t X_a + b_t)$;
            $hb =$ Bert($WbXb + bb$);
    Step 6: Output $hb$ and $ht$ is output to the forward propagation network.
```

to sentimental exogenous variables, and x_t^2 stands for the *ILLIQ* exogenous variable at time *t*. h^t represents the hidden market state at time *t*, which is determined by h^{t-1} and X_t. o^t stands for the output of the model at time *t*, which is determined only by the current hidden state h^t. L^t indicates the loss function of the model. $f(t)$ shows the actual output of the training sample sequence at time *t*. *U, W*, and *V* matrices are the linear relationship parameters of our model. We input the hidden state at time $t-1$ and the training data $X_t \in \{x^t, X_t^1, x_t^2\}$ (including market data and exogenous variables) into the forget gate; we use the activation function sigmoid to calculate the output of the forget gate, which is the probability of forgetting the hidden cell state of the last layer:

$$f^t = S\left(W_f h^{t-1} + U_f P^t(X) + b_f\right) \tag{6}$$

where W_f, U_f, b_f represent the coefficients and constants of linear relations, and *S* is the sigmoid activation function, which is given just as the feed function of *BERT* algorithm in the last section:

$$\mathrm{S}(X) = Feed\left(W_f h_a + U_f P^t(X) + b_f\right) + h_a \tag{7}$$

Therefore, since $P^t(X)$ is given by Equation (2) of NMC and the sigmoid activation function is given by *BERT*, the probability of forgetting the hidden cell state of the last layer is estimated such that the transitions in stock market status are considered on the basis of exogenous variables measuring investors' emotions and attitude toward the future stock market status. As a result, the *LSTM* model in our architecture can calculate the transition probability for each gate (past recursive node location) with the sigmoid function *σ(net)* using the exogenous sentimental and illiquid information obtained by the NMC model. Algorithm 2 shows the steps of *LSTM-X* estimation (explanations for the steps are provided between /* */ in the algorithm), whereas Figure 4 presents its architecture.

3.2.4 Reinforced Learning DQN Algorithm. Future rewards from a Markov decision process are discounted by$\gamma$to obtain the Q function in the *DQN* algorithm. After optimization, the Q-Learning update formula becomes

$$Q^*(s,a) = Q(s,a) + \alpha(\gamma + \max_{a'} Q(s',a') - Q(s,a)) \tag{8}$$

Here, we need to determine the loss function based on *Q*-Learning. The loss function of the *DQN* is determined by Equation (6), and the gradient of *L(θ)* with respect to *θ* is given by the partial derivative of the loss function as presented in Equation (7).

$$L(\theta) = E\left[(TargetQ - Q(s,a;\theta))^2\right] \tag{9}$$

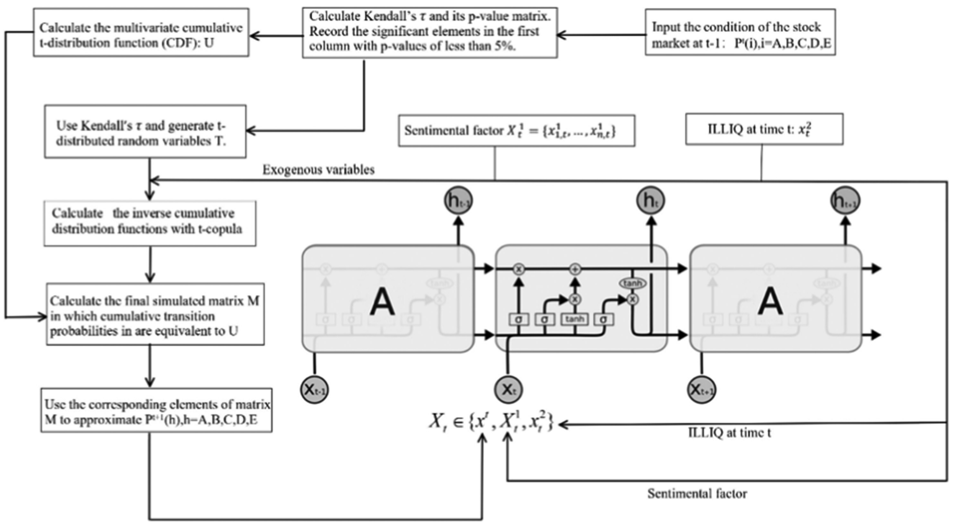
Calculate the multivariate cumulative t-distribution function (CDF): U
Calculate Kendall's τ and its p-value matrix. Record the significant elements in the first column with p-values of less than 5%.
Input the condition of the stock market at t-1: P'(i),i=A,B,C,D,E
Use Kendall's τ and generate t-distributed random variables T.
Sentimental factor $X_t^1 = \{x_{1,t}^1, ..., x_{n,t}^1\}$
ILLIQ at time t: x_t^2
Exogenous variables
Calculate the inverse cumulative distribution functions with t-copula
Calculate the final simulated matrix M in which cumulative transition probabilities in are equivalent to U
Use the corresponding elements of matrix M to approximate P^{t+1}(h),h=A,B,C,D,E
A
A
$X_t \in \{x^t, X_t^1, x_t^2\}$
ILLIQ at time t
Sentimental factor

Fig. 4. *LSTM-X* architecture.

ALGORITHM 3: Algorithm for *LSTM-X*

```
Input: $X_t \in \{x_t, X_t\}$, $h_{t-1}$ given by BERT
Output: $h_t$
Begin:
        Initialize $W, U, V$        /*the linear relationship parameters*/
                Define $tanh\,(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
        For episode = 1, $N$ do   /*construct the loop of the neural network*/
          Step1: Update forgetting export:
                $f_t = S(W_f h^{t-1} + U_f X_t + b_f)$/*concatenate and process input data using the
                sigmoid function*/
          Step2: Update the two parts of the output of the input gate:
                $i_t = S(W_i h_{t-1} + U_i x_t + b_i)$
                $a_t = tanh(W^a h_{t-1} + U^a x_t + b^a)$   /*add new information to the cell*/
          Step3: Update cell status:
                $C_t = C_{t-1} \odot f_t + i_t \odot a_t$   /*Combine the old cell information with the candidate
                cell information*/
          Step4: Update the output of the output gate:
                $o_t = S(W^o h_{t-1} + U^o x_t + b^o)$
                $h_t = o_t \odot tanh(C_t)$   /*Judge and output $h_t$ through the output gate*/
          Step5: Update the current sequence index prediction output:
```

$$\nabla_{\theta_i} L\,(\theta_i) = E_{s,a,\gamma,s'}\,[Y_i - Q(s,a|\theta_i))\nabla_{\theta_i} Q(s,a|\theta_i)] \tag{10}$$

where θ is the network parameter, whose target is

$$TragetQ = \gamma + \max_{a'} Q\,(s', a'; \theta) \tag{11}$$

Where s and s' represent sets of trading strategies, a represents trading action such as buy, hold, and sell, R represents trading rewards such as profits and loss, γ represents the discount factor, and θ represents a hyper-set of parameters, including sentimental factors.

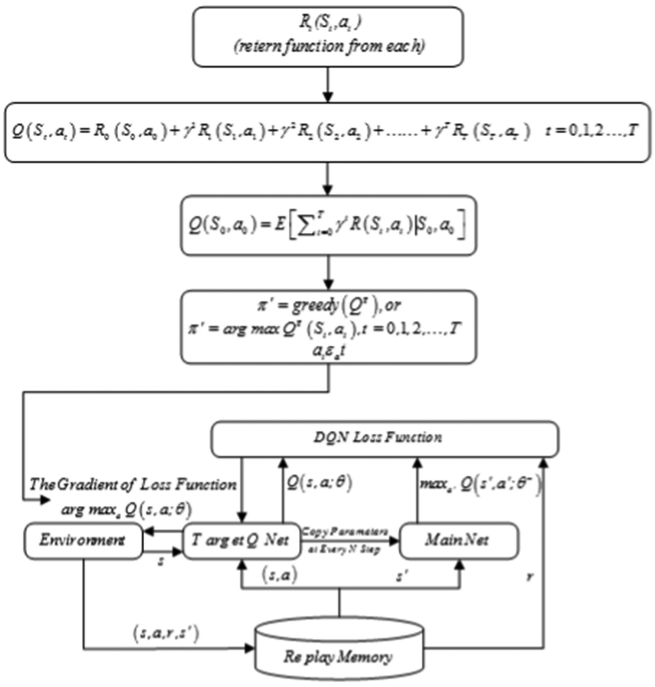
$R_t(S_t,a_t)$
(retern function from each)
$Q(S_t,a_t)=R_0(S_0,a_0)+\gamma^1R_1(S_1,a_1)+\gamma^2R_2(S_2,a_2)+......+\gamma^TR_T(S_T,a_T)\quad t=0,1,2....,T$
$Q(S_0,a_0)=E\left[\sum_{t=0}^{T}\gamma^tR(S_t,a_t)|S_0,a_0\right]$
$\pi'=greedy(Q^\pi)$, or
$\pi'=arg\,max\,Q^\pi(S_t,a_t), t=0,1,2,...,T$
$a_t\varepsilon_t t$
DQN Loss Function
The Gradient of Loss Function
$arg\,max_a\,Q(s,a;\theta)$
$Q(s,a;\theta)$
$max_{a'}\,Q(s',a';\theta^-)$
Environment
s
Target Q Net
Copy Parameters at Every N Step
Main Net
(s,a)
s'
r
(s,a,r,s')
Replay Memory

Fig. 5. The architecture of the *DQN* algorithm.

The *DQN* algorithm is based on DRL, which combines *DL* and **reinforcement learning (*RL*)** using convolutional *NNs*, along with the experience optimization mechanism (experience replay). It addresses the problem of instability when using a nonlinear functional approximator to approximate the value function. Algorithm 3 shows the steps of *DQN* estimation (explanations for the steps are provided between /* */ in the algorithm), whereas Figure 5 presents its architecture.

Where s and s' refer to sets of trading strategies, a refers to trading action such as buy, hold, and sell, R refers to trading rewards such as profits and loss, γ is the discount factor, and θ represents a hyper-set of parameters in *NMC-BERT-LSTM-X*.

4 EMPIRICAL RESULTS

4.1 Data Description

In this section, the models presented in Section 3 would be used to forecast the Shanghai-Shenzhen 300 index future (IFL09) and establish a market timing (see Appendix [4] for full trans) strategy. Shanghai-Shenzhen 300 index futures, a major futures contract underlying the Shanghai and Shenzhen 300 index (HS300) with a daily trading volume of over 60 billion RMB, are the main instrument used to hedge and speculate the stock market. We selected IFL09 daily close prices from April 16, 2010, to April 27, 2018. Figure 5 shows the trend of IFL09 daily close prices with trading days on the horizontal axis and the index points on the vertical axis.

As shown in Table 2, there are 1,953 observations, and the distribution for IFL09 daily close prices is heavily skewed to the right (skewness = 0.7551886), whereas the skewness for the normal distribution is zero (skewness = 0). The statistic is also fat-tailed such that Kurtosis is 3.471859, which means there are more extreme prices, and these prices are highly volatile.

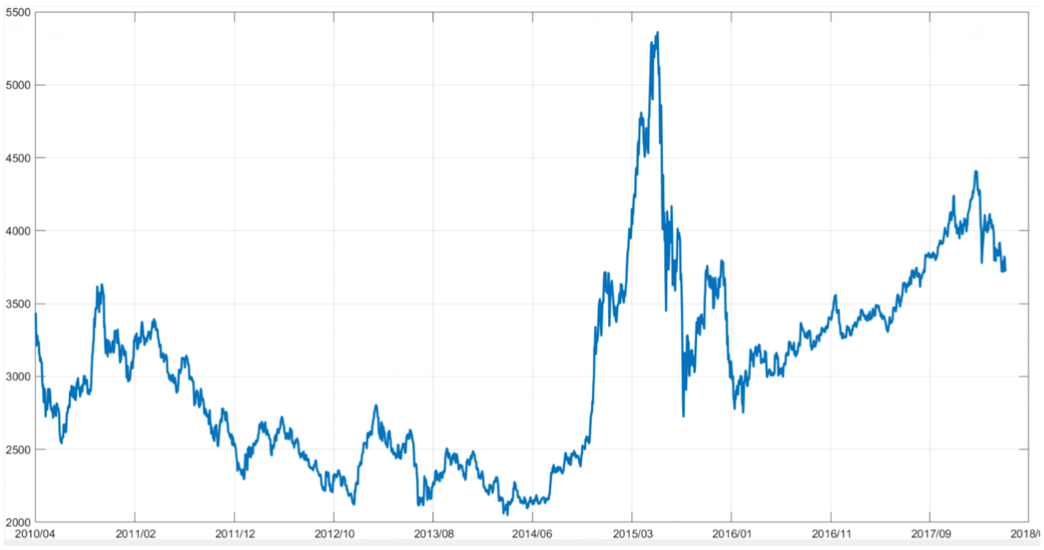

Fig. 6. IFL09 daily close prices from April 16, 2010 to April 27, 2018.

ALGORITHM 4: Deep Q-learning with experience replay.

```
Input: $Q, \theta$;
Output: $Q^*, a^*, s^*$;
Begin:
    Step 1: Initialize replay memory $D$ to capacity $M$;
    Step 2: Initialize action-value function $Q$ with exogenous variables $\theta$;
    Step 3: Initialize target action-value function $Q^*$ with $\theta\_ = \theta$;
    Step 4: For episode = $1, N$ do
    Step 5: Initialize sequences $s_1 = \{u_1\}$ and preprocessed sequence $\psi_1 = \psi(s_1)$;
    Step 6: For $t = 1, T$ do                                  /*iterate $step = 1, 2, \dots, T^*$/
    Step 7: Select a random action at with probability $\delta$;
    Step 8: select $a_t = max(a'Q(\psi(s_t), a; \theta)$;  /*use $\epsilon-greedy$ to generate action $a_t$ */
    Step 9:     Execute action at in emulator and observe reward $Rt$ and image $u_{t+1}$;
    Step 10:        Set $s_{t+1} = s_t, a_t, u_{t+1}$ and preprocess $\psi_{t+1} = \psi(s_{t+1})$;
    Step 11:        Store transition $(\psi_t, a_t, R_t, \psi_{t+1})$ in $D$;
    Step 12:        Sample random minibatch of transitions $(\psi i, ai, Ri, \psi i + 1)$;
    Step 13:        If episode terminates at step $i + 1$
    Step 14:        Set $y_i = \begin{cases} R_i \\ R_i + \gamma max_a Q^*(\Psi_{i+1} a; \theta^- \end{cases}$;
    Step 15:        Perform a gradient descent step on $(y_i, \dots, Q(a_i; \theta))$;
    Step 16:        Every $C$ steps reset $Q^* = Q$;
```

4.2 Factors Screening

We tested sentiment indexes using historical statistics to select the significant ones. **Intellectual capital (IC)** results of sentiment indexes are shown in Table 3.

Here, we provide two types of ICs. One is the normal type, and the other is calculated using residual returns. By comparing the two IC types, the following are observed: (1) these two types of ICs do not differ significantly; and (2) ICs calculated using residual returns are greater than the

Table 2. The Descriptive Statistics for IFL09 Daily Close Prices

Percentiles	Largest	Smallest	Statistic	Value
1%	2121.8	2028.6	Obs	1,953
5%	2173.2	2054.2	Mean	3028.975
10%	2275.8	2067.4	Std. Dev.	634.8356
25%	2489.8	2072.4	Variance	403016.2
50%	2973.6	3278.9	Skewness	0.7551886
75%	3403	5321.2	Kurtosis	3.471859
90%	3886.4	5328.6		
95%	4104.4	5359		
99%	4930	5368.4		

Table 3. IC Tests for Exogenous Variables

	IC Mean	Adj. IC	IC Mean	IC Std	Adj. IC	IC *t*-test
E_1MEPSRevision	0.0286	0.4275	0.0248	0.0637	0.389	0.5589
E_3MEPSRevision	0.0304	0.3122	0.0266	0.0899	0.2961	0.5673
E_6MEPSRevision	0.0277	0.2952	0.0261	0.0875	0.2985	0.5465
E_AvgEPSRevision	0.0348	0.3887	0.0306	0.084	0.3643	0.6667
E_AvgNetProfit12M	0.0196	0.1423	0.0234	0.1163	0.2012	−0.466
E_EPSF12Growth	0.0038	0.0448	0.0025	0.0789	−0.032	0.0805
E_EPSF1Std	0.0075	0.0899	0.0044	0.0739	0.0596	0.0801
E_EPSFYGrowth	0.0008	0.0084	0.0004	0.0888	0.0043	−0.031
E_F12EP	0.0653	0.4564	0.0547	0.1289	0.4244	0.9086
E_F12EPSChg	0.0257	0.2244	0.0225	0.1043	0.2154	0.5796
E_F12EPSChg6M	0.0241	0.2117	0.0235	0.1049	0.2245	0.5198
E_F12EPSStdMean	−0.009	0.0979	0.0058	0.0844	0.0691	0.1823
E_F12NetProfit3MChg	0.0204	0.1718	0.0193	0.0972	0.1989	0.2449
E_F12PE	0.0646	0.4523	0.0538	0.1299	0.4146	−0.875
E_F12ROE	0.0051	0.0458	0.0077	0.097	0.0793	0.1712
E_NetProfitChg	0.0317	0.3630	0.0312	0.0734	0.4255	0.5935
E_NetProfitF12MDiff	0.0116	0.0985	0.0135	0.1024	0.1316	0.2887
E_NPF1MaxMin	0.0193	−0.194	0.0196	0.0865	0.2262	0.5851

normal ones. As indicated by means and standard deviations of variables presented in Table 3, those with Adjusted IC's *t*-test results greater than 0.5 are selected. Following this rule, nine variables are selected.

We also report results generated using other criteria, including expected excessive returns, **inflation rate (IR)**, Sharp ratios, hit ratios, maximum drawdown (see Appendix [5] for full trans), and turnover rate of the variables" (the new appendix [5] is "Maximum Drawdown: Maximum losses") (Table 4).

Table 3 shows how variables are sorted by column. Those with the lowest ranking among the top 70% in four of six columns are combined with the selection criteria in Table 2 to form the final variable pool. Thus, a total of four variables are identified, including E_AvgEPSRevision, E_1MEPSRevision, E_F12EPSStdMean, and E_EPSF1Std. Therefore, we select the above factors with higher ranks as exogenous variables and perform model testing. The selected variables' trends

Table 4. Variable Results Generated Using Other Criteria

	Expected excessive returns	IR	Sharp ratios	Accuracy ratios	Maximum draw down	Turn over
E_1MEPSRevision	0.0475	0.4755	0.0726	0.5846	0.7071	0.7431
E_3MEPSRevision	0.0153	0.1270	−0.0092	0.5385	0.7046	0.4848
E_6MEPSRevision	0.0642	0.5002	0.1095	0.5385	0.6861	0.3883
E_AvgEPSRevision	0.0417	0.4314	0.0643	0.6154	0.6812	0.4981
E_AvgNetProfit12M	−0.0122	−0.0919	−0.0802	0.5077	0.6626	0.1963
E_EPSF12Growth	0.0069	0.0731	−0.0236	0.6000	0.6816	0.3934
E_EPSF1Std	0.0331	0.4345	0.0334	0.5846	0.7347	0.3464
E_EPSFYGrowth	0.0518	0.5015	0.0797	0.4923	0.6753	0.373
E_F12EP	0.0598	0.3487	0.1008	0.5077	0.6943	0.3539
E_F12EPSChg	−0.0029	−0.069	−0.0572	0.5385	0.7304	0.4237
E_F12EPSChg6M	0.0082	0.0424	−0.0288	0.5692	0.7084	0.3662
E_F12EPSStdMean	0.0069	−0.019	−0.0425	0.6154	0.7398	0.4699
E_F12NetProfit3MC	−0.0075	−0.0636	−0.0794	0.4462	0.6412	0.3778
E_F12PE	−0.1052	−0.8441	−0.3587	0.3538	0.7425	0.4835
E_F12ROE	−0.0208	−0.2765	−0.1058	0.4923	0.7511	0.3503
E_NetProfitChg	0.0363	0.2103	0.059	0.4462	0.6214	0.6162
E_NetProfitF12MD	0.0016	0.0189	−0.039	0.5231	0.6622	0.2211
E_NetProfitF12MG	−0.0089	−0.1104	−0.0663	0.5385	0.692	0.4694
E_NPF1MaxMin	0.0026	−0.0281	−0.0461	0.5385	0.6986	0.4773

are plotted in Figure 5. As shown in Figure 7, the most recursive nodes are the previous nine nodes; the horizontal axis represents the previous nine nodes, whereas the vertical axis shows the values of the selected variables.

4.3 Model Results

The processed data would be applied to *NMC-BERT-LSTM-DQN-X* models. Rolling windows of 10 days (in sample) are used as the training period to obtain the optimum parameters of the algorithms, and one quarter (out-of-sample) ahead as the testing period. To directly evaluate the performance of our algorithms, we simply use a trading rule for quantitative strategy investment. The trading rule is stated as follows: if the algorithm predicts the market to rise, the strategy will buy with all available funds if there is no previous position. On the other hand, the algorithm predicts the market to go down, and there is a previous position. On the other hand, the algorithm predicts the market to go down, and there is a previous position, the position would then be emptied. Furthermore, we select *LSTM, NMC-LSTM-DQN-X, BERT-X,* and *BERT-LSTM-X* as the benchmarks.

As shown in Table 5, *NMC-BRET-LSTM-DQN-X* has the highest cumulative return rate 218.84%, which is corresponding an annualized return rate of 29.25%. It also has the highest accuracy of 61.77%. On the downside, it has the second-to-one maximum loss −8.29%, only smaller that of *NMC-LSTM-DQN-X. We* suspect that such an increase in risks and losses may be because the *NMC-LSTM-DQN-X* model does not effectively reflect the impact of long-or-short-term market sentimental factors for out-of-sample future and cannot fully capture market fluctuations.

Furthermore, we segment the market conditions and study the timing effect of different segments based on the *NMC-LSTM-DQN-X* model. Our definition is as follows: if the index rises above

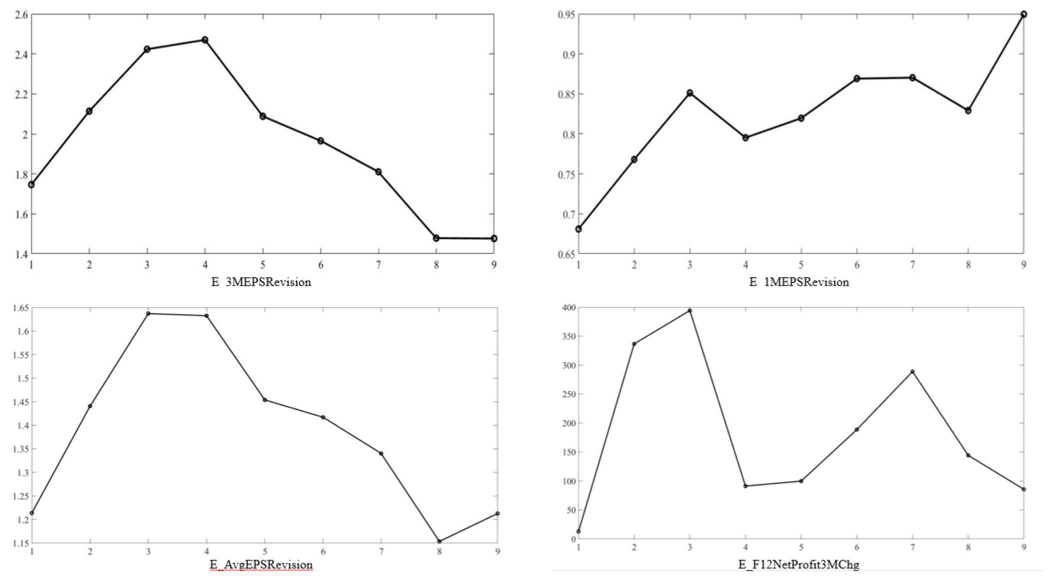

Fig. 7. Exogenous variables' trends.

Table 5. Model Comparison

	LSTM-DQN	NMC-LSTM-DQN-X	BERT-X	BERT-LSTM-X	NMC-BERT-LSTM-DQN-X
Cumulative Return Rate	106.57%	149.56%	161%	198%	218.84%
Annualized Return Rate	12.07%	21.07%	23.62%	27.93%	29.25%
Profit and Loss	1.93	2.63	2.97	3.15	3.37
Accuracy	52.58%	54.53%	57.01%	59.42%	61.77%
Maximum Loss	−8.27%	−8.58%	−8.45%	−8.06%	−8.29%
Maximum Drawdown	−9.05%	−15.17%	−16.29%	−8.73%	−13.57%

5% for the past 10 consecutive trading days, it is considered as a Bull market; if the index fluctuates in the range of −5% to 5% for the past 10 consecutive days, it is considered as a *Shock* market; if the index drops more than 5% for the past 10 consecutive days, it is considered as a Bear market.

The comparison of cumulative returns among the *LSTM-DQN, NMC-LSTM-DQN-X, NMC-BERT-LSTM-DQN-X, BERT-X, and BERT-LSTM-X* is presented in Figure 8. Figure 8 shows the cumulative returns for the *LSTM-DQN, NMC-LSTM-DQN-X, NMC-BERT-LSTM-DQN-X, BERT-X, and BERT-LSTM-X* for a fund of $1 invested in the model at the beginning of a period. The horizontal axis represents the investment time, and the vertical axis shows the fund's cumulated market value.

As indicated by Figure 8, *NMC-BERT-LSTM-DQN-X* produces higher annualized returns (see Appendix [7] for full trans). Its cumulative return curve exemplifies superiority in forecasting and timing the market. Based on all analyses above, we come to three conclusions. (1) The NMC model based on RL has great prediction performance on the stock market, suggesting that its trend is consistent with the stochastic process; (2) the *BERT*-type models can better extract sentiment variables and use sentiment variables to predict market trends; (3) the dynamic length of backtesting data combined with investors' sentimental factors increases returns remarkably; and (4) the dynamic length of backtesting data combined with investors' sentimental factors increases returns remarkably.

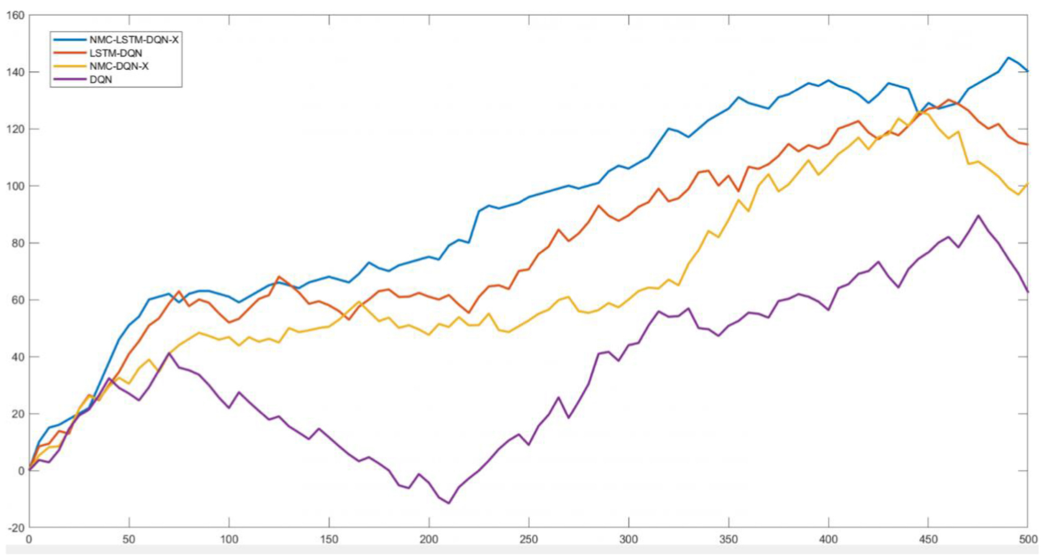

Fig. 8. Comparison of cumulative returns.

4.4 Model Evaluation and Comparison Through SOTA Test

In this subsection, we used the *SOTA*, or state of the art, methods to compare our algorithm with the time series prediction algorithm proposed in recent three years (2019–2021) to verify the correctness and efficiency of our algorithm. Our task is to predict the stock market index. While the *BERT* is proved to be *SOTA*; however, there is no absolutely optimal prediction algorithm, due to the particularity, volatility, and periodicity of each stock market. We select several sequential forecasting algorithms that have been proved to have excellent performance and good performance in predicting stock index, and compare them with our *NMC-BERT-LSTM-DQN-X* algorithm on the same dataset.

In this subsection, the method proposed in this study, *NMC-BERT-LSTM-DQN-X,* would be compared with state-of-the-art sequence prediction algorithms, namely, (1) *DeepState* [Rangapuram et al. 2018], (2) *MQR(C)NN* [Wen et al. 2017], and (3) *DeepFactors* [Nakagawa et al. 2019]. *DeepState* uses an *RNN* and a Gaussian linear space model to predict future time series, thereby improving the flexibility of the algorithm in terms of the length of backtesting data while greatly reducing the complexity of the *RNN. MQR(C)NN* can dynamically select the length of the backtesting data. It uses a nonlinear structure to effectively retain the length information of each encoder through stack pressing and stack bouncing operations. *DeepFactors* is a global–local framework prediction model. The global part is a linear combination of a group of deep factors, and these deep factors are obtained using a deep *NN*, which is highly similar to our algorithm. Both *DeepFactors* and our algorithm set multiple deep factors, which effectively reduce the prediction variance and greatly improve prediction accuracy. Compared with the traditional *RNN* predictor, *DeepState* and *MQR(C)NN* require fewer parameters to learn, whereas *DeepFactors* is mainly concerned with determining whether the embedded matrix *W* is inputted into the *NN* for learning, and the *NMC-BERT-LSTM-DQN-X* algorithm model has higher relevance and smooth convergence, which avoids the disadvantages of repeated introduction of parameters. The latter is more computationally efficient because it uses a recursive sequence to replace the matrix operation.

Table 6. Comparison Results with the *SOTA* Methods

Algorithms	MSE	RMSE	MAE	MAPE
NMC-LSTM-DQN-X	0.127	0.0636	0.153	16.6%
DeepState	0.236	0.0936	0.139	16.7%
MQR(C)NN	0.139	0.0596	0.196	15.7%
DeepFactors	0.163	0.0714	0.183	15.2%
LSTM-DQN	0.385	0.0947	0.214	18.3%
BERT-X	0.136	0.0762	0.151	16.9%
BERT-LSTM-X	0.118	0.0744	0.137	15.3%
NMC-BERT-LSTM-DQN-X	0.095	0.0739	0.104	15.1%

The algorithms are compared in terms of optimization curves (convergences), forecasting performance, and forecasting errors. Forecasting errors are evaluated via the RMSE, MAPE, **mean absolute error (MAE), mean square error (MSE)**, and coefficient of determination (R^2), which are defined below. The comparison results are presented in Table 6.

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2} \tag{12}$$

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2 \tag{13}$$

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|\hat{y}_n - y_n| \tag{14}$$

$$MAPE = \frac{1}{N}\sum_{n=1}^{N}\frac{|\hat{y}_n - y_n|}{y_n} \tag{15}$$

Table 6 indicates that although the eight methods achieve similar results based on the error test, *NMC-BERT-LSTM-DQN-X* has slight advantages for MSE, MAE. Figure 9 shows the convergence curves of *NMC-BERT-LSTM-DQN-X, LSTM-DQN, BERT-X, BERT-LSTM-X, NMC-BERT-LSTM-DQN-X, DeepState, MQR(C)NN, and DeepFactors.*

Figure 9 is calculated as follows: for each in-sample forecasting point at each training (there are 1,572 in-sample rolling window forecasting points), we average the first 30 iterations. At each iteration, we calculate the average of the *MSE.* Therefore, at every 30 points in Figure 9, the average is based on 1,572 observations. As indicated by Figure 9, *NMC-BERT-LSTM-DQN-X* has the fastest convergence rate. The convergence is reached after 16 iterations. This is faster than the 20 times convergence speed of the model without BERT.

Figure 10 shows the cumulative returns for *LSTM-DQN, NMC-LSTM-DQN-X, NMC-BERT-LSTM-DQN-X, BERT-X, BERT-LSTM-X, DeepState, MQR(C)NN, and DeepFactors* for a fund of ∃1 invested in the model at the beginning of a period. The horizontal axis represents the investment time, and the vertical axis shows the fund's cumulated market value. Table 7 presents the performance statistic. Once more, *NMC-BERT-LSTM-DQN-X* (ranked highest) achieves superior performance to *LSTM-DQN, NMC-LSTM-DQN-X, BERT-X, BERT-LSTM-X, DeepState, MQR(C)NN, and DeepFactors.*

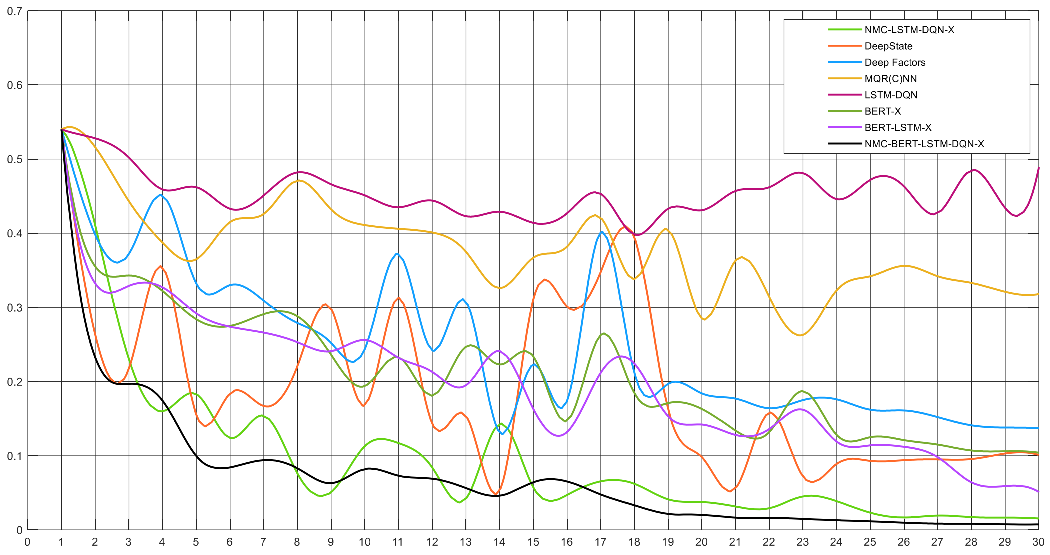

Fig. 9. Convergence comparison.

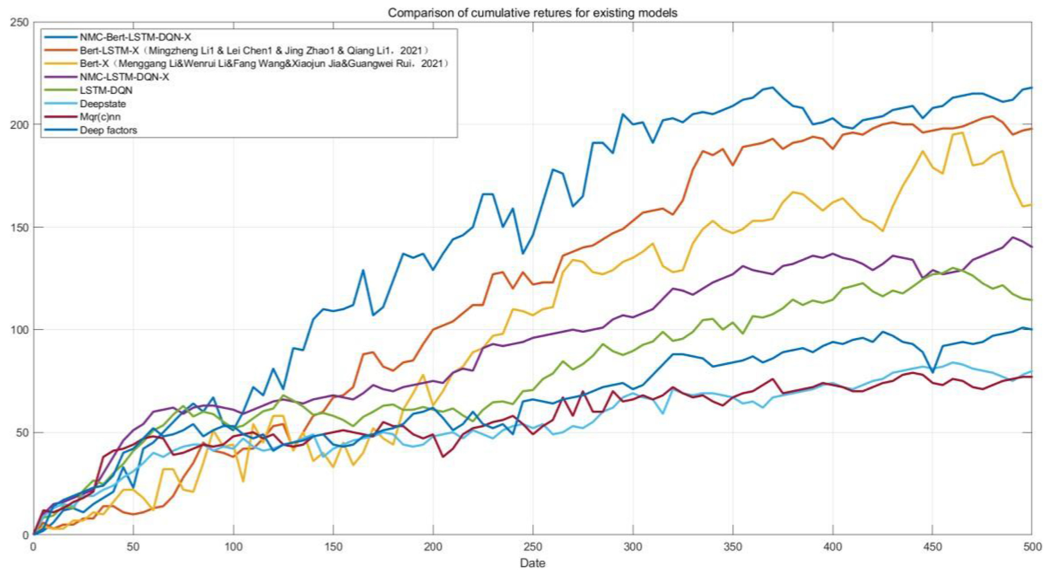

Fig. 10. Comparison of cumulative returns.

Interesting trends are observed in Table 7 and Figure 9. First, *NMC-BERT-LSTM-DQN-X* achieves the highest annualized rate of return and accuracy rate. Second, although *DeepFactors* has a high return among the sequential forecasting methods, it does have the largest loss of −26.54%. Finally, the methods are interactive because the uptrend and downtrend in the market are mutually indemnifying, as shown in Figure 10.

Table 7. Performance Statistic

Style tag	*NMC-BERT-LSTM-DQN-X*	*BERT-X*	*BERT-LSTM-X*	*DeepState*	*MQR(C)NN*	*DeepFactors*
Cumulative rate of return	218.84%	161%	198%	125.68%	120.45%	134.49%
Annualized rate of return	29.25%	23.62%	27.93%	16.96%	19.05%	15.56%
Profit and loss	3.37	2.97	3.15	2.45	2.64	2.03
Accuracy	61.77%	57.01%	59.42%	55.65%	53.78%	31.03%
Maximum loss	−8.29%	−8.45%	−8.06%	−19.14%	−20.8%	−26.54%
Maximum drawdown	−13.57%	−16.29%	−8.73%	−13.35%	−18.40%	−23.18%

5 CONCLUSION

In this study, the application of the *NMC-BERT-LSTM-DQN-X* method with sentimental and illiquid exogenous variables in quantitative trading was proposed. The method outperformed existing state-of-the-art sequential forecasting methods with an accuracy of 61.77%, annualized rate of return of 29.25%, and a maximum drawdown of −13.57%. Furthermore, among the comparison methods, the proposed method achieved the lowest forecasting error: *MSE* (0.095), *RMSE* (0.0739), *MAE* (0.104), and *MAPE* (15.1%).

The innovation and contribution of this study are in three folds: first, the study demonstrated a method for integrating comprehensive multi-dimensional information about the market through PCA, thereby achieving higher accuracy, lower forecasting error, and higher forecasting returns than existing forecasting models. Second, the study proposed a recursive algorithm based on *LSTM* to use past market information. The method is more efficient and has a faster convergence rate than the other three existing forecasting models. Finally, the study, using the NMC-X model to calculate the recursive time frame as well as the transition probabilities, incorporating a dynamic method that is well compatible with the market conditions into the forecasting process. The method is not only theoretically well-established but also logically solid.

Nonetheless, some issues should be addressed in future studies. First, computational costs exponentially increase with the orders of the NMC-X model (recursive time frame). Thus, real-time trading using the proposed method would be impossible. Second, exogenous variable data may be difficult to collect in high-frequency trading. Thus, the method is only valid for trading daily. We intended to address those limitations in our future studies. On one hand, the computational time could be significantly reduced if parallel computing architecture is designed and implemented. On the other hand, the pool of exogenous variables could be further fine-tuned to handle the forecasting of market movements at different extremely high frequencies. Predicted high frequencies market trends would then be integrated into low ones achieving great prediction performance.

A APPENDIX

[1] Sentimental factors: Investors' emotions and attitudes towards the market
[2] Illiquid stocks: Inactive stocks
[3] Quantitative trading: Using models and algorithms to place stock trading orders automatically
[4] Market timing: Market forecasts
[5] Maximum Drawdown: Maximum losses
[6] Back testing time frame: The length of the past market information
[7] Annualized return: The net gain or loss of an investment in percentage over a year
[8] Portfolio: A collection of financial assets such as stocks, bonds, and commodities

COMPLIANCE WITH ETHICAL STANDARDS

Conflict of Interest: The authors declare no conflict of interest.
Ethical approval: This article contains no research involving human or animal subjects conducted by any of the authors.

REFERENCES

R. Adhikari and R. K. Agrawal. 2014. A combination of artificial neural network and random walk models for financial time series forecasting. *Neural. Comput. Appl.* 24, 6 (2014), 1441–1449.

R. C. Ahana, A. Soheila, T. Michael, and K. Piyush. 2020. Enhancing profit from stock transactions using neural networks. *AI Commun.* 33, 1 (2020), 1–12.

Y. Amihud. 2002. Illiquidity and stock returns: cross-section and time-series effects. *J. Financ. Mark.* 5, 1 (2002), 31–56.

A. Andrikopoulos, T. Angelidis, and V. Skintzi. 2014. Illiquidity, return and risk in G7 stock markets: Interdependencies and spillovers. *Int. Rev. Financ. Anal.* 35, 1 (2014), 118–127**.**

S. Asadi, E. Hadavandi, F. Mehmanpazir, and M. M. Nakhostin. 2012. Hybridization of evolutionary Levenberg-Marquardt neural networks and data pre-processing for stock market prediction. *Knowl-Based. Syst.* 35, 15 (2012), 245–258.

C. S. Asness, T. J. Moskowitz, and L. H. Pedersen. 2013. Value and momentum everywhere. *J. Financ.* 68, 3 (2013), 929–985.

M. Baker and J. C. Stein. 2004. Market liquidity as a sentiment indicator. *J. Financ. Mark.* 7, 3 (2004), 271–299.

M. Baker and J. Wurgler. 2007. Investor sentiment in the stock market. *J. Econ. Perspect.* 21, 2 (2007), 129–151.

G. Bekaert, C. R. Harvey, and C. Lundblad. 2007. Liquidity and expected returns: Lessons from emerging markets. *Rev. Financ. Stud.* 20, 6 (2007), 1783–1831.

L. Bauwens and E. Otranto. 2016. Modeling the dependence of conditional correlations on market volatility. *J. Bus. Econ. Stat.* 34, 2 (2016), 254–268.

A. Brav, J. Wei, F. Partnoy, and R. Thomas. 2010. Hedge fund activism, corporate governance, and firm performance. *J. Financ.* 63, 4 (2010), 1729–1775.

A. Breaban and C. N. Noussair. 2018. Emotional state and market behavior. *Rev. Financ.* 22, 1 (2018), 279–309.

R. Cervell´o-Royo, F. Guijarro, and K. Michniuk. 2015. Stock market trading rule based on pattern recognition and technical analysis: Forecasting the DJIA index with intraday data. *Expert Systems with Applications* 42, 14 (2015), 5963–5975.

P. Chelley-Steeley, P. L. Lambertides, and C. S. Savva. 2013. Illiquidity shocks and the comovement between stocks: New evidence using smooth transition. *J. Empir. Financ.* 23, 5 (2013), 1–15.

H. Chen, T. T. Chong, and Y. She. 2014. A principal component approach to measuring investor sentiment in China. *Quantitative Finance* 14, 4 (2014), 573–579.

J. A. Cheng, J. B. Fu, Y. C. Kang, H. D. Zhu, and W. E. Dai. 2019. Sentiment analysis of social networks' comments to predict stock return. *Lecture Notes in Computer Science.* 2019, 11956, 67–74.

J. S Chou and T. K. Nguyen. 2018. Forward Forecast of stock price using sliding-window metaheuristic-optimized machine learning regression. *IEEE. T. Ind. Inform.* 14, 7 (2018), 3132–3142.

F. Di Martino, S. Senatore, and S. Sessa. 2019. A lightweight clustering-based approach to discover different emotional shades from social message streams. *Int. J. Intell. Syst.* 34, 7 (2019), 1505–1523.

O. M. E. Ebadati and M. T. Mortazavi. 2018. An efficient hybrid machine learning method for time series stock market forecasting. *Neural. Netw. World.* 28, 1 (2018), 41–55.

L. Fang and J. Peress. 2009. Media coverage and the cross section of stock returns. *J. Financ.* 64, 5 (2009), 2023–2052.

E. J. D. Fortuny, T. D. Smedt, D. Martens, and W. Daelemans. 2014. Evaluating and understanding text-based stock price prediction models. *Inform. Process. Manag.* 50, 2 (2014), 426–441.

X. Gong and B. Lin. 2018. Structural breaks and volatility forecasting in the copper futures market. *J. Futures. Markets.* 38, 3 (2018), 290–339.

J. Griffith, M. Najand, and J. Shen. 2020. Emotions in the stock market. *J. Behav. Financ.* 21, 1 (2020), 42–56.

E. Guresen, G. Kayakutlu, and T. U. Daim. 2011. Using artificial neural network models in stock market index prediction. *Expert. Syst. Appl.* 38, 8 (2011), 10389–10397.

Z. He and A. Krishnamurthy. 2013. Intermediary asset pricing. *Am. Econ. Rev.* 103, 2 (2013), 732–770.

A. A. Kasgari, M. Divsalar, M. R. Javid, and S. J. Ebrahimian. 2013. Prediction of bankruptcy Iranian corporations through artificial neural network and probit-based analyses. *Neural. Comput. Appl.* 23, 3–4 (2013), 927–936.

R. W. Kristjanpoller and V. K. Michell. 2018. A stock market risk forecasting model through integration of switching regime, ANFIS and GARCH techniques. *Appl. Soft. Comput.* 67, 1 (2018), 106–116.

A. Kumar. 2010. Who gambles in the stock market? *J. Financ.* 64, 4 (2010), 1889–1933.

Gourav Kumar, Sanjeev Jain, and Uday Pratap Singh. 2020. Stock market forecasting using computational intelligence: A survey. *Archives of Computational Methods in Engineering* 28, 3 (2020), 1069–1101.

M. Li, L. Chen, J. Zhao, and Q. Li. 2021. Sentiment analysis of Chinese stock reviews based on BERT model. *Appl. Intell.* 51, 7 (2021), 5016–5024. DOI: http://dx.doi.org/10.1007/s10489-020-02101-8

Daiki Matsunaga, Toyotaro Suzumura, and Toshihiro Takahashi. 2019. Exploring graph neural networks for stock market predictions with rolling window analysis. Retrieved November 1, 2021 from https://arxiv.org/abs/1909.10660.

L. Menggang, L. Wenrui, F. Wang, X. Jia, and G. Rui. 2020. Applying Bert to analyze investor sentiment in stock market. *Neural Computing and Applications* 33, 10 (2020), 4663–4676. DOI: http://dx.doi.org/10.1007/s00521-020-05411-7

K. Nakagawa, T. Uchida, and T. Aoshima. 2019. Deep factor model. In *ECML PKDD 2018 Workshops*. Springer, Cham.

T. H. Nguyen, K. Shirai, and J. Velcin. 2015. Sentiment analysis on social media for stock movement prediction. *Expert. Syst. Appl.* 42, 24 (2015), 9603–9611.

R. Nyman and P. Ormerod. 2014. Big data, socio-psychological theory, algorithmic text analysis and predicting the Michigan consumer sentiment index. Retrieved November 1, 2021 from https://arxiv.org/abs/1405.5695.

J. Patel, S. Shah, P. Thakkar, and K. Kotecha. 2015a. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert. Syst. Appl.* 42, 1 (2015), 259–268.

J. Patel, S. Shah, P. Thakkar, and K. Kotecha. 2015b. Predicting stock market index using fusion of machine learning techniques. *Expert. Syst. Appl.* 42, (2015), 2162–2172.

Prozil Pinto and Ana Filipa. 2018. A principal component approach to measure investor sentiment and its impact on ipo's underpricing. Retrieved from https://repositorio-aberto.up.pt/bitstream/10216/116313/2/294407.pdf.

R. Sadka. 2006. Momentum and post-earnings-announcement drift anomalies: The role of liquidity risk. *J. Financ. Econ.* 80, 2 (2006), 309–349.

S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski. 2018. Deep state space models for time series forecasting. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. 7796–7805

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. 2017. A multi-horizon quantile recurrent forecaster. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS'17)*.

Y. Shynkevich, T. M. Mcginnity, S. Coleman, A. Belatreche, and Y. Li. 2017. Forecasting price movements using technical indicators: investigating the impact of varying input window length. *Neurocomputing* 264, 1 (2017), 71–88.

J. L. Ticknor. 2013. A Bayesian regularized artificial neural network for stock market forecasting. *Expert. Syst. Appl.* 40, 14 (2013), 5501–5506.

P. Tsang, S. Kwok, S. Choy, R. Kwan, and S. Ng. 2007. Design and implementation of NN5 for Hong Kong stock price forecasting. *En. Appl. Artif. Intel.* 20, 4 (2007), 453–461.

B. Weng, M. A. Ahmed, and F. M. Megahed. 2017. Stock market one-day ahead movement prediction using disparate data sources. *Expert Syst. Appl.* 79, 8 (2017), 153–163.

B. Weng, L. Lu, X. Wang, F. M. Megahed, and W. Martinez. 2018. Predicting short-term stock prices using ensemble methods and online data sources. *Expert Syst. Appl.* 112, 1 (2018), 258–273.

H. C. Xu and W. X. Zhou. 2018. A weekly sentiment index and the cross-section of stock returns. *Financ. Res. Lett.* 27, 1 (2018), 135–139.

Z. Yang, O. Fu, and X. Peng. 2020. A decision-making algorithm for online shopping using deep-learning-based opinion pairs mining and q-rung orthopair fuzzy interaction Heronian mean operators. *Int. J. Intell. Syst.* 35, 5 (2020), 783–825. DOI: https://doi.org/10.1002/int.22225

P. D. Yoo, M. H. Kim, and T. Jan. 2005. Financial forecasting: advanced machine learning techniques in stock market analysis. In *Proceedings of the 2005 Pakistan Section Multitopic Conference*.

Jun Zhang, Yu-Fan Teng, and Wei Chen. 2018. Support vector regression with modified Firefly algorithm for stock price forecasting. *Applied Intelligence* 49, 5 (2018), 1658–1674.

Y. Zuo and E. Kita. 2012a. Stock price forecast using Bayesian network. *Expert Syst. Appl.* 39, 8 (2012), 6729–6737.

Y. Zuo and E. Kita. 2012b. Up/down analysis of stock index by using Bayesian network. *Engineering Management Research* 1, 2 (2012), 46–52.

Received October 2020; revised July 2021; accepted September 2021