Electronic Research Archive

http://www.aimspress.com/journal/era

ERA, 31(5): 2626–2650.
DOI: 10.3934/era.2023133
Received: 30 December 2022
Revised: 13 February 2023
Accepted: 01 March 2023
Published: 09 March 2023

Research article

MSGraph: Modeling multi-scale K-line sequences with graph attention network for profitable indices recommendation

Changhai Wang*, Jiaxi Ren and Hui Liang

Software Engineering College, Zhengzhou University of Light Industry, No.136 Science Avenue, Zhengzhou 450000, China

* **Correspondence:** Email: chw@zzuli.edu.cn; Tel: +8615538392030.

Abstract: Indices recommendation is a long-standing topic in stock market investment. Predicting the future trends of indices and ranking them based on the prediction results is the main scheme for indices recommendation. How to improve the forecasting performance is the central issue of this study. Inspired by the widely used trend-following investing strategy in financial investment, the indices' future trends are related to not only the nearby transaction data but also the long-term historical data. This article proposes the MSGraph, which tries to improve the index ranking performance by modeling the correlations of short and long-term historical embeddings with the graph attention network. The original minute-level transaction data is first synthesized into a series of K-line sequences with varying time scales. Each K-line sequence is input into a long short-term memory network (LSTM) to get the sequence embedding. Then, the embeddings for all indices with the same scale are fed into a graph convolutional network to achieve index aggregation. All the aggregated embeddings for the same index are input into a graph attention network to fuse the scale interactions. Finally, a fully connected network produces the index return ratio for the next day, and the recommended indices are obtained through ranking. In total, 60 indices in the Chinese stock market are selected as experimental data. The mean reciprocal rank, precision, accuracy and investment return ratio are used as evaluation metrics. The comparison results show that our method achieves state-of-the-art results in all evaluation metrics, and the ablation study also demonstrates that the combination of multiple scale K-lines facilitates the indices recommendation.

Keywords: indices recommendation; Chinese stock market; graph attention network; indices ranking; dynamic time warping

1. Introduction

Stock market investment is a topic that has been around for decades. As a single stock is vulnerable to business conditions and public sentiment, investing in stocks carries huge risks. A stock index is composited with multiple stocks to reflect the overall trends of the stock market or a specific industry, and indices are significantly less volatile than stocks. Investing in exchange traded funds (ETF) that follow stock indices is a very popular investment method these days. There are thousands of indices that can be invested in the Chinese stock market. However, not all of these indices are profitable over a period of time. Among these indices, choosing a certain number of indices which are worth investing in is a topic of great significance. This article focuses on the indices recommendation issue, which provides investors with the top-K profitable indices based on future trends prediction.

Machine learning and deep learning are the mainstream methods for this research field. They use the historical transaction data to predict future trends and select optimal indices for recommendation. Numerous models have been used to recommend stocks, ranging from the early days of technical analysis [1] and autoregressive models (ARIMA) [2] to the rise of convolutional neural networks (CNN) [3] and graph convolutional networks (GCN) [4] in recent years. These studies focus on identifying the temporal-spatial relationships between past transaction data from various indices and future patterns. Most of these studies [5–7] used the 1-day K-line sequence as the basic experimental data, and few studies [8] applied the minute level trading data. However, the trading data with a fixed time interval cannot yield sufficient characteristics for the index trend.

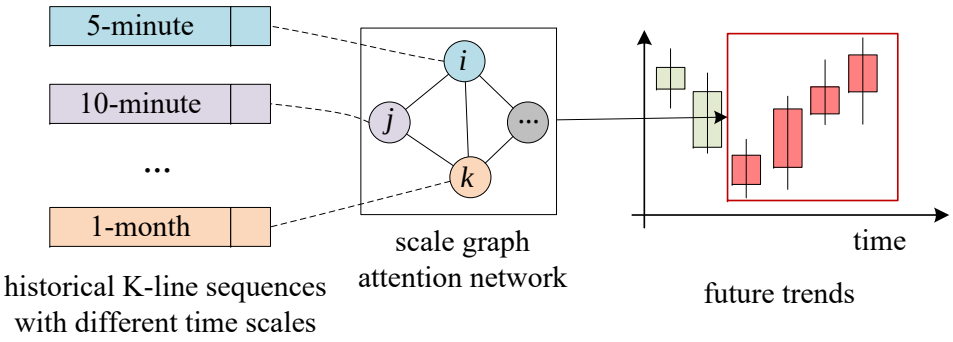


Figure 1. The correlation of multi-scale K-line sequences. The index K-line can be presented on different time scales, such as 5-minute, 10-minute, 1-month, etc. Minute-level K-lines may indicate nearby characteristics, whereas month-level data always contains long-term trends. The future trend is determined by both nearby and long historical K-lines.

According to the widely used trend-following strategies in the technical analysis of the financial market, the future trends of index points are determined by both long-term and short-term trends. As shown in Figure 1, long-term data would be the month-level K-line sequence, which covers a period of several months. The short-term data are the minute-level data, which occurs in recent days. In a long-term upward trend, a short-term downtrend would imply the end of a long-term uptrend. On the other hand, a short-term upward trend contained within a long-term negative trend frequently indicates the beginning of a long-term upward trend. These K-line sequences with different time scales contain complex correlations. It is possible to predict the index's future more precisely by fusing historical

data with various time scales in the stock market. However, current popular models cannot model this characteristic.

This paper proposes MSGraph, which abstracts the K-line sequences with different scales into a graph and applies the graph attention network to extract scale correlations. To sum up, the contributions of this article are as follows.

- We originally put forth the methodology of forecasting an index's future trend using multi-scale historical data, which was inspired by the trend-following strategy in financial investment. The graph attention network is employed in this approach to combine the influences of various scale historical sequences.
- Following our first innovative idea, we propose an index recommendation framework named MSGraph. This method first synthesizes minute-level historical K-lines into multi-scale sequences, and it uses LSTM to obtain sequential embeddings. Next, the index graph and scale graph are respectively applied to fuse the influences between different indices and different scales. Finally, the fully connected layer is utilized to map the fused embeddings to index fluctuation, and the recommended indices are obtained via the ranking.
- MSGraph is fully verified with the Chinese stock market indices. The evaluation results show that our method obtains state-of-the-art results with multiple evaluation metrics, such as reciprocal rank, precision, accuracy and return ratio. At the end of the experimental section, the effect of each component of our method is also analyzed.

This paper is organized as follows. The related work will be introduced in Section 2. The proposed MSGraph will be presented in Section 3. Section 4 presents comparison experiments, including experimental design, performance comparison, market backtesting and ablation studies. Section 5 gives a summary and further research directions.

2. Related works

In the last few decades, artificial intelligence has been used a lot in the field of investing in the financial market. Common applications include stock trend prediction, stock recommendation, etc. From the perspective of research models, it has gone through several stages such as technical analysis, classic machine learning, deep learning and graph convolution.

2.1. Technical analysis

Technical analysis is a method that emerged during the early days of stock market investment. It focuses on predicting future trends by analyzing technical indicators in stock trading data [1, 9]. Some classical technical indicators include the moving average, the Chande momentum oscillator, the Williams indicator, the relative strength index, etc. These technical indicators are simple and easy to obtain, and they are still widely used in the stock market. These indicators, on the other hand, are so simple that they cannot show the complex relationship between past transaction data and future trends. Instead, some time series models are presented. The two most famous methods are the Kalman filter [10] and the autoregressive integrated moving average model (ARIMA) [2]. These two methods are also commonly adopted in stock market investment because of their simplicity and good explanation. However, their disadvantages are also obvious. They cannot handle the non-linear correlation between

the historical and future data and also cannot express the non-Euclidean spatial correlation between multiple stocks. Another recent work is [11], which tries to capture the mutual exciting activities through a multi-dimensional Hawkes model. This approach provides a spatial correlation view on multiple stock investments, and it would be a promising research direction for technical analysis.

2.2. Machine learning

With the advancement of machine learning, more and more researchers are using machine learning models to predict stock trends and recommend profitable stocks [12]. At this stage, various machine learning models are applied to financial investments. Some typical studies include autoregressive conditional heteroskedasticity (GARCH) [13], decision trees [14], support vector machines (SVM) [15], artificial neural networks (ANN) [16], etc. On top of that, some boosting methods [17, 18] are also proposed to improve the prediction's robustness. These machine learning models are still widely used in stock market investment.

However, these models have a common characteristic, which is that their performances are always related to pre-extracted features. As these features are usually extracted with the experience of investors, and the number of features is limited, these machine learning methods cannot utilize the deep non-intuitive features. However, it is precisely these non-intuitive features that are more representative of the transaction data. This leads to the natural defects of classical machine learning models. To overcome this disadvantage, deep learning is adopted for financial investment issues.

2.3. Deep learning

The technology of deep learning has developed rapidly in the past ten years. It has achieved breakthrough results in many fields [19]. Lots of studies have applied the deep learning model to financial market issues [20]. The advantage of deep learning models is that they do not need to manually extract features, and they always use an end-to-end architecture. In these models, the historical data are taken as input, and the features are automatically extracted through the deep networks. These networks are always optimized with the gradient descent algorithm, such as AdaGrad [21], AdaDelta [22], Adam [23]. In early research, the multi-layer neural network [24] was the most widely used model. As this model contains multiple feedforward layers, it is difficult to train the optimal parameters. With the development of the convolution mechanism, many works have applied the CNN to the stock market [3, 25]. The convolutional neural network obtains promising results, and it is also widely used in financial investment.

The other most commonly used model is the sequence model, such as recurrent neural networks (RNN) [26], LSTM [27, 28], the deep transformer model [29], etc. Also, some hybrid models which combine the convolutional and sequence models are also put forward, such as [30–32]. Although deep learning models are powerful methods for the stock market, they also have deficiencies. The most obvious problem is that they cannot use correlations between different stock transaction data. However, these correlations are always important to model historical data and future trends. Thus, an improved deep model named graph neural network is proposed.

2.4. Graph neural network

The graph-based neural network was first proposed to model data with graph structure, and it has undergone rapid development in the past few years [4]. It has also been applied in many research

fields, such as traffic forecasting [33], natural language processing [34], power systems [35], etc. In the financial area, some studies [36, 37] regarded the historical K-lines as visual graphs and extracted the features for stock trend prediction. However, most works modeled the stocks as the graph nodes, and the graph convolutional network was mainly used to model the relationship between different stocks. It can capture both spatial and temporal correlations in historical transaction data of multiple stocks when combined with the time series model. Some studies using graph networks for stocks recommendation are as follows.

The Relation Stock Ranking (RSR) [5] is pioneering research for stock recommendation ranking. It utilized the sequence model to learn a stock-wise sequential embedding, and the temporal graph convolution was applied to fuse the deep features of multiple stocks. Chen et al. [9] proposed the graph convolutional feature based convolutional neural network (GC-CNN), which can simultaneously capture both stock market and individual stock features. Cheng et al. [38] proposed the attribute-driven graph attention network (AD-GAT), which introduced the attribute-driven mechanism to the graph convolution for modeling momentum spillovers. The model named graph attention long short-term memory (GALSTM) [39] was put forward to predict the future prices of stocks. Li et al. [40] proposed the LSTM relational graph convolutional network (LSTM-RGCN) to predict the overnight stock movement between the previous close price and the open price. Feng et al. [41] focused on the high return ratio stocks recommendation with the relation-aware dynamic attributed graph attention network (RA-AGAT). Xu et al. [42] aimed to predict the price-limit-hitting stocks with the hierarchical graph neural network. Hsu et al. [6] and Ma et al. [7] utilized the graph attention network and the attribute-driven fuzzy hypergraph network for profitable stocks recommendation, respectively. Wang et al. [43] combined the transformer and the graph attention network for stock investment selection.

Currently, graph-based neural networks are an emerging method for stock market prediction. However, the current study's inputs are all fixed time scale transaction data, such as 1-day K-lines, etc. Since both short-term and long-term historical data affect how a stock will move in the future, modeling the multi-scale historical data would be an interesting area of research. Meanwhile, each scale's data is influenced by the same scale data of other indices. Thus, integrating the multiple scale data of all stocks for the current index prediction is a problem that needs to be solved. This paper proposes the multi-scale graph attention network for profitable indices recommendation, and extensive experiments show that our method can obtain state-of-the-art results.

3. The proposed model

In this section, we will give the details of the proposed model. First, the problem formulation and model framework are introduced in Section 3.1. Then, each part of the proposed model is put forward in Section 3.2 to Section 3.5.

3.1. Formulation and framework

This section will first give some symbol definitions for our model and then introduce the model framework. The inputs of our model are historical data from N indices. For index i, the inputs are the minute-level K-line sequence, which is denoted as $\boldsymbol{H}_i = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots \boldsymbol{h}_T]$, where $\boldsymbol{h}_t$ is the t-th K-line and T is the sample length. In our model, the inputs $\boldsymbol{h}_t = (h_{t1}, \cdots, h_{t5})$ are the 5-minute K-line data, and 5 elements are the opening point, highest point, lowest point, closing point and trading volume at

this timestamp. Since our model needs to span multiple months, the value of T would be much larger than that of existing methods. The purpose of this article is as follows: given the historical data $\boldsymbol{H}$ of N indices for each day, predicting their movements for the next day and recommending k worth of invested indices based on prediction results. The architecture of the proposed prediction model is shown in Figure 2.

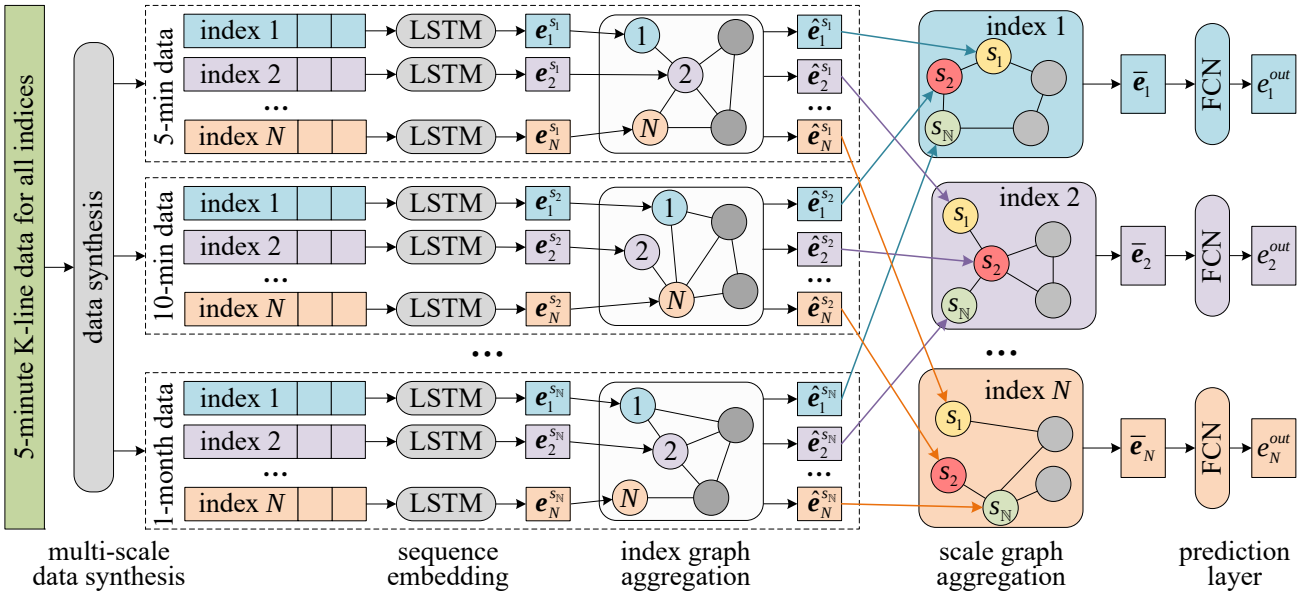


Figure 2. The framework of our prediction model. The inputs are the historical data of all indices. These data are first synthesized into multiple sequences with different scales. Then, the sequential embeddings for the same scale are modeled with an index graph. Third, the aggregated embeddings for the same index are structured as a graph attention network to embody the effect of different scales. Finally, the movement for each index is obtained through a fully connected network.

The inputs are the 5-minute K-line data for all indices, and they are first synthesized into multiple sequences with different time scales. Each scale stands for a specific time interval, such as 5-minute, 10-minute, etc. The j-th scale is represented by s_j, the scale number is written as $\mathbb{N}$, and the detailed data description will be introduced in the experimental section. The K-line synthesis algorithm will be introduced in Section 3.2. After the synthesis, the K-line sequence at scale s_j for index i is represented by $\boldsymbol{H}_i^{s_j} = [\boldsymbol{h}_1^{s_j}, \boldsymbol{h}_2^{s_j}, \cdots, \boldsymbol{h}_{T_j}^{s_j}]$, where T_j is the sequence length for scale s_j. As long sequences would cause gradient vanishing for the following sequence model, T_j is much smaller than the initial length T. Each synthesized sequence will be input into the LSTM model to get the sequential embedding, which is denoted by $\boldsymbol{e}_i^{s_j}$, where i and s_j represent index and scale, respectively.

As all indices' data at the same scale are always trends correlated, the index graph aggregation is conducted for each scale. This step contains $\mathbb{N}$ graphs, corresponding to $\mathbb{N}$ scales, and it will be put forward in Section 3.3. In these graphs, the node represents index, and the edge is index correlation. After index graph aggregations, the embeddings that combine index correlations are obtained. Each index contains $\mathbb{N}$ embeddings, and the j-th embedding for index i is written as $\hat{\boldsymbol{e}}_i^{s_j}$. According to our

motivation, the future trends of index i are determined by both long-term and short-term historical data. We utilize the graph attention network to model correlations between multiple scales, which will be brought forward in Section 3.4. This step contains N graphs corresponding to N indices. For each graph, the node depicts scale embedding, and the edge is scale relation. After the graph attention network, we can finally get the embeddings $\bar{\boldsymbol{e}}_i$, which fused the features of different indices and scales. These outputs will be input into fully connected networks to produce the movements for the next day. The output for the i-th index is denoted as e_i^{out}. The recommended k indices can be acquired by ranking the predicted results. The objective function for the prediction model and the recommendation scheme will be introduced in Section 3.5.

3.2. Data synthesis

Algorithm 1 The K-line synthesis algorithm.

Input:

$\boldsymbol{H} = [\boldsymbol{H}_1, \cdots, \boldsymbol{H}_N]$: The sample data consisting of 5-minute K-line sequences for N indices;

$\boldsymbol{n} = [n_{s_1}, \cdots, n_{s_{\mathbb{N}}}]$: The 5-minute numbers are contained in different scales;

$\boldsymbol{T} = [T_1, \cdots, T_{\mathbb{N}}]$: Sequence lengths for various scales;

Output:

$\boldsymbol{H}^s = [[\boldsymbol{H}_1^{s_1}, \cdots, \boldsymbol{H}_1^{s_{\mathbb{N}}}], \cdots, [\boldsymbol{H}_N^{s_1}, \cdots, \boldsymbol{H}_N^{s_{\mathbb{N}}}]]$: K-line sequences for all indices and scales;

Initialize $\boldsymbol{H}^s$ to $\emptyset$;
for $i = 1$ to N **do**
  Initialize $\boldsymbol{H}_i^s$ to $\emptyset$;
  for $j = 1$ to $\mathbb{N}$ **do**
    Initialize $\boldsymbol{H}_i^{s_j}$ to $\emptyset$;
    for $k = T_j$ to 1 **do**
      $tmp = \boldsymbol{H}_i[T - kn_{s_j} + 1 : T - (k-1)n_{s_j}]$;
      $\boldsymbol{h}_i^{s_j}[1] = tmp[1][1]$;
      $\boldsymbol{h}_i^{s_j}[2] = \max(tmp[:][2])$;
      $\boldsymbol{h}_i^{s_j}[3] = \min(tmp[:][3])$;
      $\boldsymbol{h}_i^{s_j}[4] = tmp[T_j][4]$;
      $\boldsymbol{h}_i^{s_j}[5] = \text{sum}(tmp[:][5])$;
      $\boldsymbol{H}_i^{s_j} = \text{append}(\boldsymbol{H}_i^{s_j}, \boldsymbol{h}_i^{s_j})$;
    end for
    $\boldsymbol{H}_i^s = \text{append}(\boldsymbol{H}_i^s, \boldsymbol{H}_i^{s_j})$;
  end for
  $\boldsymbol{H}^s = \text{append}(\boldsymbol{H}^s, \boldsymbol{H}_i^s)$;
end for
$\boldsymbol{H}^s = \text{normalize}(\boldsymbol{H}^s)$;
return $\boldsymbol{H}^s$;

As described in Section 3.1, our model uses the historical 5-minute K-lines of all indices as input.

The first phase of our model is to synthesize the initial data into K-line sequences with different scales. The synthesis method is shown as Algorithm 1, which takes three parameters as input. In addition to the raw data $\boldsymbol{H}$, there are another two parameters. $\boldsymbol{n}$ stores numbers of the 5-minute K-line contained in different scales. For example, if s_j is the scale of 1-hour, n_{s_j} equals 12, as 1 hour contains 12 5-minute intervals. The third parameter $\boldsymbol{T}$ saves sequence lengths for various scales. For each scale, all sequences share the same length. These values are much smaller than the input length T, and they will be introduced in the experimental section. The output of Algorithm 1 is all the K-line sequences, where $\boldsymbol{H}_i^{s_j}$ is the sequence of index i scaled by s_j.

Steps 5–14 are used to obtain the K-line sequences for index i scaled by s_j. Step 7 obtains a temporary 5-minute K-line sequence, which will be mapped into one K-line at scale s_j. Steps 8–12 indicate the synthesis method. According to the definitions of K-line, $\boldsymbol{h}_i^{s_j}[1]$ equals the opening point of the first K-line in the current temporary sequence. $\boldsymbol{h}_i^{s_j}[2]$ and $\boldsymbol{h}_i^{s_j}[3]$ are the maximum and minimum values during this period, respectively. $\boldsymbol{h}_i^{s_j}[4]$ is the closing point of the last K-line. $\boldsymbol{h}_i^{s_j}[5]$ is the total volume of all K-lines in this temporary sequence. At the end of this algorithm, Step 19 normalizes these sequences with Eqs (3.1) and (3.2).

$$(h_i^{s_j})'_{tk} = \frac{(h_i^{s_j})_{tk} - h_{tmp}}{h_{tmp}} \tag{3.1}$$

$$(h_i^{s_j})''_{tk} = 2\text{Sigmod}\left(\frac{(h_i^{s_j})'_{tk} - (\bar{h}_i^{s_j})'_k}{pre(\beta) - pre(100-\beta)}\right) - 1 \tag{3.2}$$

In Eq (3.1), $h_{tmp} = (h_i^{s_j})_{(t-1)4}$, if $k \in \{1, 2, 3, 4\}$. Otherwise, $h_{tmp} = (h_i^{s_j})_{(t-1)5}$. This formula is used to convert the index point to the fluctuation. The first four elements are calculated based on the previous closing point, and the fifth is determined on the basis of trading volume. As the fluctuations vary greatly from time to time, Eq (3.2) normalizes these values to a specific range. β is a value between 0 and 100, and $pre(\beta)$ is the β-th percentile of sequence $(\boldsymbol{H}_i^{s_j})'_k$. $(\bar{h}_i^{s_j})'_k$ is the average of $(\boldsymbol{H}_i^{s_j})'_k$. After normalization, all values in $\boldsymbol{H}^s$ fall between -1 and 1.

3.3. Index graph aggregation

The K-line sequences with different scales for all indices are obtained through Algorithm 1. These data contain $N \times \mathbb{N}$ sequences. The following step is modeling these sequences to get sequence embeddings. The LSTM is the most popular model for mining the time dependency, and it is adopted in our model. The sequence embedding for $\boldsymbol{H}_i^{s_j}$ is denoted as $\boldsymbol{e}_i^{s_j}$, and these embeddings are input into the index graph aggregation.

The purpose of the index graph aggregation is to integrate the characteristics between indices. Our model constructs $\mathbb{N}$ index graphs, and each graph indicates the correlations for a specific time scale. For the index graph, the node represents index, and the edge embodies relation. The first step for index graph aggregation is constructing the index graph adjacency matrix. According to [43, 44], the multi-dimensional Dynamic Time Warping (DTW) is applied to calculate the sequence similarity, and this value is regarded as the index correlation. The DTW is shown as Algorithm 2. The inputs are the K-line sequences for all indices at scale s_k and their length. The output is the corresponding adjacency matrix. In this algorithm, step 3 is used to calculate the cosine similarity between any two K-lines, and steps 5–8 are applied to obtain the similarity between two sequences using dynamic programming. The

similarities for all sequences in $\boldsymbol{H}^{s_j}$ are regarded as the adjacency matrix. On the basis of the adjacency matrix, the index graph aggregation is as

$$\hat{\boldsymbol{e}}^{s_k} = \sigma(\boldsymbol{A}^{s_k}\boldsymbol{e}^{s_k}\boldsymbol{W}_0^{s_k}), \tag{3.3}$$

where $\boldsymbol{e}^{s_k} \in \mathbb{R}^{N\times m_0}$ is the matrix of sequential embedding for scale s_k, and $\boldsymbol{W}_0^{s_k}$ is a hyperparameter matrix size of $m_0 \times m_1$. The output $\hat{\boldsymbol{e}}^{s_k}$ is the features that fused the sequence embeddings for scale s_k, and it would be input to the scale graph attention network.

Algorithm 2 The adjacency matrix construction method for index graph.

Input:

$\boldsymbol{H}^{s_k} = \{\boldsymbol{H}_1^{s_k}, \cdots, \boldsymbol{H}_N^{s_k}\}$: K-line sequences at scale s_k for all indices;

T_k: Sequence length for scale s_k;

Output:

$\boldsymbol{A}^{s_k}$: The adjacency matrix for index graph at scale s_k;

for $i, j \in [1, N]$ **do**
  for $m, n \in [1, T_k]$ **do**
    $\boldsymbol{D}_{mn} = \frac{(\boldsymbol{H}_i^{s_k})_m \cdot (\boldsymbol{H}_j^{s_k})_n}{|(\boldsymbol{H}_i^{s_k})_m||(\boldsymbol{H}_j^{s_k})_n|}$
  end for
  for $m, n = 1$ to T_k **do**
    $\hat{\boldsymbol{D}}_{mn} = \boldsymbol{D}_{mn} + \max(\hat{\boldsymbol{D}}_{(m-1)n}, \hat{\boldsymbol{D}}_{m(n-1)}, \hat{\boldsymbol{D}}_{(m-1)(n-1)})$
  end for
  $\boldsymbol{A}_{ij} = \hat{\boldsymbol{D}}_{T_kT_k}$;
end for
$\boldsymbol{A}^{s_k} = \text{normalize}(\boldsymbol{A})$;
return $\boldsymbol{A}^{s_k}$;

3.4. Scale graph aggregation

Since different scales of data have different implications for the future trends, the graph attention network is utilized to model these correlations and produce the scale aggregation. Our model contains N graphs, and each graph represents the scale relationships for one index. In each graph, the node represents scale embedding, and the edge employs the attention weight. Taking the i-th graph for example, the node j is the embedding of scale s_j for index i, and the edge will be calculated through the attention mechanism. Without loss of generality, the input for the i-th graph is denoted as $\hat{\boldsymbol{e}}_i = [\hat{\boldsymbol{e}}_i^{s_1}, \cdots, \hat{\boldsymbol{e}}_i^{s_N}]$, where $\hat{\boldsymbol{e}}_i^{s_j} \in \mathbb{R}^{m_1}$ is the feature vector for scale s_k. The output is vector $\bar{\boldsymbol{e}}_i \in \mathbb{R}^{m_3}$. The graph attention network is shown as Figure 3.

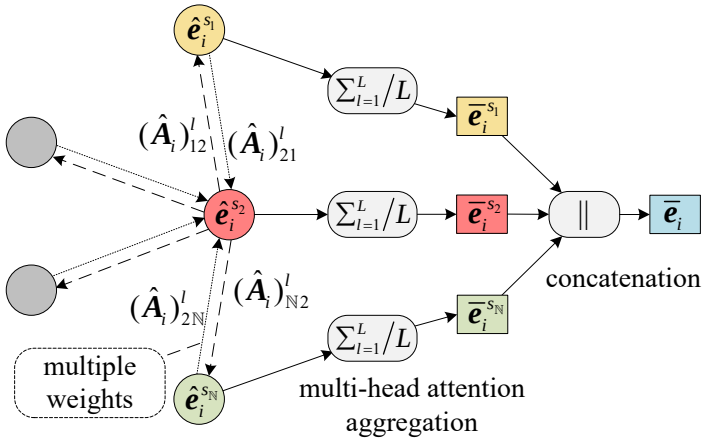


Figure 3. The graph attention network. It is a directed complete graph, and each edge consists of L attention weights. $\bar{\boldsymbol{e}}_i^{s_j}$ is obtained through multi-head attention aggregation. The output $\bar{\boldsymbol{e}}_i$ is acquired through the concatenation operator.

The scale graph is a directed complete graph. The first step for scale graph aggregation is calculating the aggregation weights. Each edge contains L weights. For the l-th weight in graph i, two hyperparameters named $\boldsymbol{W}_i^l \in \mathbb{R}^{m_2 \times m_1}$ and $\boldsymbol{\alpha}_i^l \in \mathbb{R}^{2m_2}$ are introduced. The i-th aggregation weight $(\hat{\boldsymbol{A}}_i)_{jk}^l$ from node s_k to s_j is calculated with

$$(\hat{\boldsymbol{A}}_i)_{jk}^{tmp} = \text{LeakyReLU}((\boldsymbol{\alpha}_i^l)^T[\boldsymbol{W}_i^l\hat{\boldsymbol{e}}_i^{s_j} \| \boldsymbol{W}_i^l\hat{\boldsymbol{e}}_i^{s_k}]) \tag{3.4}$$

and

$$(\hat{\boldsymbol{A}}_i)_{jk}^l = \frac{\exp((\hat{\boldsymbol{A}}_i)_{jk}^{tmp})}{\sum_{g=1}^{\mathbb{N}} \exp((\hat{\boldsymbol{A}}_i)_{jg}^{tmp})}. \tag{3.5}$$

Equation (3.4) is used to calculate a temporary weight, where $\|$ is the vector concatenation operator, and $(\boldsymbol{\alpha}_i^l)^T$ is the transpose of $\boldsymbol{\alpha}_i^l$. Equation (3.5) is utilized to normalize the temporary weight matrix. After normalizing the attention weights of all nodes, the node features can be aggregated through the graph attention layer, shown as

$$\bar{\boldsymbol{e}}_i^{s_j l} = \sum_{k=1}^{\mathbb{N}} (\hat{\boldsymbol{A}}_i)_{jk}^l \boldsymbol{W}_i^l \hat{\boldsymbol{e}}_i^{s_k}. \tag{3.6}$$

To enhance the robustness of the model, the multi-head attention aggregation is applied based on Eq (3.6), which is shown as

$$\bar{\boldsymbol{e}}_i^{s_j} = \sigma(\frac{1}{L}\sum_{l=1}^{L} \bar{\boldsymbol{e}}_i^{s_j l}), \tag{3.7}$$

where L is the attention number. The output is the embedding for scale s_j. As each graph represents an index, and our goal is to obtain the features of this index, we should merge these embeddings into one, shown as

$$\bar{\boldsymbol{e}}_i = \|_{j=1}^{\mathbb{N}} \bar{\boldsymbol{e}}_i^{s_j}. \tag{3.8}$$

The vector $\bar{\boldsymbol{e}}_i$ would be the input for the task specific layer, and its dimension is $m_3 = m_2 \times \mathbb{N}$.

3.5. Prediction layer

This section defines the prediction tasks and gives the objective function for optimization. The output $\bar{\boldsymbol{e}}_i$ of the graph attention network is first input to a fully connected network, shown as

$$\hat{y}_i = \boldsymbol{w}_i \bar{\boldsymbol{e}}_i + b_i, \tag{3.9}$$

where $\boldsymbol{w}_i \in \mathbb{R}^{1\times m_3}$ is the hidden parameter in the task specific layer, and b_i is the bias term. The output $\hat{y}_i$ is the prediction result. Since income is the most important evaluation metric for selecting profitable indices, the index return ratio for the next day is regarded as the prediction label. The label of the t-th day for index i can be acquired with

$$(y_i)_t = \frac{(\boldsymbol{H}_i^{1-day})_{(t+1)4} - (\boldsymbol{H}_i^{1-day})_{t4}}{(\boldsymbol{H}_i^{1-day})_{t4}}, \tag{3.10}$$

where $\boldsymbol{H}_i^{1-day}$ is the 1-day K-line sequence for index i, and the label is calculated with the closing point.

In this paper, we focus on the indices selection topic, in which the ranking score of each index is the essential prediction target. According to [5] and [7], the objective function, which combines regression loss and pairwise ranking-aware loss, is utilized to optimize this model. Equation (3.11) shows the objective function, where $\boldsymbol{y}$ consists of the ground-truth labels for all indices, and $\hat{\boldsymbol{y}}$ contains the prediction values.

$$\begin{aligned} l(\boldsymbol{y}, \hat{\boldsymbol{y}}) = &\sum\nolimits_{i=1}^{N} (y_i - \hat{y}_i)^2 + \\ &\alpha \sum\nolimits_{i=1}^{N} \sum\nolimits_{j=1}^{N} \max(0, -(\hat{y}_i - \hat{y}_j)(y_i - y_j)) \end{aligned} \tag{3.11}$$

The first part in Eq (3.11) is the regression loss, which illustrates the difference between the ground-truth and prediction values. The second part is the pairwise max-margin loss proposed in [45] to measure the risk of ranking. α is a parameter to balance these two losses. The total loss takes into account two factors. The ranking error considered the relative order of indices, which will facilitate the top-k indices recommendation. The regression error reduces the difference between predicted and actual values, which will help the investment decisions. For each investment day, we first predict the return ratios of the next day for all indices with this model and then rank them from high to low. The top-k indices would be the profitable ones to recommend.

4. Experiment

This section gives evaluations of our model. We compared our model with the current popular methods and also analyzed the effect of different model components. Section 4.1 describes data preparation, including dataset selection, data processing, etc. Section 4.2 lays out the experimental design, including evaluation metrics, comparison methods, parameter setting, etc. Sections 4.3 and 4.4 show the comparison results for different evaluation metrics. Section 4.5 gives the ablation study.

4.1. Data preparation

According to the design of our model, minute-level historical data should be acquired as evaluation data. Although stock market data is publicly available, minute-level data is hard to obtain. The datasets

used in the related works include Taiwan Stocks [6], shares from S&P 500, NASDAQ and NYSE [5], stocks in TOPIX-100 Index [40], Chinese stocks [9], etc. All these easily obtained data are daily-level. In order to meet our experimental needs, we bought the data interface for Chinese stock indices from a third party. This interface can provide minute-level K-lines for over 500 indices in real time. However, not all these indices are suitable for our experiments. We selected the used indices with the following rules. First, the indices which were compiled after 2005-01-04 are deleted to ensure a large number of samples. Then, we keep only one index among similar ones, as similar indices may hold comparable trends. For example, there are six indices about information technology, and we just keep the CSI Information Technology Index (SH000935). Third, some infrequently used indices such as the Equally Weighted Index (SH000971), the Cninfo 100 Index (SZ399313), etc. are removed as it is hard to find the corresponding ETFs for these indices. After filtering, we selected 60 indices for our experiment. The index list can be found in Table 1, where "SH" and "SZ" represent the Shanghai Stock Exchange and the Shenzhen Stock Exchange, respectively.

Table 1. Index list for our evaluations. It contains 60 indices. These indices were all compiled before 2005-01-04. We also delete some similar or infrequently used indices.

Number	Code	Number	Code	Number	Code	Number	Code
1	SH000001	16	SH000903	31	SH000943	46	SZ399233
2	SH000005	17	SH000904	32	SH000944	47	SZ399234
3	SH000006	18	SH000905	33	SH000949	48	SZ399235
4	SH000009	19	SH000922	34	SH000958	49	SZ399236
5	SH000016	20	SH000925	35	SH000961	50	SZ399237
6	SH000044	21	SH000928	36	SH000962	51	SZ399248
7	SH000046	22	SH000929	37	SH000963	52	SZ399300
8	SH000047	23	SH000930	38	SH000964	53	SZ399320
9	SH000063	24	SH000931	39	SH000967	54	SZ399344
10	SH000064	25	SH000932	40	SH000972	55	SZ399348
11	SH000065	26	SH000933	41	SH000980	56	SZ399351
12	SH000102	27	SH000934	42	SZ399001	57	SZ399701
13	SH000130	28	SH000935	43	SZ399004	58	SZ399967
14	SH000132	29	SH000936	44	SZ399231	59	SZ399972
15	SH000300	30	SH000937	45	SZ399232	60	SZ399986

The 5-minute K-line data that ranges from 2005-01-04 to 2022-10-31 is adopted for our model. We first split these data into prediction samples with a one-day step sliding window. Each sample contains $12,000 = 250 \times 48$ K-lines, where 250 is the number of trading days for about one year, and 48 is the number of 5-minute K-lines for each trading day. Comparing to the previous works, our input length is extremely long, and such long data cannot be efficiently handled by a single sequence model. The return ratio for the next day is considered as the prediction label.

After sample splitting, we obtain 4080 samples. These samples are divided into training, validation and testing sets in the ratio of 6:2:2. After the sample is input to the model, it is first synthesized into different scale sequences using Algorithm 1. The scale list and the corresponding parameters are

shown in Table 2. The first row in this table shows the scales, where (m, h, d, w, mon) are abbreviations of (minute, hour, day, week, month), respectively. Taking 1 h for example, it means each K-line covers one hour of trading data in the synthesized K-line sequence. As it contains 10 scales, the parameter $\mathbb{N}$ equals 10 in the model. The second row shows the number of 5-minute K-lines included in each synthesized K-line, which is the second input of Algorithm 1. Still taking the scale of 1 h as an example, as one hour consists of 12 5-minute segments, the corresponding value in the second row is 12. The third row shows the sequence length for different scales, which is the third input for Algorithm 1. The fourth row indicates the total time span for different scales. As the sequence length is 8 for the scale of 1 h, it covers $1\ h \times 8 = 8\ h$, which equals the trading time of two days in the Chinese stock market.

Table 2. Scale list and the corresponding parameters. The second row is the number of 5-minute K-lines in each synthesized K-line, and the third row is the length of the synthesized sequence. They are the second and third parameters of Algorithm 1, respectively. The fourth row is the total time that each sequence covers, and it equals the first row multiplied by the third row.

scale	5 m	10 m	30 m	1 h	2 h	1 d	2 d	1 w	2 w	1 mon
K-line number	1	2	6	12	24	48	96	240	480	960
sequence length	12	12	8	8	10	10	10	8	12	12
time span	1 h	2 h	1 d	2 d	1 w	2 w	1 mon	2 mon	6 mon	1 y

4.2. Experimental design

The PyTorch framework was adopted to implement our model. The training and validation sets were first applied to train the model, and then the prediction results of the testing set were obtained. We used the grid search strategy to determine the optimal hyperparameters. The final optimal parameters are as follows. The dimensions of embeddings m_0, m_1 and m_2 are set to 30, 25 and 20, respectively. The batch size is set to 128, and the learning rate and the drop rate are set to 0.001 and 0.2, respectively. The number of attention heads L is set to 5. The loss weight α is set to 4. The Adam [23] optimizer is applied to train our model.

When the return ratios for each day are predicted, the indices are first ranked, and the top-k indices are obtained. In order to facilitate comparison with other methods, some evaluation metrics such as mean reciprocal rank (*MRR*), precision (*Pre*) and accuracy (*Acc*) are adopted. The mean reciprocal rank is shown as

$$MRR_k = \frac{1}{k} \sum_{in_i \in \hat{\boldsymbol{I}}_k} \frac{i}{rank(in_i)}, \tag{4.1}$$

where $\hat{\boldsymbol{I}}_k$ is the set that contains recommended top-k indices, i is the predicted rank of index in_i, and $rank(in_i)$ is the ground-truth rank of index i. This indicator measures the difference between the predicted and the ground-truth index set, and it ranges from $\frac{1}{k}\sum_{j=0}^{k-1}\frac{j+1}{N-j}$ to 1. The precision is shown as

$$Pre_k = \frac{|\hat{\boldsymbol{I}}_k \cap \boldsymbol{I}_k|}{k}, \tag{4.2}$$

where $\boldsymbol{I}_k$ is the actual top-k index set. This metric ranges from 0 to 1. The accuracy (*Acc*) is the correct

ratio for predicting ups and downs, shown as

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4.3}$$

where TP, TN, FP and FN stand for true positive, true negative, false positive and false negative, respectively. For all three of these metrics, the larger the indicator is the better the results.

In addition, we employ the market backtesting comparison, which is closer to actual investment. In these evaluations, some money is supposed to be invested in index funds. For each trading day, the model predicts the top-k indices for the next day. The investment operation is performed based on the prediction results. We first sell all the shares of the indices that are not in the top-k set. Then, the indices that do not hold in the top-k set are brought in. The total money is divided into k equal parts. Each purchase costs one part. It should be noted that some investment details are ignored. First, we use the closing point as the trading price in the evaluation. However, indices in the Chinese stock market cannot be traded directly. The investor should select the corresponding ETF for investment. Second, we suppose the stock market is liquid enough, and the trading commissions are also not considered. This backtesting evaluation just adopts the simplest trading strategy. In practice, some additional strategies should be taken into account, such as quantitative timing, position management, etc. The commonly used metric is the total return ratio (Trr), shown as

$$Trr = \frac{total - initial}{initial} \times 100\%, \tag{4.4}$$

where *initial* and *total* are the money before and after the investment. We compared the proposed model with several related classical works.

- LSTM [46], which inputs the daily features to the classical LSTM model. It is the simplest method.
- SFM [47], which decomposed the historical sequence into multiple frequency components in memory cells. Future stock trends are predicted by combining these components.
- RSR [5], a state-of-the-art method that applied the LSTM to model the sequence, and utilized the graph convolution network to obtain the relational embeddings.
- AFHGN [7], which used the attribute-driven fuzzy hypergraph network to conduct the stock recommendation issue. It applied fuzzy clustering to generate the stock matrix and eliminated the industry-belonging hypergraph as the indices have no such information.
- FinGAT [6], a model which first used a hierarchical learning component to learn sequential patterns, and then used two fully-connected graphs to learn the interactions between indices.
- RA-AGAT [41], which first used the attention LSTM to get the feature matrix, and the attributed graph attention network was then applied to analyze the connections behind the finance. The mutual information and the cross-correlation analysis coefficient are adopted to construct the adjacent matrix.
- ALSP-TF [43], a model that uses Transformer to capture the sequential embeddings, and focuses on overcoming the limitations of canonical self-attention including context and being position agnostic.

It should be known that our selected indices have no background data such as sector-industry relations. We use dynamic time warping to construct the indices' relations in re-implementing RSR and FinGAT. The corresponding graph aggregation method is also modified.

4.3. Performance comparison

This section presents comparison results. Figures 4 and 5 give the results of different methods using the metrics *MRR* and *Pre*. We give results for different recommendation numbers, which are 5, 10 and 20. These figures show that our proposed MSGraph significantly outperforms the other methods. When the number of recommended indices is 5, our method improves by 1.2% and 2.5% with evaluation metrics *MRR* and *Pre*. However, when the recommended number is 10, our method improves by about 4.6% and 3.5%. As the recommended number rises to 20, our method still performs better than other methods, and the metrics improve by 1.9% and 0.9%, respectively. These results show that our method performs best when recommending 10 indices. Figure 6 plots the prediction accuracy for all indices. The main bar shows the average accuracy, and the error bar indicates the range of prediction accuracy. It indicates our method also obtains the best result and improves the accuracy by about 2.1%. The reasons for the better performance are analyzed as follows.

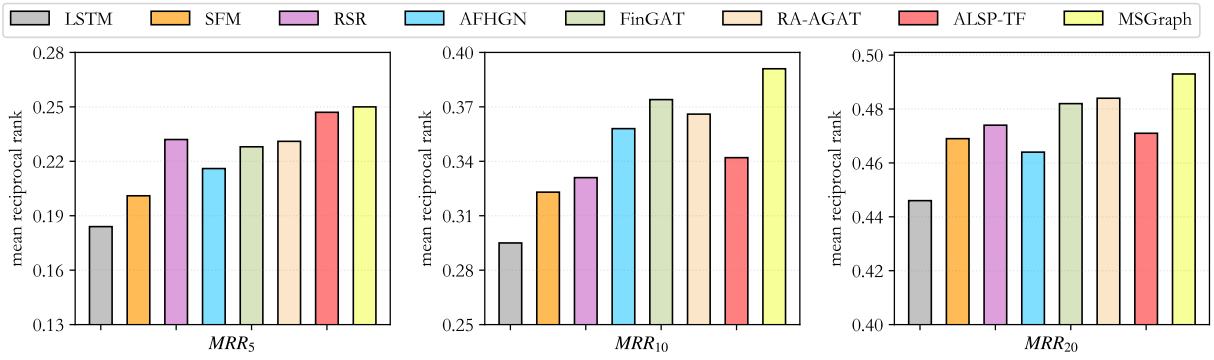


Figure 4. The comparison results for metric *MRR* with $k = 5$, 10 and 20. For all three figures, our method obtained state-of-the-art results, and improved about 1.2%, 4.6% and 1.9%, respectively.

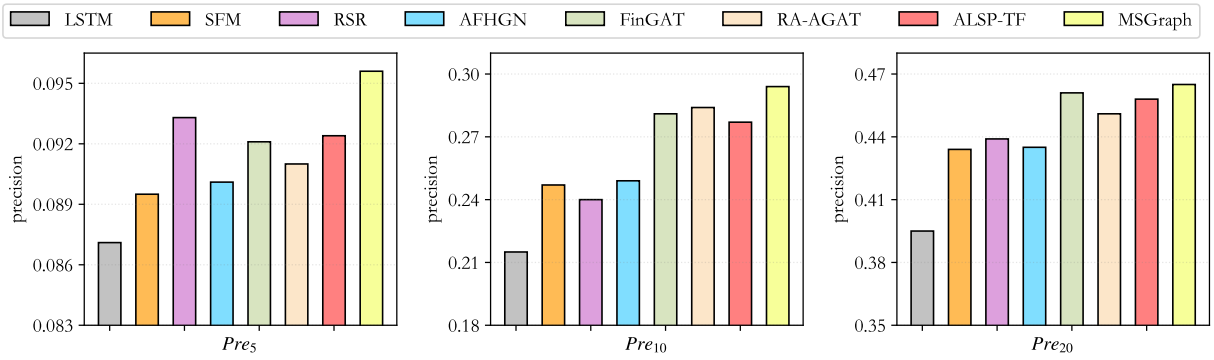


Figure 5. The comparison results for metric *Pre* with $k = 5$, 10 and 20. For these three figures, our method performs best, and improved about 2.5%, 3.5% and 0.9%, respectively.

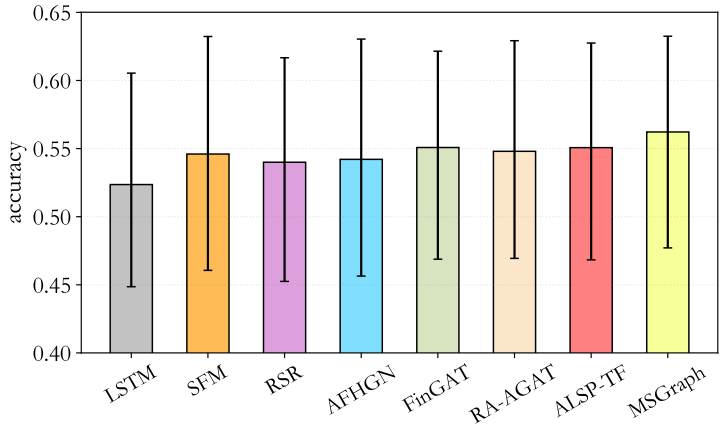


Figure 6. The comparison results for the metric of prediction accuracy. Our method is 56.2%, and improved about 2.1% more than the current best method.

The LSTM performs the worst since it does not take into account index correlations and multiple scale K-lines. The SFM considers the multiple frequencies of input data, but it loses sight of correlations between indices. It performs slightly better. These two results also indicate that utilizing the correlations between indices can significantly improve ranking performance. The RSR is the most similar scheme to our method. However, it just uses the daily level data for prediction, and it loses sight of multiple frequency data. FinGAT, RA-AGAT and ALSP-TF outperform RSR by improving the sequential embedding method with attentive RNN and Transformer. This also implies that the attention mechanism is an effective strategy for raising prediction performance. In the original article, the AFHGN tries to model the stock correlations with hypergraph. However, for the index prediction task, the hypergraph cannot be built, and it is similar to RSR. It consequently performs badly. However, these four methods do not take into account the K-line data with multiple time scales, and the scale correlations are also not considered. Thus, these methods all perform worse than our method. The result also indicates that interactions between indices and correlations between multi-scale trading data are very effective for improving the forecasting effect.

4.4. Market backtesting

This section presents the backtesting comparison results. Backtesting runs from 2020-08-03 to 2022-10-31. In these evaluations, the prediction methods are first used to get the recommended k indices for each day. If the recommended indices have already been held, we hold them still. Otherwise, we changed the indices that were held to the newly recommended ones. For each trading day, we calculate the corresponding amount of each index that is held, and the total amount is used to fetch the return ratio. Due to the randomness of the backtesting curve, we repeat the evaluation 100 times to obtain the average curves.

Figure 7 shows the investment return ratio curves for different prediction methods, in which five indices are recommended as investment targets. As AFHGN and RA-AGAT are similar to RSR and FinGAT, respectively, we did not draw this method's curve in order to simplify this figure. It is clear that our method performs much better than others and gets a 25.6% return ratio. FinGAT produced the

best results among current methods, with a return ratio of 18.4%. The proposed MSGraph improved the return ratio by about 7.2%. The worst-performing method is still the LSTM, as it takes neither index correlations nor scale associations into account. Apart from LSTM and FinGAT, the other three models produce similar curves.

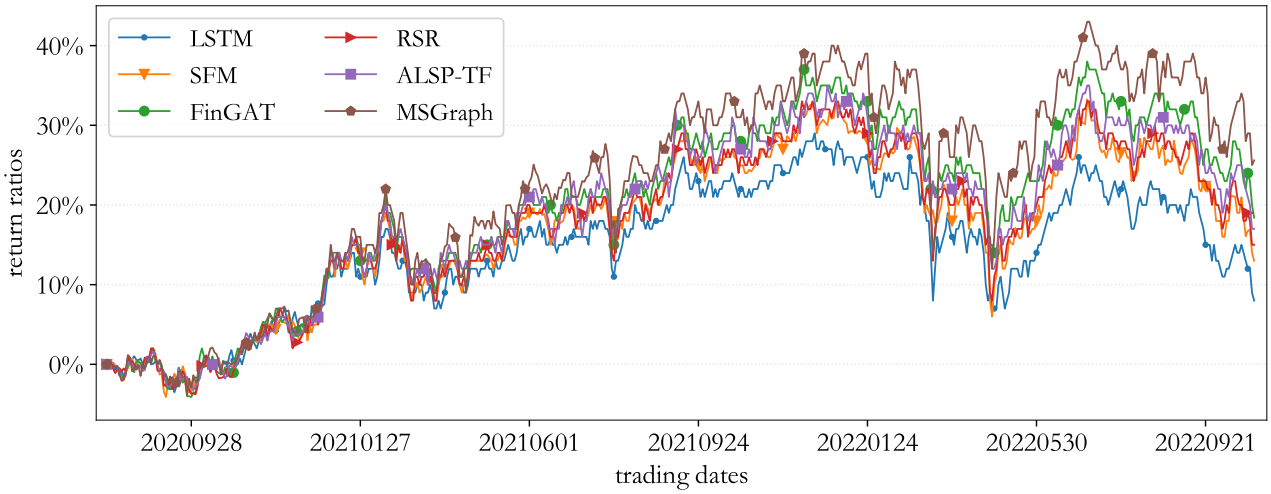


Figure 7. The backtesting curves for different methods. The current best method FinGAT yields 18.4% return ratio, and our MSGraph obtained 25.6%. Our method improves by about 7.2%.

Figure 8 shows the backtesting curves for a number of stock market indices, as well as the results of comparing different methods to make predictions. We chose five indices for comparison. The SH000001 and SZ399001 are the SSE Composite Index and SZSE Component Index, which are compiled by the Shanghai Stock Exchange and the Shenzhen Stock Exchange, respectively. The full name of SH000300 is the CSI 300 Index, which is produced by the China Securities Index Company. These three indices are the standard benchmarks, reflecting the overall trends of the Chinese stock market. SH000928 and SZ399248 are the CSI Energy Index and SZSE Culture Index, respectively, which are the best and worst performing indices in our 60 experimental data. The MSGraph5, MSGraph10 and MSGraph20 are our methods, which take the top 5, 10 and 20 indices as the investment targets, respectively.

We can acquire the following results. First, our prediction methods obviously outperform the stock market's overall trends. The composite indices are -14.1%, -25.5% and -26.5% of income, respectively. The worst performance index is a negative 46.8% return. However, our methods obtain a return of over 10%. This reveals that the indices recommendation can benefit index investment. Second, the total return ratio of our method is lower than the best index (SH000928), but our maximum drawdown ratio is also small. This demonstrates that our methods are more stable. The reason is as follows. Our investment targets contain multiple indices, and the curves are less affected by fluctuations in a single index. It is for this reason that we cannot obtain the highest income. However when it comes to investing money, a steady growth curve is more important than a curve that changes a lot. Third, among the three curves of our methods, MSGraph5 performs better than MSGraph10 and MSGraph20, which can be explained as follows. Most indices are trending downward during this backtesting period. As the

number of indices invested goes up, more downward indices are chosen, which makes the investment return ratio go down.

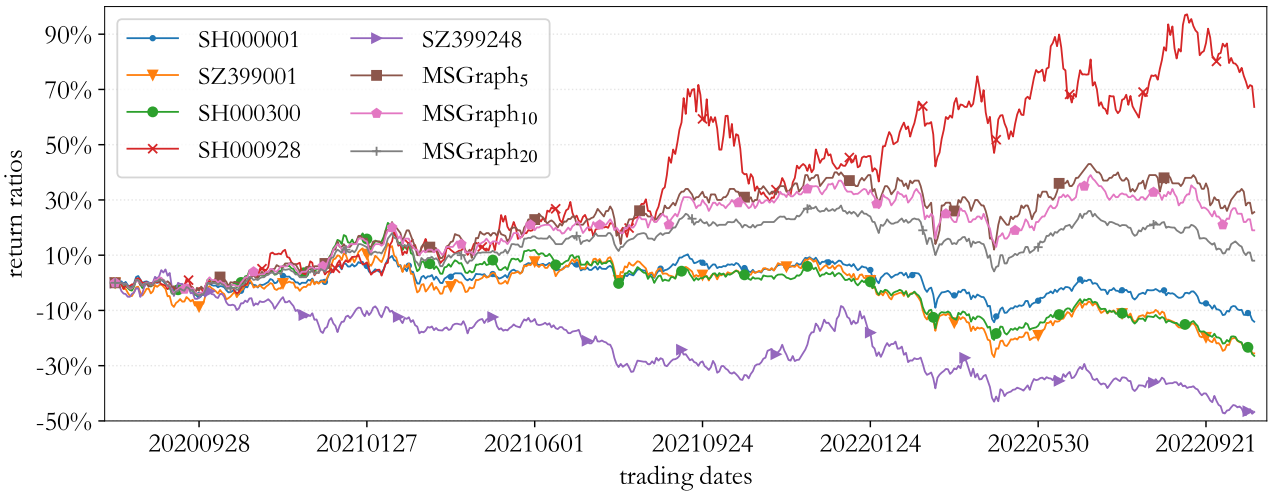


Figure 8. Comparison results with different stock market indices. SH000928 (63.7%) and SZ399248 (-46.8%) are the best and worst-performing indices in our 60 experimental data sets, respectively. SH000001 (-14.1%), SZ399001 (-25.5%) and SH000300 (-26.5%) represent the overall market trends. Although our method performs worse than the top-performing index, it significantly outperforms the overall market return.

4.5. Ablation study

This section will give the ablation study of our model in order to in-depth analyze the effectiveness of the model. Some influencing factors need to be analyzed, including various model components, different data scales and the balance parameter α in the loss function.

4.5.1. Effects of various model components

The previous sections present the results of a comparison between our method and current methods. This section will analyze the effectiveness of different components of our method. Our method contains several critical components, such as index aggregation introduced in Section 3.3, scale aggregation and multi-head attention mechanism which are put forward in Section 3.4. We eliminate these parts from our model and perform the index ranking evaluation. These modified methods are denoted as follows.

- MSGraph, our proposed model with all components.
- MSGraph-I, eliminating the index aggregation from MSGraph. In other words, the sequence embeddings $\boldsymbol{e}_i^{s_j}$ for various scales are fed directly into the scale graph attention network.
- MSGraph-S, eliminating the scale aggregation from MSGraph. That is, the 1-day level K-lines are input into the sequence model and index graph aggregation. The aggregated embeddings $\hat{\boldsymbol{e}}_i^{s_j}$ are fed into the fully connected network directly.
- MSGraph-A, eliminating the multi-head attention in the scale graph attention network, which uses just one attention weight to perform scale aggregation.

The ablation study results for different methods are shown in Table 3. The columns $k = 5$, $k = 10$ and $k = 20$ contain the ranking results. The Acc involves the classification results for all indices, where three columns represent the average, the best and the worst accuracy, respectively. From this table, we can see that all three modified methods perform worse than the original model. This means the reduced three components are all beneficial to ranking prediction. By observing the detailed results of three methods, MSGraph-S performs basically the worst. According to the design of this method, the scale aggregation is removed, and this method is essentially equivalent to RSR in Section 4.2. It also means the scale graph attention network is the most important component of our proposed model. Compared with MSGraph-S, MSGraph-I performs slightly better, but it is also significantly worse than our method. It means that the relationships between the indices are also a key part of making better predictions. In addition, MSGraph-A performs best in these three methods. The reason is that it includes both the index and scale aggregation. However it is also slightly worse than MSGraph. It implies that the multi-head attention can also facilitate prediction performance.

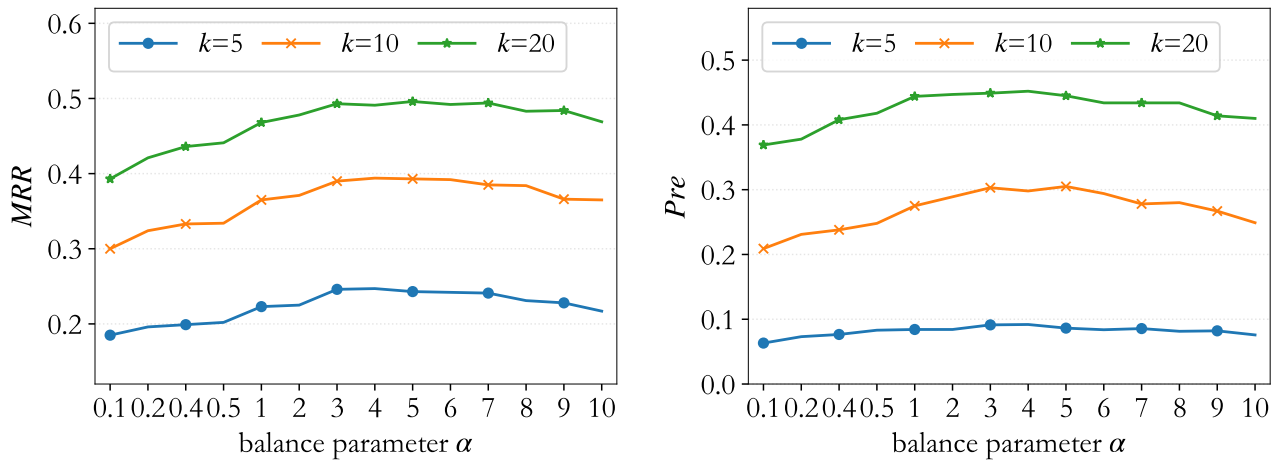
Table 3. Results on ablation study for the proposed MSGraph.

	$k = 5$		$k = 10$		$k = 20$		*Acc*		
	MRR	*Pre*	*MRR*	*Pre*	*MRR*	*Pre*	Aver	Best	Worst
MSGraph	0.250	0.0956	0.391	0.2940	0.493	0.4650	0.5622	0.6325	0.4771
MSGraph-I	0.231	0.0925	0.357	0.2668	0.459	0.4396	0.5510	0.6348	0.4595
MSGraph-S	0.213	0.0901	0.332	0.2520	0.464	0.4400	0.5410	0.6192	0.4504
MSGraph-A	0.247	0.0944	0.374	0.2761	0.478	0.4595	0.5602	0.6315	0.4731

4.5.2. Analysis on data scales

This section will address another topic: which scale is more important for ranking prediction. The previous sections used all 10 data scales introduced in Table 2. This part selects a subset of Table 2 for evaluation. Table 4 shows the best MRR_{10} results and their corresponding scale lists for different scale numbers. Taking MSGraph-5 for example, it means we select 5 scales to construct the scale graph attention network. Since there are C_{10}^5 combinations to generate the subset, we traverse all of them and train the prediction model for each combination. The best prediction result and its corresponding scale list are the final results.

From the table, we can see that the MRR_{10} increases significantly as the scale number changes from 1 to 7. Then, it generally stays the same. This reveals that multi-scale transaction data can facilitate future predictions. As the scale number is large, the expression ability of some scales becomes the same. Thus, the indicator does not improve. From the column of scale lists, we can get several recurring scales, which are 1 mon, 1 d, 2 h, 10 m, etc. Observing the MSGraph-5 to MSGraph-7, we can find the scale list contains both low and high frequency data, which indicates the combination of long-term and short-term data can substantially enhance the prediction results. This finding also offers a novel perspective on the field of study.



(a) results of *MRR* **(b)** results of *Pre*

Figure 9. The ablation results of balance parameter α with different metrics. The results show that when α is between 3 and 5, the model can get optimal results.

Table 4. MRR_{10} for different scale numbers.

	MRR_{10}	scale list
MSGraph-1	0.332	1 d
MSGraph-2	0.345	1 d, 10 m
MSGraph-3	0.353	2 w, 2 h, 10 m
MSGraph-4	0.371	1 mon, 2 d, 2 h, 10 m
MSGraph-5	0.382	1 mon, 1 w, 1 d, 2 h, 5 m
MSGraph-6	0.386	1 mon, 1 w, 2 d, 1 d, 1 h, 5 m
MSGraph-7	0.393	1 mon, 2 w, 2 d, 1 d, 2 h, 30 m, 10 m
MSGraph-8	0.388	1 mon, 2 w, 1 w, 1 d, 2 h, 30 m, 10 m, 5 m
MSGraph-9	0.392	1 mon, 2 w, 1 w, 1 d, 2 h, 1 h, 30 m, 10 m, 5 m
MSGraph-10	0.391	1 mon, 2 w, 1 w, 2 d, 1 d, 2 h, 1 h, 30 m, 10 m, 5 m

4.5.3. Selection of balance parameter α

The parameter α in formula (3.11) is used to balance the regression loss and the ranking loss. This value can also affect the performance of our proposed model. In Section 4.2, this parameter is set to 4. This section will put forward the selection of this parameter. Following the previous studies [5], we tune α within the set [0.1, 0.2, 0.4, 0.5, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10] with the other hyperparameters fixed. $\alpha < 1$ indicates that the regression loss dominates the model training, while $\alpha > 1$ means ranking loss accounts for a large proportion. The prediction results with metrics *MRR* and *Pre* are shown in Figure 9. The results show that the indicators improve first and then decline with α changes. When it comes to 3–5, we can get relatively good results. For *MRR*20 and *Pre*10, $\alpha = 5$ yielded the optimal results, while for the rest of the indicators, $\alpha = 4$ is the best. Thus, we selected $\alpha = 4$ as the optimal value in Section 4.2.

5. Conclusions

Indices recommendation is a long-standing topic in stock market investment. Improving the prediction performance of future trends is a central issue. This paper focuses on the impacts of different scales of transaction data on the prediction result. The contributions of this article are as follows. First, the MSGraph framework is proposed, which considers both index aggregation and scale aggregation. Second, the data synthesis and index aggregation are detailed. Third, the scale graph attention network is proposed to model the multi-scale correlations, where each scale is regarded as a node. In the experiment, 60 Chinese stock market indices are used to demonstrate its effectiveness, and the effects of different model components are analyzed.

The proposed MSGraph is the first study on correlations between different scales of transaction data. There are several future research directions. First, the LSTM is used as the sequence embedding model in our method. Currently, there are some more powerful sequence models that can be utilized, such as Transformer, Autoformer, etc. This part could be improved. Second, this article recommends profitable indices based on daily prediction results. Actually, investment returns are determined by both short and long-term prediction results. Redefining the prediction task and developing the trading strategy is a promising research direction. In addition, it would be interesting to learn more about how multi-scale historical data affects long-term prediction tasks.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (61872439), the Key Science and Technology Program of Henan Province (212102210096, 222102210030).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. R. D. Edwards, J. Magee, W. C. Bassetti, *Technical Analysis of Stock Trends*, CRC press, 2018. https://doi.org/10.4324/9781315115719
2. Y. Wang, Y. Guo, Forecasting method of stock market volatility in time series data based on mixed model of arima and xgboost, *China Commun.*, **17** (2020), 205–221. https://doi.org/10.23919/JCC.2020.03.017
3. S. Barra, S. M. Carta, A. Corriga, A. S. Podda, D. R. Recupero, Deep learning and time series-to-image encoding for financial forecasting, *IEEE/CAA J. Autom. Sin.*, **7** (2020), 683–692. https://doi.org/10.1109/jas.2020.1003132
4. Y. Zhou, H. Zheng, X. Huang, S. Hao, D. Li, J. Zhao, Graph neural networks: Taxonomy, advances, and trends, *ACM Trans. Intell. Syst. Technol.*, **13** (2022), 1–54. https://doi.org/10.1145/3495161

5. F. Feng, X. He, X. Wang, C. Luo, Y. Liu, T. S. Chua, Temporal relational ranking for stock prediction, *ACM Trans. Inf. Syst.*, **37** (2019), 1–30. https://doi.org/10.1145/3309547
6. Y. L. Hsu, Y. C. Tsai, C. T. Li, Fingat: Financial graph attention networks for recommending top-k profitable stocks, *IEEE Trans. Knowl. Data Eng.*, **35** (2021), 469–481. https://doi.org/10.1109/tkde.2021.3079496
7. X. Ma, T. Zhao, Q. Guo, X. Li, C. Zhang, Fuzzy hypergraph network for recommending top-k profitable stocks, *Inf. Sci.*, **613** (2022), 239–255. https://doi.org/10.1016/j.ins.2022.09.010
8. C. Wang, H. Liang, B. Wang, X. Cui, Y. Xu, Mg-conv: A spatiotemporal multi-graph convolutional neural network for stock market index trend prediction, *Comput. Electr. Eng.* , **103** (2022), 108285. https://doi.org/10.1016/j.compeleceng.2022.108285
9. W. Chen, M. Jiang, W. G. Zhang, Z. Chen, A novel graph convolutional feature based convolutional neural network for stock trend prediction, *Inf. Sci.*, **556** (2021), 67–94. https://doi.org/10.1016/j.ins.2020.12.068
10. R. Bisoi, P. K. Dash, A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented kalman filter, *Appl. Soft Comput.*, **19** (2014), 41–56. https://doi.org/10.1016/j.asoc.2014.01.039
11. S. Li, X. Jiang, J. Wu, L. Tong, K. Xu, Detection of mutual exciting structure in stock price trend dynamics, *Entropy*, **23** (2021), 1411. https://doi.org/10.3390/e23111411
12. L. Cao, Ai in finance: Challenges, techniques, and opportunities, *ACM Comput. Surv.*, **55** (2022), 1–38. https://doi.org/10.1145/3502289
13. L. Wang, F. Ma, J. Liu, L. Yang, Forecasting stock price volatility: New evidence from the garch-midas model, *Int. J. Forecasting*, **36** (2020), 684–694. https://doi.org/10.1016/j.ijforecast.2019.08.005
14. R. A. Kamble, Short and long term stock trend prediction using decision tree, in *2017 International Conference on Intelligent Computing and Control Systems*, IEEE, (2017), 1371–1375. https://doi.org/10.1109/ICCONS.2017.8250694
15. N. I. Sapankevych, R. Sankar, Time series prediction using support vector machines: a survey, *IEEE Comput. Intell. Mag.*, **4** (2009), 24–38. https://doi.org/10.1109/mci.2009.932254
16. P. Shahbaz, B. Ahmad, E. Atani Reza, J. Moghaddam Jalal, Stock market forecasting using artificial neural networks, *Eur. Online J. Nat. Social Sci. Proc.*, **2** (2014), 2404–2411.
17. M. Zolfaghari, S. Gholami, A hybrid approach of adaptive wavelet transform, long short-term memory and arima-garch family models for the stock index prediction, *Expert Syst. Appl.*, **182** (2021), 115149. https://doi.org/10.1016/j.eswa.2021.115149
18. K. K. Yun, S. W. Yoon, D. Won, Prediction of stock price direction using a hybrid ga-xgboost algorithm with a three-stage feature engineering process, *Expert Syst. Appl.*, **186** (2021), 115716. https://doi.org/10.1016/j.eswa.2021.115716
19. S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, et al., A survey on deep learning: Algorithms, techniques, and applications, *ACM Comput. Surv.*, **51** (2018), 1–36. https://doi.org/10.1145/3234150

20. E. Chong, C. Han, F. C. Park, Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies, *Expert Syst. Appl.*, **83** (2017), 187–205. https://doi.org/10.1016/j.eswa.2017.04.030

21. J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, *J. Mach. Learn. Res.*, **12** (2011), 2121–2159.

22. M. D. Zeiler, Adadelta: an adaptive learning rate method, *arXiv preprint*, arXiv:1212.5701. https://doi.org/10.48550/arXiv.1212.5701

23. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, *arXiv preprint*, arXiv:1412.6980. https://doi.org/10.48550/arXiv.1412.6980

24. A. Bose, C. H. Hsu, S. S. Roy, K. C. Lee, B. Mohammadi-ivatloo, S. Abimannan, Forecasting stock price by hybrid model of cascading multivariate adaptive regression splines and deep neural network, *Comput. Electr. Eng.*, **95** (2021), 107405. https://doi.org/10.1016/j.compeleceng.2021.107405

25. O. B. Sezer, A. M. Ozbayoglu, Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach, *Appl. Soft Comput.*, **70** (2018), 525–538. https://doi.org/10.1016/j.asoc.2018.04.024

26. X. Zhang, N. Gu, J. Chang, H. Ye, Predicting stock price movement using a dbn-rnn, *Appl. Artif. Intell.*, **35** (2021), 876–892. https://doi.org/10.1080/08839514.2021.1942520

27. D. Fister, M. Perc, T. Jagrič, Two robust long short-term memory frameworks for trading stocks, *Artif. Intell.*, **51** (2021), 7177–7195. https://doi.org/10.1007/s10489-021-02249-x

28. H. He, S. Dai, A prediction model for stock market based on the integration of independent component analysis and multi-lstm, *Electron. Res. Arch.*, **30** (2022), 3855–3871. https://doi.org/10.3934/era.2022196

29. C. Wang, Y. Chen, S. Zhang, Q. Zhang, Stock market index prediction using deep transformer model, *Expert Syst. Appl.*, **208** (2022), 118128. https://doi.org/10.1016/j.eswa.2022.118128

30. R. Zhang, Z. Yuan, X. Shao, A new combined cnn-rnn model for sector stock price analysis, in *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, IEEE, (2018), 546–551. https://doi.org/10.1109/COMPSAC.2018.10292

31. W. Lu, J. Li, J. Wang, L. Qin, A cnn-bilstm-am method for stock price prediction, *Neural Comput. Appl.*, **33** (2021), 4741–4753. https://doi.org/10.1007/s00521-020-05532-z

32. A. F. Kamara, E. Chen, Z. Pan, An ensemble of a boosted hybrid of deep learning models and technical analysis for forecasting stock prices, *Inf. Sci.*, **594** (2022), 1–19. https://doi.org/10.1016/j.ins.2022.02.015

33. W. Jiang, J. Luo, Graph neural network for traffic forecasting: A survey, *Expert Syst. Appl.*, **207** (2022), 117921. https://doi.org/10.1016/j.eswa.2022.117921

34. L. Wu, Y. Chen, K. Shen, X. Guo, H. Gao, S. Li, et al., Graph neural networks for natural language processing: A survey, *arXiv preprint*, arXiv:2106.06090. https://doi.org/10.48550/arXiv.2106.06090

35. W. Liao, B. Bak-Jensen, J. R. Pillai, Y. Wang, Y. Wang, A review of graph neural networks and their applications in power systems, *J. Mod. Power Syst. Clean Energy*, **10** (2021), 345–360. https://doi.org/10.35833/MPCE.2021.000058

36. J. Wu, K. Xu, X. Chen, S. Li, J. Zhao, Price graphs: Utilizing the structural information of financial time series for stock prediction, *Inf. Sci.*, **588** (2022), 405–424. https://doi.org/10.1016/j.ins.2021.12.089

37. S. Li, J. Wu, X. Jiang, K. Xu, Chart gcn: Learning chart information with a graph convolutional network for stock movement prediction, *Knowledge-Based Syst.*, **248** (2022), 108842. https://doi.org/10.1016/j.knosys.2022.108842

38. R. Cheng, Q. Li, Modeling the momentum spillover effect for stock prediction via attribute-driven graph attention networks, in *Proceedings of the AAAI Conference on Artificial Intelligence*, AAAI, (2021), 55–62. https://doi.org/10.1609/aaai.v35i1.16077

39. T. Yin, C. Liu, F. Ding, Z. Feng, B. Yuan, N. Zhang, Graph-based stock correlation and prediction for high-frequency trading systems, *Pattern Recognit.*, **122** (2022), 108209. https://doi.org/10.1016/j.patcog.2021.108209

40. W. Li, R. Bao, K. Harimoto, D. Chen, J. Xu, Q. Su, Modeling the stock relation with graph network for overnight stock movement prediction, in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, ACM, (2021), 4541–4547. https://doi.org/10.24963/ijcai.2020/626

41. S. Feng, C. Xu, Y. Zuo, G. Chen, F. Lin, J. XiaHou, Relation-aware dynamic attributed graph attention network for stocks recommendation, *Pattern Recognit.*, **121** (2022), 108119. https://doi.org/10.1016/j.patcog.2021.108119

42. C. Xu, H. Huang, X. Ying, J. Gao, Z. Li, P. Zhang, et al., Hgnn: Hierarchical graph neural network for predicting the classification of price-limit-hitting stocks, *Inf. Sci.*, **607** (2022), 783–798. https://doi.org/10.1016/j.ins.2022.06.010

43. H. Wang, T. Wang, S. Li, J. Zheng, S. Guan, W. Chen, Adaptive long-short pattern transformer for stock investment selection, In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, (2022), 3970–3977. https://doi.org/10.24963/ijcai.2022/551

44. Y. S. Jeong, M. K. Jeong, O. A. Omitaomu, Weighted dynamic time warping for time series classification, *Pattern Recognit.*, **44** (2011), 2231–2240. https://doi.org/10.1016/j.patcog.2010.09.022

45. Z. Zheng, K. Chen, G. Sun, H. Zha, A regression framework for learning ranking functions using relative relevance judgments, in *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ACM, (2007), 287–294. https://doi.org/10.1145/1277741.1277792

46. J. Luo, G. Zhu, H. Xiang, Artificial intelligent based day-ahead stock market profit forecasting, *Comput. Electr. Eng.*, **99** (2022), 107837. https://doi.org/10.1016/j.compeleceng.2022.107837

47. L. Zhang, C. Aggarwal, G. J. Qi, Stock price prediction via discovering multi-frequency trading patterns, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, (2017), 2141–2149. https://doi.org/10.1145/3097983.3098117

AIMS Press