A Stock Prediction Model based on LightGCN

Yixin Yang[1], Lifang Yang[2], Hui Lv[3*], Xinyue Di [2]
1. School of Computer and Cyber Sciences, Communication University of China, Beijing, China
2. School of Information and Communication Engineering, Communication University of China, Beijing, China
3. Beagledata Technology (Beijing) Co. Ltd.
yangyx@189.cn, yanglifang@cuc.edu.cn, lvhui@beagledata.com, 1941037087@qq.com

2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC) | 978-1-6654-2207-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ITAIC54216.2022.9836964

Abstract—**Stock prediction aims to predict future prices and trends of stocks in order to help investors make good investment decisions and get more profits. An advanced stock prediction model is Relational Stock Ranking (RSR), which has a great improvement in the cumulative investment return ratio (IRR) than previous studies. However, in RSR model, the relation between stocks is learned with the one-layer graph learning method, which can't comprehensively describe the dependency between different stocks. In this paper, we propose a new model for stock prediction, named L-RSR, which can effectively predict the stock ranking and help investors select stocks with higher return ratios. L-RSR contains a graph learning technique based on LightGCN with multiple layers, leading to a better extraction of the relation features. We conduct a series of experiments with the data from National Association of Securities Dealers Automated Quotations (NASDAQ) market. Experimental results show that L-RSR outperforms RSR with more than 25% improvements in IRR.**

Keywords—Multiple Layers; Stock Prediction; Graph-based Learning; LightGCN; Learning to Rank

I. Introduction

With an overall world market capitalization more than 93 trillion U.S. dollars in 2020, stock transaction grows more important these days. In order to gain higher profits, stock prediction becomes a popular technique for many investors. Though the debate over whether the stock market is predictable or not continues, some recent evidences [1, 3, 4, 5] as well as meaningfulness of the technique itself lead scholars to deeper investigations.

Traditional methods for stock prediction are based on time-series analysis models. However, these kinds of models usually fail to describe a stock comprehensively and can only focus on a few artificially chosen indicators. To overcome the drawbacks, the models based on deep neural networks [6, 10, 11, 12], especially the recurrent neural networks (RNNs), such as vanilla RNN network, Long-Short Term Memory (LSTM) network [9], Gate Recurrent Unit (GRU) network [15] and State Frequency Memory (SFM) network [5], are applied. Though these methods do achieve a good performance with a small mean square error (MSE), they overlook the dependencies between stocks and the influences of stocks ranking on stocks choosing and profits. A method with a good MSE can predict a wrong ranking, which make investors choose less profitable stocks thus receive lower profits.

An advanced solution is Relational Stock Ranking (RSR) network [7], which includes two types of RSR, named RSR_E network and RSR_I network. In its training process, RSR network optimize its model with both pointwise regression loss (specifically, the regression MSE) and pairwise ranking-aware loss, which is conducive to avoid the wrong ranking circumstance and help investors gain more benefits. Apart from that, RSR also takes the relations between different stocks into consideration, for example, the stock of a supplier company may impact the stocks of its consumer companies. This can be a big advancement from the previous models because the stocks are not considered as independence anymore. To achieve this, RSR uses the Temporal Graph Convolution (TGC) which is modified from graph convolutional network (GCN) and is architected with only one layer. However, the TGC architecture has two shortages:

- TGC with one layer can only consider the direct relations between two stocks. For example, if company A is the supplier of company B and company B is the supplier of company C, the relations between stock A and stock B, and stock B and stock C, are considered but the relation between stock A and stock C is overlooked;
- The design of the TGC is complicated in which the impact strength of the relations needs to be learned from both the relation vectors and the temporal information.

To address the aforementioned limitations of RSR, we devise the new model L-RSR, in which the TGC is replaced with a graph learning technique based on LightGCN [8]. LightGCN is one of the most advanced graph learning models, which not only simplifies TGC but also has a better performance. In addition, we revise LightGCN to fit our circumstance of stock prediction. Extensive experiments demonstrate that our L-RSR model outperforms RSR_E with 37.3% improvements in return ratio and RSR_I with 25.2% improvements in return ratio.

II. Preliminaries

A. Long Short-Term Memory

Long Short-Term Memory (LSTM) [9] is effective in processing sequence data [2, 14, 16]. It is a special kind of RNN. While RNN contains hidden states to process sequence of inputs and capture the sequential pattern of data, LSTM adds cell sates to store the long-term memory and capture the long-term dependency in a sequence. In order to decide the fraction of the information to be allowed or forgotten, LSTM consists of many kinds of gates, including input gate, forget gate and output gate. In addition, it contains sigmoid layer and tanh layer as activation functions in the process. Mathematically, LSTM architecture is defined by the following equations:

$$\begin{aligned} f_t &= \sigma\left(W_f h_{t-1} + P_f x_t + b_f\right) \\ i_t &= \sigma(W_i h_{t-1} + P_i x_t + b_i) \\ r_t &= tanh(W_r h_{t-1} + P_r x_t + b_r) \\ c_t &= f_t \odot c_{t-1} + i_t \odot r_t \\ o_t &= \sigma(W_o h_{t-1} + P_o x_t + b_o) \\ h_t &= o_t \odot tanh(c_t), \end{aligned} \tag{1}$$

in which $x_t \in R^D$ is an input vector at time t, where D is the input dimension; h_t and $c_t \in R^U$ denote a hidden state vector and a cell state vector respectively, where M is the hidden dimension;f_t, $i_t$and $o_t \in R^U$ denote output vectors of the forget gate, the input gate and the output gate respectively; r_t is an intermediate result for the cell state at time t; W_f, W_i, W_r, $W_o \in R^{U\times U}$ and P_f, P_i, P_r, $P_o \in R^{U\times D}$ are the mapping matrices; b_f, b_i, b_r, $b_o \in R^U$ are bias vectors. Among them, W, P and b are the parameters that a LSTM cell learns during training.

B. LightGCN

LightGCN [8] is one of the most advanced graph learning models with the light design and the remarkable performance in recommendation. It only remains the most essential components in GCN. Specifically, LightGCN learns the user and item embeddings at layer l through linearly propagating the embeddings at layer $l-1$ on the user-item interaction graph, and then the embeddings at all layers are weighted and added to get the final embedding. The graph convolution operation and the final results e_u and e_i, which are the final vector of user u and the final vector of item i, are defined as follows:

$$\begin{aligned} \boldsymbol{e}_u^{(l+1)} &= \sum_{i\in V_u} \frac{1}{\sqrt{|V_i||V_u|}} \boldsymbol{e}_i^{(l)} \\ \boldsymbol{e}_i^{(l+1)} &= \sum_{u\in V_i} \frac{1}{\sqrt{|V_i||V_u|}} \boldsymbol{e}_u^{(l)} \\ \boldsymbol{e}_u &= \sum_{l=0}^{L} \gamma_l \boldsymbol{e}_u^{(l)} \\ \boldsymbol{e}_i &= \sum_{l=0}^{L} \gamma_l \boldsymbol{e}_i^{(l)}, \end{aligned} \tag{2}$$

where V_i and V_u denote the set of users interacted by item i and the set of items interacted by user u respectively; l means layer l, and L is the total number of layers; γ_l is the hyper-parameters.

In addition, in order to predict the ranking score for recommendation, the model calculates the inner product of user vector $\boldsymbol{e}_u$ and item vector $\boldsymbol{e}_i$, and the final result $\hat{\mathcal{Y}}_{ui}$ is represented below:

$$\hat{\mathcal{Y}}_{ui} = \boldsymbol{e}_u \boldsymbol{e}_i^T. \tag{3}$$

III. L-RSR Model

The former section describes two important methods, LSTM and LightGCN. In this section, we will make use of these methods, and fit them to our L-RSR model.

As illustrated in Fig.1, L-RSR contains four layers: LSTM Layer, Relation Layer, Graph Layer and Prediction Layer. The detailed descriptions of each layer are as follows.

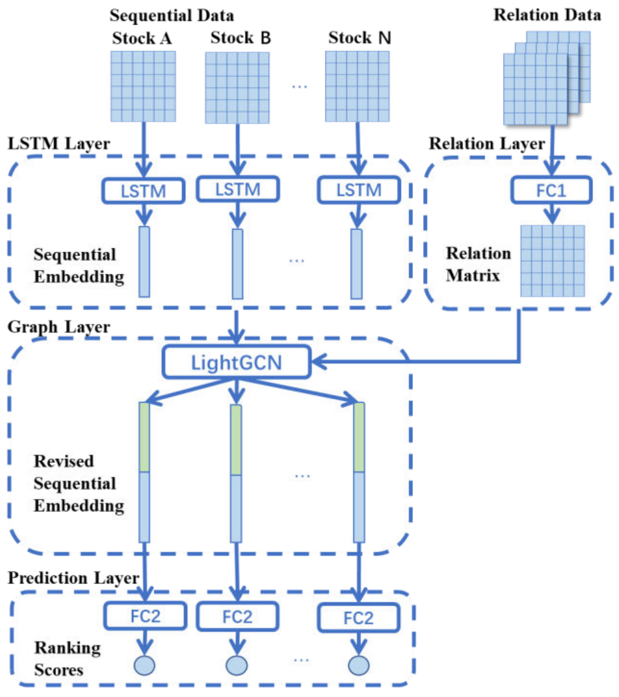

Fig. 1. L-RSR framework. In the framework, the LSTM cells and FC (Fully Connected layer) units depicted in the same layer share the same parameters.

A. LSTM Layer

In this layer, we use the method Long Short-Term Memory to grasp the inherent pattern of the sequences. The reasons are that recent studies [6] have demonstrated the effectiveness of RNN in stock prediction, and among various RNN models, LSTM has outstanding ability in storing long-term memory and capturing long-term dependency, which can be essential for stock prediction.

The input of the layer is a matrix $V \in R^{N\times U\times D}$, which contains the time sequences of N stocks, and in every time sequence, we have a D-dimensional vector for each of U

historical moment. In the LSTM, U also means the number of the hidden units in LSTM. The outputs of the layer are a matrix $E_0 = [e_1, e_2, \dots, e_N] \in R^{N\times D}$, showing the sequence features of those N stocks. In order to get the sequence features at a specific moment *t*, we have the equation as follows:

$$E_0^t = LSTM(V^t). \tag{4}$$

B. Relation Layer

While LSTM layer is used to process sequence data, Relation Layer learns the dependency between different stocks. In order to get the influence between stocks, relational data from Wikidata [13] are gotten. Then, we feed the three-dimensional relation data $G \in R^{N\times N\times K}$ into this full connection layer to get the two-dimensional relation matrix $A \in R^{N\times N}$, where N denotes the number of stock companies and K denotes the number of relation types. We use $G^k$to denote the adjacency matrix of the graph between each stock in the *k*-th relation, and use λ_k as the hyper-parameters for the *k*-th relation. The equation in this layer is as follows:

$$A = \sum_{k=0}^{K-1} \lambda_k G^k. \tag{5}$$

C. Graph Layer

We substitute the original Temporal Graph Convolution (TGN) in RSR with a LightGCN, which contains multiple layers and also simplifies the architecture of TGN. Since the graph network contain more than one layer, the relation it learns is not limited in a direct relation between A_1 and A_2, but also take the relation between A_1 and A_{L+1} into account when A_1 is related to A_2, …, and A_L is related to A_{L+1}, where L is the number of layers.

To fit the context of stock prediction, the LightGCN we use in this work has two differences from the original LightGCN in recommendation context. Firstly, since we only have stocks in stock prediction, user-item interaction graph is changed to the relation graph between different stocks, which is learned from Relation Layer and displayed by relation matrix A; secondly, since the relation is learned from a full connection layer, elements of the matrix denote strengths of relation and are within the range of real numbers while in recommendation context, the elements can only be 1 if the user and the item are interacted otherwise 0. The formula is as follows:

$$E_{l+1} = \left(D^{-\frac{1}{2}} A\, D^{-\frac{1}{2}}\right) E_l, \tag{6}$$

$$T = \left[\sum_{l=1}^{L} \lambda_l E_l\, , E_0\right]. \tag{7}$$

where $A \in R^{N\times N}$ is the relation matrix we get in *B*., and $E_l \in R^{N\times Q}$ indicates embeddings at the l-th layer of the graph network, in which E_0 is the output embedding of LSTM Layer. $T \in R^{N\times 2Q}$ is the output, and it concatenates the weighted sum of E_1 to E_L with the initial sequence feature matrix E_0. We use the weighted sum of E_1 to E_L to revise the sequence features. Since E_0 contains the original sequence features of stocks, we use concatenation to connect E_0 and the weighted sum, which is a lossless operation and can retain all the information of E_0. With Graph layer, the outputs of LSTM are converted to the revised embeddings, for the feature vectors are revised by the dependency between different stocks.

D. Prediction Layer

Finally, we feed the embeddings from Graph layer into a full connected layer to predict the ranking scores of each stock. The propose of the model is to help investors receive high profit. Therefore, when we optimize the model, both MSE and the rank of stocks need to be taken into consideration. the optimization expression are as follows:

$$loss^t = \frac{1}{N}\left(\sum_{n=0}^{N-1} \hat{\mathcal{Y}}_n^t - \mathcal{Y}_n^t\right)^2$$
$$+\alpha \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \max\left(0, -(\hat{\mathcal{Y}}_i^t - \hat{\mathcal{Y}}_j^t)(\mathcal{Y}_i^t - \mathcal{Y}_j^t)\right), \tag{8}$$

where $\hat{\mathcal{Y}}_n^t$ and $\mathcal{Y}_n^t$ are the predicted ranking score and the ground-truth ranking score of stock *n* at time *t*.

IV. Experiments

A. Experimental environment and data

1) Hardware environment: 64-bit Windows 10, CPU 11th Gen Intel(R) Core(TM) i9-11900H @ 2.50GHz, RAM 32G, hard driver 455G.

2) Software environment: Anaconda 3, Python 3.7, Tensorflow 2.3.

3) Data Source: The data we use in this work is same as the data used in the work for RSR in Temporal Relational Ranking for Stock Prediction [7]. The transaction data is collected from NASDAQ market from 01/02/2013 to 12/08/2017 and the relation data from Wikidata. Then, the data is pretreated. The final experiment datasets include the normalized close prices of each day and 5-, 10-, 20-, 30-days moving averages which represent the sequence features. The targets of the prediction are the ground-truth ranking scores of stocks which are also the 1-day return ratios and are calculated with the following equation:

$$r_n^{t+1} = \frac{p_n^{t+1} - p_n^t}{p_n^t}, \tag{9}$$

where p_n^t is the normalized close price of the stock n in day t.

B. Experimental steps

1) Step 1: Under different numbers of layers in LightGCN, running L-RSR model. Then, finding the best number of layers, which has the best performance. In order to report the performance, Mean Square Error (MSE), Mean Reciprocal Rank (MRR), and the cumulative investment return ratio (IRR) are employed as the metrics. IRR is the main metric to be considered, which is sum of the return ratios of the selected stock on each day and can directly reflect the effect of the stock prediction. A performance is better, when value of MSE is smaller, and the value of MRR and IRR is larger.

2) Step 2: To make a good comparison, we run the original RSR and L-RSR in the same environment with the same datasets, the same value of α in Equation (8) and the same hyper-parameters for LSTM Layer and Relation Layer.

C. *Experimental Results and Analysis*

1) Adjusting Layers

We adjust the number of layers in Graph layer from 1 to 5. The results are shown in Table I.

TABLE I. PERFORMANCE COMPARISON BETWEEN L-RSR WITH DIFFERENT LAYERS

NASDAQ			
	MSE	MRR	IRR
One Layer	0.0003797	**0.05186**	2.2491
Two Layers	0.0003797	0.05134	2.5823
Three Layers	0.0003796	0.05135	2.6093
Four Layers	0.0003796	0.05149	**2.6855**
Five Layers	**0.0003796**	0.05119	2.6511

In the range of 1 to 4 layers, the main metrics IRR increases with increasing layers, and the value of IRR reaches the top when the number of layers is 4. The value of IRR drops slightly with five layers. Despite one-layer L-RSR has the best MRR, its advantage in MRR is small, which is only about 0.00037 larger than the second best MRR, and its IRR is the lowest. The values of MSE in three-, four- and five-layer L-RSR are very close to each other. Thus, we have the best L-RSR model when the number of layers in Graph Layer is 4. Usually, increasing the number of layers in LightGCN can improve the performance. Since IRR is decided by the predicted ranking of stock, and a slight change in ranking can make a big variation in the IRR value, it is reasonable to find better and better performances in one to four layers, and a drop in performance with five layers.

2) Results of Different Models and Comparisons

We use the models Rank_LSTM, RSR_E, and RSR_I from Temporal Relational Ranking for Stock Prediction as the baselines. Rank_LSTM is RSR without the graph learning method, and RSR_I and RSR_E are two types of RSR model, in which the letter "I" denotes "Implicate" while "E" denotes "Explicate". Between these two RSR models, there is only a slight difference in their formulas. The previous work [7] has shown that RSR model outperforms other typical methods, for example, SFM and LSTM. Therefore, in this work, we only need to demonstrate that L-RSR has a better performance than RSR model.

From Table II, we can see that L-RSR has the best MRR and IRR results, and RSR_E has the best MSE result. However, the difference between the value of MSE in L-RSR and the best value of MSE in these four models is minor, which is only about 0.0000003. L-RSR has an improvement in the main metrics IRR: compared to RSR_E the IRR performance of L-RSR improves about 37.3%, and compared to RSR_I it improves about 25.2%. This is mainly because that the multiple layers in Graph Layer can extract more information from the relations between stocks.

TABLE II. PERFORMANCE COMPARISON BETWEEN L-RSR, RANK_LSTM, RSR_E, AND RSR_I

NASDAQ			
Model	MSE	MRR	IRR
Rank_LSTM	0.0003797	0.02968	1.0926
RSR_E	**0.0003793**	0.03864	1.9565
RSR_I	0.0003803	0.05051	2.1448
L-RSR	0.0003796	**0.05149**	**2.6855**

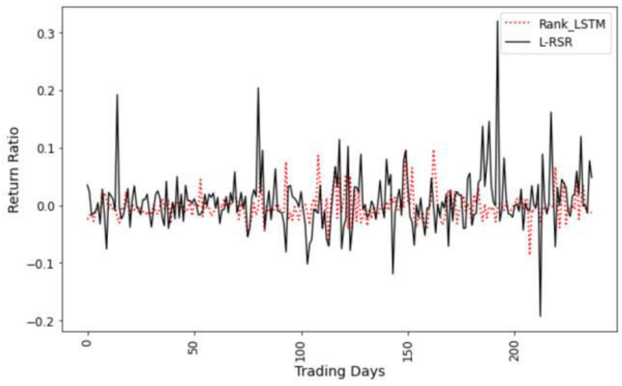

Fig. 2.THE PERFORMANCE COMPARISON OF RANK_LSTM AND L-RSR REGARDING IRR

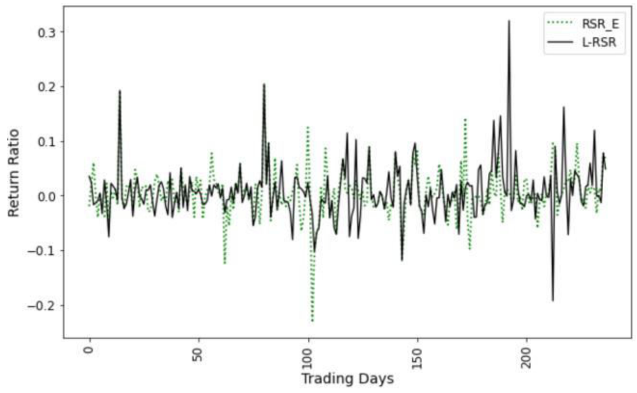

Fig. 3.THE PERFORMANCE COMPARISON OF RSR_E AND L-RSR REGARDING IRR

Fig.2, Fig.3 and Fig.4 are the performance comparison of Rank_LSTM, RSR_E, RSR_I, and L-RSR regarding IRR. From the figures, we can see that L-RSR has a better performance in most of days than the other models. In addition, we also find that L-RSR has a sharp drop on the 213th days with a return ratio of -0.19. However, this also happens in RSR_E on the 103th day. Since IRR is related to the ranking of stocks, it is reasonable that a minor error in the prediction of ranking scores can change the ranking and lead to a huge change in IRR.

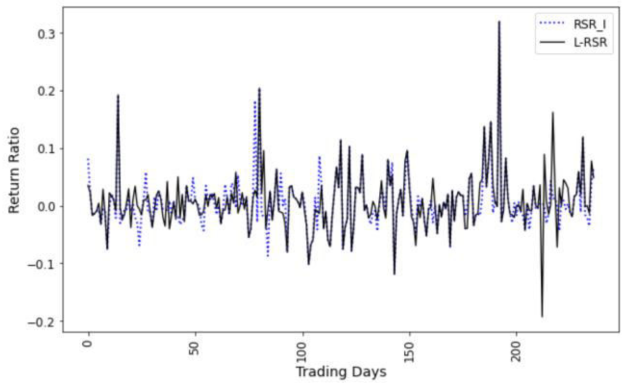

Fig. 4.THE PERFORMANCE COMPARISON OF RSR_I AND L-RSR REGARDING IRR.

V. CONCLUSION

In this paper, we propose a novel stock prediction model called L-RSR, which is an improved vision of the state-of-art RSR model. L-RSR has an excellent performance in stock selection and offers stocks with high rates of return for investors. We conduct a series of experiments with the data from NASDAQ market. The results show that L-RSR achieves better performance than Rank_LSTM model and RSR model. Further research is to merge other data, for example, financial news, national policies and international events into the predictive model. Lastly, the dependency between different stocks is constantly changing, so we will add dynamism in Relation Layer and Graph Layer.

REFERENCES

[1] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, Tie-Yan Liu. Listening to chaotic whispers: a deep learning framework for news-oriented stock trend prediction. WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, US, pp:403–412

[2] Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms." International conference on machine learning. PMLR, 2015.

[3] Li, Q., Chen, Y., Jiang, L. L., Li, P., & Chen, H. A tensor-based information framework for predicting the stock market. ACM Transactions on Information Systems (TOIS), 2016, 34(2): 1-30.

[4] Tu, W., Cheung, D. W., Mamoulis, N., Yang, M., & Lu, Z. Investment recommendation using investor opinions in social media. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, July, pp. 881-884.

[5] Zhang, Liheng, Charu Aggarwal, and Guo-Jun 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. 2017.

[6] Bao, W., Yue, J., & Rao, Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 2017, 12(7), e0180944.

[7] Feng, F., He, X., Wang, X., Luo, C., Liu, Y., & Chua, T. S. Temporal relational ranking for stock prediction. ACM Transactions on Information Systems (TOIS), 2019, 37(2), 1-30.

[8] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. Lightgcn: simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, July, pp. 639-648.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory", Neural computation, 1997, vol. 9, no. 8, pp. 1735–1780.

[10] Lu, W., Li, J., Wang, J., & Qin, L. A CNN-BiLSTM-AM method for stock price prediction. Neural Computing and Applications, 2021, 33(10), 4741-4753.

[11] Zhang, Y., Xiong, Y., Kong, X., & Zhu, Y. Learning node embeddings in interaction graphs. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, November, pp. 397-406.

[12] Zhao, S., Wang, Q., Massung, S., Qin, B., Liu, T., Wang, B., & Zhai, C. Constructing and embedding abstract event causality networks from text snippets. In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, February, pp. 335-344.

[13] Vrandečić D, Krötzsch M. Wikidata: a free collaborative knowledgebase. Communications of the ACM, 2014, 57(10): 78-85.

[14] Yan, R., Song, Y., & Wu, H. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, July, pp. 55-64.

[15] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078, 2014.

[16] Li, W., Bao, R., Harimoto, K., Chen, D., Xu, J., & Su, Q. 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, 2021, January, pp. 4541-4547.