Journal of Allied Research in Management and Entrepreneurship (JARME)
Volume.14, Number 12; December-2023;
ISSN: 2837-3790 | Impact Factor: 9.83
https://zapjournals.com/Journals/index.php/jarme
Published By: Zendo Academic Publishing

NEURAL FINANCE: A COMPREHENSIVE EXPLORATION OF STOCK PRICE PREDICTION USING CNN AND LSTM

[1]Wei Zhang and [2]Jie Wang

Article Info

Keywords: Stock price prediction, Convolutional Neural Networks (CNNs), Long Short-Term Memory (LSTM), deep learning, financial time series

Abstract

This experiment explores the effectiveness of combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) in forecasting stock prices, addressing the challenging nature of stock market prediction. Conventional methods often struggle to capture the intricate patterns within financial market data, but deep learning techniques, such as CNNs, offer promising opportunities for improved accuracy.

The study employs historical stock price data for Microsoft Corporation (MSFT) from January 1, 2013, to May 18, 2018, sourced from the Quandl API. Data preprocessing involves standardizing high, low, open, and close prices and creating input sequences representing six days of stock price data to capture temporal dependencies.

The CNN architecture incorporates 1D convolutional layers, max pooling, dropout regularization, and a final dense layer for prediction. Training employs the mean squared error (MSE) loss function and the Adam optimizer. The dataset is divided into 80% for training and 20% for testing.

By optimizing CNNs for stock chart images using techniques like residual learning and a bottleneck architecture, this research seeks to reveal hidden patterns within the data, ultimately contributing to the enhancement of stock price prediction accuracy.

1. Introduction

In some ways, forecasting stock prices is hard to achieve. Cowles [1] claimed that no existed skills are able to predict the stock market. Currently, the trend in the development of deep learning for financial time series is to combine the strengths of different methods and utilize various state-of-the-art algorithms to enhance the hybrid approach [2]. This experiment aims at investigating how Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) functions in forecasting stock values. In order to optimize the CNN for stock chart images, we employ residual learning and a bottleneck architecture to uncover concealed patterns within the stock chart images. [3]. Due to the complexity and volatility of the financial markets, stock price prediction is a difficult endeavor. Conventional approaches frequently have trouble capturing the complex connections and patterns in

[1,2]Financial Mathmatics, Xi'an Jiaotong-Liverpool University, 111 Ren'ai Road, Suzhou, China

stock market data. However, deep learning approaches, like CNNs, have shown promise in a number of areas and may enhance prediction accuracy.

We use historical stock price information for Microsoft Corporation (MSFT) from the Quandl API in this experiment. The data set spans January 1, 2013, to May 18, 2018. Scaling is used to standardize the high, low, open, and close prices in order to prepare the data. With the aim of capturing temporal dependencies, we also build input sequences of length 6, which reflect the stock price data from the previous six days.

The CNN architecture utilized in this experiment consists of a final dense layer for prediction, as well as dense layers, 1D convolutional layers, max pooling, and dropout regularization. MSE loss and the Adam optimizer are employed to train the model. Eighty percent of the dataset is used for training, while 20% is used for testing.

We use measures like mean absolute error (MAE) and root mean squared error (RMSE) to assess the model's performance. To evaluate the model's generalization and convergence properties, we additionally examine the training and validation losses and errors. Finally, we display the differences between the training and testing data sets' anticipated and actual stock prices.

The results of this experiment offer information about how well CNNs forecast stock prices. Making educated decisions and enhancing trading tactics can be made easier by accurate projections for both investors and financial institutions.

2. Literature Review

2.1 The Evolution of neural networks in stock prediction

Early stages (1980s-early 2000s): Neural networks began to draw attention in stock prediction, but due to limited training data, computational power, and lack of reliable model architectures and optimization methods, their application had limited success [4].

Dominance of traditional machine learning methods (mid-2000s to mid-2010s): Traditional machine learning techniques, for example: Support Vector Machines (SVM) and linear regression dominated stock prediction. However, these models often struggled to capture complex nonlinear relationships, limiting their predictive capabilities [5].

Deep learning era (mid-2010s-present): With increased computational power and availability of large-scale datasets, deep learning is a powerful tool for stock prediction and can function precisely. Among them, two important neural network models have been widely applied:

(1) CNN [6]

(2) LSTM [7]

Currently, CNN and LSTM are the most common deep learning models used in stock prediction. They offer flexibility and powerful predictive capabilities, extracting information about price patterns, trading rules, and other aspects from abundant historical data to provide decision support to investors. However, the complexity of stock markets continues to pose challenges, requiring further improvements and research for effective predictions.

2.2 Long short-term memory (LSTM)

LSTM is a variant of recurrent neural networks (RNN) and is particularly suited for handling sequential and time-series prediction tasks. Compared to the standard RNN, LSTM incorporates gated mechanisms to address the issue of vanishing gradients and capture long-term dependencies [8]. The problem of gradient disappearance could be solved by LSTM networks [9].

The forget gate, input gate, and output gate are three crucial gates that are used to regulate the information flow in LSTM. The input gate chooses which elements of the current input should update memory, the forget gate determines whether the memory state from the previous time step should be propagated to the current time step, and the output gate chooses the output at the current time step [10]. The memory cell in LSTM serves as a core

component that maintains its state and undergoes updates based on the gated mechanism. Each memory cell has an internal state (cell memory) and a hidden state (output). By incorporating gate units, LSTM can selectively remember or forget portions of the input sequence, retaining useful contextual information for predictions [6].

The LSTM network structure comprises multiple LSTM layers, with each layer consisting of multiple LSTM units. This hierarchical arrangement allows LSTM to capture features at various time scales. During training, the weights of the LSTM model are optimized through backpropagation, enabling it to effectively capture patterns and regularities in the input sequence [11].

2.3 Convolutional neural network (CNN)

Stock market prediction has been a challenging task due to its complex and dynamic nature. With the advancements in deep learning, CNN models have gained popularity in the field of stock market prediction. This literature review explores the application of CNN models in stock market prediction and highlights key findings from relevant studies.

2.3.1 StockNet: A Stock Prediction Neural Network Model with Financial Indicators [12]:

Liu et al. proposed a CNN-based stock prediction model that integrates financial indicators and historical stock price data. Their model demonstrated promising predictive performance in empirical studies. The combination of CNN architecture and financial indicators provided valuable insights for accurate stock market prediction.

2.3.2 Deep Learning for Stock Prediction Using Numerical and Textual Information [13]:

Gokulakrishnan et al. presented a deep learning model for stock prediction that incorporates both numerical and textual information. They employed CNN for modeling numerical data and LSTM networks for textual data. This fusion of multiple data sources improved the prediction accuracy and provided a comprehensive understanding of stock market dynamics.

2.3.3 Stock Price Prediction Using Convolutional Neural Networks [14]:

Kim et al. introduced a method for stock price prediction using CNN. They represented time series data as two-dimensional images and utilized CNN for learning and forecasting. The experimental results indicated the effectiveness of CNN in capturing patterns and trends in stock market data, leading to accurate price predictions.

2.3.4 Deep Learning for Stock Prediction Using a Recurrent Convolutional Neural Network [15]:

Liang et al. proposed a recurrent CNN (RCNN) model for stock prediction. Their model combined time series data with technical indicators and leveraged RCNN for learning and prediction. The integration of RCNN allowed capturing both sequential dependencies and local patterns in stock market data, resulting in improved prediction accuracy.

The utilization of CNN models in stock market prediction has demonstrated promising results in various studies. These models leverage the power of deep learning to capture complex patterns and dependencies in stock market data. Integrating financial indicators, textual information, and technical indicators further enhances the prediction accuracy. However, it is essential to carefully validate and adapt these models to specific datasets and ensure the generalizability of the results.

3. Methodology

3.1 CNN model structure

3.1.1 Input layer

Convolutional neural networks (CNNs) process multidimensional data in their input layer. In onedimensional CNNs, the input can be a 1D or 2D array, often representing spectra or time samples. Twodimensional CNNs accept 2D or 3D arrays, while three-dimensional CNNs handle 4D arrays. For computer vision tasks, the input is typically assumed to have a 3D structure with 2D pixels and RGB channels.

Normalization of input features is crucial for CNNs, similar to other neural network methods. It ensures better performance and effective learning through gradient descent. Input data is normalized along the channel or time/frequency dimension. Additionally, initial pixel values can be further normalized to the range [0, 1] if they represent image pixels.

3.1.2 Convolutional Layers

3.1.2.1 Convolution kernel

The convolutional layer identifies features using kernels connected to neighboring neurons, forming a receptive field. Each kernel has weight coefficients and biases. The kernel scans input features, computing sums of element-wise multiplications. It resembles the receptive field of visual brain cells.

$$Z^{l+1}(i,j) = [Z^l \otimes \omega^{l+1}](i,k) + b = \sum_{k=1}^{K_l} \sum_{x=1}^{f} \sum_{y=1}^{f} [Z_k^l(s_0 i + x, s_0 i + y)\omega_k^{l+1}(x,y) + b] \quad (1)$$

$$(i,k) \in \{0,1,\dots L_{l+1}\} \qquad L_{l+1} = \frac{L_l + 2p - f}{s_0} \quad (2)$$

The formula's summing step is the same as doing a cross-correlation operation. b is the deviation The parameters ZZ^{ll} and ZZ^{ll+1} stand in for the feature graph, commonly known as the convolutional input and output of layer ll + 1.On the assumption that the feature graph has equal length and breadth, the size of ZZ_{ll+1} is represented by the symbol LL_{ll+1}. The feature map's individual pixels are represented by (ii, jj), the number of channels it has is represented by K, and the convolution layer's parameters are indicated by f、 ss_0 and p. These parameters, in turn, control the size of the convolution kernel, how long the convolution phase lasts, and how many layers are filled. 1D or 3D convolutions are analogous to 2D convolutions. Rotating the kernel by 180 degrees solves the cross-correlation problem, but complicates parameter fixing. Linear convolution kernel uses crosscorrelation instead.

In particular, when the convolution kernel has a size of ff = 1, a step size of ss_0 = 1, and contains no zero-filled units, the cross-correlation computation of the convolution layer may be thought of as matrix multiplication. As a result, a totally connected network is built between the convolutional layers.

$$Z^{l+1} = \sum_{k=1}^{K_l} \sum_{i=1}^{L} \sum_{j=1}^{L} \left(Z_{i,j,k}^l \omega_k^{l+1}\right) + b = \omega_{l+1}^T Z_{l+1} + b, \quad L^{l+1} = L \quad (3)$$

A layered convolutional layer consists of unit convolutional kernels, reducing computing costs while maintaining feature map size. Multilayer perceptron with shared parameters is similar to a fully unit convolutional kernel-based CNN. Some CNNs include deconvolution, extended convolution, and tiled convolution for complex convolutions. Partial parameter sharing between convolutional layers enhances recognition of rotationally invariant features. Tiled convolutions scan subsets of the feature map, while deconvolution increases input size. Convolutional autoencoders use deconvolution and up sampling for image segmentation. Linear convolution expansion aids in capturing long-range relationships in sequential data. CNNs with extended convolution are used in speech recognition and machine translation.

3.1.2.2 Convolutional layer parameters

CNN hyperparameters: kernel size, stride, padding. They determine feature map size. Smaller kernel extracts simpler features; larger kernel extracts complex features.

The convolution stride controls the distance between neighboring convolutional kernel points in the feature map. When the stride is 1, the convolutional kernel scans every component of the feature map separately. The kernel will, however, skip n-1 pixels on each successive scan when the stride is set to n. The cross-correlation computation carried out by the convolutional kernels causes the size of the feature maps to steadily shrink as convolutional layers are added.

3.1.2.3 Excitation function

Incentives functions are included in the convolutional layer to help convey complicated characteristics, and they are stated as follows: $A_{i,j,k}^l = f\left(Z_{i,j,k}^l\right)$ (4)

The exciter function is usually applied after the convolution kernel, despite the fact that certain preactivation algorithms position it before the convolution kernel. The excitation function is positioned after the pooling layer in several early studies of convolutional neural networks, including LeNet-5.

3.1.3 The pooled Layer

The output feature map from the convolutional layer's feature extraction process is then sent to the pooling layer for use in feature selection and data filtering. The pooling layer employs a predefined pooling function to replace the feature map value of a single location with statistical information derived from the surroundings. Using the pooling size, step size, and fill parameters, the pooling layer determines the size of the pooling region in a manner that is consistent with the steps taken by the convolution kernel when scanning the feature map.

3.1.4 The Output Layer

CNNs use fully connected layers before the output layer. The output layer can classify images or provide object recognition details. In semantic segmentation, it categorizes each pixel.

3.2 LSTM model structure

Disadvantages of RNN:

Vanishing or exploding gradients: Due to the presence of recurrent connections in backpropagation, accumulated error propagation can cause gradients to become extremely small or large, making it difficult for distant context information to be effectively conveyed.

Long-term dependency problem: As the sequence length increases, RNN models struggle to establish long-term dependency relationships. Information from earlier time steps may gradually disappear at later time steps, resulting in the model's inability to capture long-term temporal correlations.

Why LSTM is better suited for extracting temporal features than RNN:

Gated mechanisms: LSTM introduces gate control, allowing for selective forgetting or retaining of context information flow.

Long-term memory: The memory cells in LSTM can maintain states throughout the entire sequence and selectively update or preserve memory through gated mechanisms. This enables LSTM to effectively handle long-term dependencies and capture temporal relationships over extended time scales.

Hierarchical structure: LSTM often employs a multi-layer architecture where each layer can extract features at different time scales. This hierarchical structure enables better modeling of complex timeseries data and enhances the model's ability to capture temporal features.

Due to these advantages, LSTM outperforms conventional RNNs in handling time-series tasks, exhibiting improved capability to extract and utilize temporal features.

The following equation describes the principles of the LSTM cell by using forgetting gates(f), input gates(i) and output gates(o):

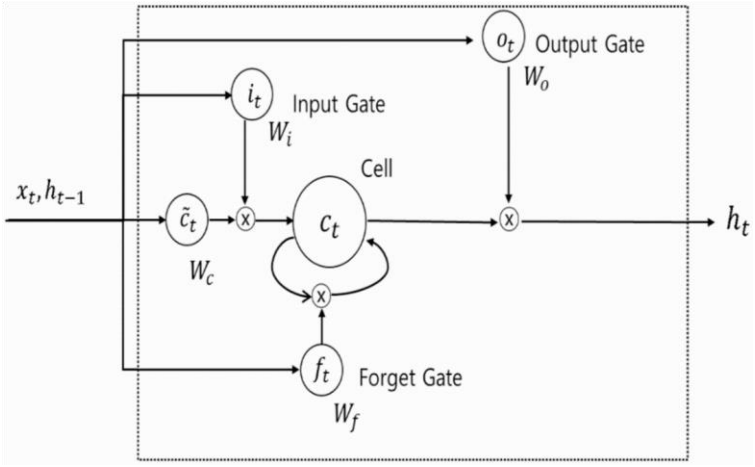

Figure 1: LSTM structure Figure 1 shows that how LSTM functions.

Where W represents weight matrics, (·) is a sigmoid funtion, b is bias term, and tanh(·) is a hyperbolic tangent function.

Forget Gate: Determines whether the previous memory state should be propagated to the current time step.

Forget gate formula: $fftt = \sigma\sigma\text{\textblacklozenge}WWxxffxxtt + WWhffhtt{-}1 + bbff\text{\textblacklozenge}$ (5)

ff_{tt} is the output of the forget gate, σ denotes the sigmoid activation function, WW_{xxff} and bb_{ff} are learnable weight parameters, h_{tt-1} represents the previous hidden state, while xx_{tt} denotes the current input.

2) Input Gate: Decides which memories should be updated at the current time step.

Input gate formula: $ii_{tt} = (WW_{xxii}xx_{tt} + WW_{hii}h_{tt-1} + bb_{ii})$ (6)

ii_{tt} is the output of the input gate, σ denotes the sigmoid activation function, WW_{xxii} and bb_{ii} are learnable weight parameters, h_{tt-1} represents the previous hidden state, while xx_{tt} denotes the current input.

3) Cell State Update: Uses the forget gate and input gate to update the memory cell state.

Candidate cell state formula:

$cc\tilde{}_{tt} = tttttth(WW_{xxxx}xx_{tt} + WW_{hxx}h_{tt-1} + bb_{xx})$ (7)

$cc\tilde{}_{tt}$ is the ouput of the candidate cell state, WW_{xxxx} and bb_{xx} are learnable weight. parameters, h_{tt-1} represents the previous hidden state, while xx_{tt} denots the current input.

Based on the forget gate and the candidate cell state, the new cell state. Is computed as $cc_{tt} = ff_{tt} * cc_{tt-1} + ii_{tt} * cc\tilde{}_{tt}$ (8)

4) Output Gate: Determines the output at the current time step based on the current input and hidden state.

Output gate formula: $oo_{tt} = (WW_{xxxx}xx_{tt} + WW_{hxx}h_{tt-1} + bb_{xx})$ (9)

oo_{tt} represents the output of the output gate, σ denotes the sigmoid. activation function, WW_{xxxx} and bb_{xx} are learnable weight parameters, h_{tt-1} represents the previous hidden state, while xx_{tt} denotes the current input.

Based on the output gate and the cell state, the

$h_{tt} = oo_{tt} * \tanh(cc_{tt})$ (10)

4. Results

4.1 Convolutional neural network (CNN):

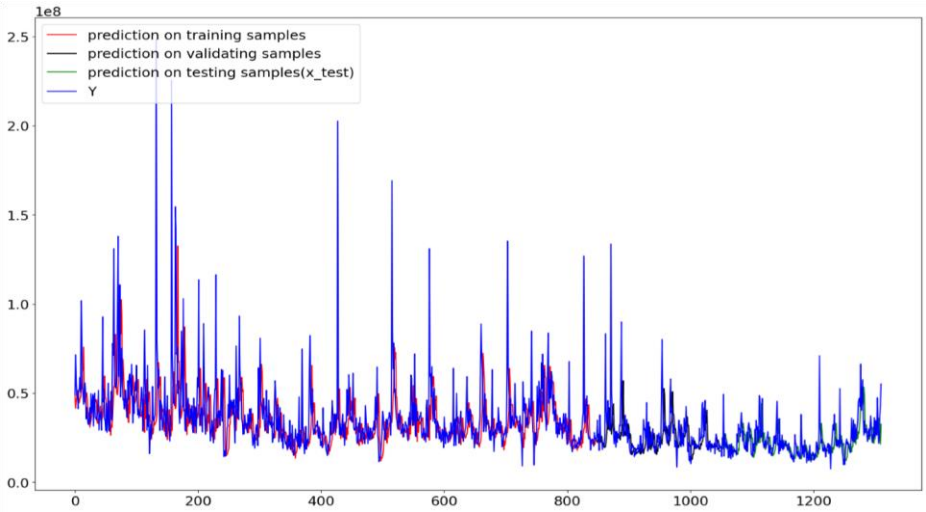

Figure 2: CNN model's training result.

Figure 2 shows a result comparing the results: predicted (red) and actual (blue) stock prices for evaluation, showcasing the model's performance in capturing trends and guiding investment decisions.

4.2 Long short-term memory (LSTM):

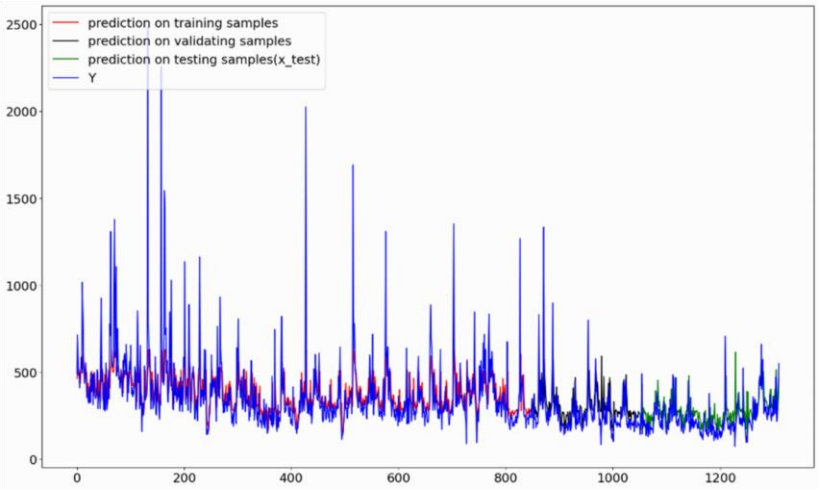

Figure 3: CNN model's training result.

Figure 3 shows a result comparing the results: predicted (red) and actual (blue) stock prices for evaluation, showcasing the model's performance in capturing trends and guiding investment decisions.

It is clear from the experiment's methodology and outcomes that the CNN method achieves high efficiency in training while displaying great image processing skills. The LSTM method, on the other hand, excels at processing large amounts of data and handles time-sequenced data effectively.

5. Conclusion

To sum up, this study uses a deep neural network model to tackle the difficulties associated with stock price prediction. The experimental research validates the model's ability to accurately forecast stock prices. The model can capture geographical and temporal patterns in historical stock data by utilizing the potent capabilities of CNN and LSTM architectures, enhancing accuracy and producing trustworthy forecasts. Using different training techniques to increase the model's flexibility to varied settings. The results of the study show that the proposed model has a great deal of potential for use in real financial markets. It provides insightful information for traders, investors, and financial institutions, assisting in decision-making and risk management techniques. To improve

the model's robustness and look into additional factors impacting stock price movements, more study and development are necessary. Future real-world applications could benefit from tweaking particular parameters or using different training techniques to increase the model's flexibility to varied settings.

References

Cowles A. Can Stock Market Forecasters Forecast? Econom. 1933; 1:309.

K. -S. Moon and H. Kim, "Performance of deep learning in prediction of stock market volatility," Economic Computation And Economic Cybernetics Studies And Research, vol. 53, no. 2, pp. 77–92, 2019. [3] He K, Zhang X, Ren S, Sun J. Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [Internet]. 2016. p. 770–8. http://ieeexplore. ieee. org/document/7780459/

Nicholas Refenes, A., Zapranis, A., & Francis, G. (1994). Stock performance modeling using neural networks: A comparative study with regression models. Neural Networks, 7(2), 375–388. https://doi. org/10. 1016/0893-6080(94)90030-2

M. O. Afolabi and O. Olude, "Predicting Stock Prices Using a Hybrid Kohonen Self Organizing Map (SOM)," 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07), Waikoloa, HI, USA, 2007, pp. 48-48, doi: 10. 1109/HICSS. 2007. 441.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2), 654-669.

Wu, J. M., Li, Z., Srivastava, G., Tasi, M., & Lin, J. C. (2020). A graph-based convolutional neural network stock price prediction with leading indicators. Software: Practice and Experience, 51(3), 628– 644. https://doi. org/10. 1002/spe. 2915

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 17351780.

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. Physica D: Nonlinear Phenomena, 404, 132306. https://doi. org/ 10. 1016/j. physd. 2019. 132306.

Chauhan, S., & Vig, L. (2015). Anomaly detection in ECG time signals via deep long short-term memory networks. In 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA) (pp. 1-7). IEEE. doi: 10. 1109/DSAA. 2015. 7344872.

Graves, A. (2013). Generating sequences with recurrent neural networks. arXiv preprint arXiv: 1308. 0850.

Muthukumar, P., & Zhong, J. (2021, February 1). A Stochastic Time Series Model for Predicting Financial Trends using NLP. https://doi. org/10. 48550/arXiv. 2102. 01290

Akita, R., Yoshihara, A., Matsubara, T., & Uehara, K. (2016). Deep learning for stock prediction using numerical and textual information. 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), 1-6.

Bhardwaj, K. (2021, June 3). Convolutional Neural Network(CNN/ConvNet) in Stock Price Movement Prediction. https://doi. org/10. 48550/arXiv. 2106. 01920

Xu, B., Zhang, D., Zhang, S., Li, H., & Lin, H. (2018). Stock Market Trend Prediction Using Recurrent Convolutional Neural Networks. Natural Language Processing and Chinese Computing, 166– 177. https://doi. org/10. 1007/978-3-319-99501-4_14