Transformer for stock price forecasting

Ying Yu, Kaihua Yu, Yuting Wen, Bingyang Gao, Yani Wei, Jinliu Yu

Abstract

Time series forecasting is a chanllenge task due to complex mix of inputs. The stock price forecasting difficulties mainly lie in the uncertainty and noise of the sample. Several deep learning methods have been proposed to learn the temporal dependence by variants of CNNs and RNNs. However, CNNs equipped with local receptive field, unable to capture context variables, and RNNs are prone to severe gradient disappearance or gradient explosion problems. Recently, Transformer encoder uses PositionEncoding function for capturing the relative position relationship of input variables and the Self-Attention mechanism , which allows the model to parallelize training andcapture global information. In this context, we use Transformer encoder to explicitly forecast the closing stock price. Instead of adopting the Encoder-Decoder architecture in the original paper , the Decoder is replaced by a full-connection layer to output the predicted value. In addition,we propose mask layer to avoid introducing future information. We evaluated our method on two datasets that Pfizer and AstraZeneca stocks prices were taken from the Yahoo Finance site. As a result , the proposed framework performance overcomes the methods in the state of the art on all experiments.

1 Introduction

Since the birth of the stock market , the main goal of investors is to accurately and reliably predict stock prices. Millions of dollars of trading occur every day, and every trader wants to profit from his or her investment. Investors who make the right buying and selling decisions will profit. Investors must take further measures to make a fair choice based on technical analysis, such as company charts, stock market ratios, newspaper and Weibo statistics. Despite many efforts in the last years, time series forecasting is still a chanllenge task[1]. Accurately predicting the stock market is a complex task because there are millions of situations that can affect it. Therefore, we need to be able to capture as many of these preconditions as possible. The stock market plays a very important role in today's financial market. In recent years, the stock market has attracted more and more attention. The prediction of stock price fluctuations has become more attractive. Many scholars in different fields have also

begun to pay attention to this field. With the development of computer science and technology, methods based on data mining and machine learning are also used to automatically predict the volatility of stock prices.For a prediction model, accurate prediction accuracy is the primary condition. Higher prediction accuracy can help market decision-makers make better judgments. In the past few years, some methods based on data mining and machine learning, such as neural networks and support vector machines, have been widely used in classification and regression problems. Neural networks and support vector machines have good performance in classification and regression problems[2]. In order to further improve the longitude of stock price prediction, some improved algorithms and learning strategies are applied to the problem of stock price prediction[1]. It has been pointed out that using multiple market data sources can achieve better prediction accuracy than using only a single data source. In recent years, some researchers have made efforts to improve the performance of stock price prediction by adding other useful data to the historical price data. Some researchers pointed out that market news can help improve the accuracy of prediction, and put forward the method of combining the historical stock price and market news to predict the stock price, while considering the market news and historical price to improve the accuracy of prediction.

Convolutional Neural Networks (CNN) is a deep feedforward neural network that focuses on convolution and pooling operations[3]. The principle of time series prediction using convolutional neural networks is to use the ability of convolutional kernels to sense the changes in historical data over a period of time and make predictions based on the changes in this historical data. Pooling operations can preserve key information and reduce information redundancy. Convolutional neural networks can effectively reduce the human resource consumption of previous algorithms for feature extraction, while avoiding the generation of human errors[4]. However, convolution is only a sliding window extraction for a small area, this can lead to limited receptive fields,and it can occur that features of the stock cannot actually be extracted simultaneously by the same convolution kernel, which means that the locality of feature extraction is also a typical "long-range dependency". Convolutional neural networks are unable to correlate information on time series, such as the inability to establish a relationship between the previous and subsequent stock data, i.e., they have no memory function, which leads to poor performance in stock price prediction tasks. Since the training parameter "convolution kernel" is a feature that we want to extract, this means that we need a large number of data sets to train these convolution kernels. There are different scales for stock data, and we need to normalize them to the same scale for training. Recurrent Neural Networks (RNN) is a deep learning model proposed by M. I. Jordan in 1990 for learning temporal dimensional features. The units of the RNN are connected together in a long chain and recursively follow the direction of sequence development. The input to the model is sequence data, which can be used to process various tasks of natural language processing (such as text emotion classification, machine translation, etc.), it can demonstrate strong learning ability when processing time series data and voice data, recognizing the sequential characteristics of the data and using previous patterns to predict[5]. RNNs neural network algorithms have been an important method for solving time series prediction tasks since they were proposed.

They have been widely used as the main model for solving time series data prediction problems until 2017. Conversely, RNN does not handle long sequences well.Tthe main reasons is that RNN is prone to gradient explosion and gradient disappearance during training , which results in gradients not being continuously transmitted over long sequences during training, making RNN unable to capture the long-term impact of stock prices.

Transformer algorithms are now widely used in various tasks in the field of artificial intelligence[6]. Building models based on Transformer can break the ability bottleneck of previous algorithms and simultaneously provide good capture performance . The ability to capture short-term and long-term dependencies , effectively solving long series prediction problems , and can be processed in parallel.

The self attention mechanism, which focuses on specific parts of the human eye, is inspired by the broad perspective of the human eye, but is limited to visual resources. It focuses on the more valuable parts of the data[7][8]. The self attention mechanism adopted by Transformer solves the problem that the input of neural networks is many vectors of different sizes , and vectors at different times often have some potential connection.When practicing , the potential connection between inputs cannot be fully captured , resulting in poor model training results . A self attention module receives n inputs and then returns n outputs , all of which interact with each other , mining out the attention points that have significant effects . The aggregation and attention score of these interactions are the outputs given by the module.

2 Framework

The transformer model is entirely based on the attention mechanism and does not have any convolutional layers or recurrent nerves layer . The transformer is composed of an encoder and a decoder[6]. The encoder and decoder of the transformer are based on the superposition of modules that require attention . The embedding table of the input sequence and output sequence will be added with position encoding , and then input to the encoder and decoder respectively . The encoder of the transformer is composed of multiple identical layers stacked together , with each layer having two sub layers (referred to as the sub layer). The second layer is the aggregation of multi head self attention ; The second layer is a position wise feed forward network . Specifically , when calculating the attention of the encoder, queries , keys , and values all come from the output of the previous encoder layer . Each layer has adopted residual connections. In the transformer , for any input $x \in R^d$ at any position in the sequence . All require full $sublayer(x) \in R^d$, so that the residual is connected to full $x+$ sublayer $(x) \in R^d$. After the addition calculation of residual connections , it is immediately followed by layer normalization . Therefore , for each position corresponding to the input sequence , the transformer encoder will output a d-dimensional table vector . The position based feedforward network makes the same multi-layer perceptron (MLP) when transforming the table of all positions in the sequence, which is why the feedforward network is position wise. The Add&Norm component is composed of residual connections and subsequent layer normalization. Both are key to building an effective deep architecture.

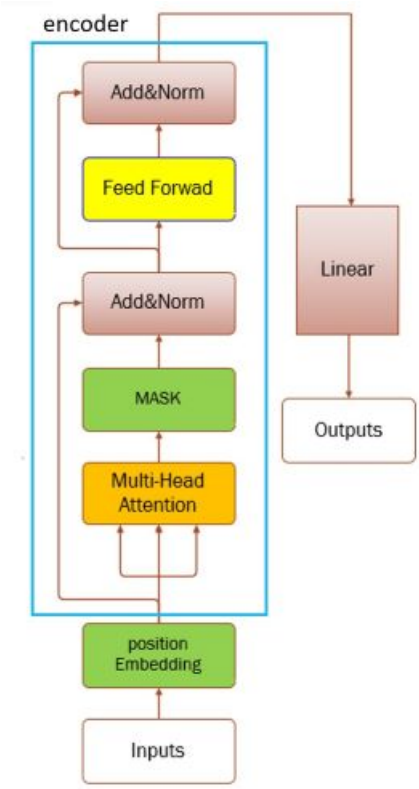

Fig. 1 Transformer

We replace the previous Decoder with a fully connected linear layer

2.1 Positional embedding

Because Transformer does not use the structure of RNN, but uses global information, it cannot utilize the sequential information of time series, which is very important. So in Transformer, position embedding is used to store the relative or absolute position of words in the sequence[8]. In order to use the sequential information of time series, we annotate absolute or relative positional information by adding positional encoding to the input table. The position encoding can be obtained through learning or directly fixed. We use fixed position encoding based on sine and cosine functions. Position encoding embeds a matrix of positions with the same shape. The elements on the i row, 2j column , and 2j+1 column of the matrix are:

$$p_{i,2j} = sin(\frac{i}{10000^{2j/d}}) \tag{1}$$

$$p_{i,2j+1} = cos(\frac{i}{10000^{2j/d}}) \tag{2}$$

2.2 Multi Head Attention

When calculating, matrices Q (query), K (key), and V (value) need to be used. In practice, Self Attention receives input or output from the previous Encoder block. And Q, K, and V are obtained through linear transformation through the input of Self Attention.If the input of Self Attention is represented by matrix X, then Q, K, and V can be calculated using the linear variable matrix W_Q, W_K,and W_V. After obtaining the matrices Q, K, and V, the output of Self Attention can be calculated using the following formula:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d^k}})V \tag{3}$$

We hope that the model can learn different behaviors based on the same attention mechanism, and then combine the different behaviors as knowledge to capture the temporal dependencies within the sequence. Therefore, the multi head attention is beneficial. Query $q \in R^{d_q}$, key $k \in R^{d_k}$ and Value$v \in R^{d_v}$. The calculation method of each attention head h_i(i=1,..., h) is as follows:

$$h_i = f(W_i^q q + W_i^k k + W_i^v v) \in R^{p_v} \tag{4}$$

Among them, the learnable parameters include:$W_i^q \in R^{p_q \times d_q}$,$W_i^k \in R^{p_k \times d_k}$,$W_i^v \in R^{p_v \times d_v}$as well as the function f representing the convergence of attention *f*. The output of multi head attention needs to undergo another linear transformation, which corresponds to the result of h head connections. Therefore, its learnable parameters are$W_o \in R^{p_o \times hp_v}$:

$$W_o H \in R^{p_o} \tag{5}$$

H=$h_1, h_2 \ldots h_h$. Each header may focus on different parts of the input, which can be represented as a simpler weighted average with more complex functions.

2.3 Mask

Mask is the operation of X and M to eliminate the effects of padding. If we directly perform softmax on the vector after padding, the padding part will also share a portion of the probability, resulting in the sum of the actual meaningful probabilities not being equal to 1. The solution is the same as for max, making the padding part small enough to make e * * x close enough to 0, so that it can be ignored.Mask processing, basically only requiring output:$X \bigotimes M$. That's it, which means keeping the padding part at 0.A mask vector (matrix) is a 0/1 vector (matrix), where 1 represents the meaningful part and 0 represents the meaningless padding part.A mask vector : M is a 0/1 vector (matrix), where 1 represents the meaningful part and 0 represents the meaningless padding part.

2.4 Add & norm

The Add & Norm layer consists of two parts: Add and Norm, and its calculation formula is as follows:

$$Y = LayerNorm(X + MultiHeadAttention(X)) \tag{6}$$

$$Y = LayerNorm(X + FeedFword(X)) \tag{7}$$

Among them, X represents the input of Multi Head Attention or Feed Forward, while Multi Head Attention (X) and Feed Forward (X) represent the output (the output and input X dimensions are the same, so they can be added). Add refers to X+MultiHeadAttention (X), which is a residual connection commonly used to solve the problem of multi-layer network training, allowing the network to only focus on the current differences. Norm refers to Layer Normalization, usually used in RNN structures. Layer Normalization converts the inputs of each layer of neurons to have the same mean and variance, which can accelerate convergence.

2.5 FeedForward

The Feed Forward layer is simple. It is a two-layer full connection layer. The activation function of the first layer is Relu, and the activation function of the second layer is not used. The corresponding formula is as follows:

$$Y = max(0, XW_1 + b_1)W_2 + b_2 \tag{8}$$

X is the input, and the dimension of the output matrix obtained by Feed Forward is consistent with X.

We replace the previous Decoder with a fully connected linear layer for output. Our model is shown in Figure 1.

3 Experiments

3.1 Data preprocessing

We evaluated the accuracy of our method on two publicly available datasets. The Pfizer and AstraZeneca stocks prices were taken from the Yahoo Finance site. The data also includes stock trading volume, opening price, closing price, adjacent closing price, and high/low. Before training the transformer model, the data is divided into a training dataset and a testing dataset. The training dataset will be used to fit the transformer model, and accuracy will be tested on the test data afterwards. Accuracy and backpropagation testing will be conducted on the test dataset. We will scale the stock value to a value between 0 and 1, so that our data calculation costs will be reduced. Scaling data is always beneficial for preprocessed applications, so that our data is not scattered in huge value. Scaling down in proportion always helps to achieve higher accuracy. Because we are using time series prediction, we use a scrolling window with a size of 20 to provide different time increments. We use the stock price from the first 20 days to predict the stock price for the next day. The data in the scrolling

window is the past stock prices. We will advance the scrolling window one day and add the next real stock price to the latest date of the scrolling window to predict the next day's stock price. We follow two approaches: 1. We use the past 14 days' Pfizer

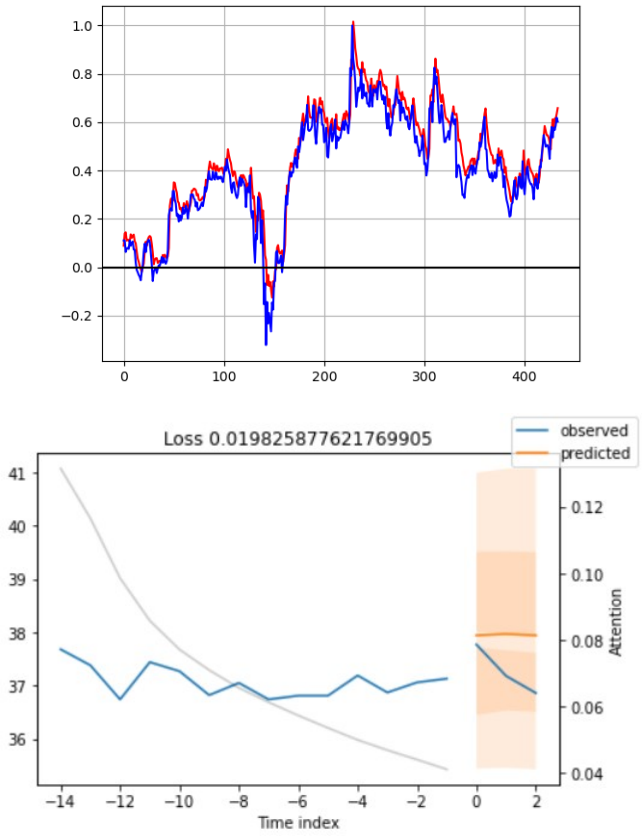

Fig. 2 Fitting diagram, output value and label comparison,the first is the transformer prediction graph, and the second is one of the tft prediction graph

closing prices to forecast the prices of the 3 following days . 2. We enhanced the first model by giving as input also the past 14 days AstraZeneca closing prices to forecast the prices closing of the 3 following days of the Pfizer stock . The dataset is challenging due to some influencing factors.

Table 1 By changing different parameters and comparing their influences on forecast results, transformer is better than tft model. " >" means larger than the original result.The result is mse loss value

Loss of transformer and tft on simple univariate datasets.				
transformer			tft[10]	
	Astra	Pfizer	Astra	Pfizer
original param(lr=0.001)			2.8718	1.0863
lr=0.005	0.0687	0.00751		
lr=0.01	0.09734	0.00918		
lr=0.001	0.00404	0.00563		
original param(num_layer=1)				
num_layer=3	>	>		
num_layer=2	>	>		
original param(feedward=49)				
feedward=60	0.0044	0.00607		
feedward=4	0.00422	0.00582		
original param(nhead=10)				
nhead=25	0.00377	0.00597		
nhead=5	0.00386	0.00601		
original param(feature_size=250)				
feature_size=100	0.00642	0.00652		
feature_size=300	0.00365	0.00647		

3.2 Loss function

The loss function we use is the mean square error loss function[9], whose basic form is:

$$J_{MSE} = \frac{1}{N}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9}$$

3.3 Contrast Ablation

We conducted comparative experiments using tft [10]and transformer.The test data output and label value fitting are shown in Figure 2. TFT uses standardized components to effectively construct feature representations for each input type (i.e. static, known, observed) to achieve high predictive performance on a wide range of problems. The tft requires too much data and there are also too many features input into the model. The results have shown that transformer performs better than tft in predicting stock closing prices. We also set different model architectures by setting different parameters for comparison:

learning rate:0.005,0.001,0.01
num_layers:1, 2, 3
feedback:49, 60, 40
n_head:5, 10, 25
feature_size:100, 250, 300

We use the loss value as the result for comparison and ablation. The experimental results are shown in Table 1. Conduct ablation experiments with or without embedding and mask layers. Experiments show that embedding can better predict the closing price of stocks,Lacking global information.Without embedding , it cannot utilize the

sequential information of time seriesleads to poor prediction results.And no mask layer will introduce future stock data, resulting in overfitting problems and weak generalization ability. The ablation results are shown in table 2.

Table 2 The ablation table of the embedding and mask layers

	Original param(0.001)	**without embedding**	**without mask layer**
Astra	0.00404	0.21743	0.00173
Pfizer	0.00563	0.06591	0.00114

4 Conclusion

In this work, we just use Transformer encoder to explicitly forecast the closing stock price. In order to overcome stock price forecasting difficulties that mainly lie in the uncertainty and noise of the sample. Instead of adopting the Encoder-Decoder architecture in the original paper,the Decoder is replaced by a full-connection layer to output the predicted value. In addition,we propose mask layer to avoid introducing future information.We trained and enhanced our method on two datasets that Pfizer and AstraZeneca stocks prices were taken from the Yahoo Finance site.As a result,the proposed framework performance overcomes the methods in the state of the art on all experiments. The significance of studying stock price forecasting is very significant, with the aim of enabling investors to benefit from stock price forecasting and helping them better allocate investment funds.

References

[1] Chen, J., Wen, Y., Nanehkaran, Y.A., Suzauddola, M., Chen, W., Zhang, D.: Machine learning techniques for stock price prediction and graphic signal recognition. Eng. Appl. Artif. Intell. **121**, 106038 (2023)

[2] WAN C, D.W.X.e.a. LI W Z: A multivariate time series forecasting algorithm based on self-evolutionary pre-training[j]. Chinese Journal of Computer **45**(03), 513–525 (2022)

[3] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning **1**, 326–366 (2016)

[4] Jiuxiang, Wang, Zhenhua, Kuen, Jason, Lianyang, Shahroudy, Amir, Shuai, Bing: Recent advances in convolutional neural networks. Pattern Recognition: The Journal of the Pattern Recognition Society (2018)

[5] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning[m]. MIT press, 363–405 (2016)

[6] Vaswani A, P.N.e.a. Shazeer N: Attention is all you need[j]. advances in neural information processing systems, 30 (2017)

[7] WANG W G, J.Y.D. SHEN J B: Review of visual attention detection. Ruan Jian Xue Bao/Journal of Software **30**(2), 416–439 (2019)

[8] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition. In: Neural Information Processing Systems (2015)

[9] Wang, Z., Bovik, A.C.: Mean squared error : Love it or leave it? IEEE Signal Processing Magazine **26**(1), 98–117 (2009)

[10] Lim B, L.N.e.a. Arık S Ö: Temporal fusion transformers for interpretable multi-horizon time series forecasting[j]. International Journal of Forecasting **37**(4), 1748–1764 (2021)