Project Final Report on

StockCoder: Stock price prediction using Transformer architecture

Submitted By:

Saiman Dahal (11873444)

Aashik Sharif Basheer Ahamed (11870531)

Course: Neural Network Design & Application

Date of submission: 2024/04/21

Abstract:

The stock market can be defined as the market where people can buy or sell their company stocks. Predicting the trends and patterns by proper value of stock can help consumers to gain maximum profitability. We have developed a system that helps to predict the price of stocks. This report provides an overview of the prediction of stock prices utilizing the transformer architecture under particular sector. Transformer architectures are well known for their implementation in LLMs, but the recent trend shows their implementation in the time series analytics and long sequences. The goal is to provide accurate forecasts to aid investors and analysts in making decisions on the stocks.

Because there are so many variables affecting stock prices and the stock market is inherently complex, predicting stock prices has proven to be a difficult undertaking. Conventional methods have frequently depended on technical analysis and statistical techniques, which might not be sufficient to adequately capture the complex patterns and dynamics of the market. But recent developments in deep learning, especially with transformer topologies, have demonstrated encouraging outcomes in capturing temporal dynamics and long-range relationships, which qualifies them for time series forecasting applications like stock price prediction. [4]

Introduction:

Prediction is an important application of machine learning. Nowadays, people interested in prediction using technology have developed to a huge extent. Implementation of deep learning in forecasting is done since long time. The evolution of models is a continuous process. Nowadays, Transformer model is covering big areas in large language models like OpenAI, ChatGPT and Google Bard. But their implementation in time series analysis is increasing. The problem associated with RNN is removed from transformer, which gives the best reason why transformer is efficient in use.

This general architecture is followed by the Transformer, which uses pointwise, connected layers for the encoder and decoder. Self-attention is an attention mechanism relating different positions of a single sequence to compute a representation of the sequence. The input sequence is split and thus the relation is checked in terms of three paraments: Query, Key and Value.

Literature Review:

Various approaches have been done since past for the application of deep learning in forecasting time series data. RNN is known as the best model for the time series analysis implementation. Applications of deep learning in financial market prediction have attracted widespread attention from investors and scholars. The article by Chaojie Wang, Yuanyuan Chen and team have explored the implementation of Transformer to predict the stock market index [1].

The introduction of the attention module in the transformer-based prediction models has created an impressive potential in forecasting. The main background to this implementation is the ability to parallel process long-range sequences. Although many current approaches have shown excellent predictive power, their study focused on the improvements in prediction from spatiotemporal correlation [3].

A comprehensive overview of the evolution of predictive models in the finance domain, particularly focusing on stock price forecasting begins with the transformative impact of Transformers in Natural Language Processing and their subsequent application in financial forecasting, highlighting key works that leverage Transformers for sentiment analysis and price prediction tasks. It also covers traditional methods like ARIMA and established neural network architectures like LSTM, offering insights into their strengths

and limitations. By summarizing various approaches and their respective accuracies, it sets the stage for the evaluation of Transformer models in stock price prediction, emphasizing the significance of this study in addressing the complexity of financial markets [6].

Traditional approaches, such as RNNs, often struggle with capturing long-term dependencies, which Transformers address effectively through self-attention mechanisms. Transformer architectures, specifically in conjunction with sentiment analysis, for stock trend prediction compares the proposed model with a baseline LSTM, demonstrating superior performance, particularly in capturing long-term trends. Additionally, a recent model called TEANet, combining Transformers and LSTMs for stock prediction, indicating comparable short-term performance but leaving room for exploration in longer time windows [7].

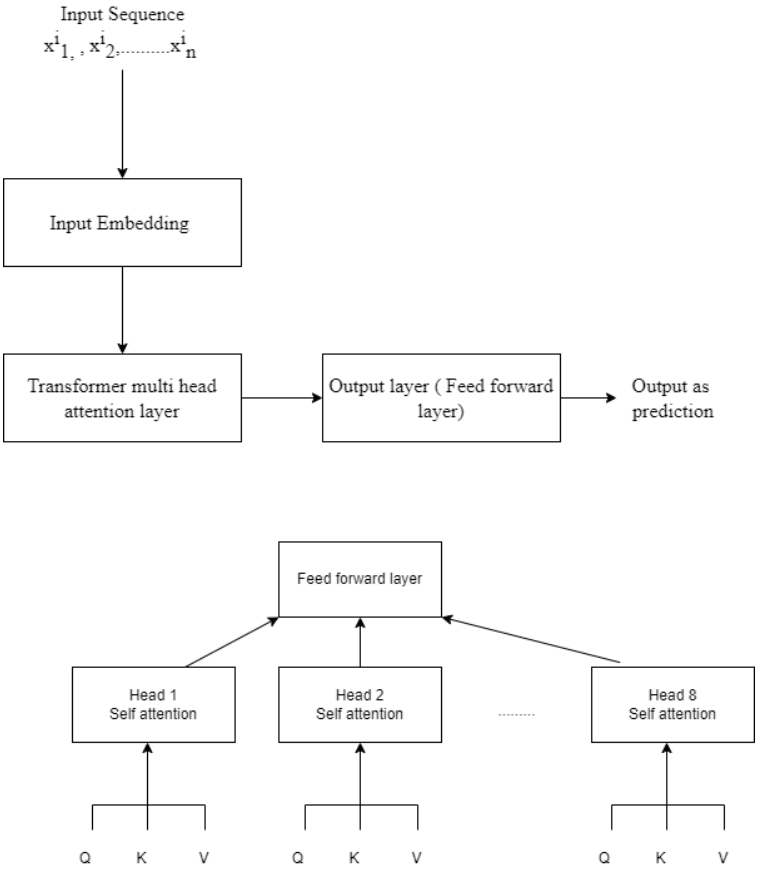

Figure 1: Overall working structure of model

Technical Plan:

Transformers use the attention mechanism for the implementation of the architecture. The main principle behind the attention mechanism is that the layer of neural networks calculates their attention to specific parts of data. In other words, attention mechanisms help to predict the dependencies of the sequence without regard to the distance. With increased parallelization the performance can be upgraded with better accuracy, thanks to the removal of vanishing gradient issue with attention module. The architecture of the encoder-decoder approach in transformer is shown below. Our main goal is to implement the encoder part of the architecture and predict the result. Further changes in the minor architecture level can be seen in the implementation. From our analysis we came to know we are skipping the decoder part and only implementing the transformer encoder part. There will be some adjustments in the feed forward layers in the architecture. The primary language we have planned to use is the Python as it is easy to implement the models, faster to run and supports multiple machine learning packages. [9]

The overview of our modified architecture is given below:

Considering X as the stock measures of n companies for m days. Here, considering X_1^i, X_2^i, …., X_n^i be the input sequence then V_i can be taken as the output showing the predicted value.

The primary mechanism in the attention module is the scaled dot product of the weights. The module introduces three parameters:

Q – Query
K – Key
V – Value

$$\text{Attention}(Q, K, V) = \text{SoftMax}\left(\frac{QK^T}{\sqrt{d_k}}\right)\text{V}$$

As per the above formula the transpose of the Key value and the Query undergo dot product. The SoftMax function makes the output in the range of 0 to 1 and finally the result is multiplied with Value to get the output from the attention block. In the multi-head attention part, several single-attention blocks are used, and the result is concatenated as the final output [2].

The detailed plan of the implementation is outlined as follows:

- Data Preprocessing: The stock market raw database collected will be filtered by removing unnecessary/corrupt data that will affect the accuracy of the model. This has been done in the dataloader.py [5] file. This file also does feature engineering to filter data based on the dates ranging older than 2004 Jan as we are considering those companies which are present before 2004 in the NASDAQ stock market.
- Feature Engineering: Prepare and create new features based on the raw data to improve performance and accuracy.
- Model Training: Training the model with the above pre-processed data. This will usually take a longer time to train. The model.py [5] file contains the model structure and will process the training as part of the generating a model with highest accuracy to predict future time-series data. For comparing transformer, we used LSTM model and the comparison is done in the model efficiencies and theoretical concepts.
- Documentation: Preparing final report and the presentation for the project review.

Implementation:

The source code of the project can be found at the GitHub repository referenced [5].

As per the plan provided in the proposal, we completed the research of the project and got the final results. This research included some feasibility works and also the motivation of the project. The project was feasible from every aspect like technical, financial and operational planning. There are lot of motivation behind the project as well. A truly accurate model with the proper data analysis of stocks prediction is always in need. We have developed a model that has long sequence of the temporal data for various stocks companies.

As per the dataset preparation step, we selected the NASDAQ dataset. NASDAQ is the stock market that has the index of over 7000+ companies in which the dataset will be hosting 7848 companies stock prices data till 2020. We will be predicting for companies from Technology sector and filtering companies whose IPO (Initial Public Offering) is older than 2004. The companies for training/testing data are also taken based on data with common dates of over 3500 days and these dates are common across all companies so that it will be easy for us to train the data. The latest data date range is till 2020 December 31 and the minimum date range starts from at least 2004 January 01.

Parameters for each stock on each day are:

- Volume : Net volume of stocks traded.
- Open : The open price of the day
- High : Highest price of the day.
- Low : Lowest price of the day.
- Close : Closing price of the day.
- Adjclose : Closing price after all adjustments of splits and dividends.

Data preprocessing:

The dataloader.py [5] file does many important functions as part of the code, it is divided into

i) Loading the data from the local device

ii) Cleaning the data: The cleaning of data especially focuses on filtering the data by date older than 2004 and creating the data with common range of date. It also focuses on pre-processing rows of data whose values are missing by initializing them to 0.

iii) Generating test and Training Data: The testing data is considered for those data whose years are 2015, 2018 and 2020 where most of the volatility has been observed in the market trends throughout the world. The data apart from the testing data for those of 2015, 2018 and 2020 are considered and taken as training data.

Model Structure and implementation:

All these temporal data processing has been created under the file data loader which sets the data for the model to train and the model structure is present in the model.py file for the transformer model and the LSTM model has been put under LSTM model file to generate another stock price prediction for LSTM model using which outputs, the transformer results will be compared. Further, we analyzed our model and have implemented the model using the encoder part of the transformer architecture. Upon various testing of different hyperparameters we have come up with best accuracy for the following hyperparameters:

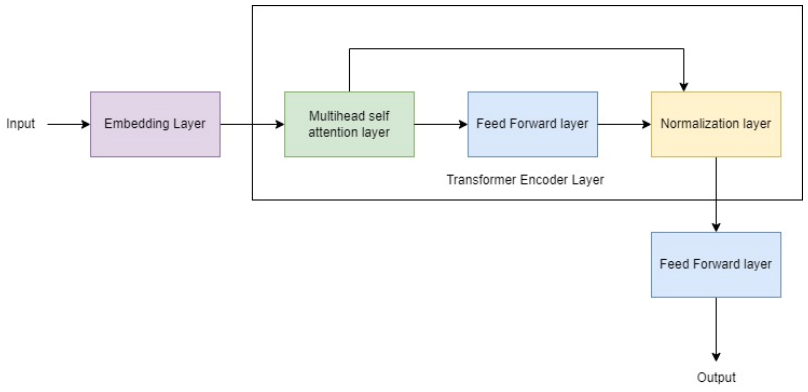

Figure 2: StockCoder Architecture

- Features : 6
- Number of companies : 102
- Model dimension : 512
- Learning rate : 0.0001
- Weight Decay : 0.0001
- Epoch number : 32
- Dropout rates : 0.05 -0.2

- Under the transformer model:
  - No. of Transformer layers: 16
  - No. of head : 8

The model.py [5] has the structure of model coded in it. The layers included are encoding layer, Transformer-encoder Layer, Feed Forward Layer and other linear layers which compiles all the model layers. The loss function utilized here is also total mean square error (MSE) loss. To optimize, the Adam optimizer (this yields better training loss) is used.

The model.py [5] can be broken into:

Embedding layer:

The input data undergoes embedding transformation through two linear layers (input_embed_1, input_embed_2). These layers covert raw inputs into higher dimension features suitable for processing by the subsequent layers. Post that, a dropout layer (input_dropout) is added to prevent overfitting during the training.

Transformer Layers:

The main section of the model consists of transformer layers in the StockCoder class and they form the backbone of model architecture by facilitating the extraction of hierarchical features from the input sequence. Each transformer layers consist of multi-head self-attestation and feed-forward sublayers enabling the model to capture local and global dependencies within the data.

Dimension Reduction:

Followed by transformer layers, dimension reduction layers are utilized to reduce the feature dimensionality and extract high level representations from the encoded sequences. The dimension reduction process involves multiple linear layers each followed by activation function (mostly ReLU) and dropout regularization to prevent overfitting [8].

Activation and Dropout Functions:

Throughout the model multiple activation functions has been tested (ReLU, GeLU, TanH, Identity, Sigmoid) and out of all, ReLU and GeLU proved to be effective for this model in introducing non-linearity and facilitating feature transformation function.

Final Results:

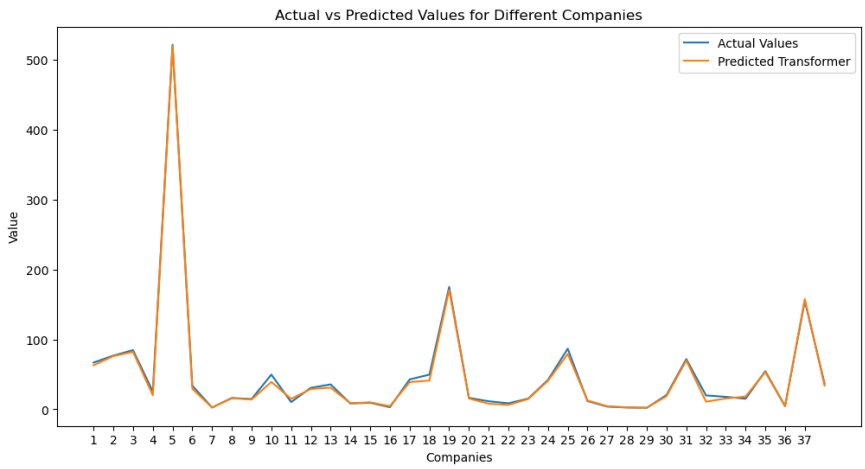

Figure 3: Mean closing value of companies for all testing years (2015, 2018, 2020)

The model is built as per the structure given above. All the layers and blocks are implemented in the model.py file. The result is taken from 38 companies. While testing the actual values are quite similar to the predicted values from our model. This graph shows the mean closing value of each company for the testing dataset years which are 2015, 2018 and 2020.

The plot below shows the training loss in transformer and LSTM model. It is clearly observable that transformer has performed much better than the LSTM.

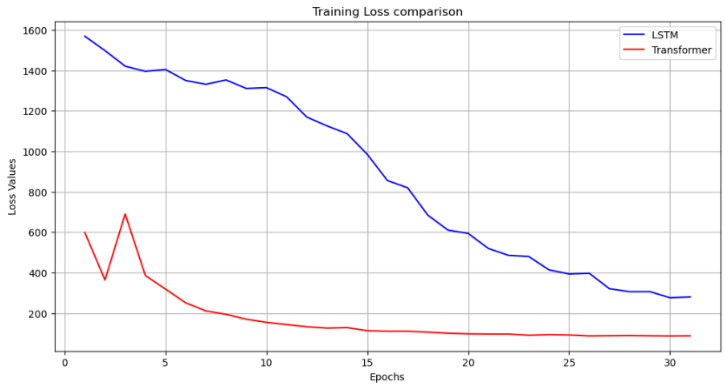

Figure 5: The loss value comparison between Transformer and LSTM models

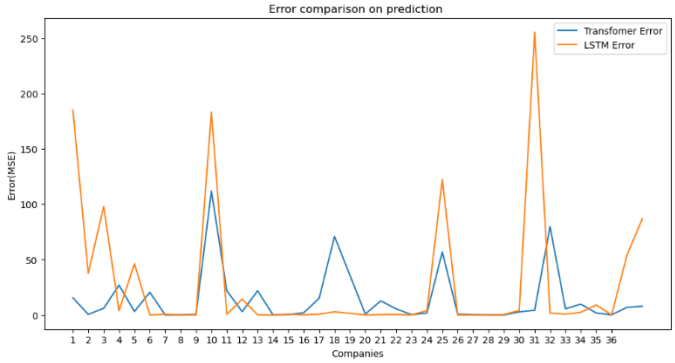

Figure 4: MSE error for Transformer vs LSTM comparison

The plot in figure 5 shows the MSE error on prediction while testing in transformer and LSTM.

The overall testing accuracy in transformer is 88 percent.

It is also observed from the figure 5 that the MSE for LSTM is way larger for many companies compared to Transformer model thus will reduce losses to the stock brokers using the model.

Future Works:

As per the proposal provided previously, we have completed the research and got the best accuracy for the model. This model can be further implemented in a front-end application to predict the data for the new companies in NASDAQ and might need tuning accordingly.

This same process can be expanded to train other different models from different datasets of various stock markets around the world and can be trained into sector wise to tune it accordingly to the markets in such a way it can only be able to predict the market for those particular stock markets with higher accuracy since companies under similar sectors have same trends mostly.

This also opens up the concept of finding the relation of price and volume in stock as companies with larger volumes and same sectors are more likely to show closing price trends to the similar value and the same applies with companies with smaller volume and market capital under same sectors.

Once the above are to be implemented, the hyper-parameter can be tuned to the other models accordingly based on the sector as the architecture will remain the same but the parameters won't for the better accuracy and lowest mean square error possible.

References:

[1] Chaojie Wang , Yuanyuan Chen b, Shuqi Zhang b, Qiuhui Zhang. 2022. Stock market index prediction using deep Transformer model. *ScienceDirect*

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkorei . 2017. Attention Is All You Need. *https://arxiv.org/abs/1706.03762*

[3] Yuan Gao, Shohei Miyata, Yuki Matsunami, Yasunori Akashi. 2023. Spatio-temporal interpretable neural network for solar irradiation prediction using transformer. *ScienceDirect*

[4] Mingxing Xu, Wenrui Dai, Chunmiao Liu, Xing Gao, Weiyao Lin, Guo-Jun Qi, Hongkai Xiong. 2021. Spatial-Temporal Transformer Networks for Traffic Flow Forecasting. https://arxiv.org/abs/2001.02908

[5] Saiman dahal, Aashik Sharif. 2024. https://github.com/saimandahal/StockCoder

[6] Lorenzo D. Costa, Alexei M. C. Machado. 2023. Prediction of Stock Price Time Series using Transformers. https://doi.org/10.5753/bwaif.2023.230239

[7] Harsimrat Kaeley, Ye Qiao, Nader Bagherzadeh. 2023. Support for Stock Trend Prediction Using Transformers and Sentiment Analysis. https://doi.org/10.48550/arXiv.2305.14368

[8] Krishu K Thapa, Bhupinderjeet Singh, Supriya Savalkar, 2023. Attention-based Models for Snow-Water Equivalent Prediction. https://arxiv.org/pdf/2311.03388.pdf

[9] StockFormer: A Swing Trading Strategy Based on STL Decomposition and Self-Attention Networks. https://arxiv.org/html/2401.06139v1