Predicting Stock Market Performance from Financial News Reports and Past Performance sing Transformer Architecture

Name	ID	Campus E-Mail
Assaf Levanon	209125483	assafl@campus.technion.ac.il
Tal Machavariani	211601406	tal.machavar@campus.technion.ac.il

Date: 07/11/2024

1. Introduction

- The goal of this project is to demonstrate that financial news reports contain valuable insights that can be leveraged to better understand future market performance. This information can be explicit, such as revenues, open lawsuits, layoffs, and bankruptcies, or implicit, like analysis opinions, public attitudes, and policymakers' beliefs and ideas. To test this hypothesis, we compared a traditional LSTM model that utilizes only past market performance data with an LSTM model enhanced by BERT[1], which extracts key insights from financial news articles.
- Predicting stock market performance is inherently challenging and widely recognized as a task with no guaranteed success. Numerous variables and unpredictable events influence market movements, making accurate forecasting a complex endeavor. Historically, the best predictor for future stock performance has often been the historical mean return, as it captures the average behavior of the market over time. This project seeks to explore whether integrating financial news can provide a more nuanced and potentially more accurate prediction than relying on historical data alone.

2. Methodology

- This project involves developing a neural network to predict stock market performance using datasets of financial news headlines and historical market data. The architecture features an embedding layer to convert textual data into numerical form, followed by LSTM layers that capture temporal dependencies and sequential patterns in the news headlines. These layers are then succeeded by fully connected layers that map the processed text features to stock market gains or losses. The model employs advanced text representation techniques and sequential processing to understand the complex relationships between news events and market movements. Additionally, we plan to build on previous projects[2] and utilize pre-trained modules to better suit our needs.
- For training and testing the Plain LSTM model, we used close price data about the S&P500 index and the 40 biggest companies that make up the index, from 2005 to 2020 (acquired from yfinance).

---

[1] https://huggingface.co/docs/transformers/en/model_doc/bert

[2] https://github.com/tomer9080/Stock-Prediction-Using-RWKV/tree/main?tab=readme-ov-file
https://github.com/tamirhaver129/Deep-Learning-Course-046211-Project-Predictinting-Stock-Prices?tab=readme-ov-file

- Financial News Headlines: Used for training and testing the LSTM with BERT module.
- We started with an LSTM model that predicted the next day's closing price based solely on a sequence of previous closing prices[3]. Additionally, we utilized a BERT transformer model to analyze the day's financial news headlines and obtain sentiment-driven insights. These two sub-models are then aggregated with a concatenation layer for the final output.

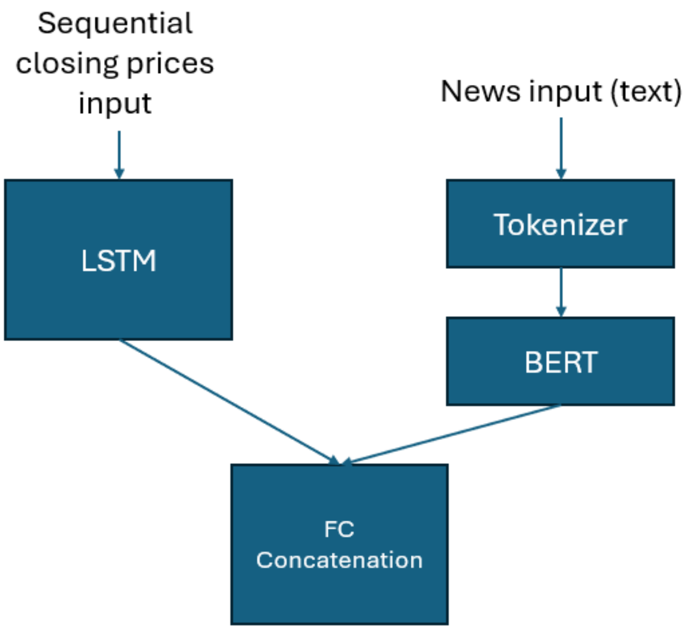


Figure 1: general structure of the components of the module

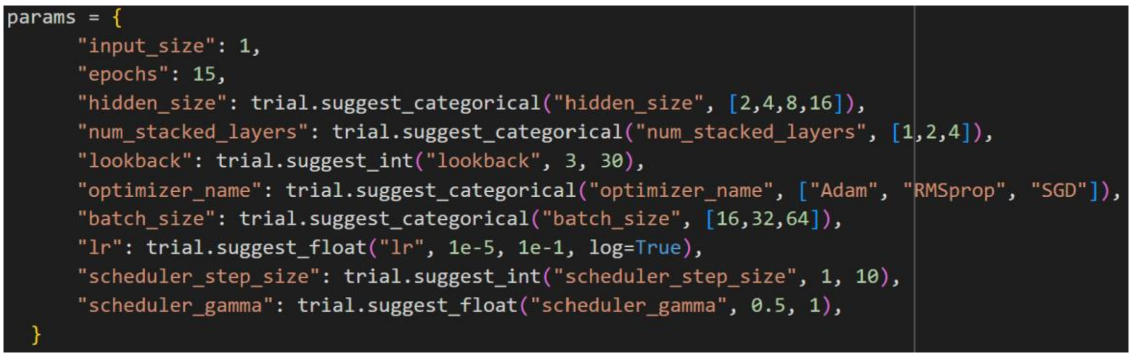
- After further research, we concluded that it would be beneficial to predict the Gain/Loss (measured as a percent of the stock price) of each consecutive day rather than the closing price of each day. Those values are more recurrent and achieve superior results while offering the same insight.

- Hyperparameter tuning was made using the Optuna library. The optimizer type (Adam, RMSprop, SGD) was a hyperparameter determined by Optuna. Both the Plain and the BERT versions used StepLR scheduler, with the scheduler step size and gamma as hyperparameters. The hidden size and number of layers in the LSTM modules were also optimized by Optuna. The loss criterion chosen for assessing our results is mean square error (MSE), as it is widely used for regression problems of this sort.

[3] We relied on Greg Hoggs "Stock Price Prediction & Forecasting with LSTM Neural Networks in Python" YouTube tutorial for the initial LSTM structure and syntax

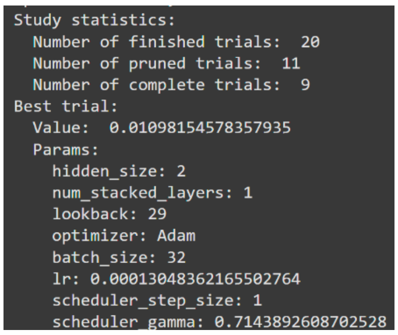
3. experiments and results

For the plain LSTM, we used the following initialization for the Optuna Study:



```
params = {
      "input_size": 1,
      "epochs": 15,
      "hidden_size": trial.suggest_categorical("hidden_size", [2,4,8,16]),
      "num_stacked_layers": trial.suggest_categorical("num_stacked_layers", [1,2,4]),
      "lookback": trial.suggest_int("lookback", 3, 30),
      "optimizer_name": trial.suggest_categorical("optimizer_name", ["Adam", "RMSprop", "SGD"]),
      "batch_size": trial.suggest_categorical("batch_size", [16,32,64]),
      "lr": trial.suggest_float("lr", 1e-5, 1e-1, log=True),
      "scheduler_step_size": trial.suggest_int("scheduler_step_size", 1, 10),
      "scheduler_gamma": trial.suggest_float("scheduler_gamma", 0.5, 1),
  }
```

Figure 2: Plain LSTM hyper parameters range for Optuna study



```
Study statistics:
  Number of finished trials:  20
  Number of pruned trials:  11
  Number of complete trials:  9
Best trial:
  Value:  0.01098154578357935
  Params:
    hidden_size: 2
    num_stacked_layers: 1
    lookback: 29
    optimizer: Adam
    batch_size: 32
    lr: 0.00013048362165502764
    scheduler_step_size: 1
    scheduler_gamma: 0.7143892608702528
```

Figure 3: Plain LSTM study results

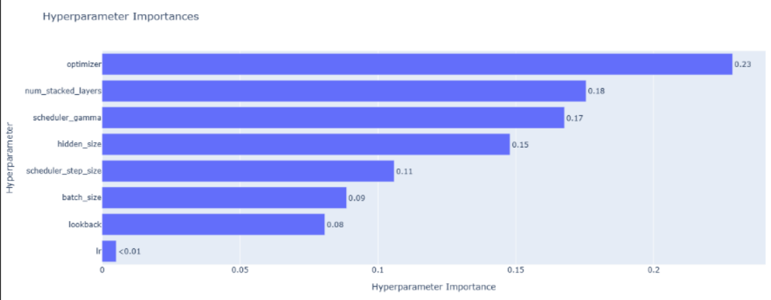


Figure 4: Plain LSTM hyper parameter importance

After finding the hyper parameters, we trained our model with them. The training and validation loss vs. epoch of the training is shown in Figure 5.

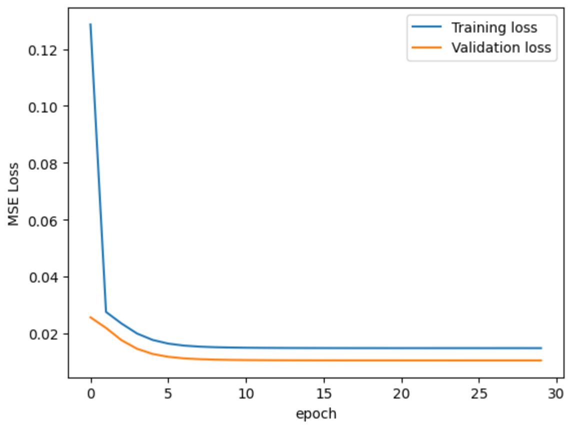
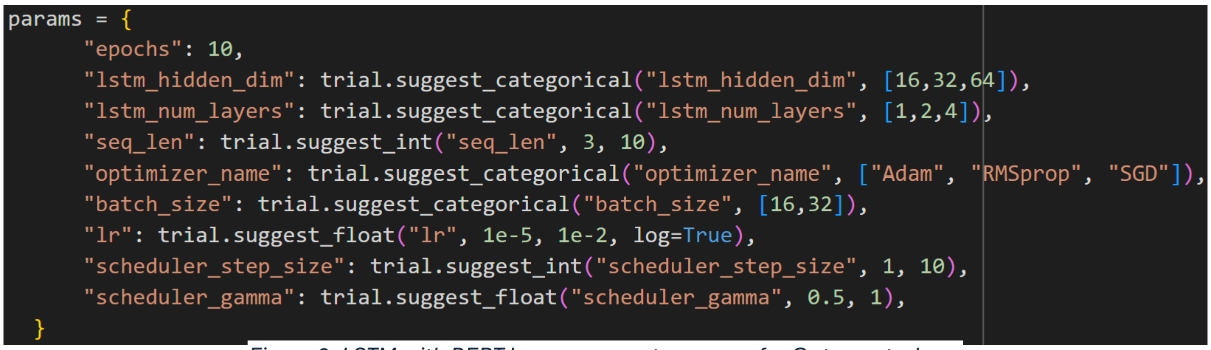


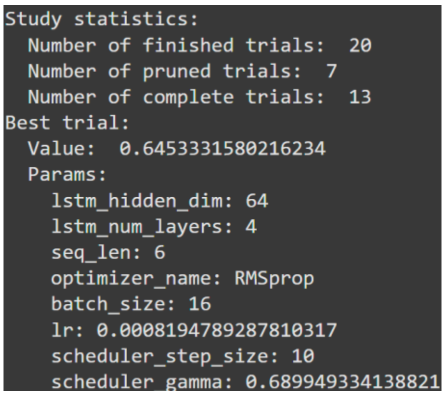
Figure 5: MSE loss vs Epoch - Plain LSTM module

For LSTM with BERT module, we used the following initialization:



```
params = {
        "epochs": 10,
        "lstm_hidden_dim": trial.suggest_categorical("lstm_hidden_dim", [16,32,64]),
        "lstm_num_layers": trial.suggest_categorical("lstm_num_layers", [1,2,4]),
        "seq_len": trial.suggest_int("seq_len", 3, 10),
        "optimizer_name": trial.suggest_categorical("optimizer_name", ["Adam", "RMSprop", "SGD"]),
        "batch_size": trial.suggest_categorical("batch_size", [16,32]),
        "lr": trial.suggest_float("lr", 1e-5, 1e-2, log=True),
        "scheduler_step_size": trial.suggest_int("scheduler_step_size", 1, 10),
        "scheduler_gamma": trial.suggest_float("scheduler_gamma", 0.5, 1),
  }
```

Figure 6: LSTM with BERT hyper parameters range for Optuna study



```
Study statistics:
  Number of finished trials:  20
  Number of pruned trials:  7
  Number of complete trials:  13
Best trial:
  Value:  0.6453331580216234
  Params:
    lstm_hidden_dim: 64
    lstm_num_layers: 4
    seq_len: 6
    optimizer_name: RMSprop
    batch_size: 16
    lr: 0.0008194789287810317
    scheduler_step_size: 10
    scheduler_gamma: 0.689949334138821
```

Figure 7: LSTM with BERT study results

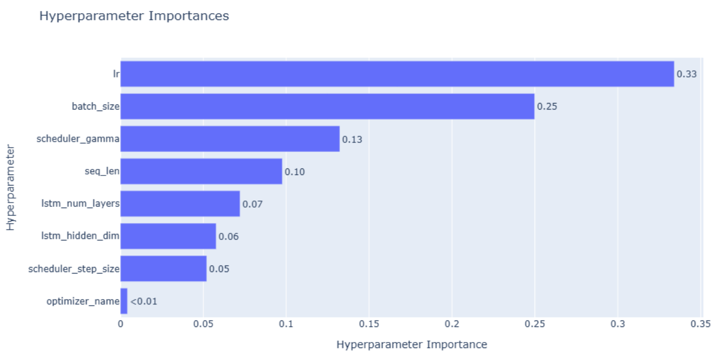


Figure 8: LSTM with BERT hyper parameter importance

After finding the optimal hyperparameters, we trained our model with them. The training and validation loss vs. epoch of the training is shown in Figure 9.

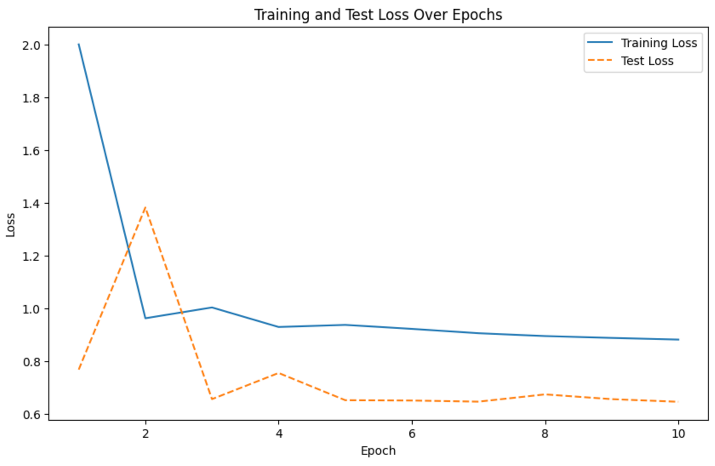


Figure 9: MSE loss vs Epoch - LSTM with BERT module

The predictions of the trained modules on the test set are shown in Figures 10, 11

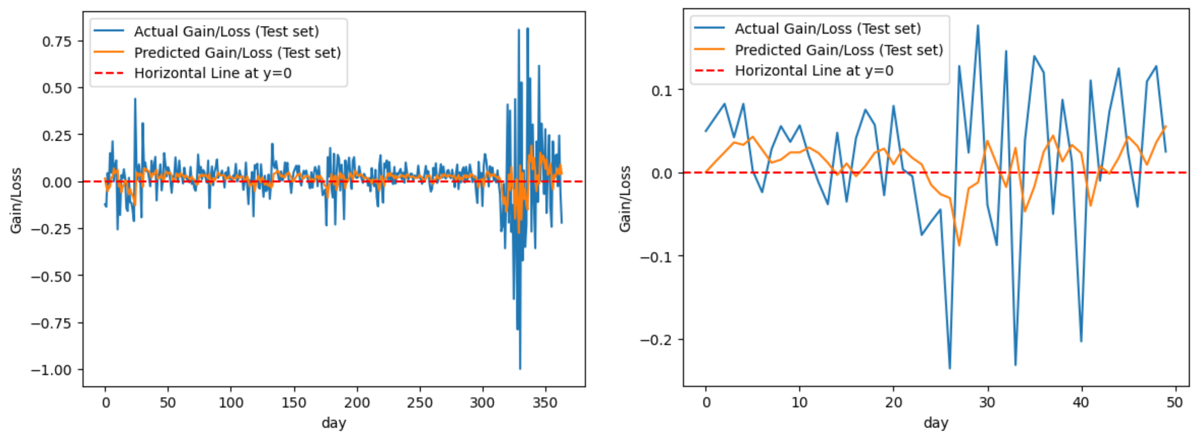


Figure 10: Actual Gain/Loss vs. Predicted by Plain LSTM module, on test set, with zoom-in (right graph)

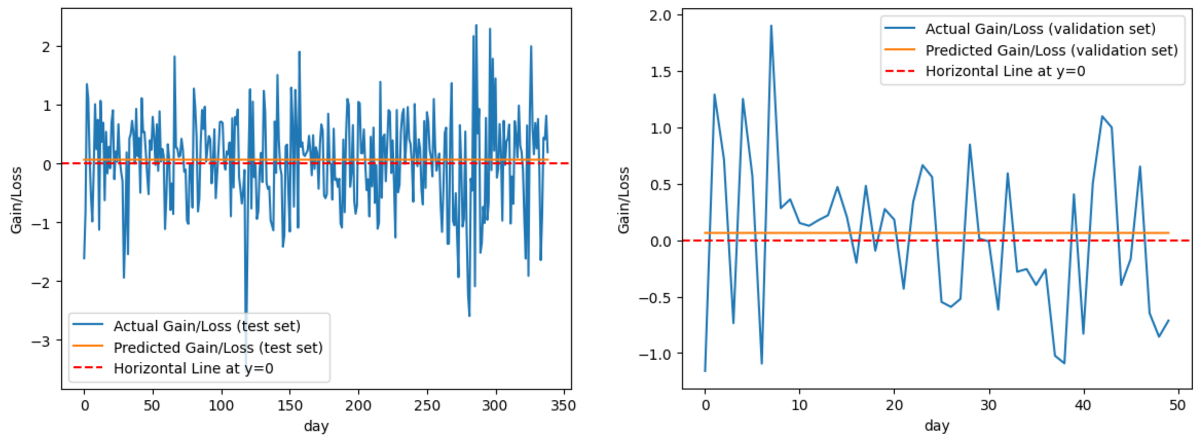
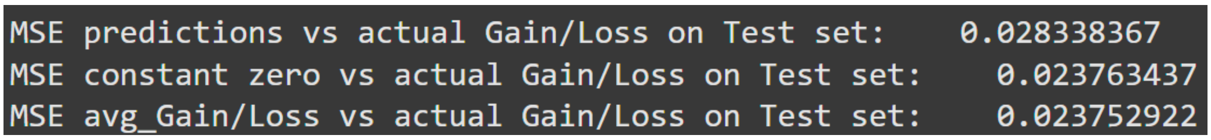


Figure 11: Actual Gain/Loss vs. Predicted by LSTM with BERT module, on test set, with zoom-in (right graph)

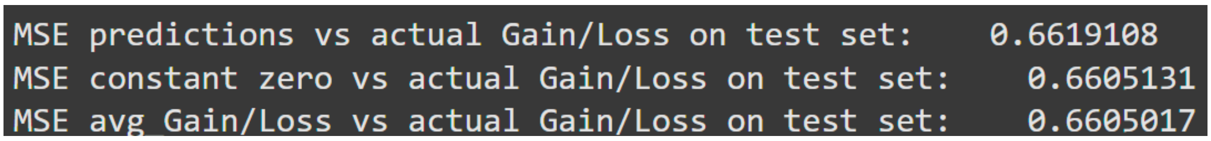
After examining the results, it seems like the Plain LSTM module achieves mediocre results. The predictions follow the general trend of the market on some occasions and produce what seems like random noise on other occasions. On the other hand, the LSTM with BERT module seems to produce a constant prediction for the gain. This falls in line with our knowledge about the unpredictability of the stock market but means that we have failed to extract useful insights from the news.

We decided to check the performance of our module vs. two very naïve modules: one of constant market prices (Gain = zero), and the other of constant market gain equivalent to the average stock market return (10% annually). The metric used was MSE over the test set.



```
MSE predictions vs actual Gain/Loss on Test set:    0.028338367
MSE constant zero vs actual Gain/Loss on Test set:    0.023763437
MSE avg_Gain/Loss vs actual Gain/Loss on Test set:    0.023752922
```

Figure 11 – MSE comparison of Plain LSTM Module and two naïve modules



```
MSE predictions vs actual Gain/Loss on test set:    0.6619108
MSE constant zero vs actual Gain/Loss on test set:    0.6605131
MSE avg_Gain/Loss vs actual Gain/Loss on test set:    0.6605017
```

Figure 12 – MSE comparison of LSTM with BERT Module and two naïve modules

We shall note that the absolute values of the MSE should not be compared because they were achieved on different datasets. It seems that the Plain LSTM module performs significantly worse than LSTM with BERT compared to the best module available for each dataset (which is constant gain of 10% annually). Yet, we can confidently say that LSTM with BERT performs relatively well because it “collapsed” to be close to the trivial solution, which is still the best so far.

4. conclusion

The project has failed to produce a module that outperforms the naive approach of Constant Gain. Yet, there is much room for further improvement and research. Our computational resources were scarce, and we gathered a relatively small amount of data. We believe that a deeper examination of the problem, cross-referencing multiple datasets, and combining other types of data (e.g., historical interest rates or US national debt) into the encoder-decoder architecture may produce superior results.

We shall note a considerable fault of our research – the Plain LSTM and LSTM with BERT modules were trained on different stock market price datasets. This negatively affected our ability to compare the results of the two modules and should be avoided in future projects. The use of Optuna was extremely beneficial and will be of great importance in our future research.

5. Ethics Statement

Student names: Assaf Levanon & Tal Machavariani

Project Title: Predicting Stock Market Performance Using Transformers and LSTM

Summery:

- Our project aims to predict future stock market performance by leveraging both historical market data and relevant financial news reports. We use a Transformer architecture for integrating text data from news articles with past market performance and compare it to a traditional LSTM model that solely considers historical market data.

Stakeholders: Investors, Financial Analysts, Stock Market Regulators

What would be the Implications for each stakeholder:

- **Investors:** Our system provides enhanced insights into potential future stock movements by analyzing both historical data and real-time financial news. This can help investors make more informed decisions and potentially increase their returns.
- **Financial Analysts:** The model offers an advanced tool that integrates diverse data sources to predict market trends. This can support analysts in providing more accurate forecasts and recommendations to their clients.
- **Stock Market Regulators:** The project ensures transparency and fairness by providing an additional layer of analysis that could highlight unusual market activities influenced by news, helping to maintain market integrity.

Who is responsible for giving the explanation to each stakeholder?

- **Investors:** The responsibility lies with financial advisors and investment firms who adopt this technology. They need to communicate how the model works and its benefits.

- **Financial Analysts:** Data scientists and developers working on this project should provide detailed explanations to financial analysts about the model's capabilities, data sources, and predictive accuracy.

- **Stock Market Regulators:** Regulatory bodies should receive comprehensive reports from the development team, detailing how the model functions, its data sources, and any potential implications for market surveillance.

Reflect on the AI output

- To make the explanation more ethical, the responses should emphasize the importance of addressing potential biases in the data. Specifically, social biases present in the news may affect our model. For example, our model may provide biased evaluations to certain companies based on location, local culture, and race. This, in turn, may affect the stakeholders and negatively impact those companies, thereby reinforcing existing social biases.