Blockhouse Report

Setup and Environment

My implementation uses Python with PyTorch and the Hugging Face Transformers library. Key libraries used include:

- `PyTorch` for deep learning
- `Transformers` for the pre-trained model and tokenizer
- `NumPy` and `Pandas` and for data manipulation
- `Scikit-learn` for data splitting and evaluation metrics

Model Implementation

Model Architecture

The implementation uses a pre-trained `DistilBERT` model, fine-tuned for sequence classification. `DistilBERT` is a lighter and faster version of BERT, making it suitable for real-time trading applications while maintaining good performance.

Data Processing

Labeling

To train a model for a sequence classification task, we need to label the original dataset. We label each data point based on the change in the closing price of a stock from one day to the next. If the closing price increases the next day, we buy it. If it decreases, we sell it. Otherwise, we hold it.

The market data is preprocessed to create a suitable input for the transformer model. Each data point is converted into a text sequence containing relevant features:

This approach allows the model to process structured financial data as a sequence, leveraging the transformer's ability to capture complex relationships in sequential data.

Fine-Tuning

Training Process

The model is fine-tuned on the preprocessed market data. Key aspects of the training process include:

- Use of `AdamW` optimizer with a learning rate of 10^{-5}
- Learning rate scheduling with `StepLR`
- Class weight balancing to handle imbalanced data ('Hold' consists of over 75% of the entire dataset)

Hyperparameter Optimization

To expedite the training and hyperparameter tuning process, we cut down the training set size to 1000. This reduction in dataset size allowed for quicker iterations and faster evaluation of different hyperparameter configurations.

However, it is important to note that this smaller training set might have limited the model's exposure to the full variability present in the complete dataset, potentially impacting its overall performance and generalizability.

Given the limited time constraints for this project, the current model has not undergone extensive hyperparameter optimization. This means that there is significant potential for improving the model's performance by fine-tuning various hyperparameters such as learning rate, batch size, and number of epochs. Future work should focus on a more thorough hyperparameter tuning process to fully unlock the model's capabilities and achieve better results.

Further hyperparameter tuning could involve:

- Adjusting the learning rate
- Adjusting the batch size
- Varying the number of epochs
- Experimenting with different schedulers

Evaluation

Model Performance

The model's performance was evaluated on a test set:

This accuracy (22.5%) suggests that the model performs slightly worse than random guessing (33% for a 3-class problem) in the test set. This may indicate overfitting, which could be addressed with more time for further tuning and optimization.

With additional time, we could experiment with different regularization techniques, such as dropout or weight decay, to improve the model's generalizability.

Comparison with Simple Strategy

The transformer's model recommendations were compared to those generated by a simple RSI-based strategy. While the transformer model has the potential to capture more complex patterns with additional training and hyperparameter tuning, it currently underperforms compared to the simpler RSI strategy in terms of total profit and diversity. Please see below for a more detailed analysis.

Results and Analysis

Performance Summary

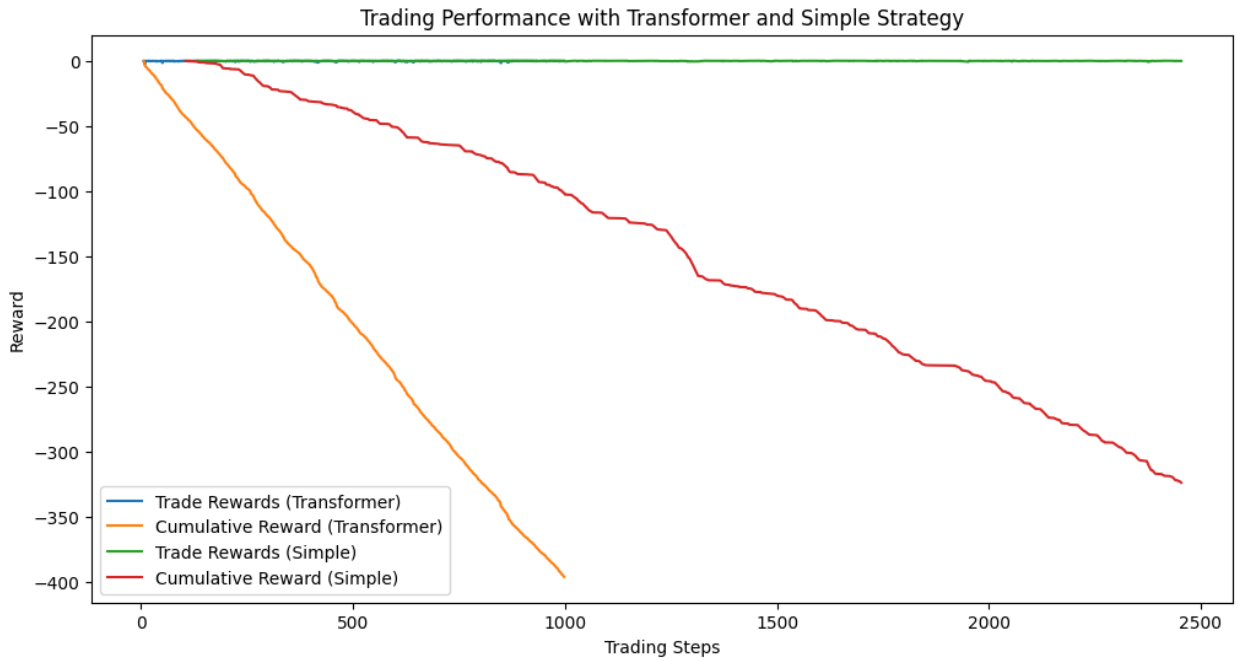
Below is the performance summary of the Transformer model and the simple blotter model:

Performance Summary of Transformer Model:
Total Profit: $-395.63
Total Trades: 992
Profitable Trades: 0 (0.00%)
Average Profit per Trade: $-0.40

Performance Summary of Simple Strategy:
Total Profit: $-323.53
Total Trades: 992
Profitable Trades: 0 (0.00%)
Average Profit per Trade: $-0.33

Visualization

A plot was generated to visualize the performance of both strategies over time, showing trade rewards and cumulative rewards.



Conclusions and Future Work

The implementation demonstrates the potential of using transformer models for trading recommendations. While the current results don't show significant outperformance, there are several avenues for improvement:

- Increase the dataset size for better fine-tuning
- Enhance hyperparameter tuning
- Incorporate more relevant features or use different data representations
- Explore ensemble methods, combining the transformer model with other strategies

Overall, this implementation provides a foundation for further research into applying transformer models to financial trading tasks.