1 Introduction

This project introduces a novel of Reinforcement Learning (RL) agent characterized by its ability to modulate risk tolerance due to a Natural Language Processing (NLP) module.

Central to this agent's decision-making framework is an NLP-based Large Language Model (LLM), specifically a fine-tuned BERT model [1], which interprets an investor's risk preference from natural language inputs and assigns a corresponding risk level which modifies the agent's behaviour. The agent engages with three distinct assets: Google stock ("GOOG"), the S&P500 index ("ĜSCP"), and Gold ("GC=F"). It is programmed to execute one of three actions —buy, hold, or sell— for each asset at every decision point, provided it has enough remaining balance to perform the action.

The training of the BERT model involved a bespoke-created dataset comprising 1,000 phrases likely to be used by potential users, allowing for classification of input phrases into one of three risk categories: low, medium, or high.

The objective of this paper is to show how varying input phrases influence the agent's behavior. This is achieved through a series of simulations that explore different risk paths and their consequent impact on asset volatility. Furthermore, we investigate the incorporation of the Sharpe ratio, a metric for assessing the risk-adjusted return, and its effects on the agent's decision-making process.

The paper is structured into Methodology, Data, Experiment, and Conclusion chapters, detailing the agent and NLP model's architecture, dataset rationale, experimental findings, and the implications of merging NLP with RL for financial strategies, alongside future research directions.

2 Data

In this chapter, we will go over the datasets used to train both the Reinforcement Learning (RL) agent and the Natural Language Processing (NLP) model. The goal is to give you a clear picture of the data behind the decisions and the language understanding of the project.

2.1 Agent Data

To train the agent we used data relative to three distinct assets: Google stock (”GOOG”), the S&P500 index (”ĜSCP”), and Gold (”GC=F”). The data contained daily price for each asset between ”2016-01-01” and ”2022-12-31”. After weeding out days when not all assets were traded, we ended up with data for 1,255 trading days. For each trading day the following data features were available: Open, High, Low, Close, Adjusted Close (Adj Close), and Volume. To add more information, the daily returns and 3,6,9 day moving Adj Close averages were computed.

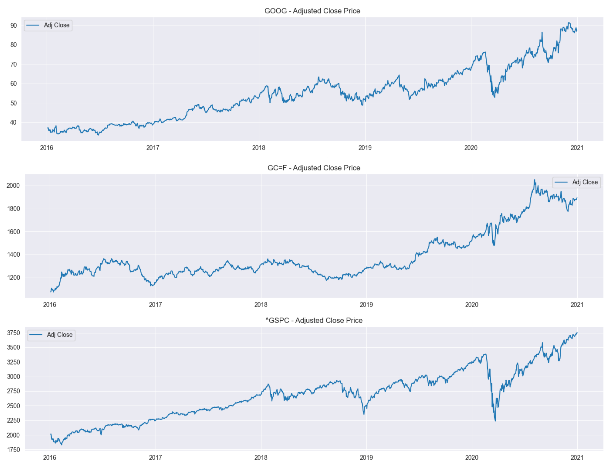


Figure 1 ***Assets' Adj Close Price*** - Asset price in the specified time-frame.

When the daily returns were analysed [2], it was clear which asset was the most volatile: Google stock showed the highest volatility, followed by the S&P500, with Gold being the most stable. The summarized volatility insights can be seen in Table 1.

Asset	**Std. Dev. Returns**	**Min Adj Close**	**End Price**
GOOG	1.669	33.412	91.399
S&P500 (GSPC)	1.226	1829.080	3756.070
GOLD (GC=F)	0.922	1073.900	2051.500

Table 1: Asset volatility in timeframe

The data was also split in training and two test sets: the time period from ”2016-01-01” to ”2020-12-31” was used as training data (1005 records), the data from ”2021-01-01” to ”2021-12-31” as the first test data (250 timesteps) while from ”2022-01-01” to ”2022-12-31” for the second test dataset.

2.2 NLP Data

As mentioned in the previous section the NLP part of the project was composed by a fine-tuned BERT model. To fine-tune the LLM, an ad-hoc dataset had to be created to mimic the various inputs an investor might give. Thus, we manually wrote down a list of 20 phrases an investor might say, along with 10 possible phrase beginnings, 5 possible phrase endings related to their risk-tolerance and 5 related to their time-horizon preference. The lists were extended with generative AI to 45 complex full phrases, 25 phrase beginnings, 11 risk-tolerance endings and 9 time-horizon phrase components. Each phrase or phrase ending related to risk tolerance was tagged with a label: low, medium, or high. The final dataset comprised these 45 complex phrases plus a mix-and-match of beginnings and endings, alongside time-horizon components, to round out to 1,000 labeled phrases. In anticipation of future studies, we also tagged each phrase with a time-horizon label for completeness.

Phrase	**Risk-level**	**Time-horizion**	**Method**
”I am using my life’s saving for a stable retirement”	0	2	Complete
”I prefer investments that are a bit adventurous but not too wild”	1	1	Complete
”Eager to explore” + ”safe investment options” + ”over the long haul”	0	2	Mix-and-match

Table 2: Example of dataset phrases

3 Methodology

In this section, we detail the methodology behind our project and go over the approach of combining the RL agent with NLP.

3.1 Agent Model

The main agent code is an extension of the framework provided by Prof. Tomaso Aste, Dr. Paolo Barucca and T.A. Antionio Briola [3]. It operates across three assets -up from the original single asset- and employs a straightforward two-layer neural network to approximate the Q-function, a decision driven by computational constraints, with potential enhancements left for future work.

The learning process is iterative: the agent with probability ϵ selects a random action a or, with probability 1 - ϵ, selects the action that maximizes the Q-value (a = argmax

Q(s,a)). It executes action a and store the tuple (s,a,r,s') in the replay buffer. Network training is performed by sampling a random mini-batch of transitions from the replay buffer. For every transition in the batch, it computes y = r if the episode has ended, or y = r + γ max Q*(s',a') otherwise and uses y for computing the loss and updating the Neural Network to better approximate Q(s,a). And this process is repeated every N steps.

As we will see in the following subsection, the reward r is modified based on the risk level.

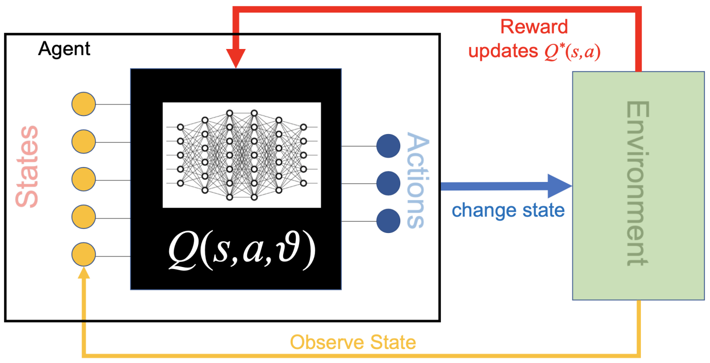
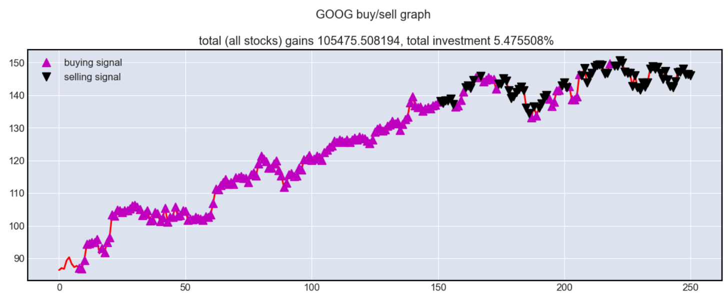


Figure 2 ***Network architecture*** *[4]* Q-Learning Process

The action space consists of 27 possible actions—combinations of buying, selling, or holding for each asset. An action decoder executes these transactions, modifying the agent's inventory based on available funds. The state vector s is composed of the 10 data features presented in the Data section for each asset, making the state vector of total length 30.

The agent has internal variables that estimate and track the volatility of each asset and their Sharpe ratio, at each time-steps the agent updates it's arithmetic mean estimates.



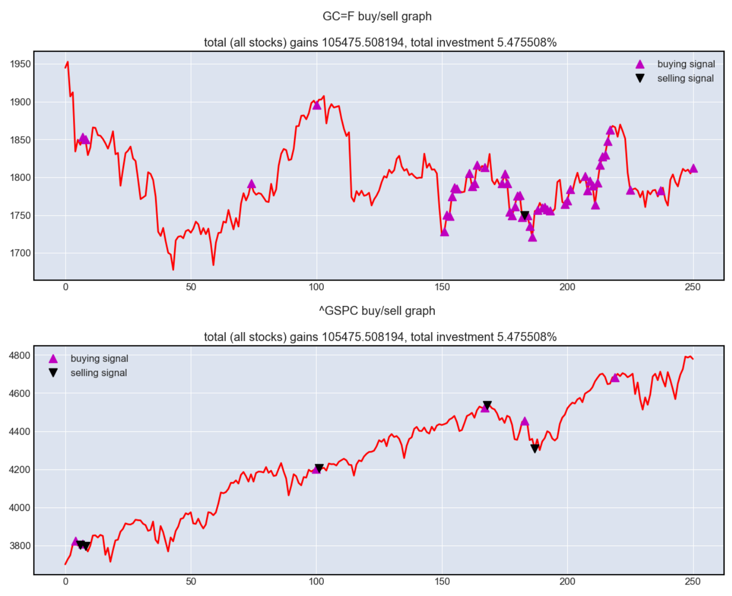


Figure 3 ***Sample behaviour of agent on test set*** - Buy / Sell actions on assets on test set (final return considers inventory).

3.2 NLP Model

The NLP model is a fine-tuned BERT 'base-uncased' model along with it's tokenizer [5]. The training was performed on the aforementioned bespoke dataset on a three class classification task assigning low (0), medium (1) or high (2) risk-level to each input [6]. Given the size and performance of the LLM model, even by training on a relatively small dataset, after 30 epochs of training, test accuracy reached 99%.

3.3 Integration of Agent model and NLP model

The agent contains a risk-level variable that can be changed and is set by giving an input to the NLP model. The output label of the NLP model is then set as the agent's risk level.

The risk level is integrated in the agent's act function where if risk-level is low then the action's probability -the corresponding logit- is boosted by up to 0.15, proportional to the number of hold actions the agent has to perform to execute the action (ex. action [hold, hold, hold] on all three assets' probability is boosted the most while actions like [sell, buy, hold] are boosted less). If risk level is high the inverse happens and hold actions are penalized. This is to reflect a situation where high risk level encourages more trading to try and "beat" the market while low risk is related to holding good assets.

The impact of risk levels extends to how rewards are calculated and assigned for each action. [7] Specifically, the reward at each time-step is the sum of the reward after executing actions on each stock and the reward for an action on each asset is as follows: a ”sell” action garners a reward equivalent to the profit earned from the transaction, a ”buy” action maintains a neutral reward position (zero or negative transaction cost), and a ”hold” action incurs a penalty reflective of negative unrealized gains compared to the previous time-step. These rewards are then modified by the risk level: for high-risk level reward is multiplied by 20% if operating on a high volatility stocks while if risk-level is low, the reward is increased by 20% when operating on a low volatility asset. An asset is considered high (low) volatility if the internal estimate of it’s volatility by the agent surpasses (is below) a certain threshold.
In pseudo-code formulation the reward is as such:

```
reward():
    reward_list = []
    for action_taken, asset in actions_taken_this_timestep:
        if action_taken == "buy":
            r = -transaction_cost
        elif action_taken == "sell":
            r = profit = current_price - bought_price
        elif action_taken == "hold":
            # As if bought and sold immediately
            r = unrealized_gains = - return_on_day
        # Reward multiplier based on risk level and asset volatility
        if risk_level == "high" and asset_vol_estimate > vol_threshold:
            r *= 1.2
        elif risk_level == "low" and asset_vol_estimate < vol_threshold:
            r *= 1.2
        reward_list.append(r)
    total_reward = sum(reward_list)
```

4 Experiments

Two main experiments have been conducted. The first experiment investigates how varying risk levels influence the agent's strategic behavior, while the second experiment assesses the impact of incorporating the Sharpe ratio into the decision-making process.

4.1 Experiment 1 - Agent Behaviour

In this experiment we will show how the different internal risk level change the agent's action and how they implement it's policy. The objective was not to benchmark performance but to understand whether the agent adopts different strategies across risk levels. To achieve such goal, for each risk level (set using a corresponding NLP input), the agent was trained on the training set and then simulated the performance on two different test sets 50 different times. The first test set contained data from "2021-01-01" to "2021-12-31", while the second one from "2022-01-01" to "2022-12-31". Due to the randomness in selecting the action all paths are subject to differences, so we analyzed their distribution and characteristics. As can be seen in the graph below, there is significant difference in the path taken by the agent at different risk levels.
The following image shows the agent's total asset path on the test set simulation divided by colour. Each colour represents a different risk level (red represents high risk level, blue low risk level and green medium risk level).

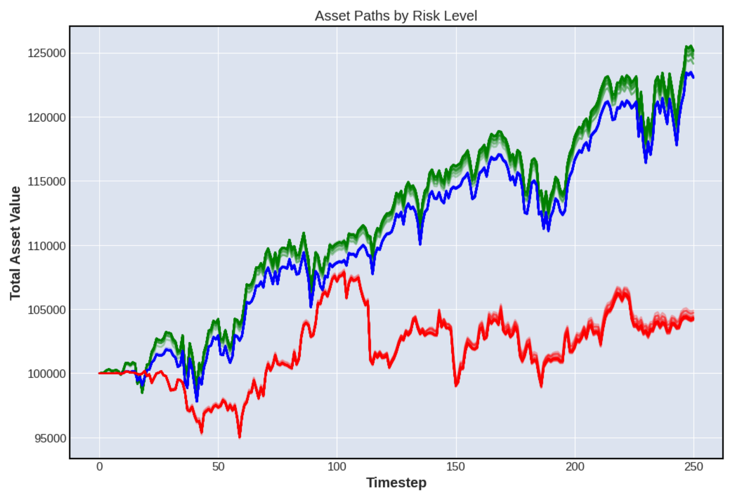


Figure 4 ***Asset paths by risk level on test set 1*** - Total agent asset value paths, differnt colors imply different risk levels.

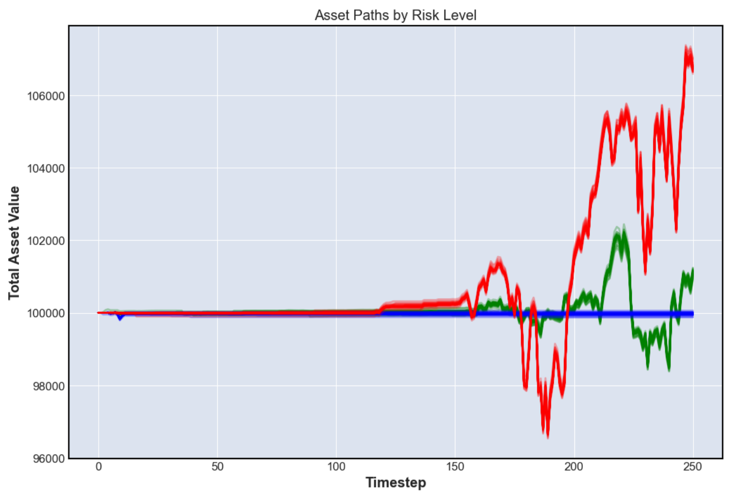


Figure 5 ***Asset paths by risk level on test set 2*** - Total agent asset value paths, different colors imply different risk levels.

By analyzing these simulation we can see that the agent's decision-making policy indeed adapts to different risk tolerances, although optimizing these policies for specific objectives remains a task for future research. Further confirmation was found using a statistical test: an ANOVA statistical test on the volatility confirmed that the null hypothesis of all -note all- three different risk levels were sampled from the same distribution was false.
It's important to note that while the agent performed well on our test set, it was largely due to having chosen very good underlying assets for the selected test period, since our focus was strictly on the agent's behavioral dynamics rather than financial outcomes.

4.2 Experiment 2 - Sharpe Ratio

In our second experiment, we incorporated the Sharpe ratio, a metric for risk-adjusted returns, into the decision-making for a medium risk level. As mentioned in the Methodology section, the agent updates its Sharpe ratio estimates for each asset at every timestep and to leverage this a reward function tweak was introduced: a 120% multiplier for actions with high Sharpe ratio assets applicable if the agent's risk-level is medium. This aims to explore how Sharpe ratio estimates affect agent behavior, encouraging a balance between risk and return. Similar to the first experiment, we trained the agent across risk levels and ran 40 simulations on the test set.

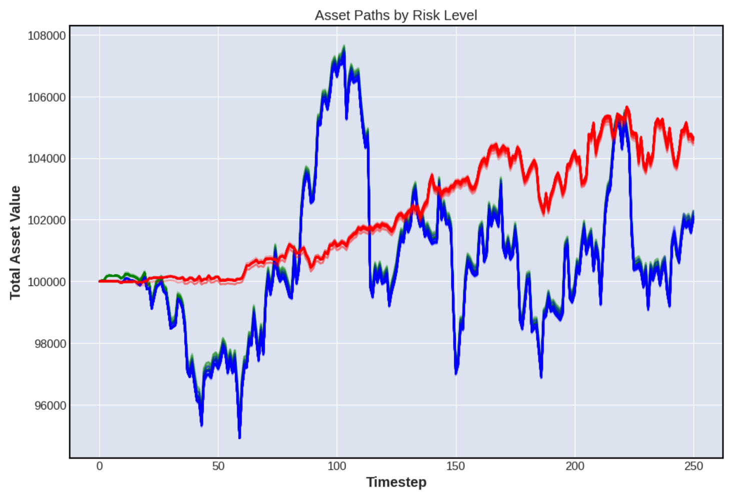


Figure 6 ***Asset paths by risk level on test set*** - Green corresponds to risk level medium, blue to low and red to high.

Results indicate this adjustment aligns the agent's behavior with the low risk setting, likely because the assets with high Sharpe ratios are also those with low volatility. Nonetheless, it confirms that incorporating the Sharpe ratio affects behavior, suggesting further refinement for its integration into the reward and action functions is an avenue for future research.

5 Conclusion

In conclusion, our exploration into integrating a Reinforcement Learning agent with Natural Language Processing (NLP) presents an interesting proof of concept. This novel agent demonstrates the potential to move beyond maximizing profits towards dynamically adapting its strategies based on user input regarding risk tolerance and investment time horizons. While the agent and its underlying models are not yet fully optimized, they pave the way for innovative approaches in designing adaptive financial decision-making tools.

5.1 Critical Areas & Future Work

The constraints of time and computational resources have highlighted several critical areas for further development:

- **Neural Network Architecture:** Upgrading from the current simple two-layer neural network to a more complex architecture could enhance the agent's learning efficiency and decision-making capabilities.

- **Data and Computational Resources:** Expanding the dataset and state-space to allow the agent to operate on more assets (and more frequent data) while securing more powerful computational resources would allow for more thorough training of the model.
- **Risk Level Dynamics:** Further investigation into how different risk levels affect the agent's behavior, especially regarding the modification of rewards with better integration of the volatility system and Sharpe ratio, could refine its responsiveness to varying market conditions and user preferences.
- **Exploration of agents adaptive behaviour** : To better understand the behaviour of the agent, new experiments could be conducted, such as giving it a new NLP input mid test cycle to see if behaviour changes (since act function is also influenced) or more detailed analysis on various test sets. Moreover, the statistical test analysis can be strengthened with paired tests between risk levels.
- **Time-Horizon Integration:** Given the dataset already includes labels for investment time horizons and an internal time-horizon variable has been partially set up, incorporating this dimension into the agent's decision-making process by modifiying act/reward functions represents a promising avenue for research.

5.2 Closing Remarks

This project highlights the potential of adaptive agents in customizing financial strategies to investor profiles, integrating RL and NLP for deeper user insight and paving the way for personalized investment tools. Although this proof of concept shows promise, enhancing the agent's framework, expanding the dataset, and integrating additional user inputs are crucial next steps to advance the goal of developing bespoke financial decision-making systems.

Bibliography

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[2] pouyaardehkhani. FinancialAnalysis, 2022. URL `https://github.com/pouyaardehkhani/FinancialAnalysis`. Inspiration for Data Analysis and Sentiment Analysis NLP; accessed Mar-2024.

[3] Dr. Paolo Barucca Prof. Tomaso Aste and T.A. Antionio Briola. Rl agent notebook, 2023.

[4] Prof. Tomaso Aste, Dr. Paolo Barucca, and T.A. Antionio Briola. Rl course slides, 2023. URL `https://moodle.ucl.ac.uk/mod/resource/view.php?id=5960505`.

[5] Vincent Tan. Fine-tuning pretrained NLP models with Huggingface's Trainer, 2019. URL `https://towardsdatascience.com/fine-tuning-pretrained-nlp-models-with-huggingfaces-trainer-6326a4456e7b`. Inspiration for Data Analysis; accessed Mar-2024.

[6] theartificialguy. Multi class classification tf bert, 2021. URL `https://github.com/theartificialguy/NLP-with-Deep-Learning/tree/master/BERT/Multi-Class%20classification%20TF-BERT`. Source for understanding BERT fine-tuning; accessed Feb-2024.

[7] Edward Lu. Q-Trader. URL `https://github.com/edwardhdlu/q-trader`. Inspiration for Reward Function and Agent; accessed Feb-2024.