Spotlight News Driven Quantitative Trading Based on Trajectory Optimization

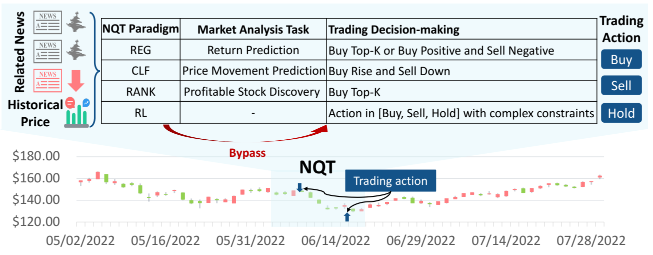
Mengyuan Yang[1], **Mengying Zhu**[1*], **Qianqiao Liang**[2], **Xiaolin Zheng**[1], **MengHan Wang**[3]
[1]Zhejiang University, Hangzhou, China
[2]MYbank, Ant Group, Hangzhou, China
[3]eBay Inc., Shanghai, China
{yangmy412, mengyingzhu, xlzheng, wangmengh}@zju.edu.cn, liangqianqiao.lqq@mybank.cn

Abstract

News-driven quantitative trading (NQT) has been popularly studied in recent years. Most existing NQT methods are performed in a two-step paradigm, i.e., first analyzing markets by a financial prediction task and then making trading decisions, which is doomed to failure due to the nearly futile prediction task. To bypass the financial prediction task, in this paper, we focus on reinforcement learning (RL) based NQT paradigm, which leverages news to make profitable trading decisions directly. In this paper, we propose a novel NQT framework `SpotlightTrader` based on decision trajectory optimization, which can effectively stitch together a continuous and flexible sequence of trading decisions to maximize profits. In addition, we enhance this framework by constructing a spotlight-driven state trajectory that obeys a stochastic process with irregular abrupt jumps caused by spotlight news. Furthermore, in order to adapt to non-stationary financial markets, we propose an effective training pipeline for this framework, which blends offline pretraining with online finetuning to balance exploration and exploitation effectively during online tradings. Extensive experiments on three real-world datasets demonstrate our proposed model's superiority over the state-of-the-art NQT methods.

1 Introduction

News-driven quantitative trading (NQT) has attracted great attention in recent years, which leverages news to capture the influence over market dynamics for making profitable quantitative trading (QT) decisions. Most existing NQT methods perform trading in a two-step paradigm: they first leverage news to analyze the market dynamics with some auxiliary financial prediction task, e.g., price regression [Schumaker and Chen, 2009], movement classification [Du and Tanaka-Ishii, 2020], and stock ranking [Sawhney *et al.*, 2021b]; and then make trading decisions based on their analyze. The trading performance of the two-step paradigm methods often relies on the predictive performance, however, which



NQT Paradigm	Market Analysis Task	Trading Decision-making
REG	Return Prediction	Buy Top-K or Buy Positive and Sell Negative
CLF	Price Movement Prediction	Buy Rise and Sell Down
RANK	Profitable Stock Discovery	Buy Top-K
RL	-	Action in [Buy, Sell, Hold] with complex constraints

Figure 1: Different paradigms of NQT.

may hardly be accurate enough in practice [Millea, 2021; An *et al.*, 2022]. In this paper, we focus on a more practical NQT paradigm, i.e., reinforcement learning (RL) based NQT paradigm, which bypasses the financial prediction and optimizes trading decisions directly, as shown in Figure 1.

The RL-based NQT paradigm models the complex sequential decision-making processes from the experience of interaction with the market to maximize overall profit directly [Ye *et al.*, 2020; Sawhney *et al.*, 2021c]. Although existing RL-based NQT methods produce remarkable results, they focus on learning effective news representations to enhance the market state rather than improving the learnability of RL to generate a series of coherent and profitable trading actions under the non-stationary financial markets, making them inapplicable in the online trading environment. Compared to the aforementioned approaches, we focus on addressing these prevalent issues in the RL-based NQT paradigm, which corresponds to the following three questions.

(**Q1**) *How can we construct an effective RL-based NQT method to make continuous and flexible trading actions?*

(**Q2**) *How can we design an efficient mechanism that can leverage genuinely valuable information from a large amount of chaotic news to guide trading actions?*

(**Q3**) *How can we adapt the RL-based NQT method to the non-stationary online trading environment?*

Keeping this in mind, our works in this paper are all around answering the above three questions.

(A1) Novel RL-based NQT Framework. In quantitative practice, a multi-step profitable trading strategy is often a flexible action combination with several holding periods, e.g., short-term, medium-term, or long-term hold between consecutive buy and sell actions. Forming such a strategy re-

*Corresponding Author

quires ensuring trading action continuity, i.e., shifting trading actions frequently are not allowed to break the consistency of judgments and objectives of previous actions. To ensure action continuity, existing RL-based QT methods [Liu *et al.*, 2020] introduce behavior cloning to learn from expert experience, which, however, may not be able to handle situations beyond the scope of the domain knowledge, especially in non-stationary markets. In addition, it is necessary for the multi-step profitable trading strategy to comprehensively consider the actions in the long history, so as to flexibly adjust the actions in different periods and achieve the long-term goal. Recently, in the RL community, a trajectory optimization framework originating from Decision Transformer [Chen *et al.*, 2021], treats the RL problem as a sequence modeling problem under conditional behavior cloning, and applies transformer architecture to model a decision trajectory. Such a framework learns offline decision trajectories at a cheap knowledge cost and releases the powerful long-sequence modeling capability of the transformer architecture, which inspires our work. Therefore, we propose an RL-based NQT method named `SpotlightTrader` inheriting the trajectory optimization framework, which models the trading decision trajectory and captures the multi-scale intrinsic relations between trading actions, market states, and rewards to form a continuous and flexible sequence of trading actions.

(A2) Novel spotlight-driven state trajectory modeling. For NQT, a common approach to leverage news is fusing the information from all news no matter whether or not the news is influential to stocks [Ye *et al.*, 2020]. In reality, only a fraction of the massive and chaotic news stream holds the potential to influence stock trends [Sawhney *et al.*, 2021c; Sawhney *et al.*, 2021a], which we term as *spotlight news*. Considering all news equally not only dilutes the influence of spotlight news but also introduces noise to fool trading actions. Meanwhile, the occurrence of spotlight news will fleetingly and unanticipated break the stock trends that originally has the characteristics of random walk, and cause some irregular jumps in stock trends (cf. Figure 3). For example, once the spotlight news occurs, keen speculators will seize the profitable opportunity and quickly enter the market, resulting in abrupt jumps in the stock trend. Inspired by the above analysis, we innovatively propose a spotlight-driven state trajectory module, which first filters spotlight news by a sparse attention mechanism and then models a state trajectory following a stochastic process with irregular abrupt jumps caused by spotlight news to guide the trading actions.

(A3) Effective training pipeline for online adaption. The online RL-based models improve the capability of decision-making by interacting with the environment, which may not be suitable for trading, a scenario where online trial and error are expensive. By contrast, the offline RL-based models may be more practical [Levine *et al.*, 2020], which leverages previously collected offline datasets to learn optimal policies without accessing the real market. However, the non-stationary markets face the severe *distribution shift problem* [Sun *et al.*, 2021], which may render previously effective strategies ineffective if only exploiting historical data without any new exploration. Therefore, in order to adapt to the online environment, we propose a novel training pipeline, which blends offline pretraining for exploitation with online finetuning for exploration to enhance `SpotlightTrader`. In our training pipeline, we customize effective components, e.g., trading rollout mechanism, trajectory-level replay buffer, prioritized sampling policy, and hindsight return recalculation, to efficiently generate, store, and sample data in a trajectory-level way. The training pipeline prolongs `SpotlightTrader`'s lifelong experience, i.e., new rollout decision trajectory immediately becomes part of the growing training set, to further improve our model's behavior in a continual online fashion.

In conclusion, we have proposed a novel RL-based NQT framework for making profitable trading decisions, followed by an effective spotlight-driven state trajectory modeling to enhance our proposed framework and an effective training pipeline for adapting the framework to online environments. *To the best of our knowledge, this is the first work to plug the RL-based NQT model into the trajectory optimization framework.* To verify the effectiveness of `SpotlightTrader`, we have conducted comprehensive experimental evaluations on three real-world datasets to demonstrate our model's superiority over the state-of-the-art NQT methods in terms of profitability and drawdown risk control.

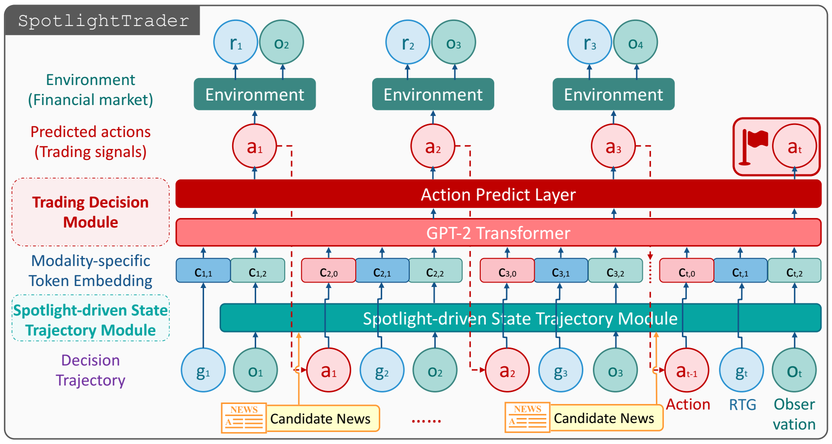
2 Related Work

2.1 News-driven Quantitative Trading

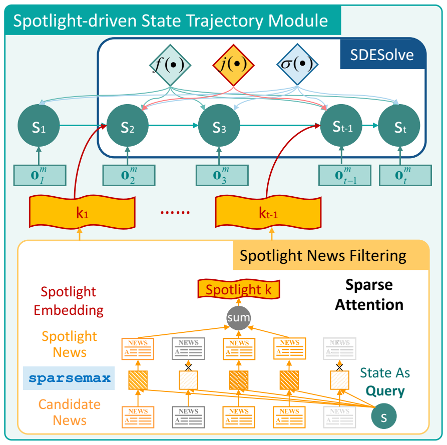
NQT has attracted extensive research focus from the AI community, which incorporates natural language processing (NLP) techniques to translate raw unstructured textual data into meaningful insights for supporting NQT model. Early works [Ding *et al.*, 2014; Hu *et al.*, 2018] formulate NQT as a text classification problem by directly predicting the rise or fall of stock prices based on the extracted features, which hardly considers how to execute the trading decision. In recent years, sentiment-based and event-based NQT methods model the impact of news on trading. Sentiment-based NQT methods [Nan *et al.*, 2022; Chen and Huang, 2021] regard the news articles' sentiments as the trading signal, which would be affected by the author's personal standpoints or writing styles, thus too subjective to assist trading decision making. Event-based NQT methods [Wu, 2020; Zhou *et al.*, 2021] extract events from news articles and then regard events as the trading signal, which heavily relies on the knowledge extracted from specific events. Nevertheless, their works are orthogonal to our work, because instead of focusing on how to model complex news structure or content, we pay more attention to how news, as an important type of market observation, participates in and guides trading.

2.2 RL based Quantitative Trading

In recent years, RL has gained huge attention in the field of quantitative trading (QT) because of its superiority in complex sequential decision making. RL-based QT methods can be mainly categorized into two kinds in terms of the model design target. The first kind of method focus on designing various formulation of MDP to adapt to complex practical trading rules, e.g., reducing transaction cost [Zhang *et al.*, 2022], refined trading operations [Wang *et al.*, 2021], and


SpotlightTrader
Environment (Financial market)
Predicted actions (Trading signals)
Trading Decision Module
Modality-specific Token Embedding
Spotlight-driven State Trajectory Module
Decision Trajectory
Environment
Action Predict Layer
GPT-2 Transformer
Spotlight-driven State Trajectory Module
Candidate News
Action
RTG
Obser vation

(a) The architecture of SpotlightTrader.



(b) Spotlight-driven state trajectory module.

Figure 2: Overview of the proposed SpotlightTrader.

allowing short [Asodekar *et al.*, 2022], and then directly apply classical RL model, e.g., DPG-based [Jiang *et al.*, 2017], DDPG-based [Sawhney *et al.*, 2021c], PPO-based [Yang *et al.*, 2020] method, which spends little effort in modeling the complex relationship between financial environment and trading actions. The second kind of method spends a lot of effort on modeling complex states, i.e., extracting effective feature representations from financial environments by applying powerful sequential models, under the traditional RL framework [Xu *et al.*, 2021; Sawhney *et al.*, 2021c]. However, without careful consideration of the optimization of the RL model itself, these methods suffer from the instability issue [Hu *et al.*, 2022] empirically when directly applying the modeled state to the decision process. Besides, Liu et al. [2020] formulate the QT process as a POMDP and learn expert behaviors based on behavioral cloning model, which is affected by the quality of the expert strategy to be imitated.

3 Problem Formulation

3.1 NQT: News-driven Quantitative Trading

NQT refers to the process of constantly buying, holding, or selling N stocks during T trading rounds for profit by analyzing recent historical price features and news information. At each round t, we denote $\mathbf{p}^c_t = \left[p^c_{t,1}, , \cdots, p^c_{t,N}\right]$ as the close price of all stocks; $\mathbf{h}_t \in \mathbb{R}^N$ as holding amount vector of all stocks; $b_t \in \mathbb{R}_+$ as account balance; and r_t as account profit that corresponds to reward in the RL term.

Initially, before actual trading, we set b_0 to be the initial fund available for trading. At each trading round, a trading agent looks back at the stock price features and the stock-related news of the past period for making trading decisions, aiming to maximize the cumulative account profits $\sum_{t=1}^{T} r_t$ at the end of the trading period.

3.2 POMDP for News-driven Quantitative Trading

According to the NQT process and the fact that the states of a financial market can only be observed partially, it is suitable to model the whole NQT process as a partially observable Markov Decision Process (POMDP). At each trading round t, a trading agent leverages a trading policy $\pi_\theta(\cdot)$ to take an action $\mathbf{a}_t \in \mathcal{A}$ in a particular market state $\mathbf{s}_t \in \mathcal{S}$, which results in reward r_t and the transition of state changes to state $\mathbf{s}_{t+1}$. However, only part of the state, i.e., observation $\mathbf{o}_t \in \mathcal{O}$, can be obtained by the agent. Hence, the key components of the NQT problem formulating as a POMDP are as follows:

Observation. At a round t, $\mathbf{o}_t = [\mathbf{o}^a_t, \mathbf{o}^m_t]$ is an account-market observation. The account observation $\mathbf{o}^a_t$ comprises the account balance b_t and the amount of holdings $\mathbf{h}_t$ at round t. The market observation $\mathbf{o}^m_t$ comprises the Opening-High-Low-Closing (OHLC) price features $\mathbf{p}^{ohlc}_{t-1} = \left[\mathbf{p}^o_{t-1}, \mathbf{p}^h_{t-1}, \mathbf{p}^l_{t-1}, \mathbf{p}^c_{t-1}\right]$ of all stocks and stock-relevant news embeddings $\mathbf{n}_{t,1:n^{\text{news}}_t}$ in $[t-1, t)$ of n^{news}_t pieces of news.

Trading actions. At a round t, $\mathbf{a}_t \in [-1, 1]^N$ is an action vector on all stocks. We re-scale this action using a constrain $K_{\max}$ as described in [Kabbani and Duman, 2022], which represents the maximum allocation (buy/sell shares) and we denote the re-scaled action as $\hat{a}_{t,i}$. The available trading action $\hat{a}_{t,i}$ of each stock i includes selling, buying, and holding, which results in decreasing, increasing, and no change of the holdings h, respectively, as follows: (1) *Buying*: when $\hat{a}_{t,i} \in [-b_t/p^c_{t,i}, -1]$, buy $\hat{a}^i_t$ shares, and it leads to $h_{t+1,i} = h_{t,i} + \hat{a}_{t,i}$; (2) *Selling*: when $\hat{a}_{t,i} \in [1, h_{t,i}]$, sell $\hat{a}_{t,i}$ shares, and it leads to $h_{t+1,i} = h_{t,i} + \hat{a}_{t,i}$; (3) *Holding*: when $\hat{a}_{t,i} \in (-1, 1)$ shares, hold the shares, and it leads to no change in h^i_t. It should be noted that all bought stocks should not result in a negative balance on the portfolio value. That is, without loss of generality, we assume that selling orders are made on the first d_1 stocks and the buying orders are made on the last d_2 ones, and that a_t should satisfy $\mathbf{p}^c_{t,[1:d_1]}{}^\top \hat{\mathbf{a}}_{t,[1:d_1]} + b_t + \mathbf{p}^c_{t,[N-d_2:N]}{}^\top \hat{\mathbf{a}}_{t,[N-d_2:N]} \geq 0$. The remaining balance is updated as $b_{t+1} = b_t + \mathbf{p}^c_t{}^\top \hat{\mathbf{a}}_t$.

Reward. We define the reward r_t as the change in the value when an action $\mathbf{a}_t$ is taken as:

$$r_t = \left(b_{t+1} + \mathbf{p}^c_{t+1}{}^\top \mathbf{h}_{t+1}\right) - \left(b_t + \mathbf{p}^c_t{}^\top \mathbf{h}_t\right) - c_t \quad (1)$$

where c_t denotes the transaction costs incurred at round t.

Return-to-go (RTG). The returns-to-go refers to the future desired returns, which is an important component of the trajectory optimization framework. We calculate the returns-to-go by $g_t = \sum_{t'=t}^{T} r_{t'}$. In order to generate actions based on future desired returns, we model the returns-to-go instead of directly modeling the rewards.

4 Method

The architecture of `SpotlightTrader` is presented in Figure 2(a). In this section, we first define the decision trajectory, i.e., a sequence of return-to-go, observation, and action, which is a basic target in decision autoregressive training and generation (**§4.1**). We then present the spotlight-driven state trajectory (`SST`) module, which recovers the state trajectory from the partial observation trajectory (**§4.2**). With the state trajectory, we present how to make trading decisions in the trading decision (`TD`) module (**§4.3**). After that, to adapt to an online trading environment, we present a pretrain-finetune pipeline (**§4.4**). Finally, we present the model optimization for model training and finetuning (**§4.5**).

4.1 Decision Trajectory

In order to keep the continuity of trading actions, `SpotlightTrader` inherits Decision Transformer [Chen *et al.*, 2021] to learn the policy under the conditional behavior cloning framework. Here, we term a decision trajectory $\boldsymbol{\tau}$ as a sequence of (RTG, observation, action) tuples collected from any period of consecutive trading rounds, which are amenable to autoregressive training and generation:

$$\boldsymbol{\tau} = (g_1, \mathbf{o}_1, \mathbf{a}_1, ..., g_L, \mathbf{o}_L, \mathbf{a}_L), \tag{2}$$

where L is the length of the decision trajectory.

We introduce the collection methods of decision trajectories in the offline training phase and online trading phase. In the training phase, we collect expert decision trajectories from existing strategies or replayed trajectories generated via the partially-trained `SpotlightTrader` to form trajectory-wise offline data $\mathcal{J}_{\text{train}} = \{\boldsymbol{\tau}^i\}_{i=1}^{|\mathcal{J}_{\text{train}}|}$, where $\boldsymbol{\tau}^i$ is a decision trajectory with L consecutive (RTG, observation, action) tuples. In the trading phase, we rollout trading actions to generate a brand new trajectory with every L trading round, which will be elaborated in **§4.4**.

4.2 `SST`: Spotlight-driven State Trajectory Module

To precisely characterize state trajectories, `SST` models the dynamics of state trajectory, which consists of the stochastic observation-controlled continuous state trajectory with perturbation by the inherent spotlight-driven jump as shown in Figure 3. With the jump driven by spotlight news, we term the state trajectory as a spotlight-driven state trajectory.

Dynamics of state trajectory. In `SST`, each stock has its own spotlight-driven state trajectory, which will be merged together as the final state of the trading decision. `SST` model spotlight-driven state trajectory as a hybrid system including a stochastic process with jumps, which extends the neural ordinary differential equations (Neural ODEs) [Chen *et al.*, 2018] with two terms: (1) a stochastic latent dynamics term

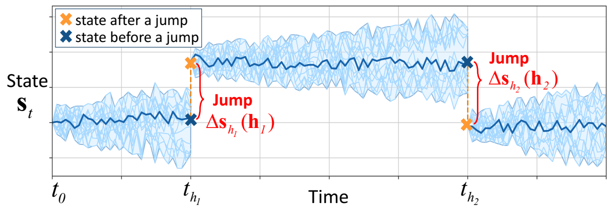


Figure 3: An illustration of the spotlight-driven state trajectory.

that models the stochastic of observation $\mathbf{o}^m$ and (2) a spotlight jump term that models the discrete spotlight news.

For each stock's spotlight-driven state trajectory, a Neural ODE defines a continuous-time transformation of variables, and we specify the transformation of states at round t by a stochastic differential equation (SDE) as follows:

$$d\mathbf{s}_t = \underbrace{f_\psi(t, \mathbf{s}_t, \mathbf{o}_t^m) \cdot dt + \sigma_\phi(t, \mathbf{s}_t) \cdot dW_t}_{\text{stochastic latent dynamics with observation controlling}} + \underbrace{\omega(t) j_\varphi(t, \mathbf{s}_t, \mathbf{k}_t) dt}_{\text{spotlight-driven jump}}, \tag{3}$$

where f_ψ and σ_ϕ are the drift and diffusion functions, respectively, and j_φ is the jump function. To overcome the discontinuity of the spotlight jumps, we set an indicator function $\omega(t)$, where $\omega(t) = \mathbf{1}$ when the spotlight jump exists at t.

Starting from an initial state $\mathbf{s}_0$, the transformed state at any time t is given by integrating an SDE forward by time:

$$\mathbf{s}_t = \mathbf{s}_0 + \int_{t'=0}^{t} \frac{d\mathbf{s}_{t'}}{dt'} dt'. \tag{4}$$

We elaborate the neural networks in $d\mathbf{s}_t$, which defines the stochastic latent dynamics and spotlight jump as follows.

Stochastic latent dynamics. We elaborate on the two terms in stochastic latent dynamics, i.e., dynamics drift function and dynamics diffusion function as follows.

Dynamics drift function $f_\psi(t, \mathbf{s}_t, \mathbf{o}_t^m)$ model the deterministic of stochastic process. Given that the observation trajectory reflects part of the state trajectory, it is natural to learn the drift function based on the observation trajectory. We parameterize the internal state dynamics based on observation $\mathbf{o}_t^m$ by a multi-layer perceptron (MLP). This constrains the internal state dynamics to the observation space and improves the stability of the state trajectory.

Dynamics diffusion function $\sigma_\phi(t, \mathbf{s}_t)$ is necessary to incorporate stochasticity into the stochastic latent dynamics, which we parameterize by an additional MLP.

Spotlight jump. The spotlight jump is modeled by first filtering the spotlight news from the noisy news stream and then applying it as the control point of the jump function.

Firstly, we introduce the spotlight news filtering mechanism. Here, we develop a novel spotlight news filtering layer based on a sparse attention mechanism [Martins and Astudillo, 2016], which is beneficial for focusing on the spotlight news driving market state changes. As shown in Figure 2(b), at each round t, we obtain a set of candidate news $\mathbf{n}_{t,1:n_t^{\text{news}}}$, in which each piece of news i is embed by BERT [Kenton and Toutanova, 2019] into a news embedding $\mathbf{n}_{t,i}$. The sparse attention mechanism is adopted to filter spotlight news and aggregate the representations of all spotlight news to obtain one spotlight representation $\mathbf{k}_t$. Specifically, we first generate query $\mathbf{Q}_t$, keys $\mathbf{K}_t$, and values $\mathbf{V}_t$ from the state and candidate news representations as:

$$\mathbf{Q}_t = \mathtt{Linear}_Q(\mathbf{s}_t), \mathbf{K}_t = \mathtt{Linear}_K(\mathbf{n}_{t,1:n_t^{\text{news}}}), \\ \mathbf{V}_t = \mathtt{Linear}_V(\mathbf{n}_{t,1:n_t^{\text{news}}}). \tag{5}$$

Then, we leverage the query controlled by the current state to guide the spotlight representation learning as follows:

$$\mathbf{k}_t = \mathtt{sparsemax}(\frac{\mathbf{Q}_t \cdot \mathbf{K}_t^\top}{\sqrt{d}})\mathbf{V}_t, \tag{6}$$

where the $\mathtt{sparsemax}(\cdot)$ function retains most of the important properties of $\mathtt{softmax}(\cdot)$ function and automatically prune the large volume of news.

Secondly, we introduce the spotlight jump function $j_\varphi(t, \mathbf{s}_t, \mathbf{k}_t)$. The spotlight news introduces a jump $\Delta\mathbf{s}_t$ to the spotlight-driven state trajectory. The jump is parameterized by an MLP that takes the spotlight representation $\mathbf{k}_t$ and internal state $\mathbf{s}_t$ as input. Our architecture also assumes that the spotlight representation does not directly interrupt the internal state.

Final state embedding. For final state embedding at round t, we first apply a CNN layer for fusing all stocks' state embeddings learned by `SST` to obtain the spotlight-driven state $\mathbf{s}_t$, then concatenate the trading-account state $\mathbf{o}_t^a$, to form an overall state representation $\mathbf{s}_t^{\text{final}} = [\mathbf{s}_t, \mathbf{o}_t^a]$.

4.3 `TD`: Trading Decision Module

In our POMDP setting for NQT, the trading agent needs to generate a trading action $\mathbf{a}_t$ at each round t. Similar to Decision Transformer, we feed a decision trajectory into GPT-2 Transformer [Radford *et al.*, 2019], which applies a causal mask to enforce the autoregressive structure of decision trajectory. To obtain token embeddings, we embed each element $\mathbf{a}_i$, g_i, $\mathbf{s}_i^{\text{final}}$ in the decision trajectory by a linear layer. Additionally, a position embedding for each trading round is learned and added to each token, as one timestep corresponds to three tokens. The token embeddings are then processed by a GPT-2 Transformer, which predicts the latest action token embedding via autoregressive modeling. Finally, we added an action prediction layer, followed by a `tanh` activation function to make the latest action fall within the specified range, which outputs actions to buy, hold or sell the shares of each stock. With the latest action from the `TD`, `SpotlightTrader` makes a trading decision and obtains the reward according to Eq. (1).

4.4 Pretain-Finetune Pipeline

RL policies trained on purely offline datasets are typically sub-optimal because the offline trajectories may cover only a limited part of the state space under a non-stationary financial market. Therefore, it is in high demand to finetune the pretrained RL policy based on the newly collected trajectories during online trading. The key property of such pretrain-finetune pipeline is to balance the exploration-exploitation trade-off, so we devise a sample-efficient online finetune mechanism, which takes exploration into consideration and includes four key components introduced as follows.

Algorithm 1: `SpotlightTrader`'s pipline

```
Input: Trajectories in training data $\mathcal{J}_{\text{train}}$, trading rounds $T$,
       exploration RTG $g_{\text{online}}$, trajectory length $L$, replay
       buffer size $K$, mini-batch size $B$.
Output: Trading actions $\mathbf{a}_{1:T}$.
/* Pretrain stage                                          */
1  while not converged do
2      Sample $B$ trajectories $\boldsymbol{\tau}^{i_1,\dots,i_B}$ out of $\mathcal{J}_{\text{train}}$.
3      Update the policy $\pi_\theta$ by $\boldsymbol{\tau}^{i_1,\dots,i_B}$.
/* Finetune stage                                          */
4  Initialize trajectory-level replay buffer: $\mathcal{J}_{\text{replay}} \leftarrow$ top $K$
     trajectories in $\mathcal{J}_{\text{train}}$.
5  for $t = 1, L+1, 2L+1, ..., T$ do
6      Trading rollout a new trajectory $\boldsymbol{\tau}_t$.
7      Replace the oldest trajectory by $\boldsymbol{\tau}_t$ in $\mathcal{J}_{\text{replay}}$.
8      PS: Compute sampling probability $P(\boldsymbol{\tau})$ and sample $B$
         trajectories out of $\mathcal{J}_{\text{replay}}$ by $P(\boldsymbol{\tau})$.
9      for each sampled trajectory $\boldsymbol{\tau}$ do
10         HRR: Recalculate the RTG sequence $\mathbf{g}_{1:L}$ by
             $g_i = \sum_{j=i}^{L} r_j, 1 \le i \le L$ and assemble the new
             trajectory $\hat{\boldsymbol{\tau}}$ with the hindsight RTG sequence.
11         Update the policy $\pi$ by $\hat{\boldsymbol{\tau}}$.
```

Trading rollout. Firstly, we need to collect online trajectories during actual trading, which depends on the trading rollout of a learned policy. Specifically, given a pre-defined desired RTG g_t and an initial observation $\mathbf{o}_t$ at round t, `TD` generates the action $\mathbf{a}_t = \pi_\theta(\mathbf{o}_t, g_t)$ for trading. Then, we obtain a reward r_t and the next observation $\mathbf{o}_{t+1}$, which gives us the next RTG as $g_{t+1} = g_t - r_t$. As before, `TD` generates $\mathbf{a}_{t+1}$ based on $\mathbf{o}_{t:t+1}$, $g_{t:t+1}$, and $\mathbf{a}_t$. This process is repeated to generate an online trajectory until a specified length of the trajectory is reached.

Trajectory-level replay buffer (TRB). Secondly, we need a replay buffer to store the trajectories that are used for online finetuning. Initially, we set a prioritized replay buffer $\mathcal{J}_{\text{replay}}$, which is filled with trajectory-level data with the highest returns from offline data, see Line 4 in Algorithm 1. Every time we completely rollout a trajectory with the current policy, we refresh the replay buffer in a first-in-first-out manner periodically. Afterward, we update the policy and rollout again.

Prioritized sampling (PS). Thirdly, we need to select the data from TRB to form the mini-batch used for online finetuning. Specifically, we enable each mini-batch to consist of decision trajectories from offline data and rollout data, which are sampled by a prioritized sampling mechanism [Schaul *et al.*, 2016], see Line 8 in Algorithm 1. The prioritized sampling mechanism encourages replaying trajectory with higher expected learning progress more frequently. For each trajectory $\boldsymbol{\tau}^i \in \mathcal{J}_{\text{replay}}$, we calculate it's trajectory priority as $p_{\boldsymbol{\tau}^i} = \mathcal{L}_{\boldsymbol{\tau}^i} + \epsilon_D$, where $\mathcal{L}_{\boldsymbol{\tau}^i}$ is loss calculated by Eq. (7), and ϵ_D is a positive constant for trajectory from offline data to increase the probability of getting sampled. The probability $P(\boldsymbol{\tau}^i)$ of the trajectory $\boldsymbol{\tau}^i$ is proportional to its priority $p_{\boldsymbol{\tau}^i}$, namely, $P(\boldsymbol{\tau}^i) = \frac{p_{\boldsymbol{\tau}^i}}{\sum_j p_{\boldsymbol{\tau}^j}}$. To consider changes in the sample distribution, which may affect model optimization, trajectories for the network updates are weighted with importance sampling weights, i.e., $w_{\boldsymbol{\tau}^i} = (\frac{1}{K} \cdot \frac{1}{P(\boldsymbol{\tau}^i)})^\beta$, where β is an annealing parameter [Schaul *et al.*, 2016]. In this way, the prioritized replay buffer controls the ratio of data between the offline data and new rolling data.

Hindsight return recalculation (HRR). The return achieved during a policy rollout and the induced RTG can differ from the intended RTG, which harms the model learning. To make each sampled trajectory $\boldsymbol{\tau}$ more similar to the expert trajectories and thus boost learning, we apply HRR to generate new trajectories $\hat{\boldsymbol{\tau}}$ with the achieved RTG, as opposed to the intended RTG with a manually set initial RTG g_{online}, see Line 10-11 of Algorithm 1.

4.5 Model Optimization

In this subsection, we aim to train our model to make profitable trading decisions, a prime purpose of the model, which is equivalent to maximizing the high-return stochastic optimal policy distribution $\pi^*_\theta(\mathbf{a}_t|g_{\leq t}, \mathbf{o}_{\leq t}, \mathbf{a}_{<t})$. Further, in order to equip the ability of `SpotlightTrader` to balance the exploration-exploitation trade-off in online trading, we apply the negative log-likelihood (NLL) loss with a policy entropy [Zheng *et al.*, 2022]. Thus, the loss function based on the decision trajectories' distribution $\mathcal{T}$ is calculated by:

$$\mathcal{L}_\tau = \frac{1}{L}\mathbb{E}_{(\mathbf{a}^\tau,\mathbf{o}^\tau,\mathbf{g}^\tau)\sim\mathcal{T}}[-\sum_{l=1}^{L}(\log \pi_\theta(\mathbf{a}^\tau_l|\mathbf{g}^\tau_{\leq l}, \mathbf{o}^\tau_{\leq l}, \mathbf{a}^\tau_{<l}) + \lambda H^{\mathcal{T}}_\theta(\mathbf{a}^\tau_{<l}|\mathbf{g}^\tau_{\leq l}, \mathbf{o}^\tau_{<l}, \mathbf{a}^\tau_{<l}))], \tag{7}$$

where λ serves the role of temperature variable and $H^{\mathcal{T}}_\theta$ is the policy entropy. Note that in the offline training phase, the temperature $\lambda = 0$. The loss of mini-batch can be calculated by $\mathcal{L} = \sum_{i=1}^{B} w_{\tau^i}\mathcal{L}_{\tau^i}$, where B is the mini-batch size.

5 Experiments

We conduct extensive experiments to answer the following questions:

Q1: How does `SpotlightTrader` perform on profitability and risk over the trading period?

Q2: How effective are constructing state trajectories with different information sources?

Q3: How do the key components contribute to the performance of `SpotlightTrader`?

Q4: How do the major parameters affect the performance and how to choose optimal values?

Q5: How does `SpotlightTrader` makes decision?

5.1 Experimental Settings

Dataset Descriptions. We conduct experiments on three real-world datasets[1]. **Twitter-SP500** is a public dataset consist stocks in S&P 500 index from U.S. market and tweets from Twitter. **THS-CSI300** is our collected China market dataset covering stocks from the CSI 300 Index and news from iFinD. **COVID-SP500** is our collected U.S. stock market dataset covering stocks in the S&P 500 index and news from Aylien during the COVID-19 pandemic period. Following [Jiang *et al.*, 2017; Sawhney *et al.*, 2021c], we preprocess the price data by missing values filling and normalization, and we embed the English and Chinese news text by pre-trained FinBERT [Araci, 2019] and Chinese-FinBERT[2]. We divide each dataset into non-overlapping offline and online datasets, and the statistics of the datasets are presented in Table 1. We elaborate on the generation of trajectory-level offline data and provide the offline dataset in our GitHub repository[3]. The parameter setting and implement details will be presented in a longer version of this paper.

Dataset	Twitter-SP500	THS-CSI300	COVID-SP500
Offline start date	01/01/2014	01/01/2020	01/01/2020
Offline end date	09/30/2015	12/31/2020	04/30/2020
Online start date	10/01/2015	01/01/2021	05/01/2020
Online end date	03/31/2016	06/30/2021	07/31/2020
Online trading rounds	565	118	145
Total number of assets	85	229	488
Total number of news	116,278	51,571	20,542,912
Max news per round	1,815	2,499	587,510
Average news per round	206	143	141,675

Table 1: Datasets descriptions

Baselines. We compare SpolightTrader with nine NQT methods in four groups: (1) *Regression (REG) based method* (i.e., **AZFinText** [Schumaker and Chen, 2009] and **Game** [Ang and Lim, 2022]) predicts asset returns and trades assets that are greater than or equal to 1%; (2) *Classification (CLF) based method* (i.e.,**StockNet** [Xu and Cohen, 2018], **Stock-Emb** [Du and Tanaka-Ishii, 2020], and **HTLSTM** [Sawhney *et al.*, 2021a]) classifies prices trends and trades assets where prices are expected to rise; (3) *Ranking(RAN) based method* (i.e., **FAST** [Sawhney *et al.*, 2021b] and **HISN** [Wang *et al.*, 2022]) ranks assets and trades the most profitable assets; (4) *Reinforcement Learning (RL) based method* (i.e., PG-based **SARL** [Ye *et al.*, 2020] and DDPG-based **Profit** [Sawhney *et al.*, 2021c]) directly makes news-driven trading decisions.

Evaluation Metrics. Following [Sawhney *et al.*, 2021c], we adopt four widely used metrics. Annualized Percentage Yield (APY) and Maximum Drawdown (MDD) measure the returns and risks, respectively. Sharpe Ratio (SR) and Sortino Ratio (StR) measure the trader's risk-adjusted returns.

5.2 Performance Comparison and Analysis

To answer **Q1**, in Table 2, we compare `SpotlightTrader` with methods from four different NQT paradigms.

Result 1: Performance on Profitability. In general, we observe that RL-based methods are typically more profitable, i.e., achieve higher APY values, compared to the two-step methods. This is because the RL trading agents optimize every trading action for profit generation directly, unlike two-step methods, where the trading mechanism is only to select the predicted profitable stocks to trade and the overall profits are not optimized as a reward. These observations validate the necessity of formulating quantitative trading as an

[1]Data is collected from https://github.com/yumoxu/stocknet-dataset, http://www.51ifind.com.cn/, and https://aylien.com/.

[2]https://github.com/valuesimplex/FinBERT

[3]https://github.com/Yangmy412/SpotlightTraderOfflineDataset

		Twitter-SP500				THS-CSI300				COVID-SP500			
Categories	Methods	APY	MDD	SR	StR	APY	MDD	SR	StR	APY	MDD	SR	StR
Market	Index[4]	0.15	0.13	0.90	1.41	0.07	0.15	0.32	0.46	0.57	0.07	2.49	2.78
REG	AZFinText	0.04 ± 0.000	0.09 ± 0.009	0.80 ± 0.000	2.07 ± 0.000	0.04 ± 0.000	0.14 ± 0.000	0.21 ± 0.000	0.32 ± 0.000	0.32 ± 0.000	0.12 ± 0.000	1.07 ± 0.000	1.37 ± 0.000
	Game	0.16 ± 0.007	0.12 ± 0.024	1.06 ± 0.220	1.84 ± 0.449	0.08 ± 0.009	0.23 ± 0.025	0.39 ± 0.023	0.59 ± 0.032	0.66 ± 0.008	0.11 ± 0.000	2.17 ± 0.028	2.83 ± 0.037
CLF	StockNet	0.15 ± 0.001	0.13 ± 0.000	0.94 ± 0.007	1.60 ± 0.012	0.12 ± 0.022	0.15 ± 0.008	0.51 ± 0.250	0.78 ± 0.355	0.72 ± 0.114	0.10 ± 0.005	2.51 ± 0.188	3.19 ± 0.252
	StockEmb	0.16 ± 0.026	0.13 ± 0.011	0.99 ± 0.186	1.71 ± 0.339	0.05 ± 0.026	0.14 ± 0.004	0.26 ± 0.142	0.40 ± 0.225	0.67 ± 0.022	0.12 ± 0.002	2.18 ± 0.082	2.85 ± 0.097
	HTLSTM	0.17 ± 0.027	0.12 ± 0.034	1.01 ± 0.280	1.74 ± 0.559	0.09 ± 0.016	0.15 ± 0.014	0.48 ± 0.091	0.76 ± 0.137	0.69 ± 0.070	0.12 ± 0.010	2.19 ± 0.060	2.95 ± 0.371
RAN	FAST	0.19 ± 0.035	0.09 ± 0.023	1.26 ± 0.228	2.06 ± 0.387	0.17 ± 0.102	0.24 ± 0.045	0.52 ± 0.311	0.78 ± 0.460	0.84 ± 0.146	0.16 ± 0.041	1.93 ± 0.316	2.94 ± 0.402
	HISN	0.17 ± 0.015	0.11 ± 0.015	1.10 ± 0.109	1.77 ± 0.210	0.11 ± 0.012	0.18 ± 0.006	0.47 ± 0.052	0.67 ± 0.076	0.85 ± 0.050	0.08 ± 0.003	3.65 ± 0.261	4.20 ± 0.292
RL	SARL	0.20 ± 0.017	0.10 ± 0.008	1.27 ± 0.113	1.91 ± 0.268	0.19 ± 0.090	0.14 ± 0.010	0.56 ± 0.078 *	0.83 ± 0.111	0.93 ± 0.158	0.07 ± 0.015 *	3.27 ± 0.592	4.63 ± 0.987 *
	Profit	0.21 ± 0.021 *	0.09 ± 0.009 *	1.28 ± 0.185 *	2.21 ± 0.295 *	0.20 ± 0.047 *	0.13 ± 0.008 *	0.54 ± 0.134	0.85 ± 0.207 *	0.97 ± 0.051 *	0.08 ± 0.003	4.04 ± 0.286 *	4.53 ± 0.454
	`Sportlight-Trader`	**0.28 ± 0.038**	**0.08 ± 0.007**	**1.75 ± 0.274**	**2.71 ± 0.374**	**0.32 ± 0.043**	**0.12 ± 0.014**	**0.92 ± 0.102**	**1.32 ± 0.125**	**1.09 ± 0.067**	**0.05 ± 0.001**	**4.65 ± 0.452**	**6.15 ± 0.693**
Improvement[5] (%)		34.380	7.601	37.194	22.656	61.659	8.178	65.102	54.326	12.390	24.874	15.100	32.852
p-value[6]		0.000	0.007	0.000	0.005	0.000	0.008	0.000	0.000	0.000	0.003	0.003	0.001

Table 2: Results of all methods on four metrics (mean ± std, computed across 10 runs).

Information Sources	Twitter-SP500		THS-CSI300		COVID-SP500	
	APY	MDD	APY	MDD	APY	MDD
A (Account)	0.04	0.17	0.03	0.22	0.35	0.10
A+P (Price)	0.16	0.16	0.17	0.16	0.64	0.06
A+N (News)	0.18	0.12	0.11	0.19	0.50	0.10
A+P+N	0.24	0.09	0.27	0.15	1.01	0.07
A+P+SN (Spotlight News)	**0.28**	**0.08**	**0.32**	**0.12**	**1.09**	**0.05**

Table 3: Results of different information sources.

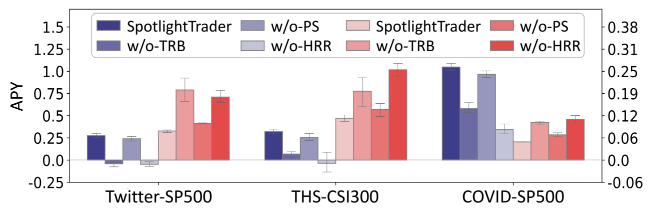


Figure 4: Performance of model variants. Purple- and red-shade show results on APY and MDD, respectively.

RL problem. Furthermore, among the RL-based methods, our method achieves the highest APY values with an average improvement of 36.14% over the best-performing baselines. Our method benefits from not only its novel decision trajectory optimization framework to make flexible long-term and short-term profitable trading decisions, but also its effective spotlight news filtering and modeling mechanisms to enhance the decision trajectory optimization framework.

Result 2: Performance on Risk. From the results in Table 2, we observe that our model achieves the lowest MDD values with an average improvement of 13.55%, indicating its ability to respond to bearish markets. In addition to the superior RL-based trading framework in our model over the two-step strategies, such performance is attributable to our model's novel training pipeline design, which adapts the RL-based framework to online trading environments for avoiding volatile stocks timely during such turbulent environments. Our proposed strategy also achieves the best performance on the SR and StR metric, which indicates its superiority in balancing return and risk.

[4]The index is S&P 500 (^GSPC) in Twitter-SP500 and COVID-SP500 datasets, and CSI 300 (000300.SH) in THS-CSI300.

[5]The improvement over the best-performing baselines.

[6]The improvement is significant based on paired t-test at the significance level of 0.05.

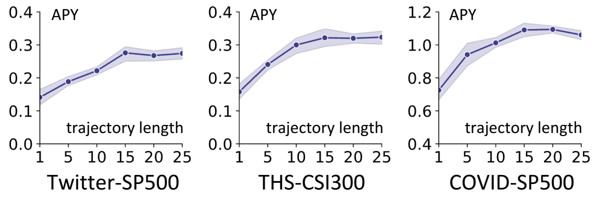


Figure 5: Performance on different trajectory lengths.

5.3 Ablation Study

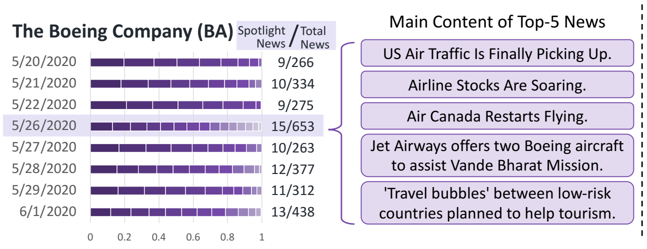
Result 3: Effect of Absorbed Information. To answer **Q2**, we evaluate the effectiveness of different information sources on stock profiling. We report `SpotlightTrader`'s performance under five cases, which cover all possible combinations of the price feature and news information, in Table 3, from which we have two observations. Firstly, significant APY and MDD improvements are noted by blending price features with news information, indicating the contributions of both news and price features. This is mainly because price features and news are indispensable stochastic processes and jumps, respectively, in the state trajectory, which can well support the generation of profitable trading actions. Secondly, we find that considering all pieces of news from the news stream unexpectedly hurts the model's performance, which demonstrates that the spotlight news filtering mechanism is very necessary to reduce the noise from irrelevant news.

Result 4: Effect of Model Variants. To answer **Q3**, we conduct an ablation analysis on the adaptability of our model to online environments. We build three variants of `SpotlightTrader`, including w/o-TRB, w/o-PS, and w/o-HRR, which corresponds to our model without TRB, PS, and HRR, respectively. From Figure 4, we can conclude that interactively finetuning using online trading decision trajectories is crucial to our model, and the priority sampling mechanism clearly benefits the model's adaptability. Besides, failure to perform hindsight return recalculation will confuse the model's estimation of the cumulative return of the trajectory, making it impossible to seek the optimal trajectory.

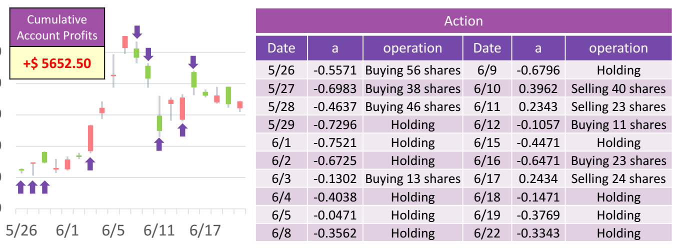
5.4 Parameter Analysis: Probing Sensitivity

To answer **Q4**, we conduct sensitivity experiments on two important parameters, i.e., trajectory length and online RTG.

Result 5: Effect of trajectory length. The trajectory length L determines how long the agent's current action de-



(a) Sparse-attention weights and corresponding Spotlight News.



Action					
Date	a	operation	Date	a	operation
5/26	-0.5571	Buying 56 shares	6/9	-0.6796	Holding
5/27	-0.6983	Buying 38 shares	6/10	0.3962	Selling 40 shares
5/28	-0.4637	Buying 46 shares	6/11	0.2343	Selling 23 shares
5/29	-0.7296	Holding	6/12	-0.1057	Buying 11 shares
6/1	-0.7521	Holding	6/15	-0.4471	Holding
6/2	-0.6725	Holding	6/16	-0.6471	Buying 23 shares
6/3	-0.1302	Buying 13 shares	6/17	0.2434	Selling 24 shares
6/4	-0.4038	Holding	6/18	-0.1471	Holding
6/5	-0.0471	Holding	6/19	-0.3769	Holding
6/8	-0.3562	Holding	6/22	-0.3343	Holding

(b) Trading actions from `SpotlightTrader`.

Figure 6: Case study on Boeing: the filtered spotlight news on May 26, 2020 and the corresponding trading actions afterwards.

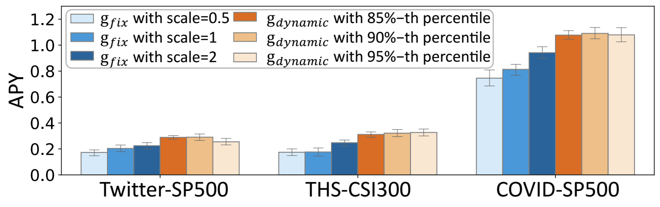


Figure 7: Performance on two online RTG mechanisms.

pends on the past. We depict `SpotlightTrader`'s performance on APY under different trajectory lengths L varying in $\{1, 5, 10, 15, 20, 25\}$ in Figure 5. When the trajectory length is 1, an edge case where each action only depends on the current RTG and state and our POMDP formulation can be reduced to a Markov decision process (MDP), our model performs the worst, indicating that the MDP formulation of NQT is detrimental to trading performance. With the increase in trajectory length, our model achieves better performance, because considering long-term historical actions can contribute to achieving the long-term goal by ensuring trading action continuity, which supports the analysis as those in the main text. In addition, it is worth mentioning that a relatively large trajectory length, i.e., $L = 15$, is enough to gather enough information to make a trading decision, and a too-large trajectory length is unnecessary and oversaturated for modeling.

Result 6: Effect of online RTG. The online RTG g_{online} is used to set g_1 during every trading rollout. Because g_{online} may seriously affect the stitching ability of our model on the trajectory during trading rollout, we discuss two mechanisms for pre-defined g_{online} [Zheng *et al.*, 2022], i.e., fixed and dynamic RTG. Firstly, in the fixed RTG mechanism, we set the RTG g_{online} as the fixed RTG g_{fix}, which is the scale of the maximum RTG obtained in the pretraining stage. The scale varies in $\{0.5, 1, 2\}$. Secondly, in the dynamic RTG mechanism, we adjust the online exploration RTG g_{online} to be a dynamic RTG g_{dynamic}, which is calculated based on the cumulative rewards of the trajectories, i.e., achieved RTG in each trajectory, stored in the replay buffer. Specifically, we set dynamic RTG as the q-th percentile of these non-stationary achieved RTGs. The percentile q varies in $\{85\%, 90\%, 95\%\}$. We report `SpotlightTrader`'s performance on APY under different g_{online} in Figure 7, which shows that the dynamic RTG mechanism achieves higher returns compared to the fixed RTG mechanism, demonstrating the superiority of our model in adapting to non-stationary financial markets.

5.5 Case Study

We conduct an extended analysis in Figure 6 to elucidate on `SpotlightTrader`'s spotlight news filtering result and its practical applicability to real-world quantitative trading for answering **Q5**. Here, we study the U.S. market from May 20, 2020 to June 20, 2020. We visualize the sparse-attention weights of Boeing (BA), which is an aircraft manufacturer, and excerpt the briefing of spotlight news to analyze how our proposed model trade BA on May 26.

Result 7: Analyzing trading actions. In order to explore how our model forms trading actions, we report BA's trading actions for 20 consecutive days from May 26, 2020. On each day, the sparse attention mechanism in our `SST` module filters out a large amount of news and only retains a handful of more influential ones. For example, on May 26, 2020, we found that the top 4 news with the highest weights were all about the loosening of flying bans during COVID-19, which is very likely to have a positive impact on BA, leading to BA's price rises afterward. This case study indicates that `SpotlightTrader` has the ability to filter spotlight news, that is, to find news that can drive price changes to guide transactions. Under the positive influence of the spotlight news, our model chose the Buy signal for 3 consecutive days after May 26, continuously expanding BA's position. Afterward, it chose to hold BA for a period of time, and our model did not sell the position of BA until June 10, 2020, when there was an obvious downward trend. The above continuous and explainable trading behavior indicates our model's powerful modeling ability for action trajectory, which ensures the consistency and coherence of trading behavior.

6 Conclusions and Future Work

In this paper, we study the NQT task and propose a novel NQT framework `SpotlightTrader`, which can release the powerful long-sequence modeling capability of the transformer and effectively stitch together a continuous and flexible sequence of trading decisions. Extensive experiments demonstrate our model's superiority in terms of both profitability and drawdown risk control. In the future, we plan to extend our work into two potential directions, i.e., modeling complex relations between stocks to coordinate stock trading decisions and leveraging more multi-modal information, e.g., earning calls, to capture the market state more completely.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (No. 72192823), the "Ten Thousand Talents Program" of Zhejiang Province for Leading Experts (No. 2021R52001), and the Cooperation Project of MYbank, Ant Group.

References

[An *et al.*, 2022] Bo An, Shuo Sun, and Rundong Wang. Deep reinforcement learning for quantitative trading: Challenges and opportunities. *IEEE Intelligent Systems*, 37(2):23–26, 2022.

[Ang and Lim, 2022] Gary Ang and Ee-Peng Lim. Guided attention multimodal multitask financial forecasting with inter-company relationships and global and local news. In *Proc. of ACL*, pages 6313–6326, 2022.

[Araci, 2019] Dogu Araci. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063*, 2019.

[Asodekar *et al.*, 2022] Eeshaan Asodekar, Arpan Nookala, Sayali Ayre, and Anant V Nimkar. Deep reinforcement learning for automated stock trading: Inclusion of short selling. In *International Symposium on Methodologies for Intelligent Systems*, pages 187–197, 2022.

[Chen and Huang, 2021] Yu-Fu Chen and Szu-Hao Huang. Sentiment-influenced trading system based on multimodal deep reinforcement learning. *Applied Soft Computing*, page 107788, 2021.

[Chen *et al.*, 2018] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proc. of NeurIPS*, pages 6572–6583, 2018.

[Chen *et al.*, 2021] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Proc. of NeurIPS*, pages 15084–15097, 2021.

[Ding *et al.*, 2014] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proc. of EMNLP*, pages 1415–1425, 2014.

[Du and Tanaka-Ishii, 2020] Xin Du and Kumiko Tanaka-Ishii. Stock embeddings acquired from news articles and price history, and an application to portfolio optimization. In *Proc. of ACL*, pages 3353–3363, 2020.

[Hu *et al.*, 2018] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proc. of WSDM*, pages 261–269, 2018.

[Hu *et al.*, 2022] Shengchao Hu, Li Shen, Ya Zhang, Yixin Chen, and Dacheng Tao. On transforming reinforcement learning by transformer: The development trajectory. *arXiv preprint arXiv:2212.14164*, pages 1–8, 2022.

[Jiang *et al.*, 2017] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, pages 1–8, 2017.

[Kabbani and Duman, 2022] Taylan Kabbani and Ekrem Duman. Deep reinforcement learning approach for trading automation in the stock market. *IEEE Access*, pages 93564–93574, 2022.

[Kenton and Toutanova, 2019] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[Levine *et al.*, 2020] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, pages 1–8, 2020.

[Liu *et al.*, 2020] Yang Liu, Qi Liu, Hongke Zhao, Zhen Pan, and Chuanren Liu. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proc. of AAAI*, pages 2128–2135, 2020.

[Martins and Astudillo, 2016] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proc. of ICML*, pages 1614–1623, 2016.

[Millea, 2021] Adrian Millea. Deep reinforcement learning for trading—a critical survey. *Data*, 6(11):119, 2021.

[Nan *et al.*, 2022] Abhishek Nan, Anandh Perumal, and Osmar R Zaiane. Sentiment and knowledge based algorithmic trading with deep reinforcement learning. In *International Conference on Database and Expert Systems Applications*, pages 167–180, 2022.

[Radford *et al.*, 2019] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):1–24, 2019.

[Sawhney *et al.*, 2021a] Ramit Sawhney, Shivam Agarwal, Megh Thakkar, Arnav Wadhwa, and Rajiv Ratn Shah. Hyperbolic online time stream modeling. In *Proc. of SIGIR*, pages 1682–1686, 2021.

[Sawhney *et al.*, 2021b] Ramit Sawhney, Arnav Wadhwa, Shivam Agarwal, and Rajiv Shah. Fast: Financial news and tweet based time aware network for stock trading. In *Proc. of ACL*, pages 2164–2175, 2021.

[Sawhney *et al.*, 2021c] Ramit Sawhney, Arnav Wadhwa, Shivam Agarwal, and Rajiv Shah. Quantitative day trading from natural language using reinforcement learning. In *Proc. of ACL*, pages 4018–4030, 2021.

[Schaul *et al.*, 2016] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *International Conference on Learning Representations*, pages 1–8, 2016.

[Schumaker and Chen, 2009] Robert P Schumaker and Hsinchun Chen. Textual analysis of stock market prediction using breaking financial news: The azfin text system.

ACM Transactions on Information Systems (TOIS), pages 1–19, 2009.

[Sun *et al.*, 2021] Shuo Sun, Rundong Wang, and Bo An. Reinforcement learning for quantitative trading. *arXiv preprint arXiv:2109.13851*, pages 1–8, 2021.

[Wang *et al.*, 2021] Rundong Wang, Hongxin Wei, Bo An, Zhouyan Feng, and Jun Yao. Commission fee is not enough: A hierarchical reinforced framework for portfolio management. In *Proc. of AAAI*, pages 626–633, 2021.

[Wang *et al.*, 2022] Heyuan Wang, Tengjiao Wang, Shun Li, Shijie Guan, Jiayi Zheng, and Wei Chen. Heterogeneous interactive snapshot network for review-enhanced stock profiling and recommendation. In *Proc. of IJCAI*, pages 3962–3969, 2022.

[Wu, 2020] Xianchao Wu. Event-driven learning of systematic behaviours in stock markets. In *Proc. of ACL*, pages 2434–2444, 2020.

[Xu and Cohen, 2018] Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *Proc. of ACL*, pages 1970–1979, 2018.

[Xu *et al.*, 2021] Ke Xu, Yifan Zhang, Deheng Ye, Peilin Zhao, and Mingkui Tan. Relation-aware transformer for portfolio policy learning. In *Proc. of IJCAI*, pages 4647–4653, 2021.

[Yang *et al.*, 2020] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

[Ye *et al.*, 2020] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proc. of AAAI*, pages 1112–1119, 2020.

[Zhang *et al.*, 2022] Yifan Zhang, Peilin Zhao, Qingyao Wu, Bin Li, Junzhou Huang, and Mingkui Tan. Cost-sensitive portfolio selection via deep reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, pages 236–248, 2022.

[Zheng *et al.*, 2022] Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *Proc. of ICML*, pages 27042–27059, 2022.

[Zhou *et al.*, 2021] Zhihan Zhou, Liqian Ma, and Han Liu. Trade the event: Corporate events detection for news-based event-driven trading. In *Proc. of ACL*, pages 2114–2124, 2021.