NATURAL LANGUAGE REINFORCEMENT LEARNING

Xidong Feng[1*†], **Bo Liu**[2*†], **Ziyu Wan**[3†], **Haotian Fu**[4†],
Mengyue Yang[5], **Girish A. Koushik**[6], **Zhiyuan Hu**[2], **Ying Wen**[3], **Jun Wang**[1]
[1]University College London, [2]National University of Singapore,
[3]Shanghai Jiao Tong University, [4]Brown University, [5]University of Bristol, [6]University of Surrey
{fengxidongwh}@gmail.com

ABSTRACT

Reinforcement Learning (RL) mathematically formulates decision-making with Markov Decision Process (MDP). With MDPs, researchers have achieved remarkable breakthroughs across various domains, including games, robotics, and language models. This paper seeks a new possibility, **Natural Language Reinforcement Learning (NLRL)**, by extending traditional MDP to natural language-based representation space. Specifically, NLRL innovatively redefines RL principles, including task objectives, policy, value function, Bellman equation, and policy iteration, into their language counterparts. With recent advancements in Large Language Models (LLMs), NLRL can be practically implemented to achieve RL-like policy and value improvement by either pure prompting or gradient-based training. Experiments over Maze, Breakthrough, and Tic-tac-toe games demonstrate the effectiveness, efficiency, and interpretability of the NLRL framework among diverse use cases.

1 INTRODUCTION

Reinforcement Learning (RL) (Sutton & Barto, 2018) provides a rigorous framework – Markov Decision Process (MDP) for solving general decision-making problems. It transforms the policy learning problem into a mathematical optimization task. While RL has achieved breakthroughs across various domains, several challenges remain. For example, traditional RL algorithms generally lack task-specific prior knowledge, requiring extensive sampling to approximate environment dynamics. RL policy also lacks interpretability. Even in superhuman-performing models like AlphaZero (Silver et al., 2017), strategic reasoning remains elusive, even to professional players. RL training is also unstable (Zheng et al., 2023; Andrychowicz et al., 2020) due to its reliance on scalar rewards as the sole supervision signal. This one-dimensional feedback is particularly limiting in real-world scenarios where richer, multi-modal signals are naturally available, such as textual feedback (Bai et al., 2022; Madaan et al., 2024), visual demonstrations (Bousmalis et al., 2023; Xu et al., 2024).

To tackle these challenges, we seek for a new RL paradigm shift, inspired by language-centric decision-making. Unlike traditional RL, which relies heavily on formalized mathematical modeling, humans can leverage natural language to interpret tasks, devise strategies, and communicate their reasoning. This language-driven approach enables rapid generalization using text-based prior knowledge, enhances interpretability through explicit reasoning, and provides access to rich, informative signals from linguistic data. Thus, natural language represents a largely untapped resource for improving the efficiency, stability, and interoperability of RL systems. The recent success of language-based transformers (Vaswani, 2017) further opens new avenues to integrate language into the RL framework. Trained in a vast corpus of text data, Large Language Models (LLMs) have demonstrated unprecedented proficiency in generating, understanding, and processing language-based information.

Building upon language-centric decision-making and advancement of LLMs, we introduce **Natural Language Reinforcement Learning** (NLRL), a novel RL paradigm that combines RL's mathematical rigor with the representational richness of natural language. In NLRL, core RL compo-

*Equal contribution, correspondence to fengxidongwh@gmail.com
†Core contributors.

nents—such as task objectives, policies, value functions, and the Bellman equation—are reinterpreted as language-based constructs. The medium of natural language largely facilitates the integration of prior knowledge stored in LLMs, and effectively translates decision-making processes into a form that is both intuitive and interpretable. NLRL also provides a systematic solution for leveraging rich textual feedback in sequential decision-making tasks, enabling stable training. Specifically, our key contributions are three-folded:

Framework. NLRL framework enables LLMs to learn language policy and critic purely from environment feedback, requiring no labeled data from humans or advanced models.

Algorithms. Under the NLRL framework, We present three algorithms, to show how LLM critique and planning ability can be boosted through pure prompting (Sec. 4.1), natural language value function training (Sec. 4.2) or natural language actor-critic training (Sec. 4.3).

Experiments. We empirically validate these three algorithm instances in Maze, Breakthrough, and Tic-tac-toe environments (Sec. 5), demonstrating NLRL's interpretability, effectiveness, and superiority over traditional RL.

2 Preliminary of Reinforcement Learning

Reinforcement Learning models the decision-making problem as a Markov Decision Process (MDP), defined by the state space $\mathcal{S}$, action space $\mathcal{A}$, probabilistic transition function $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$, discount factor $\gamma \in [0, 1)$ and reward function $r : \mathcal{S} \times \mathcal{A} \to [-R_{\max}, R_{\max}]$. The goal of RL aims to learn a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which measures the action a's probability given the state s: $\pi(a|s) = \Pr(A_t = a \mid S_t = s)$. In decision-making tasks, the optimal policy tends to maximize the expected discounted cumulative reward: $\pi^*(a|s) = \arg\max_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(\mathrm{s}_t, \mathrm{a}_t)\right]$. The state-action and state value functions are two key concepts that evaluate states or state-action pairs by measuring the cumulative reward starting from them: $Q_\pi(s_t, a_t) = \mathbb{E}_{(s,a)_{t+1:\infty} \sim P_\pi}\left[\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i) \mid s_t, a_t\right]$, $V_\pi(s_t) = \mathbb{E}_{a_t,(s,a)_{t+1:\infty} \sim P_\pi}\left[\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i, a_i) \mid s_t\right]$, where P_π is the trajectory distribution given π and P.

Given the definition of $V_\pi(s_t)$, the relationship between temporally adjacent state's value (e.g., $V(s_t)$ and $V(s_{t+1})$) forms the Bellman expectation equation (Bellman et al., 1965). An one-step Bellman expectation equation can be:

$$V_\pi(s_t) = \mathbb{E}_{a_t}\Big[r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V_\pi(s_{t+1})]\Big], \tag{1}$$

Given these basic RL definitions and equations, many RL algorithms fall in the scope of generalized policy iteration (GPI) (Sutton & Barto, 2018). GPI is an iterative process over policy evaluation and policy improvement.

Policy Evaluation. The target of the policy evaluation process is to estimate $V_\pi(s)$ or $Q_\pi(s, a)$ for a given policy π. For simplicity, we only utilize $V_\pi(s)$ in the following illustration. Two common value function estimation methods are the Monte Carlo (MC) estimate and the Temporal Difference (TD) estimate (Sutton, 1988). Starting from the definition of $V_\pi(s_t)$, the Monte Carlo (MC) estimate uses sampling over complete trajectories to calculate an unbiased estimate: $V_\pi(s_t) \approx \frac{1}{K}\sum_{n=1}^{K}\left[\sum_{i=t}^{\infty} \gamma^{i-t} r(s_i^n, a_i^n)\right]$ where we average the cumulative rewards over multiple full paths starting from s_t. The Temporal Difference (TD) estimate, on the other hand, builds on the temporal relationship between states without requiring complete trajectories. It estimates $V_\pi(s_t)$ using the immediate reward and the estimated value of the next state: $V_\pi(s_t) \approx \frac{1}{K}\sum_{n=1}^{K}\left[r(s_t, a_t^n) + \gamma V_\pi(s_{t+1}^n)\right]$. This approach effectively uses a "bootstrap" by depending on the estimated value of the next state to approximate $V_\pi(s_t)$.

Policy Improvement. The policy improvement process aims to update and improve policy according to policy evaluation results. Specifically, it updates the old policy π_{old} to the new one π_{new} to increase the expected return: $V_{\pi_{\text{new}}}(s_t) \geq V_{\pi_{\text{old}}}(s_t)$. In the environment with small, discrete action spaces, such improvements can be achieved by greedily choosing the action that maximizes $Q_{\pi_{\text{old}}}(s_t, a)$: $\pi_{\text{new}}(\cdot \mid s_t) = \underset{\bar{\pi}(\cdot|s_t) \in \mathcal{P}(\mathcal{A})}{\arg\max}\ \mathbb{E}_{a \sim \bar{\pi}}\left[Q_{\pi_{\text{old}}}(s_t, a)\right]$.

Another improvement method involves applying policy gradient ascent (Sutton et al., 1999). It parameterizes the policy π_θ with θ and we can derive the analytical policy gradient: $\nabla_\theta V_{\pi_\theta}(s_t) =$

$\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s_t) Q_{\pi_\theta}(s_t, a)\right]$, By choosing a small step-size $\alpha > 0$ to conduct gradient ascent: $\theta_{\text{new}} = \theta + \alpha \nabla_\theta V_{\pi_\theta}(s_t)\big|_{\theta=\theta_{\text{old}}}$, we have the policy improvement: $V_{\pi_{\text{new}}}(s_t) \geq V_{\pi_{\text{old}}}(s_t)$.

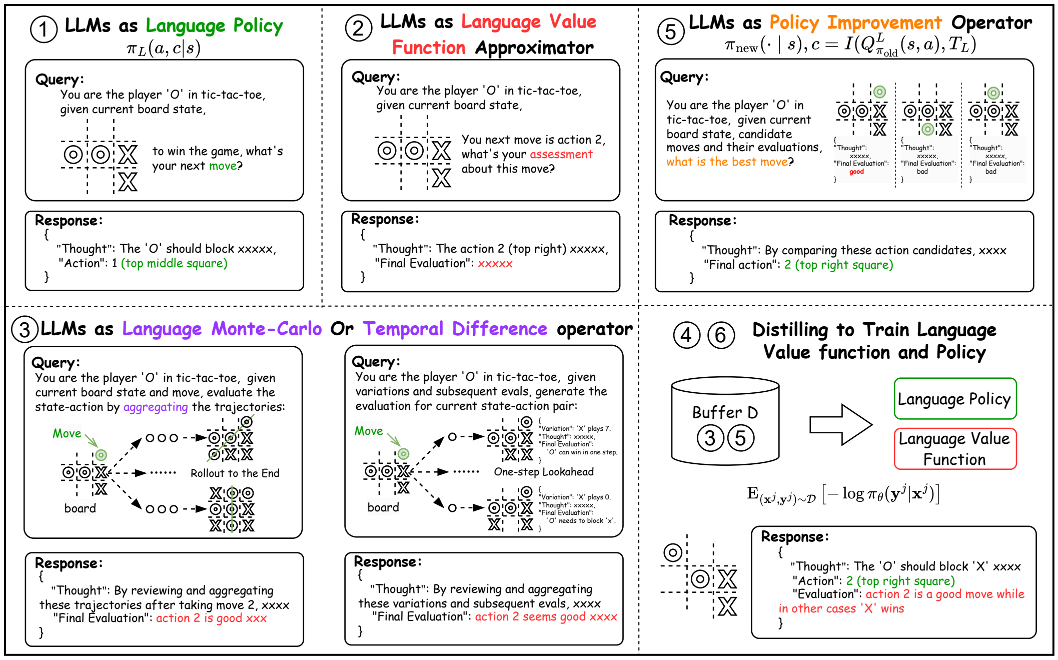


Figure 1: Practical pipeline for implementing NLRL in the Tic-tac-toe game. LLMs can serve as the language policy ①, the language-based value function approximator ②, the language Monte Carlo or Temporal Difference operator ③, and the language policy improvement operator ⑤. By distilling (④, ⑥) the improved evaluations from ② and the enhanced actions from ⑤, the NLRL agent can iteratively refine its language policy and evaluation capabilities.

3 Natural Language Reinforcement Learning

In contrast to the precise statistical models used in traditional RL, NLRL frames all elements—including task objectives, value evaluations, and strategic policies—within the form of natural language. In this section, we demonstrate:

- How to navigate decision-making tasks using natural language, **aligning it with traditional RL concepts, definitions, and equations**. Due to the inherent ambiguity of natural language, the equations presented are analogical and based on empirical insights from original RL concepts. We leave rigorous definitions and analysis for future work.
- **How to effectively implement NLRL components**. We utilize LLMs—models adept at understanding, processing, and generating language—to emulate human behavior and simulate NLRL language-based algorithmic components.

We provide visualizations in Fig 1 to illustrate these concepts (① to ⑥) and implementations.

3.1 RL Concept Analogies

We start with concept analogies in RL to model NLRL.

Text-based MDP: To conduct RL in natural language space, we convert traditional MDP to the text-based one, which leverages text descriptions to represent MDP's basic concepts, including state s, action a, and environment feedback (state transitions P and reward r).

Language Task instruction: For decision-making tasks, NLRL defines a natural language task instruction T_L, like "reaching the goal" or "opening the door". Then, we denote a metric by F that measures the completeness of the task instruction given the trajectory description $D_L(\tau_\pi)$, where

D_L is a language descriptor that can transform the trajectory distribution τ_π into its corresponding language description $D_L(\tau_\pi)$. The objective of NLRL is reformulated as

$$\max_\pi F(D_L(\tau_\pi), T_L) \tag{2}$$

That is, NLRL is trying to optimize the policy so that the language description of the trajectory distribution τ_π can show high completeness of the task instruction.

Language Policy ①: Instead of directly modeling action probability, NLRL determines the action with chain-of-thought process (Wei et al., 2022b), including strategic thoughts, logical reasoning, and planning. Thus, we represent the policy on language as $\pi_L(a, c|s) = \pi_L(c|s)\pi_L(a|c, s)$, which will first generate such thought process $\pi_L(c|s)$, then output the final action probability $\pi_L(a|c, s)$. A straightforward way to achieve such language policy is to leverage **LLMs as language policy (π_L).** Many works adopted LLMs as the decision-making agent (Wang et al., 2023a; Feng et al., 2023a; Christianos et al., 2023; Yao et al., 2022) with Chain-of-thought process (Wei et al., 2022b). By setting proper instructions, LLMs can leverage natural language to describe their underlying thought for determining the action, akin to human strategic thinking.

Language Value Function ②: Similar to the definition of Q and V in traditional RL, NLRL leverages language value function, relying on natural language evaluation to assess the policy. The language state value function V_π^L and language state-action value function Q_π^L are defined as:

$$Q_\pi^L(s_t, a_t) = D\left((s, a)_{t+1:\infty} \sim P_\pi \mid s_t, a_t, T_L\right), \tag{3}$$

$$V_\pi^L(s_t) = D\left(a_t, (s, a)_{t+1:\infty} \sim P_\pi \mid s_t, T_L\right) \tag{4}$$

Given the current state s_t or state-action (s_t, a_t), Q_π^L and V_π^L leverage language descriptions instead of scalar value to demonstrate the effectiveness of policy for achieving the task objective T_L. The language value functions are intuitively information-rich compared with the traditional scalar-based value. It can represent the evaluation results from different perspectives, consisting of the underlying logic/thoughts, prediction/analysis of future outcomes, comparison among different actions, etc.

To practically implement it, we utilize **LLMs as language value function approximator ② (Q^L, V^L).** The original idea of value function approximation (Sutton et al., 1999) is to use a parameterized one-dimensional output function to replace the big value table and serve as a value function approximator. Such design largely helps RL to handle high-dimensional and large-scale decision-making problems. Similarly, in NLRL, we can leverage (multi-modal) LLMs, to evaluate the state s or state-action pair (s, a), and serve as the language value function approximator for Q^L, V^L. This exactly corresponds to what (multi-modal) LLMs are capable of – they are designed to take in the features from the task state, such as low-dimension statistics, text, or images, and output the corresponding language understanding. By further prompting or fine-tuning over evaluation dataset, LLMs can generate language assessment.

Language Bellman Equation: In the traditional Bellman equation (Equ. 1), the state evaluation value $V_\pi(s_t)$, can be decomposed into two parts: : (1) the intermediate transition, which include immediate a_t, reward r_t, and next state s_{t+1}. (2) the next state evaluation $V_\pi(s_{t+1})$. Based on such decomposition intuition, we argue that the language value function V_π^L should analogically satisfy the language Bellman equation Equ.5 for all states $s_t \in \mathcal{S}$:

$$V_\pi^L(s_t) = G_1^{a_t, s_{t+1} \sim P_\pi}\left(G_2\left(d\left(a_t, r_t, s_{t+1}\right), V_\pi^L(s_{t+1})\right)\right), \tag{5}$$

where $d\left(a_t, r_t, s_{t+1}\right))$ depicts the language description of intermediate transition, while G_1 and G_2 serves as two information aggregation functions. By drawing an analogy to Equ. 1, G_2 mimics the add the summation operation '+' in the original Bellman equation, aggregating information from intermediate transition's description d and future evaluation $V_\pi^L(s_{t+1})$. Meanwhile, G_1 serves the role of the expectation operator $\mathbb{E}$, aggregating information accross different (a_t, s_{t+1}) pairs from the transition distribution P_π.

3.2 Language Generalized Policy Iteration

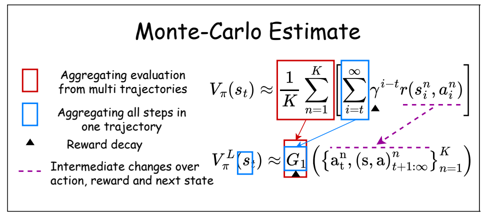
In this part, we introduce how language GPI is conducted. Similar to traditional GPI, language GPI also consists of language policy evaluation and language policy improvement.

3.2.1 Language Policy Evaluation

Language policy evaluation aims to estimate language value function V_π^L and Q_π^L for each state. We present how MC and TD estimate work in language policy evaluation.

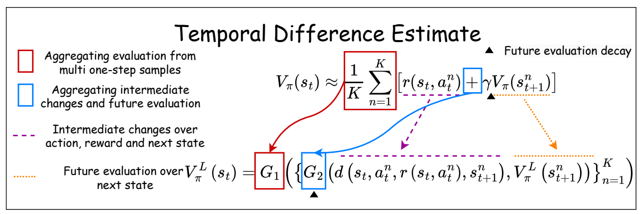
Language Monte Carlo Estimate ③. Starting from the state s_t, MC estimate is conducted over text rollouts (i.e. K_{MC} full trajectories $\{a_t, (s,a)_{t+1:\infty}\}$) given the policy π. Since we cannot take the average operation in language space, we instead leverage language aggregator G_1 to fuse information, approximating the expected evaluation:

$$V_\pi^L(s_t) \approx G_1\left(\left\{\mathrm{a_t^n}, (\mathrm{s},\mathrm{a})_{t+1:\infty}^n\right\}_{n=1}^{K_{MC}}\right), \tag{6}$$



Language Temporal Difference Estimate ③. Language TD estimate mainly relies on the one-step language Bellman equation illustrated in Equ. 5. Similar to the language MC estimate, we aggregate K_{TD} one-step samples to approximate the expected evaluation for all states $s_t \in \mathcal{S}$:

$$V_\pi^L(s_t) \approx G_1\left(\left\{G_2\big(d(s_t, a_t^n, r_t^n, s_{t+1}^n), V_\pi^L(s_{t+1}^n)\big)\right\}_{n=1}^{K_{TD}}\right), \tag{7}$$



where d, G_1 and G_2 has the same meaning for that in Equ. 5. We provide two illustrations to show how language MC and TD operator correspond to traditional MC and TD operator. Also Check Appendix E for more discussions.

Language MC estimate is free from estimation "bias" * as it directly utilizes samples from complete trajectories. However, the MC method is prone to high "variance" considering the significant number of variations (Large K) in the long-term future steps. Such variability poses a challenge for the language aggregator G_1 in Equ. 6 to effectively extract crucial information from long-context and diverse trajectories. On the contrary, while the inaccuracy of the next state evaluation $V_\pi^L(s_{t+1})$ can bring estimation "bias" to TD estimate, they effectively reduce "variance" and aggregation difficulty by discarding future variations. G_1 and G_2 are only required to conduct simple one-step information aggregation with limited variations.

LLM's summarization (Zhang et al., 2023), extraction (Xu et al., 2023), reflection (Shinn et al., 2024) and aggregation ability **make it a perfect match for the language Monte Carlo and TD operator** ③. Specifically, LLMs can serve as G_1, G_2 by prompting them to summarize and aggregate multiple rollout trajectories (language MC), or multiple few-step transitions and future states' evaluation (language TD).

In addition, ③ provides an unsupervised and scalable way to generate language evaluation data through environment interaction, which can be further leveraged to train our language value function approximator in ②. We can **distill language value estimation into language value function approximator.** (④ in Fig. 1) This corresponds to traditional critic training for value-based RL, e.g, DQN (Mnih et al., 2015), or actor-critic algorithms, e.g, PPO (Schulman et al., 2017), but happens in natural language space.

3.2.2 Language Policy Improvement

Similar to traditional policy improvement, language policy improvement also aims to select actions that maximize the task completeness function F for all states $s_t \in \mathcal{S}$:

$$\pi_{\text{new}}(\cdot \mid s) = \underset{\bar{\pi}(\cdot|s)\in\mathcal{P}(\mathcal{A})}{\arg\max}\; F(Q_{\pi_{\text{old}}}^L(s,a), T_L), \tag{8}$$

*We use quotes for "bias" and "variance" to indicate that we draw on their concepts, not their strict statistical definitions

The task completeness F is difficult to quantify for general language-based tasks, as it largely relies on human textual prior knowledge. Given this complexity, instead of mathematically optimizing F, we also refer to the chain-of-thought process–NLRL leverages a language analysis process I that can generate the thought process c to guide policy optimization and action selection for all states $s_t \in \mathcal{S}$.

$$\pi_{\text{new}}(\cdot \mid s), c = I(Q^L_{\pi_{\text{old}}}(s,a), T_L), \bar{\pi}(\cdot \mid s) \in \mathcal{P}(\mathcal{A}), \tag{9}$$

Language policy improvement conducts strategic analysis c to determine the most promising action for task completion as the new policy $\pi_{\text{new}}(\cdot \mid s)$. Ideally, this analysis is mainly based on the correlation judgment between the language evaluation $Q^L_{\pi_{\text{old}}}(s,a)$ and task objective T_L.

With the chain-of-thought process and prior knowledge about the world, **LLMs can serve policy improvement operator** I (⑤ in Fig. 1) to determine the most promising action $\pi_{\text{new}}(\cdot \mid s)$ by taking language analysis c over the correlation of language evaluation $Q^L_{\pi_{\text{old}}}(s,a)$ and task objective T_L. The underlying idea also aligns with some recent works (Kwon et al., 2023a; Rocamonde et al., 2023) that leverage LLMs or Vision-language models as the reward–they can accurately model the correlation. Specifically, for a given state s, we prompt the LLM with several action candidates and their corresponding language evaluations Q^L, to obtain the improved action with a chain-of-thought process analyzing different actions' evaluations.

Similar to distillation in ④, **we can train our language policy in ① by supervised-finetuning** (⑥ in Fig. 1) over the improved chain-of-thought based language policy data from ⑤. This corresponds to the traditional policy training in policy-based RL, e.g, REINFORCE (Williams, 1992) or actor-critic algorithms, e.g, PPO (Schulman et al., 2017).

4 NLRL Algorithms

Building upon these components and procedures, we can create various NLRL applications. Here we illustrate three algorithms, though many more possibilities exist beyond.

4.1 Language GPI boosts LLM's critic and policy by prompting (Combining ①, ②, ③, ⑤)

Our first case utilizes language GPI to enhance LLM's critic and policy solely through prompting, which can be particularly beneficial for improving proprietary models such as GPT-4 (OpenAI, 2023) or Gemini (Team et al., 2023). Specifically, we first combine ①, ②, ③ to build a language policy evaluation pipeline. Take language TD shown in Equ. 7 as an example. We prompt LLMs to (1) evaluate the subsequent state's value $V^L_\pi(s_{t+1})$ (②) (2) serve as the TD operator G_1, G_2 (③). By performing a one-step look-ahead with $a_t \sim \pi$ (①) and leveraging G_1, G_2 to aggregate information from intermediate transition d alongside the subsequent evaluation $V^L_\pi(s_{t+1})$, language TD can generate new and improved language evaluation, $V^L_\pi(s_t)_{\text{new}}$. Building upon this new evaluation, LLM-based policy improvement operator I (⑤) converts the evaluation into a better policy π_{new}. Refer to Algorithm 1 for detailed procedures.

4.2 Training natural language value function for a given policy (Combining ②, ③, ④)

Our second case aims to train an LLM critic capable of evaluating any given state with natural language explanations, similar to a chess annotator who provides insightful commentary on boards, moves, and strategies. For example, we can build an iterative language TD pipeline by combining ②, ③, and ④. First, we leverage a tunable LLM A and prompt it to become a language value function approximator ②. Combined with the look-ahead transitions $(a_t, s_{t+1}) \sim P_\pi$ by taking rollouts with policy π, and the subsequent state evaluation $V^L_\pi(s_{t+1})$ generated by A, we prompt LLM B for the language TD estimate (③), similar to Sec 4.1. The model A is further finetuned by such language TD estimates (④) and will be plugged back for $V^L_\pi(s_{t+1})_{\text{new}}$ in a new iteration. Iteratively, we can obtain the final converged natural language value function. Check Algorithm 2 for more details.

4.3 Natural Language Actor-Critic Learning (Combining ①, ②, ③, ④, ⑤, ⑥)

Last, we combine all these procedures and build the full natural language actor-critic pipeline. Similar to traditional actor-critic (Barto et al., 1983; Sutton et al., 1999), our natural language actor-critic

Table 1: Language GPI (LGPI) results with ablations on look-ahead steps N and variations number K_{TD} on Double-T and Medium Maze. We include language Policy (①) and language value function (LVF) + policy Improvement I (②,⑤) baselines.

Avg Reward	Double-T	Medium
Language Policy $\pi^L(s)$ *(①)*	-27.3 ± 4.4	-27.1 ± 5.3
LVF $Q^L(s,a) + I$ *(②,⑤)*	-18.3 ± 6.1	-33.6 ± 14.4
LGPI (K=1, N=3) (②, ③, ⑤)	-17.9 ± 3.7	-20.9 ± 7.6
LGPI (K=4, N=1) (②, ③, ⑤)	-17.5 ± 4.5	-12.7 ± 4.7
LGPI (K=4, N=3) (②, ③, ⑤)	-12.7 ± 4.5	-15.1 ± 4.4
LGPI (K=6, N=3) (②, ③, ⑤)	-12.2 ± 3.0	-15.4 ± 5.0
LGPI (K=8, N=3) (②, ③, ⑤)	$\mathbf{-11.20 \pm 2.9}$	$\mathbf{-12.2 \pm 4.5}$

framework simultaneously learns both a language policy and a language critic through unsupervised environment interactions. For each iteration, we first use language policy ① to take rollouts in the environment. With new trajectories, we update our language value model ② by language MC or TD ③ and train it with supervised-finetuning loss (④). For language policy improvement, we query our updated language value to evaluate action candidates for states extracted from the rollout trajectories. Further LLM-based improvement operator ⑤ brings us a stronger policy, which will be used to train our language policy (⑥). Check Algorithm 3 for more details.

5 Experiments

5.1 Language GPI by prompting (Sec 4.1)

Our first experiments explore Sec 4.1 and leverage language GPI to improve LLM capability with pure prompting. Specifically, we choose the maze games of LMRL (Abdulhai et al., 2023), aiming to validate that **Language TD Estimate** and **Language Policy Improvement** can benefit the evaluation and further improve policy. We use the original settings of LMRL Gym Maze, where the agent is required to navigate to a goal position in a "Double T" or "Medium" maze. We consider the fully-observable setting, where the agent's observation (described by text) includes the agent's current position in the maze, the agent's action history, the walls' position around the agent (if any), and the goal position. The action space is discrete, including moving up / down / right / left. We evaluate the performance on 30 different initial positions, each with 3 random seeds. Refer to Appendix C.1.2 for more details and visualizations.

For language TD estimate in Equ. 7, we prompt *gpt-4o-mini-2024-07-18* as the language aggregator G_1, language state-action value aggregator G_2, and language state value function V_π^L respectively. Specifically, given an environment state, for each candidate action, we use a fully random policy to rollout N steps into the future and use the language state value function V_π^L to evaluate the look-ahead state. For each state we repeat this process for K times and use G_2 to aggregate into a state-action value estimation. For language policy improvement in Equ. 9, the same GPI-4o-mini is leveraged as the improvement operator I.

We compare Language GPI with a few different baselines, including prompt-based language policy π_L (①), as well as prompt-based language value function $Q^L(s,a)$ + language policy improvement I. As shown in Table 1, Language TD produces better performance than the prompt-based language value in both mazes, and more variations & look ahead steps can have even better results. Language policy improvement can only benefit stably for language-TD-enhanced language value (② and ③).

5.2 Training Natural Language Value Function with language TD (Sec 4.2)

Our second experiment aims to train a language value function with language TD in the 5x5 breakthrough board game (Fig 5). Breakthrough is a turn-based board game for white and black players. Each player has a few lines of pawns. Black moves first and in each turn, the player chooses one piece to move one square forward (straight or diagonal) onto empty squares or capture diagonally. The goal is to break through the opponent's line and reach their home row.

As mentioned in Sec 4.2, this value function can serve as a reliable board evaluator/annotator. However, evaluating board states in this game is challenging for LLMs. First, the 5x5 breakthrough

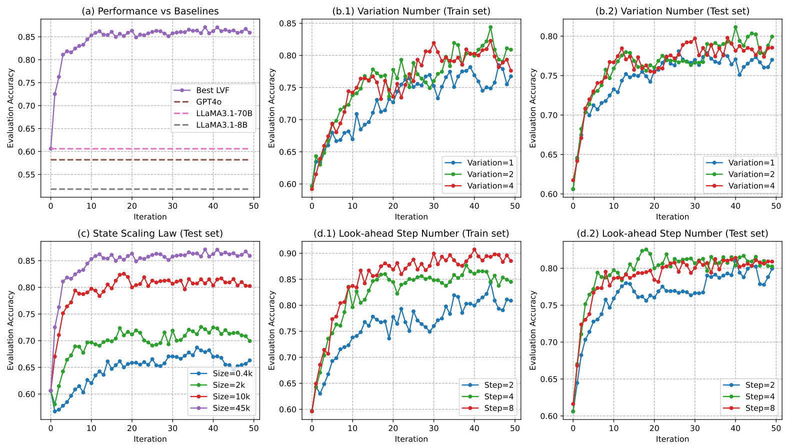


Figure 2: Breakthrough experiment results. (a) Performance comparison with baselines. (b,d) Ablation study on look-ahead step number and variation number K_{TD}. (c) Results for scaling state size.

variant is niche, leading to sparse pre-training data on board evaluations. Most LLMs have minimal knowledge of it. Second, despite the small board size, its state space complexity can reach 10^8 (Saffidine et al., 2012), making natural language assessments significantly more difficult. Refer to Appendix C.2 for more experimental details and result visualizations.

5.2.1 Experiment Setup and Methods

Text-Breakthrough. As mentioned in the text-based MDP, we textualize the game, including board, pieces, moves, positions, and information about capturing or termination.

Policy π in $V_\pi^L(s)$. Our first task is to determine the π in $V_\pi^L(s)$ since $V_\pi^L(s)$ measures the value for a specific policy, according to Equ 3. To train a reliable board annotator, we select a strong or near-optimal policy—analogous to the way we place greater trust in a grandmaster's annotations. In our experiment, our π is the Monte-Carlo tree search (MCTS) algorithm (Kocsis & Szepesvári, 2006) from OpenSpiel (Lanctot et al., 2019), with a high number of simulations and rollouts to ensure an extensive search.

Building State Dataset $s \sim P(s)$. Since $V_\pi^L(s)$ is over state s, our second step is to build a state dataset D_s–the distribution $P(s)$ that our V_π^L works on. To ensure that V_π^L can assess positions across diverse levels of gameplay, we build a mixed-state dataset by collecting rollouts between different MCTs configurations. Then we can easily split D_s to build training state set D_s^{train} and test state set D_s^{test}.

Building TD training dataset. The TD training dataset is built on $D_s^{train}$by conducting look-ahead expansion with rollout policy π. For each state s_t, we take a few times l-step look-ahead rollout and deduplicate to K distinct variations: $\{(s_{t+i}, a_{t+i}, r_{t+i}, s_{t+i+1})_{i=0}^{l-1}\}_K$.

Models. For the language TD operator (G_1, G_2), we prompt the large *LLaMA-3.1-70B-Instruct* (Dubey et al., 2024) model since it preserves stronger aggregation and reasoning ability. The language value function $V_\pi^L(s)$ is trained from the small *LLaMA-3.1-8B-Instruct* model.

Evaluation. An automatic evaluation metric is challenging even for advanced LLMs like GPT-4o, given their limited domain knowledge. We adopt a coarser metric—judging which side has the advantage. For any given state, we estimate win rates using Monte Carlo simulations to the game outcome. The ground-truth label is assigned to the side with a win rate exceeding a predefined threshold.

5.2.2 Experimental Results

Performance vs Baselines. Our Fig 2.a shows the evaluation accuracy comparison on D_s^{test}, between our best language value function's training curve and prompting-based LLMs (*LLaMA-3.1-70B*, *LLaMA-3.1-8B* and *GPT-4o-08-06*), which clearly demonstrates the necessity of language TD training. Due to the lack of domain knowledge, all prompting-based LLMs perform poorly. The best accuracy (0.61) is only slightly higher than the random guess (0.5), while our trained language value function dominates the task and the accuracy reaches 0.87 on the test set D_s^{test}.

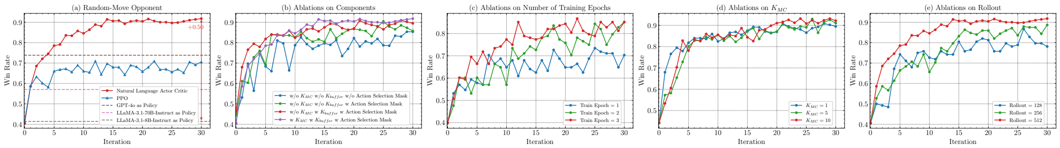


Figure 3: Natural Language Actor Critic Pipeline training results. (a) Training results against the Random-Move Opponent. (b) Ablation study on components (K_{MC}, K_{buffer}, and Action Selection Mask). These results demonstrate that **our proposed Natural Language Actor Critic pipeline can stably improve under stochastic dynamics**. (c) - (e) Ablation studies on number of training epochs, Monte Carlo sample size K_{MC}, and number of rollout trajectories.

Look-ahead Ablation. Fig 2.b and Fig 2 shows further ablation study over variation number (m) and look-ahead steps k. The test set performance is shown in Fig 2.b.2 and Fig 2.d.2, while Fig 2.b.1 and Fig 2.d.1 present a subset of in-training state evaluations. The ablation study indicates that determining the appropriate m and k is crucial for optimal performance. Notably, in Fig 2.d.1 and Fig 2.d.2, using 8 steps resulted in significantly higher in-training accuracy but lower test set accuracy compared with 4. This suggests that longer look-ahead steps may lead to overfitting, potentially hindering generalization to novel states evaluation.

State Scaling Law. We also examine how scaling the training state size affects performance in Fig 2.c, with datasets sizes 0.4k, 2k, 10k, and 45k. The results demonstrate robust scalability of our algorithm, with consistent improvements in test set performance as the training set size increases.

5.3 Natural Language Actor-Critic Learning (Sec 4.3)

For validating Sec 4.3, we implement the natural language actor-critic in a model-free setting (Sutton & Barto, 2018) in the Tic-tac-toe game (Ju, 2024), where the system learns purely from sampled trajectories without access to dynamics. As a supplement to language TD in Sec. 5.1 and 5.2, we utilize the language MC (Equ. 6) for the policy evaluation.

5.3.1 Experiment Setup and Methods

Text-Tic-tac-toe. Similar to text-breakthrough, we textualize all information in the Tic-tac-toe.

Model Architecture. Our implementation uses three language models: one *LLaMA-3.1-70B-Instruct* and two *LLaMA-3.1-8B-Instruct* models. The 70B model is for language aggregator (G_1) and policy improvement operator I. The two 8B models implement our trainable components: language policy π_L that generates actions through chain-of-thought reasoning, and language value function Q_π^L that evaluates state-action pairs.

Evaluation. To thoroughly evaluate our approach, we test against two different types of opponents. The results against a deterministic opponent that always takes the first available action can be found in Appendix C.3.5. For our main experiments, we focus on a stochastic opponent that plays randomly, presenting a more challenging case for policy evaluation. We measure performance through metrics including win/loss/tie rates.

5.3.2 Stabilizing language value function

Despite the soundness of Sec 4.3, we identify training instability during initial experiments. We observed that V_π^L can easily hallucinate when evaluating unseen state-action pairs, leading to incorrect language policy improvement.

We take two techniques to mitigate it. First, we enlarge the value training data size by merging the most recent K_{buffer} iteration's value data buffers, which helps stabilize value training. Second, we add an action selection mask to restrict language policy improvement only to moves that are likely to be chosen by the language policy π_L during rollout. We sample the language policy N_{sample} times and build an action candidate list. The language policy improvement is conducted from the list's top-m moves. Such action selection mask can help constrain policy updates within a limited range where language value function can be more reliable.

5.3.3 Experimental Results

Performance vs Baselines. The learning curves in Fig. 3(a) demonstrate our system's superiority in the random-move opponent setting. Our algorithm outperforms all other baselines (prompting-based method using *GPT-4o-08-06*, *LLaMA-3.1-8B/70B-Instruct* with chain-of-thought reasoning), including the PPO (Schulman et al., 2017) fine-tuned *LLaMA-3.1-8B-Instruct* baseline, which helps verify the effectiveness and efficiency brought by NLRL. The improvement is substantial (0.4 to 0.9), demonstrating NLRL's robustness despite the stochastic dynamics. As shown in Fig. 3(b), each component (Monte Carlo sampling number K_{MC}, buffer merging size K_{buffer}, and action selection mask) enhances the system's performance, with the full system incorporating all three components achieving the highest and most stable win rate.

Ablation Studies We conduct comprehensive ablation studies to investigate key hyperparameters. First, training with 128 trajectories per iteration shows that increasing from 1 to 3 epochs significantly improves stability and adaptation, as shown in Fig. 3(c). Second, our investigation of Monte Carlo sampling with K_{MC} values of 1, 5, and 10 shows in Fig. 3(d) that increased sampling with 512 trajectories leads to more stable training. Finally, Fig. 3(e) shows how the number of rollout trajectories affects training, with 512 trajectories per iteration yielding the most robust learning curve. Additional ablation studies on action selection mask parameter top-m and effect of experience buffer size K_{buffer} can be found in Appendix C.3.6, where we demonstrate that larger m values enhance exploration while maintaining stable learning and $K_{buffer} = 3$ helps maintain consistent performance by preserving past experiences. Refer to Appendix C.3 for more experimental details and result visualizations.

6 Related Work

Language Model Based Autonomous Agent. We are inspired by the strong emergent capabilities of LLMs in complex reasoning and planning scenarios (Brown et al., 2020; Anil et al., 2023; OpenAI, 2023; Feng et al., 2023b; Yao et al., 2022). The field of language model-based autonomous agents (Feng et al., 2023a; Christianos et al., 2023; Zhang et al., 2024) has witnessed a growing trend of leveraging LLMs for high-level planning purposes. Reflexion (Shinn et al., 2023) is built upon ReAct (Yao et al., 2022) with self-reflection, to generate tips given online feedback. (Zhong et al., 2024) proposes to distill such reflection ability from the large model to train a small policy feedback model that can guide the policy. Their underlying ideas share similarities with language MC, while NLRL formally formulates it–we extract core information by sampling multiple trajectories and leverage them to train our language value function.

Learning from Language Feedback. Our work is also related to research on learning from language feedback. While Cheng et al. (2023) focuses on benchmarking algorithms, we aim to propose a new algorithmic framework. Studies such as (Yang et al., 2023; Yuksekgonul et al., 2024; Cheng et al., 2024) introduce an LLM-based optimization paradigm that leverages natural language to represent optimization operators like gradients and backpropagation, achieving end-to-end generative optimization. NLRL, on the other hand, represents a parallel approach, offering a generative framework specifically designed for RL optimization problems.

LLMs as evaluation function. Our language value function aligns with recent efforts in NLP that leverage LLMs as generative evaluators or verifiers, as seen in (Wang et al., 2023b; Li et al., 2023; Jiang et al., 2023; Gao et al., 2024; Zhang et al., 2024; Mahan et al., 2024), or that utilize LLMs' self-evaluation capabilities to enhance task-solving performance (Putta et al., 2024; Yao et al., 2023; Hao et al., 2023). These approaches often rely on the LLM's self-assessment (which can be unreliable), or distill on stable supervision signals like human annotations or guidance from stronger LLMs. In contrast, NLRL's training of a language value function offers a novel unsupervised approach.

7 Conclusion

We propose NLRL to open a new door for implementing RL algorithms in natural language space, improving agent's effectiveness, efficiency, and interpretability. NLRL is also compatible with the current language model and LLM agent, offering potential ways to generate high-quality language synthetic data for policy and critic. We leave discussions of limitation and future work in Appendix F.

REFERENCES

Marwa Abdulhai, Isadora White, Charlie Snell, Charles Sun, Joey Hong, Yuexiang Zhai, Kelvin Xu, and Sergey Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning with language models. *arXiv preprint arXiv:2311.18232*, 2023.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? a large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

Richard Bellman, Robert E Kalaba, et al. *Dynamic programming and modern control theory*, volume 81. Citeseer, 1965.

Konstantinos Bousmalis, Giulia Vezzani, Dushyant Rao, Coline Manon Devin, Alex X Lee, Maria Bauza Villalonga, Todor Davchev, Yuxiang Zhou, Agrim Gupta, Akhil Raju, et al. Robocat: A self-improving generalist agent for robotic manipulation. *Transactions on Machine Learning Research*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Ching-An Cheng, Andrey Kolobov, Dipendra Misra, Allen Nie, and Adith Swaminathan. Llf-bench: Benchmark for interactive learning from language feedback. *arXiv preprint arXiv:2312.06853*, 2023.

Ching-An Cheng, Allen Nie, and Adith Swaminathan. Trace is the next autodiff: Generative optimization with rich feedback, execution traces, and llms. *arXiv preprint arXiv:2406.16218*, 2024.

Filippos Christianos, Georgios Papoudakis, Matthieu Zimmer, Thomas Coste, Zhihao Wu, Jingxuan Chen, Khyati Khandelwal, James Doran, Xidong Feng, Jiacheng Liu, et al. Pangu-agent: A fine-tunable generalist agent with structured reasoning. *arXiv preprint arXiv:2312.14878*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Devleena Das, Sonia Chernova, and Been Kim. State2explanation: Concept-based explanations to benefit agent learning and user understanding. *arXiv preprint arXiv:2309.12482*, 2023.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural computation*, 5(4):613–624, 1993.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. Chessgpt: Bridging policy learning and language modeling. *arXiv preprint arXiv:2306.09200*, 2023a.

Xidong Feng, Ziyu Wan, Muning Wen, Ying Wen, Weinan Zhang, and Jun Wang. Alphazero-like tree-search can guide large language model decoding and training. *arXiv preprint arXiv:2309.17179*, 2023b.

Mingqi Gao, Xinyu Hu, Jie Ruan, Xiao Pu, and Xiaojun Wan. Llm-based nlg evaluation: Current status and challenges. *arXiv preprint arXiv:2402.01383*, 2024.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *Proceedings of the 2017 ACM/IEEE international conference on human-robot interaction*, pp. 303–312, 2017.

Ying Ji, Yu Wang, and Jien Kato. Spatial-temporal concept based explanation of 3d convnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15444–15453, 2023.

Dongfu Jiang, Yishan Li, Ge Zhang, Wenhao Huang, Bill Yuchen Lin, and Wenhu Chen. Tiger-score: Towards building explainable metric for all text generation tasks. *arXiv preprint arXiv:2310.00752*, 2023.

Kim Jeong Ju. gym-tictactoe: A tic tac toe environment for openai gym, 2024. URL https://github.com/haje01/gym-tictactoe. GitHub repository.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. *arXiv preprint arXiv:2303.00001*, 2023a.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023b.

Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019. URL http://arxiv.org/abs/1908.09453.

Junlong Li, Shichao Sun, Weizhe Yuan, Run-Ze Fan, Hai Zhao, and Pengfei Liu. Generative judge for evaluating alignment. *arXiv preprint arXiv:2310.05470*, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. Generative reward models. *arXiv preprint arXiv:2410.12832*, 2024.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. Agent q: Advanced reasoning and learning for autonomous ai agents. *arXiv preprint arXiv:2408.07199*, 2024.

Juan Rocamonde, Victoriano Montesinos, Elvis Nava, Ethan Perez, and David Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

Abdallah Saffidine, Nicolas Jouandeau, and Tristan Cazenave. Solving breakthrough with race patterns and job-level proof number search. In *Advances in Computer Games: 13th International Conference, ACG 2011, Tilburg, The Netherlands, November 20-22, 2011, Revised Selected Papers 13*, pp. 196–207. Springer, 2012.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Sarath Sreedharan, Utkarsh Soni, Mudit Verma, Siddharth Srivastava, and Subbarao Kambhampati. Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with inscrutable representations. *arXiv preprint arXiv:2002.01080*, 2020.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

Yidong Wang, Zhuohao Yu, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, et al. Pandalm: An automatic evaluation benchmark for llm instruction tuning optimization. *arXiv preprint arXiv:2306.05087*, 2023b.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022b.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, and Enhong Chen. Large language models for generative information extraction: A survey. *arXiv preprint arXiv:2312.17617*, 2023.

Jie Xu, Hanbo Zhang, Xinghang Li, Huaping Liu, Xuguang Lan, and Tao Kong. Sinvig: A self-evolving interactive visual agent for human-robot interaction. *arXiv preprint arXiv:2402.11792*, 2024.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic" differentiation" via text. *arXiv preprint arXiv:2406.07496*, 2024.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024.

Tianyi Zhang, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. Benchmarking large language models for news summarization. *arXiv preprint arXiv:2301.13848*, 2023.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*, 2023.

Victor Zhong, Dipendra Misra, Xingdi Yuan, and Marc-Alexandre Côté. Policy improvement using language feedback models. *arXiv preprint arXiv:2402.07876*, 2024.

CONTENTS

A Detailed Related work

Language Model Based Autonomous Agent. Inspired by the strong emergent capabilities of LLMs in complex reasoning and planning scenarios (Brown et al., 2020; Wei et al., 2022a; Anil et al., 2023; OpenAI, 2023; Feng et al., 2023b). The field of language model-based autonomous agents (Feng et al., 2023a; Christianos et al., 2023; Zhang et al., 2024) has witnessed a growing trend of leveraging LLMs for high-level planning purposes. For text agents, ReAct(Yao et al., 2022) leverages chain-of-thought planning about the intermediate goals and task-specific generation via few-shot prompting. Reflexion (Shinn et al., 2023) is built upon ReAct (Yao et al., 2022) with self-reflection, named verbal reinforcement learning, to generate tips given online feedback. Such tips strengthen agent's capability through in-context learning. (Zhong et al., 2024) proposes to distill such reflection ability from the large model to train a small policy feedback model that can guide the policy. Their underlying ideas share similarities with NLRL's language Monte Carlo estimate, while NLRL formulates this process more formally and incorporates it into training–we extract core information/concepts by sampling multiple trajectories and leverage them to train our natural language value function. Another line of work like Zhang et al. (2023); Xu et al. (2023) leverages Large Language Models to summarize and extract information to help fuse intermediate changes. They are related to the "aggregation" idea of language value function training.

Interpretable Reinforcement Learning. One of the major purposes of interpretable RL is to automatically seek explanations from non-AI experts. For instance, methods of concept-based explanations for sequential decision-making tasks. Ji et al. (2023) provide a concept-based explanation for 3D action recognition CovNets by clustering learned human interpretable features. Sreedharan et al. (2020) formulates concept-based explanations upon state preconditions and action costs, representing any factual statement a user associates with a given state. Similarly, Hayes & Shah (2017) uses logical formulas to summarize policies. Additionally, Das et al. (2023) trains a joint embedding model for state-action pairs and concept-based explanations.

Learning from Language Feedback. Our work is also related to research on learning from language feedback. While Cheng et al. (2023) focuses on benchmarking algorithms, we aim to propose a new algorithmic framework. Studies such as (Yang et al., 2023; Yuksekgonul et al., 2024; Cheng et al., 2024) introduce an LLM-based optimization paradigm that leverages natural language to represent optimization operators like gradients and backpropagation, achieving end-to-end generative optimization. NLRL, on the other hand, represents a parallel approach, offering a generative framework specifically designed for RL optimization problems.

LLMs as evaluation function. Our language value function aligns with recent efforts in NLP that leverage LLMs as generative evaluators or verifiers, as seen in (Wang et al., 2023b; Li et al., 2023; Jiang et al., 2023; Gao et al., 2024; Zhang et al., 2024; Mahan et al., 2024), or that utilize LLMs' self-evaluation capabilities to enhance task-solving performance (Putta et al., 2024; Yao et al., 2023; Hao et al., 2023). These approaches often rely on the LLM's self-assessment (which can be unreliable), or distill on stable supervision signals like human annotations or guidance from stronger LLMs. In contrast, NLRL's training of a language value function offers a novel unsupervised approach: we can train a generative evaluator solely by aggregating language feedback from the environment, without distilling any strong model–experimental results in Sec. 5.2 show that our language value function's performance is much better than any LLMs we utilize during the training.

Successor Features. Successor features (Dayan, 1993; Barreto et al., 2017) aim to learn a high-dimensional value function representation, where each component captures the expected future occurrence of all state representations under a fixed policy. As shown in (Barreto et al., 2017), successor features help decouple the environment dynamics from the rewards, facilitating transfer learning in RL. The language value function in NLRL is conceptually similar to successor features since it also represents state values in a high-dimensional space and satisfies the language Bellman equation.

B Algorithms

In this section, we provide the full pseudo-code for three cases mentioned in Sec. 4.1, 4.2, and 4.3.

B.1 Language GPI by prompting

Algorithm 1 Language GPI by Prompting

Require: Initial language policy π, language value function V_π^L, LLM implementing operators G_1, G_2, policy improvement operator I, number of lookahead steps N, number of variations K
for each state s_t **do**
  // Language TD estimation
  for $i = 1$ to K **do**
    Sample action $a_t^{(i)}$ (①)
    Simulate N steps ahead to get trajectory $\tau^{(i)}$
    Prompt LLM to evaluate $V_\pi^L(s_{t+N}^{(i)})$ (②)
    Generate intermediate description $d^{(i)} = d(a_t^{(i)}, r_t^{(i)}, s_{t+1}^{(i)})$
    Compute value estimate $V^{(i)} = G_2(d^{(i)}, V_\pi^L(s_{t+N}^{(i)}))$ (③)
  end for
  Aggregate value estimates $V_\pi^L(s_t) = G_1(\{V^{(i)}\}_{i=1}^K)$ (③)
  // Policy improvement
  Use $V_\pi^L(s_t)$ and prompt LLM as operator I to select the optimal action a_t (⑤)
end for

B.2 Train natural language language value function with language TD

Algorithm 2 Train natural language language value function with language TD

Require: Pre-defined policy π, LLM-based language value model V_π^L, language descriptor d, language aggregator G_1, G_2, state buffer D_s, number of iterations T, number of trajectories N, number of MC trajectories K
// Collect few-step rollout data using fixed policy for TD data buffer
Initialize TD data buffer $\mathcal{B} \leftarrow \{\}$
for state s_t in state buffer D_s **do**
  Starting from s_t, run policy π for a few l-step look-ahead rollouts and collect K distinct variations: $V_{s_t} = \{(s_{t+i}, a_{t+i}, r_{t+i}, s_{t+i+1})_{i=0}^{l-1}\}_K$
  $\mathcal{B} \leftarrow \mathcal{B} \cup V_{s_t}$
end for
// Language TD learning
for iteration $t = 1$ to T **do**
  // Language TD estimate ②, ③
  $\mathcal{D}_V \leftarrow \{\}$ {Value training data}
  for each look-ahead data (s_t, V_{s_t}) in $\mathcal{B}$ **do**
    For each variation n, generate the final state evaluation $V_\pi^L(s_{t+l}^n)$ with V_π^L, and utilize d to describe intermediate transitions $d_n = d(s_t, a_t^n, r_t^n, s_{t+1}^n, ..., s_{t+l-1}^n, a_{t+l-1}^n, r_{t+l-1}^n)$
    $V_\pi^L(s_t)_{new} \leftarrow G_1\left(\left\{G_2\big(d_n, V_\pi^L(s_{t+1}^n)\big)\right\}_{n=1}^K\right)$ {l-step, k-variation Language TD}
    $\mathcal{D}_V \leftarrow \mathcal{D}_V \cup V_\pi^L(s_t)_{new}$
  end for
  // Update value function ④
  Train V_π^L using language TD estimate dataset $\mathcal{D}_V$ with supervised loss
end for

B.3 Natural language actor critic

Algorithm 3 Natural Language Actor-Critic Learning

Require: Initial language policy model π_L, language value model V_π^L, large language model for evaluation $G_1/G_1,G_2$, improvement operator I
Require: Number of iterations T, number of trajectories N, number of MC trajectories K

Initialize replay buffer history $\mathcal{B} \leftarrow \{\}$
for iteration $t = 1$ to T **do**
  // Collect trajectories using language policy ①
  $\tau \leftarrow \{\}$ {Initialize trajectory buffer}
  for $i = 1$ to N **do**
    Run policy π_L to collect trajectory: $\tau_i \leftarrow \{(s_j, a_j, r_j, s_{j+1})_{j=0}^{H_i}\}$
    $\tau \leftarrow \tau \cup \{\tau_i\}$
  end for
  // Language value estimation ②, ③
  $\mathcal{D}_V \leftarrow \{\}$ {Value training data}
  for each state-action pair (s, a) in τ **do**
    if using MC estimate **then**
      Sample K_{MC} trajectories starting from (s, a): $\{\tau_k\}_{k=1}^{K_{MC}}$
      $v \leftarrow G_1(\{\tau_k\}_{k=1}^{K_{MC}})$ {Language MC estimate}
    else if using TD estimate **then**
      Sample K_{TD} transitions starting from (s, a): $\{(s_k, a_k, r_k, s'_k)\}_{k=1}^{K_{TD}}$
      $v \leftarrow G_1(\{G_2(d(s_k, a_k, r_k, s'_k), V_\pi^L(s'_k))\}_{k=1}^{K_{TD}})$ {Language TD estimate}
    end if
    $\mathcal{D}_V \leftarrow \mathcal{D}_V \cup \{(s, a, v)\}$
  end for
  // Update value function ④
  $\mathcal{B} \leftarrow \mathcal{B} \cup \{\mathcal{D}_V\}$ {Add to buffer history}
  Train V_π^L on merged data from last K_{buffer} buffers in $\mathcal{B}$ with supervised loss
  // Language policy improvement ⑤
  $\mathcal{D}_\pi \leftarrow \{\}$ {Policy training data}
  for each state s in τ **do**
    Sample actions $\{a_i\}_{i=1}^{N_{sample}}$ from $\pi_L(s)$
    Select top-m actions $\mathcal{A}_k$ based on sampling frequency
    Query value estimates: $Q^L \leftarrow \{V_\pi^L(s, a) | a \in \mathcal{A}_k\}$
    $\pi_{new}, c \leftarrow I(Q^L, T_L)$ {Language improvement operator}
    $\mathcal{D}_\pi \leftarrow \mathcal{D}_\pi \cup \{(s, \pi_{new}, c)\}$
  end for
  // Update policy ⑥
  Train π_L on $\mathcal{D}_\pi$ with supervised loss
end for

C Experimental details

C.1 Maze Experiment

C.1.1 Experiment Details

Our Maze experiment is conducted using LMRL Gym (Abdulhai et al., 2023). We use the original codebase for the environment. We consider the fully-observable setting, where the agent's observation (described by text) includes the agent's current position in the maze, the agent's action history, the walls' position around the agent (if any), and the goal position. The action space is discrete, including moving up / down / right / left. We evaluate the performance on 30 different initial positions, each with 3 random seeds.

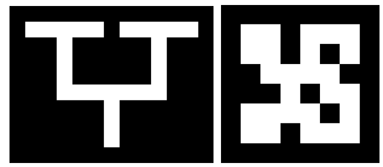


Figure 4: The "double T" maze and the "medium" maze. At the beginning of each test, the agent randomly respawns in white space.

We prompt *gpt-4o-mini-2024-07-18* (temperature = 0.7) as the language aggregator G_1, language state-action value aggregator G_2, and language state value function V_π^L respectively.

C.1.2 Language GPI Visualization

We present the language GPI's prompt and response visualization for language state value function V_π^L and language state-action value aggregator G_2 in the LMRL Maze experiment.

Maze prompts

Role	**Content**
SYSTEM	You are an expert maze solver. You only respond in json.
USER (V_π^L)	You are playing a simple maze game. An agent is moving in the maze and the objective is to reach the goal in as few steps as possible. The possible actions are "move up", "move down", "move left", "move right". You are a highly skilled evaluator in this game and is expected to function similar to state value function in reinforcement learning. At each step I will provide you with the move history of the agent (from old to new), including where the goal is, your current position, the walls that surround the agent. Your output evaluation should be a json array that includes the following concepts: - "thoughts": Let's think step by step. Generate your detailed thought process and evaluation of the agent's position and the distance towards the goal. - "final evaluation": Concisely judge how good the agent's current position is compared to all the possible positions in the maze, in terms of reaching the goal. Let's start a new game. Now, please give your evaluation of the current state given the move history of the agent: The goal is at position 8, 6. Your current position is at position 5, 7. There are walls above you, below you. move up The goal is at position 8, 6. Your current position is at position 5, 7. There are walls above you, below you. move down

	The goal is at position 8, 6. Your current position is at position 5, 7. There are walls above you, below you. move right The goal is at position 8, 6. Your current position is at position 5, 8. There are walls above you, below you. move right The goal is at position 8, 6. Your current position is at position 5, 9. There are walls to your right, below you.
ASSISTANT	’thoughts’: ["The goal is located at (8, 6) and the agent’s current position is (5, 9).", ’To evaluate the distance to the goal, I will calculate the Manhattan distance, which is the sum of the absolute differences of the coordinates: |8 - 5| + |6 - 9| = 3 + 3 = 6.’, "The agent’s current position (5, 9) has walls directly below (6, 9) and to the right (5, 10). This limits movement options.", ’The possible actions for the agent are to move left to (5, 8) or move up to (4, 9). Moving down or right is not possible due to walls.’, ’Moving left to (5, 8) would bring the agent back to a previous position, while moving up to (4, 9) opens potential new pathways.’, ’Overall, the agent is relatively far from the goal, with limited movement options due to surrounding walls.’], ’final evaluation’: "The agent’s current position (5, 9) is not ideal as it is 6 steps away from the goal and has limited movement options. Moving up to (4, 9) is the best immediate option to explore new paths."

Role	**Content**
SYSTEM	You are an expert maze solver. You only respond in json.
USER (G_2)	You are playing a simple maze game. An agent is moving in the maze and the objective is to reach the goal in as few steps as possible. The possible actions are "move up", "move down", "move left", "move right". You are a highly skilled evaluator in this game, particularly adept at making accurate assessments through look-ahead of the current maze position after taking the given action. At each step I will provide you with the move history of the agent (from old to new), including where the goal is, your current position, the walls that surround the agent, the action that the agent is going to take, *along with several key variations of trajectory pieces after taking this action (and the corresponding natural language evaluations of the trajectory pieces)*. Your task is to understand these look-ahead information and summarize, derive non-trivial analysis and understanding the *the agent's position after taking the action*. Your output evaluation should be a json array with the following *two* concepts: "thoughts": Let's think step by step. Summarize the look-ahead information of the variations after taking action move down. "final evaluation": Now Concisely judge how good the chosen action is, in terms of reaching the goal. Now, please give your evaluation given action move down, the *current environment state*: The goal is at position 8, 6. Your current position is at position 5, 7. There are walls above you, below you and the look-ahead information of different variations after taking action move down: **AGGREGATION OF VALUE ESTIMATES FOR ALL VARIATIONS**
ASSISTANT	'thoughts': ["The goal is located at (8, 6), which is three rows down and one column to the left of the current position (5, 7).", "The agent attempts to move up from (5, 7) but cannot because there is a wall above.", "Each variation shows the agent getting stuck in a loop of ineffective moves after trying to move up, consistently returning to (5, 7) or moving to (5, 9), which is still not closer to the goal.", "The walls surrounding the agent restrict movement, making it impossible to progress toward the goal efficiently.", "In all variations, the agent finds itself either returning to previous positions or moving to areas blocked by walls.", "The agentś progression towards the goal becomes increasingly limited as it cannot move up, down, or directly toward the goal from its current position."], 'final evaluation': "{ "current position": "(5, 7)", "goal position": "(8, 6)", "manhattan distance": 6, "possible moves": ["move down", "move left"], "blocked moves": ["move up", "move right"], "evaluation": "The chosen action ḿove upís ineffective as it does not change the agentś position, and the surrounding walls create a dead end, making it impossible to reach the goal efficiently."}"

C.1.3 THE EFFECT OF LANGUAGE TD ON THE VALUE FUNCTION

Below we show a comparison of the Language state-action values with/without Language TD. Comparing the two Language State-action Value, we can see that by analyzing various potential outcomes, Language TD identifies both immediate and future benefits or drawbacks of actions, which is reflected in its more detailed thoughts and final evaluation. The inclusion of varied trajectory information allows Language TD to provide a more balanced and informed assessment, mitigating over-reliance on immediate next-step evaluations.

Language State-action Value (Language TD)

```
For action "move down", {"thoughts": ["The agent's current position
is (5, 7) with the goal at (8, 6).", "After attempting to move
down, the agent remains at (5, 7) due to walls above and below.",
"In Variation 1, the agent eventually moves to (6, 6), which is
one step away from the goal at (8, 6).", "Variation 2 shows that
moving down leads to blocked paths, necessitating lateral movement,
but ultimately keeps the agent away from the goal.", "Variation 3
allows the agent to successfully reach (7, 6), positioning it just
one step away from the goal.", "Variation 4 also confirms that moving
down to (6, 6) is effective, placing the agent two steps away from
the goal.", "Overall, the evaluations indicate that while the agent
faces wall constraints, moving down leads to favorable outcomes that
progress toward the goal."], "final_evaluation": {"current_position":
"(5, 7)", "distance_to_goal": 3, "proposed_action": "move down",
"evaluation": "Although the move down is initially blocked,
subsequent evaluations indicate that this action can lead to optimal
positions in the following moves. The agent can reach the goal within
two more steps if it navigates correctly."}}
```

Language State-action Value (without Language TD)

```
For action "move down", {
  "thoughts": [
    "The agent's current position is (5, 7) and the goal is at (8,
6).",
    "The possible action is to move down to (6, 7).",
    "There are walls above (4, 7) and below (6, 7), which means the
agent can only move down if it is not blocked.",
    "Moving down to (6, 7) is valid since there is no wall at that
position.",
    "After moving down, the new position will be (6, 7).",
    "The distance to the goal can be calculated as follows: the
horizontal distance is |8 - 6| = 2 and the vertical distance is |6
- 7| = 1.",
    "The total distance to the goal after moving down would be 2 + 1
= 3."
  ], "final_evaluation": {
    "current_position": "(6, 7)",
    "distance_to_goal": 3,
    "evaluation": "The agent's position after moving down is better
than the previous one, as it moves closer to the goal. There are no
walls blocking further movement towards the goal from this position."
  }
}
```

C.2 BREAKTHROUGH EXPERIMENT

C.2.1 EXPERIMENT DETAILS

Our breakthrough experiment is conducted with Openspiel (Lanctot et al., 2019) 5x5 breakthrough game and MCTS policy. We slightly

modified Openspiel's codebase so the 5x5 breakthrough game supports two lines of pieces for each side.

For the policy π, as mentioned in the paper, we choose an MCTS policy with $uct_c = 1$, 1000 simulations for search, and 100 rollouts for the random rollout evaluator. For the state dataset, we define the MCTS policies using a grid of configurations based on the number of simulations and rollouts. Specifically, we use the following values for these parameters:

- **Simulation numbers:** 2, 10, 100, 1000
- **Rollout numbers:** 1, 10, 100, 1000

This results in $4 \times 4 = 16$ unique MCTS policies, each characterized by a specific combination of simulation and rollout numbers. To generate the mixed state dataset, we pair every possible combination of these 16 policies (including self-pairing), leading to $16 \times 16 = 256$ policy pairs.

For each pair of policies, we perform the same number of rollouts where both policies interact within the environment. The states encountered during these rollouts are recorded, ensuring that the dataset captures a diverse distribution of positions. By merging the data from all policy pairs, we create the final mixed state dataset D_s, which is designed to represent states arising from a wide range of gameplay strategies and skill levels.

We use vLLM (Kwon et al., 2023b) for any LLM inference used in the experiment, including language value function inference and aggregation inference.

We sample 3000 states from the state dataset for evaluation to serve as the D^s_{test}. When computing prediction accuracy, we only count state evaluation which can pass our rule-based parser to extract the advantageous side judgment.

Here we present the hyperparameters used in our experiment:

Rollout

Parameter	Value
Parallel Environments	192
Lookahead Step	4
Lookahead Rollout Number	4
Deduplicate State	True

Table 4: Rollout Parameters

Prompting

LLM Sampling Parameter	Value
Temperature	1.0
Top K	50
Top P	0.95
Max Tokens	512

Table 5: LLM sampling parameters for prompting.

Training

Parameter	Value
Max Sequence Length	1024
Warmup Ratio	0.03
Learning Rate	2e-5
Learning Rate Scheduler	Constant
Dtype	bfloat16
Per Device Train Batch Size	4
Gradient Accumulation Step	8
Training Epoch	2
Number of GPUs	4
Distributed Framework	FSDP

Table 6: Data Collection Parameters

Evaluation. We use the same LLM sampling parameters as the prompting process.

C.2.2 LANGUAGE TD VISUALIZATION

Here we present an example of how language TD works for query and response.

Language State-action Value (Before)

Role	Content
SYSTEM	Here is the rule for the Breakthrough board game: The game is played on an 5x5 board for 2 players (white and black), with each player starting with 10 pawns. white pawns are on the first two rows and black pawns are on the last two rows. Black moves first. In each turn, players can move one of their pieces one square forward, diagonally forward if the target square is empty. Or it can capture an opponent's piece if that square is one step diagonally forward. The game ends when one player successfully break through the opponent lines -- either move a piece to the opposite last row of the board or captures all of the opponent's pieces. For board representation, we use b for black pieces, w for white pieces, and . for empty squares. (1-5) and (a-e) are used to show the rows and columns index respectively. You are a highly skilled evaluator in this game, particularly adept at making accurate assessments through look-ahead analysis of the current board position. I will provide you with current board representation, *along with several key variations starting from this position (and their corresponding natural language evaluations of the subsequent positions)*. Your task is to aggregate and compare these look-ahead information, to summarize, derive non-trivial analysis about the *current board*. Your output should include the following concepts: 1. *Tactical Considerations*: Any immediate threats, potential tactics, or vulnerabilities in the position.

	2. *Positional Evaluation*: Consideration of pawn structure, piece activity, control of key squares, and game safety. 3. *Suggested Moves*: One or two strong candidate moves for the side to move, along with a brief rationale for comparing different moves. 4. *Advantage*: Based on all previous rationale, determine if white or black takes advantage. Use <white> or <black> to represent the winning side. Your response should be informative and concise.
USER	"*The board you need to evaluate:* 5bb.b. 4b..b. 3..bw. 2w.w.. 1wwwww abcde It is White's turn. White pieces are at: d3, a2, c2, a1, b1, c1, d1, e1. Black pieces are at: a5, b5, d5, a4, d4, c3. Here are the look-ahead variations from the current board position: *Key Variations and Subsequent Evaluation:*: *Variation 1:* Description of variation's move sequence: The action sequence is: d3e4,d5e4*,a2b3,a4b3*. Move 1:White moves piece from d3 (Column d, Row 3) to e4 (Column e, Row 4). Move 2:Black moves piece from d5 (Column d, Row 5) to e4 (Column e, Row 4), capturing White piece. Move 3:White moves piece from a2 (Column a, Row 2) to b3 (Column b, Row 3). Move 4:Black moves piece from a4 (Column a, Row 4) to b3 (Column b, Row 3), capturing White piece. Subsequent position evaluation: The subsequent board is: 5bb... 4...bb 3.bb.. 2..w.. 1wwwww abcde It is White's turn. White pieces are at: c2, a1, b1, c1, d1, e1. Black pieces are at: a5, b5, d4, e4, b3, c3. The evaluation of this subsequent board is: **Current Board Analysis:** **Tactical Considerations:**

```
White's most significant tactical consideration is
the potential to capture Black's pieces on d4 and e4,
which would gain a significant advantage.  White's
pieces are well-positioned to launch a decisive attack.
Black's pieces on a5 and b5 are somewhat isolated and
vulnerable to potential attacks.

**Positional Evaluation:**
The current position slightly favors White due to their
piece activity, control of key squares, and potential
to create a strong pawn center.  White's pawns on a1
and d1 provide potential support for the central pawns.
Black's pawns are somewhat isolated, but they still
have a chance to reorganize.

**Suggested Moves:**
Based on the look-ahead analysis, two strong candidate
moves for White are:

1.  **e1-d2**:  This move captures Black's potential
piece on d2 (in case Black plays d4-d2) and creates a
strong initiative.
2.  **c2-b3**:  This move attacks Black's position and
prepares to break through to the opposite side, gaining
a strong initiative.

**Advantage:**
<white>

Overall, White has a slight advantage in this position,
with multiple ways to break through Black's lines
and gain a significant advantage.  However, Black's
position is not yet critical, and they can still
reorganize their pieces to defend.  White's advantage
is mainly due to their piece activity, control of key
squares, and potential to create a strong pawn center.

*Variation 2:*
Description of variation's move sequence:
The action sequence is:  c1b2,c3d2,e1d2*,d4e3.
Move 1:White moves piece from c1 (Column c, Row 1) to
b2 (Column b, Row 2).
Move 2:Black moves piece from c3 (Column c, Row 3) to
d2 (Column d, Row 2).
Move 3:White moves piece from e1 (Column e, Row 1) to
d2 (Column d, Row 2), capturing Black piece.
Move 4:Black moves piece from d4 (Column d, Row 4) to
e3 (Column e, Row 3).

Subsequent position evaluation:
The subsequent board is:

5bb.b.
4b....
3...wb
2wwww.
1ww.w.
abcde

It is White's turn.
White pieces are at:  d3, a2, b2, c2, d2, a1, b1, d1.
Black pieces are at:  a5, b5, d5, a4, e3.
```

	The evaluation of this subsequent board is: **Tactical Considerations:** The current position presents a strong opportunity for White to create a breakthrough on the 'd' file, as demonstrated by the two variations. Black's pawn structure is somewhat disjointed, making it difficult to respond to White's advances on the 'd' and 'e' files. **Positional Evaluation:** White's pawn structure is more compact and cohesive, controlling key central squares. The pawns on 'a' and 'b' files provide additional support for the potential breakthrough on 'd' and 'e'. Black's pieces on 'a5' and 'd5' are somewhat isolated, making it difficult to coordinate a defense. **Suggested Moves:** Considering the look-ahead variations, the strongest candidate move for White is **d3d4**. This move puts pressure on Black's position and sets up the potential breakthrough on the 'd' and 'e' files. The rationale for this move is that it creates an immediate threat and forces Black to respond, potentially weakening their position. Another possible move could be **d2d3**, but it is less aggressive and does not put as much pressure on Black. **Advantage:** Based on the analysis, the advantage is **<white>**. The two variations demonstrate a clear path to victory for White, and Black's position appears to be under significant pressure. Please provide your analysis and understanding of the current board position based on the provided look-ahead information. Your response should be informative and concise.
ASSISTANT	"**Current Board Analysis:** **Tactical Considerations:** The primary tactical consideration for White is to utilize the pawns on the 'd' file to create a breakthrough, potentially capturing Black's pieces on d4 and e4. Additionally, White should focus on controlling the key squares and putting pressure on Black's somewhat isolated pieces on a5 and b5. Black needs to reorganize their pawns and pieces to defend against White's central pawn advance. **Positional Evaluation:** The position slightly favors White due to their more compact and cohesive pawn structure, piece activity, and control of key central squares. White's pawns on the 'a' and 'b' files provide additional support for the potential breakthrough on the 'd' and 'e' files. Black's pawns are somewhat isolated, making it challenging to coordinate a defense. **Suggested Moves:**

```
Based on the look-ahead analysis, two strong candidate
moves for White are:

1.  **d3d4**:  This move puts pressure on Black's
position, sets up a potential breakthrough on the 'd'
and 'e' files, and forces Black to respond.
2.  **e1-d2**:  This move captures Black's potential
piece on d2 and creates a strong initiative.

**Advantage:**
<white>

Overall, White has a slight advantage in this position,
with multiple ways to break through Black's lines and
gain a significant advantage.  The two variations
demonstrate a clear path to victory for White, and
Black's position appears to be under significant
pressure.  White's advantage is mainly due to their
piece activity, control of key squares, and potential
to create a strong pawn center.

The provided look-ahead information suggests that White
should prioritize controlling the central squares,
putting pressure on Black's position, and utilizing
the pawns on the 'd' file to create a breakthrough."
```

C.2.3 Comparison of language value function

Here we present another visualization comparison between the initial *LLaMA-3.1-8B-Instruct* Model and our trained model. The evaluation is conducted on the board 5..b.b 4.bb.. 3.w... 2.w... 1ww..w abcde, where white takes large advantage in this place. By our language TD training, the language value function can accurately evaluate this board and generate insightful annotation, largely decreasing the hallucination.

Language State Value (Before)

Board: 5..b.b 4.bb.. 3.w... 2.w... 1ww..w abcde, white turn

```
 Based on the current board position, here's the evaluation for
White's turn:

*Tactical Considerations:*

- Black has an opportunity to attack White's pawn at e1 with a pawn
from e5 or c5, which might put White in a difficult position.
- White's pawn at b3 and b2 is a target for Black's pawns at b4 and
c4.
- White needs to focus on safely advancing their pawns and creating a
barrier for Black's pieces.

*Positional Evaluation:*

- White has a relatively weak pawn structure, with some pawns on the
first row.
- The center is somewhat controlled by White, but Black has a solid
presence on the fifth row.
- Black has a strategic advantage in pawn structure, but White has
the opportunity to influence the game with proper planning.

*Suggested Moves:*

1. Move the pawn from b2 to b3. This move has the goal of preparing
to defend and possibly create a barrier against Black's pieces.
```

```
2. Move the pawn from a1 to a2. This move aims to create more space
and put pressure on Black's pieces, which will make it harder for
them to maneuver.

*Advantage:*
Based on the current board position, <black> is favored.
```

Language State Value (After training)

Board: 5..b.b 4.bb.. 3.w... 2.w... 1ww..w abcde, white turn

```
**Analysis and Evaluation of the Current Board Position**

**Tactical Considerations:**
The key variation shows an immediate threat of White breaking through
Black's lines. White's piece on b3 is strategically positioned to
capture Black's piece on c4, which creates an opportunity for a
decisive push.

**Positional Evaluation:**
White has a significant advantage in terms of piece activity and
control of key squares. With pieces on the third and second rows,
White is poised to push forward and break through Black's lines.
Black's pieces are concentrated on the fifth and fourth rows, which
limits their mobility and ability to respond to White's attack.

**Suggested Moves:**
Two strong candidate moves for White are:

1. b3-c4*: This move captures Black's piece and creates a weakness in
Black's position, allowing White to potentially break through.
2. b2-c3: This move develops another White piece and prepares to
support the potential breakthrough on the c-file.

However, given the look-ahead variations, the most promising move is
b3-c4*, as it leads to a terminal state where White wins.

**Advantage:**
Based on the analysis, White has a decisive advantage. The ability
to capture Black's piece on c4 and break through Black's lines gives
White a strong initiative and a clear path to victory. Therefore,
<white> is the winning side.
```

C.3 TIC-TAC-TOE EXPERIMENT

Our experiments are conducted on the Tic-tac-toe environment (Ju, 2024), a standard two-player game environment that provides a clear testbed for evaluating sequential decision-making. The implementation uses several key components: `https://github.com/haje01/gym-tictactoe` for the Tic-tac-toe environment, FSDP (Fully Sharded Data Parallel) for distributed model training, vLLM (Kwon et al., 2023b) for efficient language model inference, and a custom parallel framework for trajectory collection. The environment is wrapped with a parallel execution layer to enable efficient batch processing of multiple games simultaneously.

For policy evaluation, we employ Monte Carlo sampling to estimate state-action values, requiring complete game trajectories until terminal states. Policy improvement utilizes a structured sampling approach with temperature-controlled exploration and frequency-based action selection.

To manage computational resources efficiently, we employ gradient checkpointing and use the FSDP strategy across our GPU cluster. This configuration allows us to train larger language models while maintaining reasonable memory requirements and training speed.

The parallel data collection framework is designed to maximize GPU utilization during training. It maintains a queue of active environments and processes state-action transitions in batches, significantly reducing the overall training time compared to sequential execution.

C.3.1 TRAINING INFRASTRUCTURE

Data Collection Pipeline

For each training iteration, we collect data using the following configuration:

Parameter	**Value**
Parallel Environments	64
Trajectories per Iteration	512
Monte Carlo Samples (K_{MC})	5
Policy Samples per State (N_{sample})	10
Top-k Actions	10

Table 8: Data Collection Parameters

Model Architecture and Training

Both policy and value networks are trained from *LLaMA-3.1-8B-Instruct* using the following parameters:

Hyperparameter	**Value**
Learning Rate	1e-5
Learning Rate Schedule	Constant
Training Epochs per Iteration	2
FSDP Configuration	Full Sharding
Gradient Checkpointing	Enabled
Batch Size	8
Max Sequence Length	1024
Training Hardware	4 × H100 GPUs

Table 9: Model Training Hyperparameters

Value Function Buffer Management

To prevent catastrophic forgetting, we maintain experience buffers with the following configuration:

Parameter	**Value**
Buffer History Length (K_{buffer})	3 iterations
Merging Strategy	Equal sampling
Buffer Content	State-action pairs with MC estimates

Table 10: Buffer Management Configuration

C.3.2 ALGORITHM IMPLEMENTATION DETAILS

Policy Evaluation Details

For Monte Carlo evaluation, we use the following configuration:

Parameter	**Value**
MC Trajectories per State-Action	5
Trajectory Completion	Full game
Value Aggregation	Average over returns
Sampling Temperature	0.7
Action Space Size	9 positions (0-8)

Table 11: Policy Evaluation Configuration

C.3.3 VISUALIZATION

Below we show a comparison of the Language state-action values before and after Language MC estimation. The evaluation is conducted on the following board:

O	X	3
4	O	O
7	X	9

Through analyzing multiple complete game trajectories, Language MC helps the value function develop a comprehensive understanding of long-term consequences of actions, rather than just immediate rewards. This is particularly evident in the enhanced evaluation quality after MC estimation, where the model provides more strategic insights and nuanced assessment of game positions.

Language State-action Value (Before)

Role	**Content**
SYSTEM	You are a player of the game of Tic Tac Toe. The game goal is to get multiple of one's own symbols in a row, either horizontally, vertically, or diagonally, before the opponent does. If all nine squares are filled and no player has three in a row, the game is a draw. The board consists of "O", "X" and grid numbers. The grid number indicates an empty grid. You are learning how to evaluate a (board, action) pair in the Tic Tac Toe game by playing the game given the (board, action) pair and reflecting on the playing history. The playing history depicts a game in progress with a final result. Your answer consists of two elements: - "thought": Let's think step by step. Generate your detailed evaluation over the (board, action) pair by merely reflecting the playing history after this pair from different perspectives. You should only rely on the playing history as context and shouldn't evaluate the game with your own judgment. Your evaluation should contain the following elements: Win probability, Threat, and Potential strategies. - "final_evaluation": After all of your thoughts, judge which side takes advantage. 1 means you think 'O' takes advantage, -1 means you think 'X' takes advantage, and 0 means you think the game will be a draw. You should output your answer in the JSON format. **EXAMPLE:** user: The (board, action) to evaluate is O's turn: Board: O | O | X --------- 4 | X | 6 ---------

	7 | 8 | 9
	Action: The O's move is 4.
	Below is the rollout sequence after this (board, action):
	After O's move 4, the board position is:
	O | O | X
	--------- 	
	O | X | 6
	--------- 	
	7 | 8 | 9
	After X's move 7, the board position is:
	O | O | X
	--------- 	
	O | X | 6
	--------- 	
	X | 8 | 9
	The game is over. X wins. X wins by occupying the positions 3, 5, 7.
	assistant:
	{
	"thought": {
	"Reflection": "It appears that the initial board position
	O | O | X
	--------- 	
	4 | X | 6
	--------- 	
	7 | 8 | 9
	and action 4 were not favorable for O, as X was able to block on O's moves and ultimately win the game.",
	"Win probability": "The win probability for X is large, while the win probability for O is low.",
	"Threat": "X has played center 5 and corner 3. X can win by playing corner 7. O was able to occupy 1, 4, 7 and create a potential winning opportunity.",
	"Potential strategies": "Potential strategies for O include playing the corner 7 to block X as opposite corner and win by occupying 1, 4, 7. X could have occupied 3, 5, 7 to win the game. X has already occupied 3, 5, and there is 1 step to complete the plan."
	},
	"final_evaluation": -0.9
	}

USER	The board to evaluate is O's turn: Board: O | X | 3 --------- 4 | O | 6 --------- 7 | X | 9 Action: The O's move is 3. Below is the rollout sequence 1 after this (board, action): After O taking action 3, the board position is: O | X | O --------- 4 | O | 6 --------- 7 | X | 9. After X taking action 4, the board position is: O | X | O ---------

	X | O | 6 --------- 7 | X | 9. After O taking action 7, the board position is: O | X | O --------- X | O | 6 --------- O | X | 9. The game is over. O wins. O wins by occupying the positions [3, 5, 7]. Now generate your evaluation for the (board, action) pair.

ASSISTANT	{
	"thought": {
	"Reflection": "The initial board position
	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9
	and action 3 were favorable for O, as O was able to occupy a strategic position and ultimately win the game.",
	"Win probability": "The win probability for O is high, while the win probability for X is low.",
	"Threat": "O has played center 5 and corner 3. O can win by playing corner 7. X has played center 5's opposite corner 9 but failed to block O's winning line.",
	"Potential strategies": "Potential strategies for O include playing the corner 7 to win by occupying 3, 5, 7. X could have occupied 4, 6, 9 to block O's winning line but failed to do so."
	},
	"final_evaluation": 0.9
	}

Language State-action Value (After)

Role	Content
SYSTEM	You are a player of the game of Tic Tac Toe. The game goal is to get multiple of one's own symbols in a row, either horizontally, vertically, or diagonally, before the opponent does. If all nine squares are filled and no player has three in a row, the game is a draw. The board consists of "O", "X" and grid numbers. The grid number indicates an empty grid. You are learning how to evaluate a (board, action) pair in the Tic Tac Toe game by playing the game given the (board, action) pair and reflecting on the playing history. The playing history depicts a game in progress with a final result. Your answer consists of two elements: - "thought": let's think step by step. Generate your detailed evaluation over the (board, action) pair by merely reflecting the playing history after this pair from different perspectives. You should only rely on the playing history as context and don't evaluate the game with your own judgment. Your evaluation should contain the following elements: Win probability, Threat, and Potential strategies.

<table>
<tr><td></td><td>- "final_evaluation": After all of your thoughts, judge which side takes advantage. 1 means you think 'O' takes advantage, -1 means you think 'X' takes advantage, and 0 means you think the game will be a draw.
You should output your answer in the JSON format.
EXAMPLE:
user: The (board, action) to evaluate is O's turn:
Board:
O | O | X
----
4 | X | 6
----
7 | 8 | 9
Action: The O's move is 4.
Below is the rollout sequence after this (board, action):
After O's move 4, the board position is:
O | O | X

O | X | 6
----
7 | 8 | 9
After X's move 7, the board position is:
O | O | X

O | X | 6

X | 8 | 9
The game is over. X wins. X wins by occupying the positions 3, 5, 7.
assistant: {
"thought": {
"Reflection": "It appears that the initial board position
O | O | X
----
4 | X | 6
----
7 | 8 | 9
and action 4 were not favorable for O, as X was able to block on O's moves and ultimately win the game.",
"Win probability": "The win probability for X is large, while the win probability for O is low.",
"Threat": "X has played center 5 and corner 3. X can win by playing corner 7. O was able to occupy 1, 4, 7 and create a potential winning opportunity.",
"Potential strategies": "Potential strategies for O include playing the corner 7 to block X as opposite corner and win by occupying 1, 4, 7. X could have occupied 3, 5, 7 to win the game. X has already occupied 3, 5, and there is 1 step to complete the plan."
},
"final_evaluation": -0.9
}</td></tr>
<tr><td>USER</td><td>The board to evaluate is O's turn:
Board:
O | X | 3
----
4 | O | 6
----
7 | X | 9</td></tr>
</table>

	Action: The O's move is 9. Below is the rollout sequence 1 after this (board, action): After O taking action 9, the board position is: O | X | 3 --------- 4 | O | 6 --------- 7 | X | O. The game is over. O wins. O wins by occupying the positions [1, 5, 9]. Below is the rollout sequence 2 after this (board, action): After O taking action 9, the board position is: O | X | 3 --------- 4 | O | 6 --------- 7 | X | O. The game is over. O wins. O wins by occupying the positions [1, 5, 9]. Below is the rollout sequence 3 after this (board, action): After O taking action 9, the board position is: O | X | 3 --------- 4 | O | 6 --------- 7 | X | O. The game is over. O wins. O wins by occupying the positions [1, 5, 9]. Below is the rollout sequence 4 after this (board, action): After O taking action 9, the board position is: O | X | 3 --------- 4 | O | 6 --------- 7 | X | O. The game is over. O wins. O wins by occupying the positions [1, 5, 9]. Below is the rollout sequence 5 after this (board, action): After O taking action 9, the board position is: O | X | 3 --------- 4 | O | 6 --------- 7 | X | O. The game is over. O wins. O wins by occupying the positions [1, 5, 9]. Now generate your evaluation for the (board, action) pair.

ASSISTANT	{
	"thought": {
	"Reflection": "It appears that the initial board position

	O | X | 3 --------- 4 | O | 6 --------- 7 | X | 9 and action 9 were highly favorable for O, as O was able to win the game immediately after taking action 9 in all rollout sequences.", "Win probability": "The win probability for O is extremely high, while the win probability for X is extremely low.", "Threat": "There were no threats from X in any of the rollout sequences, as O was able to win the game immediately after taking action 9.", "Potential strategies": "The potential strategy for O was to occupy the corner 9 to win the game by occupying the positions [1, 5, 9]. This strategy was successful in all rollout sequences." }, "final_evaluation": 1 }

Policy Improvement Details

Below we show a comparison of the Language Policy Improvement procedure at the beginning of training and at the end. The evaluation is conducted on the following board:

O	X	3
4	O	6
7	X	9

The evolution in policy quality demonstrates how our framework effectively leverages value function assessments to enhance decision-making. The improved policy shows more sophisticated strategic reasoning, considering not just immediate moves but also their implications for future game states. This transformation is achieved while maintaining our restriction against using the model's pre-trained game knowledge, instead relying purely on learned evaluation capabilities.

Language Policy Improvement (Beginning)

Role	**Content**
SYSTEM	You are playing the game tic-tac-toe on a 3*3 board. Tic Tac Toe is a two-player game played on a grid. Players take turns marking a space with their respective symbol. The goal is to get multiple of one's own symbols in a row, either horizontally, vertically, or diagonally, before the opponent does. If all nine squares are filled and no player has three in a row, the game is a draw. The board consists of "O", "X" and grid number. The grid number indicates an empty grid. Your task is to determine the best move for the next player based on the given board position and the next player. The evaluations of (board, action) pairs after possible moves are given. **DO NOT** judge the board based on your knowledge, only use the evaluations to determine the best move. The evaluation for the next board is in the format of a JSON, consisting of two elements: - **"thought"**: Evaluation of the board and action pair.

	- **"final_evaluation"**: Judge which side takes advantage. 1 means 'O' takes advantage, -1 means 'X' takes advantage, and 0 means the game will be a draw.

USER	Here is the board position and the next player is O:
	Board:
	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9.
	The possible moves are [3, 7, 4, 9].
	The following are the boards after each possible move:
	### Evaluation for taking action 3:
	{
	"thought": {
	"Reflection": "The initial board position
	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9
	and action 3 were favorable for O, as O was able to win the game in all rollout sequences.",
	"Win probability": "The win probability for O is high, while the win probability for X is low.",
	"Threat": "O has played corner 1 and center 5. O can win by playing corner 3, 4, 7 or 9. X has played corner 2 and 8 but failed to block O's winning lines.",
	"Potential strategies": "Potential strategies for O include playing the corner 3 to create a potential winning opportunity. O could have occupied 1, 4, 7 or 1, 5, 9 or 3, 5, 7 to win the game. X could have blocked O's winning lines by playing 4 or 7 but failed to do so."
	},
	"final_evaluation": 0.9
	}
	### Evaluation for taking action 7:
	{
	"thought": {
	"Reflection": "The initial board position
	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9
	and action 7 were favorable for O, as O was able to win the game in all rollout sequences.",
	"Win probability": "The win probability for O is high, while the win probability for X is low.",
	"Threat": "O has played corner 1 and center 5. O can win by playing corner 7 and then 9, or 4 and then 7. X has played corner 3 and 8 but failed to block O's winning lines.",
	"Potential strategies": "Potential strategies for O include playing the corner 7 to create a potential winning opportunity and then playing 9 or 4 to win the game. X could have occupied 3, 6, 9 to block O's winning lines but failed to do so."
	},
	"final_evaluation": 0.9
	}
	### Evaluation for taking action 4:

```
{
"thought": {
"Reflection": "The initial board position
O | X | 3
---------
4 | O | 6
---------
7 | X | 9
and action 4 were favorable for O, as O was able to win
the game in all rollout sequences.",
"Win probability": "The win probability for O is high,
while the win probability for X is low.",
"Threat": "O has played center 5 and corner 1. O can
win by playing corner 7 or 9. X has played center 6
and corner 3 but failed to block O's winning lines.",
"Potential strategies": "Potential strategies for O
include playing the corner 7 or 9 to win the game. X
could have occupied 3, 6, 9 to block O's winning lines
but failed to do so."
},
"final_evaluation": 0.9
}
```

Evaluation for taking action 9:

```
{
"thought": {
"Reflection": "It appears that the initial board
position
O | X | 3
---------
4 | O | 6
---------
7 | X | 9
and action 9 were favorable for O, as O was able to win
the game in both rollout sequences.",
"Win probability": "The win probability for O is high,
while the win probability for X is low.",
"Threat": "O has played center 5 and corner 1. O
can win by playing corner 9. X has played corner 3
and center 6 is empty, but X was not able to block O's
moves and ultimately lost the game.",
"Potential strategies": "Potential strategies for O
include playing the corner 9 to create a potential
winning opportunity and then playing 3 or 7 to win
the game. X could have occupied 3, 6, 7 to block O's
winning opportunity, but X was not able to do so."
},
"final_evaluation": 0.9
}
```

Now, please give your evaluation and the best move for
O based on the given board position:

```
O | X | 3
---------
4 | O | 6
---------
7 | X | 9.
```

You should output your answer in the JSON format. Your
answer consists of two elements:
- **"thought"**: Let's think step by step. Generate
your detailed reflection by analyzing the next board
positions and their evaluations.

	- **"best_move"**: The best move for the next player. The move should be in the format of a number from 1 to 9, indicating the position on the board. And the move should be in the available positions [3, 7, 4, 9]. Don't output extra information except for the JSON format.

ASSISTANT	{
	"thought": "Analyzing the given board position
	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9 and the evaluations of the possible moves, we can see that all four moves (3, 7, 4, 9) have the same final evaluation of 0.9, indicating that O has a high chance of winning with any of these moves. However, we should consider the potential strategies and threats for each move.
	Move 3 and 7 allow O to create a potential winning opportunity by playing the corner, while move 4 allows O to play the center and create a potential winning opportunity by playing the corner 7 or 9. Move 9 allows O to play the corner and create a potential winning opportunity by playing 3 or 7.
	Considering the potential strategies, all four moves seem to have similar advantages. However, move 7 seems to have a slight advantage as it allows O to create a potential winning opportunity and then play 9 to win the game, while also putting pressure on X to block the winning line.
	Therefore, the best move for O is 7.",
	"best_move": 7
	}

Language Policy Improvement (End)

Role	**Content**
SYSTEM	You are playing the game tic-tac-toe on a 3*3 board. Tic Tac Toe is a two-player game played on a grid. Players take turns marking a space with their respective symbol. The goal is to get multiple of one's own symbols in a row, either horizontally, vertically, or diagonally, before the opponent does. If all nine squares are filled and no player has three in a row, the game is a draw. The board consists of "O", "X" and grid numbers. The grid number indicates an empty grid. Your task is to determine the best move for the next player based on the given board position and the next player. The evaluations of (board, action) pairs after possible moves are given. **DO NOT** judge the board based on your knowledge, only use the evaluations to determine the best move. The evaluation for the next board is in the format of a JSON, consisting of two elements: - **"thought"**: Evaluation of the board and action pair. - **"final_evaluation"**: Judge which side takes advantage. 1 means 'O' takes advantage, -1 means 'X' takes advantage, and 0 means the game will be a draw.
USER	Here is the board position and the next player is O: Board:

	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9.
	The possible moves are [9].
	The following are the boards after each possible move:
	### Evaluation for taking action 9:
	{
	"thought": {
	"Reflection": "It appears that the initial board position
	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9
	and action 9 were highly favorable for O, as O was able to win the game immediately after taking action 9 in all rollout sequences.",
	"Win probability": "The win probability for O is extremely high, while the win probability for X is extremely low.",
	"Threat": "There were no threats from X in any of the rollout sequences, as O was able to win the game immediately after taking action 9.",
	"Potential strategies": "The potential strategy for O was to occupy the corner 9 to win the game by occupying the positions [1, 5, 9]. This strategy was successful in all rollout sequences."
	},
	"final_evaluation": 1
	}
	Now, please give your evaluation and the best move for O based on the given board position:
	O | X | 3
	--------- 	
	4 | O | 6
	--------- 	
	7 | X | 9.
	You should output your answer in the JSON format. Your answer consists of two elements:
	- **"thought"**: Let's think step by step. Generate your detailed reflection by analyzing the next board positions and their evaluations.
	- **"best_move"**: The best move for the next player. The move should be in the format of a number from 1 to 9, indicating the position on the board. And the move should be in the available positions [9].
	Don't output extra information except for the JSON format.

ASSISTANT	{ "thought": "Based on the evaluation of the board position after taking action 9, it appears that this move is highly favorable for O. The win probability for O is extremely high, while the win probability for X is extremely low. There were no threats from X in any of the rollout sequences, as O was able to win the game immediately after taking action 9. The potential strategy for O was to occupy the corner 9 to win the game by occupying the positions [1, 5, 9]. This strategy was successful in all rollout sequences. Therefore, taking action 9 is the best move for O.",

```
"best_move": 9
}
```

C.3.4 Baseline Implementation Details

For our PPO baseline, we implement a custom PPO agent adapted for language models. The agent consists of two key components: an actor network based on LLaMA-3.1-8B-Instruct for action selection, and a critic network for value estimation.

Architecture Details The actor network uses *LLaMA-3.1-8B-Instruct*'s architecture with added special tokens for moves 1-9. The critic network consists of a two-layer MLP (Multi-Layer Perceptron) that takes the last hidden state from the actor as input:

- Input dimension: *LLaMA-3.1-8B* hidden size
- Hidden layer: ReLU activation
- Output layer: Single value with Tanh activation

Training Configuration The PPO training uses the following hyperparameters:

Parameter	Value
Learning Rate (Actor)	1e-5
Learning Rate (Critic)	1e-5
PPO Epochs	1
Batch Size	16
Gradient Accumulation Steps	2
Value Loss Coefficient	0.5
Entropy Coefficient	0.01
PPO Clip Range	0.2
Discount Factor (γ)	0.99
GAE Parameter (λ)	0.95

Table 16: PPO Training Hyperparameters

Implementation Details For each training iteration:

- We collect 512 complete game trajectories using 8 parallel environments
- Actions are selected using a temperature-controlled sampling ($T = 0.7$) over the logits
- The policy proposes $N_{sample} = 1$ candidate actions per state and selects top-k ($k = 1$) based on sampling frequency
- We use 1 A100 GPU for training
- Gradient checkpointing is enabled to optimize memory usage

Both networks are trained using AdamW optimizer with weight decay of 0.01. We maintain experience buffers with a history length of 1 iterations for on-policy optimization.

C.3.5 Additional Experiments

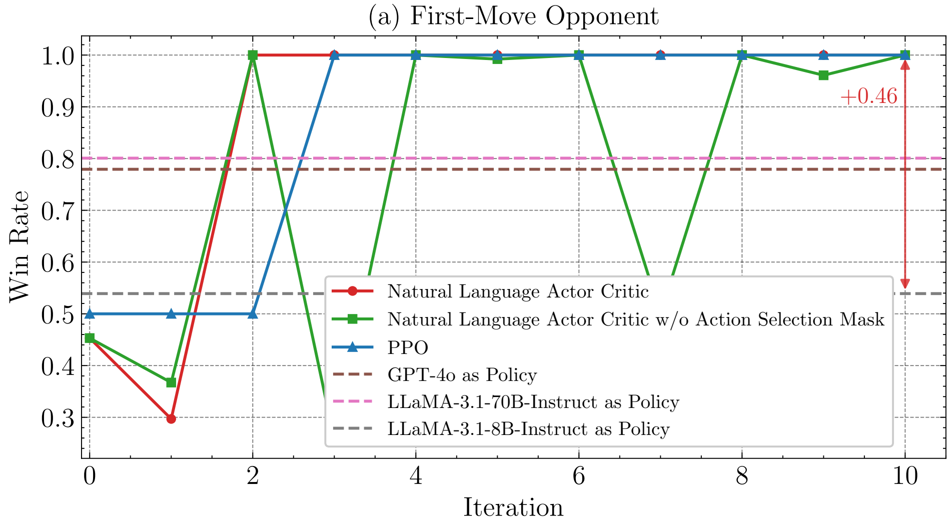


Figure 6: Training results against the deterministic first-move opponent.

C.3.6 Additional Ablation Studies

Experience Buffer Size K_{buffer} Study We examine the effect of experience buffer size K_{buffer} on learning in Fig. 3(c), demonstrating that when $K_{buffer} = 3$ helps maintain consistent performance by preserving past experiences.

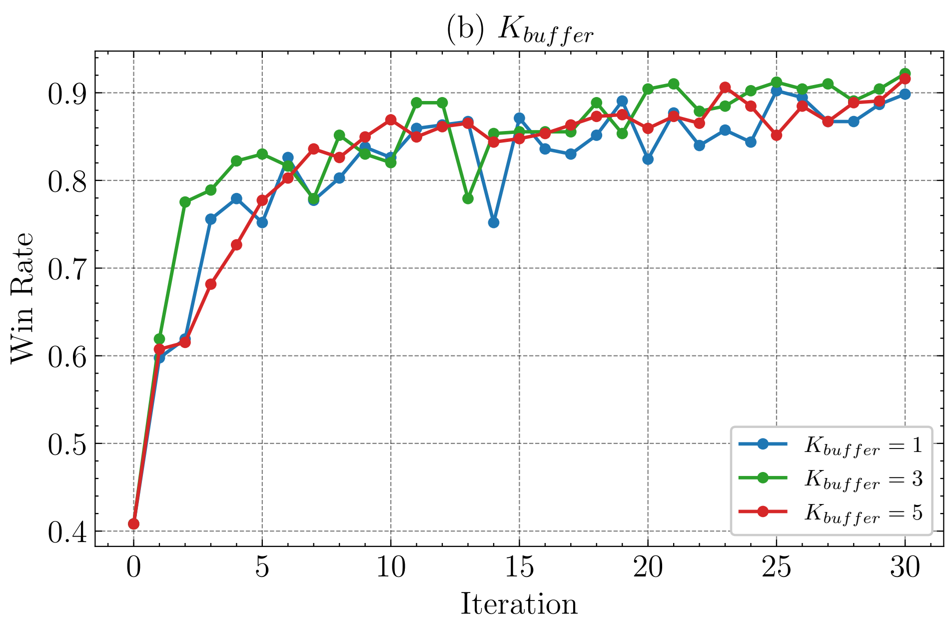


Figure 7: Ablation studies on experience buffer size K_{buffer}.

Action Selection Mask Parameter Study To complement the main ablation studies presented in Section 5.3.3, we conduct additional experiments on the action selection parameter top-m. Our results demonstrate that increasing m from 2 to 10 enhances exploration capabilities while maintaining training stability. Specifically, larger m value (10) show more stable training compared to smaller values (2-5).

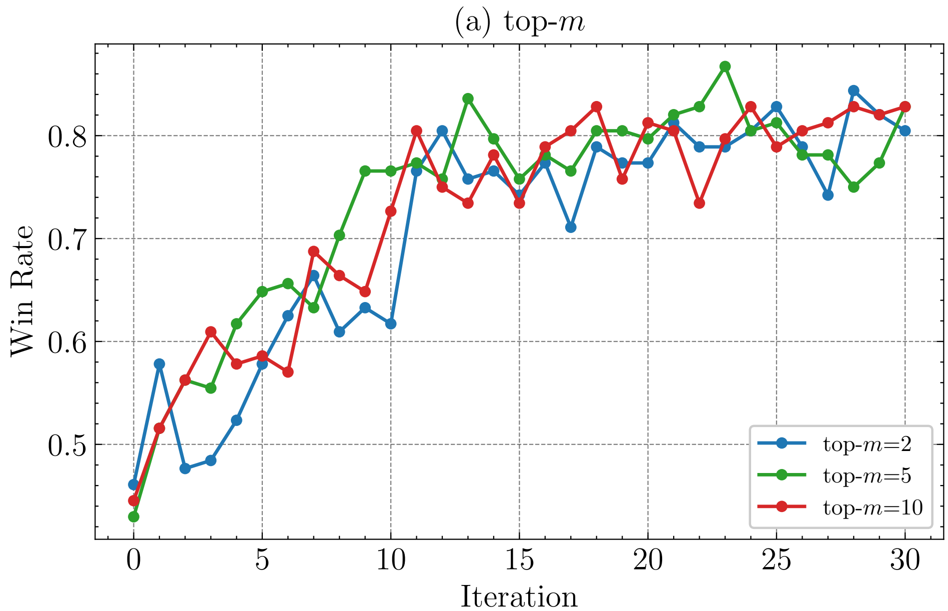


Figure 8: Ablation studies on action selection parameter top-m.

D PROMPTS

Here we provide all the prompt templates in three experiments.

D.1 MAZE EXPERIMENT

Language Value/Evaluation Prompt

```
EVAL_USER_PROMPT_S_V = f"""\
You are playing a simple maze game. An agent is moving in the maze
and the objective is to reach the goal in as few steps as possible.

The possible actions are "move up\n", "move down\n", "move left\n",
"move right\n".

You are a highly skilled evaluator in this game and is expected to
function similar to state value function in reinforcement learning.
At each step I will provide you with the move history of the
agent (from old to new), including where the goal is, your current
position, the walls that surround the agent. Your output evaluation
should be a json array that includes the following concepts:
- "thoughts": Let's think step by step. Generate your detailed
thought process and evaluation of the agent's position and the
distance towards the goal.
- "final_evaluation": Concisely judge how good the agent's current
position is compared to all the possible positions in the maze, in
terms of reaching the goal.

Let's start a new game. Now, please give your evaluation of the
current state given the move history of the agent:
```
{{game_content}}
```

"""
```

Language TD Prompt

```
EVAL_USER_PROMPT_S_TD_G2_new = f"""\
You are playing a simple maze game. An agent is moving in the maze
and the objective is to reach the goal in as few steps as possible.

The possible actions are "move up\n", "move down\n", "move left\n",
"move right\n".

You are a highly skilled evaluator in this game, particularly adept
at making accurate assessments through look-ahead of the current maze
position after taking the given action. At each step I will provide
you with the move history of the agent (from old to new), including
where the goal is, your current position, the walls that surround
the agent, the action that the agent is going to take, *along with
several key variations of trajectory pieces after taking this action
(and the corresponding natural language evaluations of the trajectory
pieces)*.
Your task is to understand these look-ahead information and
summarize, derive non-trivial analysis and understanding the *the
agent's position after taking the action*. Your output evaluation
should be a json array with the following *two* concepts:
- "thoughts": Let's think step by step. Summarize the look-ahead
information of the variations after taking action {{chosen_action}}.
- "final_evaluation": Now Concisely judge how good the chosen action
is, in terms of reaching the goal.

Now, please give your evaluation given action {{chosen_action}}, the
*current environment state*:
```
{{game_content}}
```

and the look-ahead information of different variations after taking
action {{chosen_action}}:
"""
```

Policy Improvement Prompt

```
POLICY_IMPROVEMENT_PROMPT_TD = f"""\
You are playing a simple maze game. An agent is moving in the maze
and the objective is to reach the goal in as few steps as possible.

Your task is to determine the best action for the next time step
given the current state (the move history of the agent (from old to
new), including where the goal is, your current position, the walls
that surround the agent).

Your possible actions are "move up\n", "move down\n", "move left\n",
"move right\n".

The evaluations of the agent after possible actions are given. Each
of them consists of two elements:
- "thoughts": Summarization of the look-ahead information of the
variations after taking the chosen action.
- "final_evaluation": Judge how good the chosen action is, in terms
of reaching the goal.

DO NOT judge the action based on your exterior knowledge, only use
the given evaluations to determine the best move.

Here are the evaluations of each possible action:
```

```
For action "move up", {{evaluations_up}}
,
For action "move down", {{evaluations_down}}
,
For action "move left", {{evaluations_left}}
,
For action "move right", {{evaluations_right}}

Return the best action (choose only one from the possible actions)
given the evaluations in a json array with a key "action".
"""
```

D.2 BREAKTHROUGH EXPERIMENT

Language Value/Evaluation Prompt

```
GAME_RULE_PROMPT = """Here is the rule for the Breakthrough board
game:
The game is played on an 5x5 board for 2 players (white and black),
with each player starting with 10 pawns. white pawns are on the first
two rows and black pawns are on the last two rows.
Black moves first. In each turn, players can move one of their pieces
one square forward, diagonally forward if the target square is empty.
Or it can capture an opponent's piece if that square is one step
diagonally forward.
The game ends when one player successfully break through the opponent
lines -- either move a piece to the opposite last row of the board or
captures all of the opponent's pieces.

For board representation, we use b for black pieces, w for white
pieces, and . for empty squares. (1-5) and (a-e) are used to show
the rows and columns index respectively."""

EVAL_SYSTEM_PROMPT = f"""{GAME_RULE_PROMPT}

You are a highly skilled evaluator in this game. I will provide you
with specific board information representing the current board. Your
output should include the following concepts:
1. *Tactical Considerations*: Any immediate threats, potential
tactics, or vulnerabilities in the position.
2. *Positional Evaluation*: Consideration of pawn structure, piece
activity, control of key squares, and game safety.
3. *Suggested Moves*: One or two strong candidate moves for the side
to move, along with a brief rationale for comparing different moves.
4. *Advantage*: Based on all previous rationale, determine if white
or black takes advantage. Use <white> or <black> to represent the
winning side.

Your response should be informative and concise."""

EVAL_USER_PROMPT = """*The board you need to evaluate:*
{board}"""
```

Language TD Prompt

```
TD_SYSTEM_PROMPT = f"""{GAME_RULE_PROMPT}

You are a highly skilled evaluator in this game, particularly adept
at making accurate assessments through look-ahead analysis of the
current board position.
I will provide you with current board representation, *along with
several key variations starting from this position (and their
```

```
corresponding natural language evaluations of the subsequent
positions)*.

Your task is to aggregate and compare these look-ahead information,
to summarize, derive non-trivial analysis about the *current board*.
Your output should include the following concepts:
1. *Tactical Considerations*: Any immediate threats, potential
tactics, or vulnerabilities in the position.
2. *Positional Evaluation*: Consideration of pawn structure, piece
activity, control of key squares, and game safety.
3. *Suggested Moves*: One or two strong candidate moves for the side
to move, along with a brief rationale for comparing different moves.
4. *Advantage*: Based on all previous rationale, determine if white
or black takes advantage. Use <white> or <black> to represent the
winning side.

Your response should be informative and concise."""

TD_USER_PROMPT = """*The board you need to evaluate:*

{board}

Here are the look-ahead variations from the current board position:
*Key Variations and Subsequent Evaluation:*:

{variations}

Please provide your analysis and understanding of the current board
position based on the provided look-ahead information.
Your response should be informative and concise."""

VARIATION_PROMPT = """*Variation {i}:*
Description of variation's move sequence:
{move_desc}

Subsequent position evaluation:
{subsequent_eval}"""

SUBSEQUENT_PROMPT = """The subsequent board is:

{sub_board}

The evaluation of this subsequent board is:

{sub_eval}"""
```

D.3 TIC-TAC-TOE EXPERIMENT

Policy Inference Prompt

```
POLICY_INFERENCE_SYSTEM_PROMPT = """\
You are an expert agent at playing the game tic-tac-toe on a 3*3
board. Tic Tac Toe is a two-player game played on a grid.
Players take turns marking a space with their respective symbol.
The goal is to get multiple of one's own symbols in a row, either
horizontally, vertically, or diagonally, before the opponent does.
If all nine squares are filled and no player has three in a row, the
game is a draw.

Your task is to choose the best move given board position.
The board consists of "O", "X" and grid number. The grid number
indicates empty grid.
You should output your answer in the json format. Your answer
consists of two elements:
```

```
- "thought": let's think step by step. Generate your detailed thought
process and plan for the next move.
- "best_move": the best move for the next player. The move should be
in the format of a number from 1 to 9, indicating the position on the
board. And the move should be in the available positions."""

POLICY_INFERENCE_EXAMPLE_USER_PROMPT = """\
Here is the board position and the next player is O:
O | O | X
---------
4 | X | 6
---------
7 | 8 | 9
The available move positions are 4, 6, 7, 8, 9.
"""

POLICY_INFERENCE_EXAMPLE_ASSISTANT_PROMPT = """\
{"thought": "It appears that the initial board position
O | O | X
---------
4 | X | 6
---------
7 | 8 | 9 was favorable for X, as X has occupied the positions 3,
5 and X can win by occupying the position 7. O has occupied the
positions 1, 2. Therefore, the best move for O is to occupy the
position 7 to block X and create a potential winning opportunity by
occupying the positions 1, 4, 7.", "best_move": 7}
"""

POLICY_INFERENCE_USER_PROMPT = """\
Here is the board position and the next player is {next_player}:
{state}. The available move positions are {available_positions}.
"""
```

Policy Improvement Prompt

```
POLICY_IMPROVEMENT_SYSTEM_PROMPT = """\
You are playing the game tic-tac-toe on a 3*3 board. Tic Tac Toe is a
two-player game played on a grid.
Players take turns marking a space with their respective symbol.
The goal is to get multiple of one's own symbols in a row, either
horizontally, vertically, or diagonally, before the opponent does.
If all nine squares are filled and no player has three in a row, the
game is a draw.
The board consists of "O", "X" and grid number. The grid number
indicates empty grid.
Your task is to determine the best move for the next player based on
the given board position and the next player.
The evaluations of (board, action) pairs after possible moves are
given.
DO NOT judge the board based on your knowledge, only use the
evaluations to determine the best move.
The evaluation for the next board is in the format of a json format,
consisting of two elements:
- "thought": evaluation of the the board and action pair.
- "final_evaluation": Judge which side takes advantage. 1 means 'O'
takes advantage, -1 means 'X' takes advantage, and 0 means the game
will be a draw.
"""

POLICY_IMPROVEMENT_USER_PROMPT = """\
Here is the board position and the next player is {next_player}:
{state}. The possible moves are {available_positions}.
```

```
The following are the boards after each possible move:
{next_states}

Now, please give your evaluation and the best move for {next_player}
based on the given board position {state}.
You should output your answer in the json format. Your answer
consists of two elements:
- "thought": let’s think step by step. Generate your detailed
reflection by analyzing the next board positions and their
evaluations.
- "best_move": the best move for the next player. The move should
be in the format of a number from 1 to 9, indicating the position
on the board. And the move should be in the available positions
{available_positions}.
Don’t output extra information except for the json format.
"""
```

Policy Evaluation Prompt

```
POLICY_EVALUATION_SYSTEM_PROMPT = """You are a player of the game of
Tic Tac Toe. \nThe game goal is to get multiple of one’s own symbols
in a row, either horizontally, vertically, or diagonally, before the
opponent does. If all nine squares are filled and no player has three
in a row, the game is a draw. \nThe board consists of \"O\", \"X\"
and grid number. The grid number indicates empty grid. \nYou are
learning how to evaluate a (board, action) pair in the tic tac toe
by playing the game given the (board, action) pair and reflect the
playing history. \nThe playing history depicts a game in progress
with a final result. Your answer consists of two elements:
- "thought": let’s think step by step. Generate your detailed
evaluation over the (board, action) pair by merely reflecting the
playing history after this pair from different perspectives. You
should only rely on the playing history as context and don’t evaluate
game with your own judgement. Your evaluation should contain the
following elements: Win probability, Threat, Potential strategies.
- "final_evaluation": After all of your thoughts, judge which side
takes advantage. 1 means you think ’O’ takes advantage, -1 means you
think ’X’ takes advantage, and 0 means you think the game will be a
draw.
You should output your answer in the json format."""

POLICY_EVALUATION_EXAMPLE_USER_PROMPT = """The (board, action) to
evaluate is O’s turn:
Board:
O | O | X
---------
4 | X | 6
---------
7 | 8 | 9
Action:
The O’s move is 4.

Below is the rollout sequence after this (board, action):
After O’s move 4, the board position is:
O | O | X
---------
O | X | 6
---------
7 | 8 | 9
After X’s move 7, the board position is:
O | O | X
---------
O | X | 6
---------
```

```
X | 8 | 9
The game is over. X wins. X wins by occupying the positions 3, 5, 7.
"""

POLICY_EVALUATION_EXAMPLE_ASSISTANT_PROMPT = """
{"thought": {"Reflection": "It appears that the initial board
position
O | O | X
---------
4 | X | 6
---------
7 | 8 | 9
and O's move 4 were not favorable for O, as X was able to block on
O's move at 7 and ultimately win the game.", "Win probability": "The
win probability for X is large, while the win probability for O is
low.", "Threat": "X has played at 5 and 3. X can win by move 7. O
can occupy 1, 4, 7, and create a potential winning opportunity. X
occupies 5, which is a key position to win the game.", "Potential
strategies": "Potential strategies for O include playing at 7 to
block X and create a potential win by occupying 1, 4, 7. X could have
occupied 3, 5, 7 to win the game. X has already occupied 3, 5, and
needs only 1 move to complete the win."}
"final_evaluation": -0.8}
"""

POLICY_EVALUATION_USER_PROMPT = """The board to evaluate is
{player}'s turn:
Board:
{board}
Action: The {player}'s move is {action}.

Below is the rollout sequence 1 after this (board, action):
After {player} taking action {action}, the board position is:
{rollout_board_1}
The game is over. {winner_1} wins. {winner_1} wins by occupying the
positions {positions_1}.

...

Below is the rollout sequence N after this (board, action):
After {player} taking action {action}, the board position is:
{rollout_board_N}
The game is over. {winner_N} wins. {winner_N} wins by occupying the
positions {positions_N}.

Now generate your evaluation for the (board, action) pair."""
```

E ANALOGY

E.1 LANGUAGE MC

As mentioned in the main paper and figure, language MC is an analogy of traditional MC. Specifically, the mean and sum operation in traditional MC estimate can be replaced by the new lew language aggregator G_1. r can correspond to the intermediate transition for action, reward, and next state. The discount factor γ can also have an interesting analogy in G_1 if G_1 acts as a lossy information compressor. Specifically, just as the discount factor reduces the weight of future rewards, G_1 can perform a lossy compression of future information, selectively retaining only the recent and most relevant aspects.

E.2 Language TD

Most Analogies between language TD and traditional TD share similarities with that in the language MC setting. G_1 is a language aggregator over multiple lookahead variations, corresponding to the trajectory mean operation, while G_2 combines immediate change/intermediate transition and future value evaluation, akin to the sum operation.

F Limitation and future work

Limitations. Currently, NLRL still faces several limitations. First, our experiments demonstrate its effectiveness primarily in environments with discrete action spaces and low-dimensional (textualisable) states. Its performance in environments involving continuous action spaces and high-dimensional states (e.g., low-level robotic control tasks) remains largely unexplored. Second, our experiments are primarily conducted in small-scale settings due to the substantial computational cost associated with invoking Large Language Models. The time efficiency of the current NLRL approach is significantly lower than that of traditional RL methods using smaller networks.

Future work. Given NLRL's potential, there are several promising directions for future research. First, a theoretical foundation for the entire framework in the language space is needed to move beyond analogical reasoning towards a more robust, principled basis. This would help formalise the framework and enhance its generalisability. Another important direction is exploring how NLRL can be integrated more organically with existing research on self-evaluation, self-improvement, LLM agent framework, and LLM's planning. Such integration could unlock new capabilities and efficiency improvements. Additionally, in terms of application domains, there is significant potential to extend NLRL beyond the current setups–we are actively exploring extending NLRL to general domains beyond games, such as reasoning tasks (Cobbe et al., 2021; Xin et al., 2024) and code (Chen et al., 2021).