Learning to Play Like Humans: A Framework for LLM Adaptation in Interactive Fiction Text-Based GAMEs

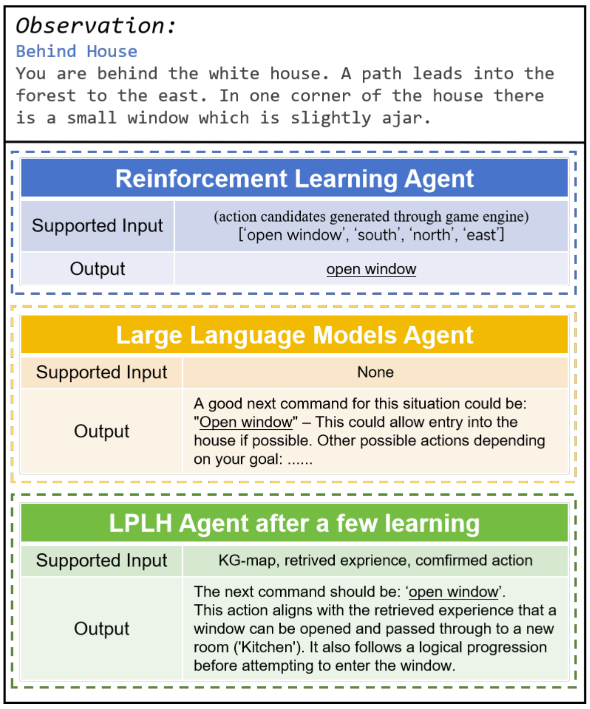
Anonymous ACL submission

Abstract

Interactive Fiction text-based adventure games (IF games) are where players interact through natural language commands. While recent advances in Artificial Intelligence agents have reignited interest in IF games as a domain for studying decision-making, existing approaches prioritize task-specific performance metrics over human-like comprehension of narrative context and gameplay logic. This work presents a cognitively inspired framework that guides Large Language Models (LLMs) to learn and play IF games systematically. Our proposed **L**earning to **P**lay **L**ike **H**umans (LPLH) framework integrates three key components: (1) structured map building to capture spatial and narrative relationships, (2) action learning to identify context-appropriate commands, and (3) feedback-driven experience analysis to refine decision-making over time. By aligning agent behavior with narrative intent and commonsense constraints, LPLH moves beyond purely exploratory strategies to deliver more interpretable, human-like performance. Crucially, this approach draws on cognitive science principles to more closely simulate how human players read, interpret, and respond within narrative worlds. As a result, LPLH reframes the IF games challenge as a learning problem for LLMs-based agents, offering a new path toward robust, context-aware gameplay in complex text-based environments.

1 Introduction

Interactive Fiction games (IF games), originating in the 1970s (Spring, 2015; Aarseth, 1995), demand abstract reasoning, implicit world inference, and narrative reconstruction from textual cues alone. Unlike visual or auditory games, IF games rely solely on language and imagination. Successful play involves iterative exploration, learning, and adaptation, guided by intuition, pattern recognition, and experience-driven generalization (Zander et al., 2016). Consequently, IF games offer a rich testbed for agent problem-solving, shedding light on core mechanisms of exploration and learning.



```
Observation:
Behind House
You are behind the white house. A path leads into the
forest to the east. In one corner of the house there
is a small window which is slightly ajar.
```

Reinforcement Learning Agent

Supported Input	(action candidates generated through game engine) ['open window', 'south', 'north', 'east']
Output	open window

Large Language Models Agent

Supported Input	None
Output	A good next command for this situation could be: "Open window" – This could allow entry into the house if possible. Other possible actions depending on your goal: ......

LPLH Agent after a few learning

Supported Input	KG-map, retrived exprience, comfirmed action
Output	The next command should be: 'open window'. This action aligns with the retrieved experience that a window can be opened and passed through to a new room ('Kitchen'). It also follows a logical progression before attempting to enter the window.

Figure 1: Example of RL approach, Basic LLM approach, our learning to Play Like Humans (LPLH) approach

DRRN (He et al., 2016) sparked a growing interest in RL settings where states and actions are expressed in natural language. Consequently, IF games have become a core testbed for integrating RL and natural language understanding (NLU) (Guo et al., 2020; Hausknecht et al., 2020). Early RL agents rely heavily on action filters and simplistic policies (Yao et al., 2020; Guo et al., 2020; Ammanabrolu et al., 2020). Although subsequent RL approaches include more sophisticated techniques (Ammanabrolu et al., 2020; Yao et al., 2021; Peng et al., 2022), their score-centric objectives still constrain nuanced narrative reasoning. Recent works have begun leveraging large language models (LLMs) (Tsai et al., 2023; Ma et al., 2024), but no system LLMs work on playing IF games yet.

Human intuition underpins how players navigate IF games, shaping engagement with complex systems. Bartle (1996) taxonomy highlights distinct player motivations that drive varied strategic and imaginative play, while Koster (2013) positions "fun" as the joy of pattern recognition—an intrinsically intuitive process of mapping challenges to learned solutions. Tekinbas and Zimmerman (2003) further show that well-crafted rules and exploratory freedom foster creative problem-solving, underscoring the pivotal role of intuitive, context-aware reasoning in game design and analysis.

In contrast to prior work using IF games primarily for RL-based NLU assessment, we propose leveraging LLMs' context-aware reasoning and decision-making to play them systematically. LLMs have demonstrated remarkable progress in multi-step reasoning and context management in other domains, showcasing advanced narrative understanding capabilities (Huang et al., 2024; Zhang and Long, 2025). And, IF games' narrative-rich, text-based design resonates with human intuition and demands more robust reasoning than score-centric RL paradigms, exposing gaps in current RL strategies. This language-driven setting thus provides a fertile ground for developing more context-aware and adaptable agents (Tsai et al., 2023).

Motivated by these insights, we introduce a novel LLMs-driven external-knowledge-training-free framework, the **L**earning to **P**lay **L**ike **H**umans (LPLH) framework. The LPLH framework simulates human playing trends to play IF game dynamically. It combines three key modules: 1) The **dynamic map** building provides a high-accuracy game map to guild agents to follow the correct location, just like most human players do when drawing a map. 2) **Action space learning** indicates learning all verbs and intractable objects once they are verified to be valid as long-term memory of a human. 3) Every time the agents solve the puzzle or fail, the LPLH will automatically **summarize the helpful experience** and fallback saving for the future, toward human players' reflection. So, for decision-making, the LPLH uses the current game's information and combines the previous related experiences to predict the next step, aligning with human players' behaviors.

By integrating these key modules, the LPLH framework entirely forgoes reliance on external knowledge to pre-train agents. Instead, it fosters a self-innovative learning process that closely mirrors how human players approach new games: constructing detailed maps, incrementally exploring valid actions, and reflecting on experience to inform future decisions. We apply this approach to various IF games from Jercho dataset (Guo et al., 2020). The main contributions are as follows: 1) We propose the LPLH framework, designed to simulate human players' behaviors, which, to our knowledge, is the first system to work to leverage LLMs for playing IF games. 2) We demonstrate the robustness and efficiency of LPLH framework by playing games on different baselines. 3) Empirical results confirm that modeling human-like decision-making processes enhances LLM performance on IF game tasks, indicating the effectiveness of our human-player simulation strategy.

2 Related Work

IF games provide a structured environment that drives research on language-based agents. These text-based simulations (Hausknecht et al., 2020) integrate challenges. Many studies adapt RL methods to handle their vast, partially observable state-action spaces. KG-A2C (Ammanabrolu and Hausknecht, 2020) builds a knowledge graph (KG) to represent game states and constrain action spaces, addressing the complexity of natural language actions. Guo et al., 2020 reformulates gameplay as a multi-passage reading comprehension task, using context-query attention and structured prediction to enhance action generation and mitigate partial observability.

Another strand of research targets the exploration challenges unique to IF games. Q*BERT and MC!Q*BERT (Ammanabrolu et al., 2020) employ knowledge-graph-based intrinsic motivation and strategic exploration to overcome bottlenecks in sparse-reward environments. Complementing these efforts, Tuyls et al., 2022 dissects the explore-vs.-exploit dilemma by decomposing each episode into distinct exploitation and exploration phases, achieving notable improvements in normalized game scores across multiple environments.

In parallel to the RL-centric approaches, recent works have explored integrating pre-trained language models to enhance agents' semantic understanding and action generation capabilities. The CALM (Yao et al., 2020) is trained on human gameplay data to produce a compact set of action candidates. CALM significantly improves in-game

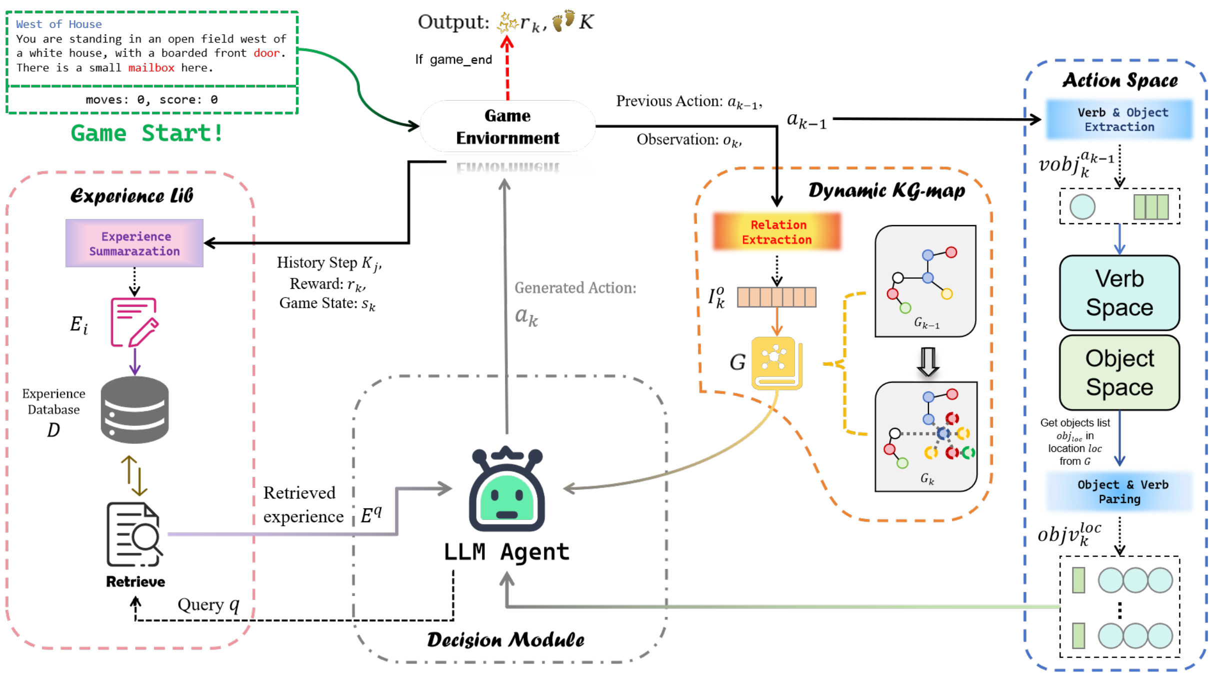


Figure 2: LPLH framework. The **Dynamic KG-map** builds a knowledge-graph map dynamically by extracting the key items in observation. The **Action Space** split valid action into verb and object phases, then pares the possible verbs with objects to the agent. The **Experience Lib** automatically grape worth steps and summarizes them as 'experience' for future guides.

scores, even on unseen games, when combined with an RL agent for action re-ranking. Similarly, Singh et al., 2022 leverages transformer-based models to inject rich semantic priors.

In contrast to approaches that integrate RL with NLU, a notable study (Tsai et al., 2023) investigates the performance of LLMs on text games without additional RL components. It evaluated ChatGPT (OpenAI, November 30, 2022) and GPT-4 (OpenAI et al., 2024) on Zork1—the only work to focus on leveraging LLMs in this context. Their findings reveal that, although ChatGPT performs competitively with existing systems, it exhibits significant limitations: it fails to construct a coherent model and struggles to incorporate pre-existing world knowledge. These shortcomings highlight critical open questions at the intersection of IF game agents and LLMs, suggesting that further research is needed to fully realize the potential of LLM-only approaches in interactive fiction environments. Thus, our LPLH framework attempts to fill this gap by simulating human playing behaviors.

3 LPLH Framwork

This section will go through the overall LPLH framework and each key module in LPLH. Figure 2 shows the architecture of LPLH framework.

3.1 Problem Define

The interaction between an autonomous agent and a text-based game environment can be formulated as a Partially Observable Markov Decision Process (POMDP) (Spaan, 2012), represented as a tuple (S, T, A, O, R) as follows: The agent issues text commands $a \in A$, selecting from a space of natural language actions; It receives text-based observations $o \in O$ describing the environment state in a limited scope; The environment provides scalar rewards $r = R(s, a)$, often sparse, to guide learning; The underlying game state $s \in S$ encodes KG-map G but is partially observable through textual feedback; The transition function $s' = T(s, a)$ updates the game state based on the agent's action, following the game's internal logic.

Unlike the existing RL approaches (He et al., 2016), which learns a value function by selecting from a predefined set of actions to maximize game rewards, our LPLH framework more closely mimics human decision-making by integrating multiple sources of information when generating the next command. Specifically, LPLH introduces a structured method for semantic understanding and decision-making in text-based games. At each time step k, LPLH receives the current game observation o_k and the previous action a_{k-1} and updates the knowledge graph G through the dynamic knowledge-graphs map module. Suppose the last action is valid (i.e., the game state changes from s to s'); the action space module stores that action by splitting it into its constituent verb and objects. Meanwhile, by evaluating the reward r_k at each step, LPLH summarizes the current game state s_k in conjunction with historical information K_j, thus producing a helpful experience E_i, where j denotes the history length and i indicates experience index.

For the generation part, our LLM-based agent LLM will put the map G, confirmed action list for each object in the current state $objv$, and retrieved experience E to predict the best suitable action.

Following sections are details of LPLH framework integrating **Dynamic knowledge-graphs (KG) map**, **Action Space**, and **Experience Lib** enables adaptive and robust learning in interactive fiction environments. This can potentially enhance agent in complex, language-driven tasks.

3.2 Dynamic knowledge-graphs map

Creating a KG to store information is widely used for long-term memory solutions in LLM research (Tsai et al., 2023; Zhu et al., 2024). In the IF game task, the KG-map serves as a continually updated map of in-game entities and their relations, thereby guiding the RL agent's action selection (Ammanabrolu and Hausknecht, 2020). However, while previous approaches often treat the KG-map as a static structure or update it only when new object relations are discovered, our dynamic KG-map module continuously modifies the graph after each change.

Concretely, as the agent interacts with the environment, the textual observations are parsed to capture newly discovered objects, places, or entities and any relationship changes (e.g., "the key is now inside the box"). These updates ensure that the KG-map aligns with the evolving state of the game world. The LLMs-based agent can more accurately retrieve relevant information when constructing its following action by maintaining a synchronized, real-time representation of the current environment. This dynamic process resolves inconsistencies (such as outdated item locations) and enriches the agent's contextual awareness, allowing for more robust decision-making in text-based games.

We employ a *verb & object extraction*, a fine-tuned model fm_{re}, to identify relational triples (location and objects) from the observation o_k as relation extraction. Formally, we define:

$$I_k^o = fm_{re}(a_{k-1}, o_k) \tag{1}$$

where a_{k-1} is the action taken at the previous step, and I_k^o denotes the set of extracted relations.

Subsequently, the module integrates these newly extracted relations I_k^o along with the preceding action a_{k-1} to update the knowledge graph G:

$$G_k = kg(G_{k-1}, a_{k-1}) \oplus kg(G_{k-1}, I_k^o) \tag{2}$$

where $kg(\cdot)$ is a dynamic function that incrementally updates the knowledge graph based on the provided information, and the operator $\oplus$ combines the updated states from both the action and the newly extracted relations.

3.3 Action Space Learning

LPHP framework learns all valid actions within a dedicated action space to emulate human player behavior. This space is decomposed into two phases: verb and object. The valid actions learned in this manner are retained as executable commands. Specifically, after observing the last action a_{k-1}, a *verb & object extraction* model fm_{vo} determines whether it remains valid based on the updated observation o_k. If a_{k-1} is valid, the model decomposes it into a set of verbs and objects:

$$\begin{aligned} vobj_k^{a_{k-1}} &= fm_{vo}(a_{k-1}) \quad \text{iff } a_{k-1} \text{ is valid}, \\ AS &= AS \cup vobj_k^{a_{k-1}} \end{aligned} \tag{3}$$

where $AS \in \mathbb{R}^{\{n,m\}}$ denotes the recognized action space, and n and m represent the maximum numbers of verbs and objects, respectively, in the game environment. The term $vobj_k^{a_{k-1}}$ contains exactly one verb (e.g., *"put * in *"*) followed by a list of corresponding objects.

In the subsequent reasoning phase, the framework then employs an *object & verb pairing* procedure to integrate objects in the current location (obj^{loc}) with candidate actions:

$$objv_k^{loc} = \text{pairing}\big(obj^{loc}, AS\big) \tag{4}$$

where the function $\text{pairing}(\cdot)$ searches the recognized action space to find all actions compatible with current location's objects. The $objv_k^{loc}$ is a list of viable action–object pairs for decision-making.

3.4 Experience Lib

The experience library serves as a crucial component for simulating human player behavior. Prior research demonstrates that humans can draw insights from multiple past trajectories to improve performance in subsequent attempts (Fazey et al., 2005). To model this process, we introduce an experience summarization module, denoted by LLM_{es}, which condenses critical information from a fixed number of historical steps K_j and the corresponding reward changes r_k, given the current game state s_k. Formally, the summarized experience E_i is generated as follows:

$$E_i = LLM_{es}(K_j, r_k, s_k),$$

where j indicates the fixed length of the history, i indexes the summarized experience, and LLM_{es} is a LLM prompted in a one-shot template.

Then, each summarized experience is stored in a vector database D to guide future decision-making and facilitate long-term policy optimization. Subsequently, we employ a simple retrieval-augmented generation (RAG) strategy (Lewis et al., 2020), which takes a query q as input and retrieves relevant information E^q from the stored experiences. We employ RAG to efficiently leverage relevant contextual knowledge from the vector database, thereby improving both factual accuracy and overall robustness in the model's decision-making process. This integration of historical insights allows the model to refine its actions and adapt its policy updates based on accumulated knowledge.

3.5 Zore-shot Decision-making

To generate the next command a_k at step k, LPLH use an LLMs-based agent, denoted by LLM_a, in a zero-shot prompting setup. Specifically, the agent first constructs a query q based on the current game context and uses it to retrieve a relevant experience set E^q. It then aggregates the current observation o_k, the retrieved experiences E^q, the structured KG-map representation G, and the confirmed action set $objv_k^{loc}$. This combined context is provided as input to the LLMs-based agent, which produces the next command a_k:

$$a_k = LLM_a\big([G,\, objv_k^{loc},\, E^q],\, o_k\big) \qquad (5)$$

By integrating observations and relevant knowledge, LPLH framework enables the agent to issue commands in a flexible, zero-shot (Kojima et al., 2022) way without task-specific fine-tuning.

The proposed LPLH framework integrates zero-shot prompting, retrieval of relevant experiences, action parings, and KG-map structures to guide an LLMs-based agent in command generation. This approach offers a robust and adaptable solution that can seamlessly accommodate diverse contexts by eliminating the need for fine-tuning. In doing so, it is the first system methodologies attempt for IF game environments through LLMs-driven techniques, starting the way for more human-behavior-aligned agent interactions.

4 Experiment

This section will briefly introduce our chosen **Dataset**, **Baselines**, and **Experiment setup**.

4.1 Dataset

We evaluate our method on a collection of IF games made available through Jericho, an open-source Python-based environment (Guo et al., 2020). The games in Jericho cover diverse genres (e.g., dungeon crawl, mystery, horror) and include both classic Infocom titles (like *Zork1*) and community-developed works (such as *Afflicted*). Most IF games employ a point-based scoring system, which serves as the primary reward signal for learning agents. While Jericho natively supports scoring detection for a curated set of games, it also offers the flexibility to run unsupported games without these features. While all games run under *'verbose'* model, which always gives the maximum observation of room.

4.2 Baselines

We choose some previous RL models to compare with LLMs approaches. These models represent advancements in integrating structured knowledge representations and natural language processing techniques to improve agent performance and interpretability in complex environments. The chosen RL models are **DRRN** (He et al., 2016), **KG-A2C** (Ammanabrolu and Hausknecht, 2020) and **DBERT-DRRNL** (Singh et al., 2022).

Also, several LLM models will be the baseline and foundation for the LPLH framework: **Qwen2.5-7B-Instruct** (Team, 2024; Yang et al., 2024), **Qwen2.5-14B-Instruct** (Team, 2024; Yang et al., 2024), **GPT-4o-mini** (OpenAI, 2025a) and **GPT-o3-mini** (OpenAI, 2025b).

Noticeably, RL approaches select the possible action candidates supported by the game engine. Meanwhile, all LLM approaches will generate action, the processes of which are more complicated but closer to the human player's ways.

4.3 Experiment setup

We assess the proposed LPLH framework on 10 games of varying difficulty levels (Guo et al., 2020). Each LLM-based agent runs for 10 epochs (250 steps per epoch). At every step, only the observation from the Jericho game engine is provided to the agent[1]. We record both the average and maximum score after all epochs for each game. As a comparison, we also implement an LHLP framework under the same experimental conditions but designate only the final three epochs as "learning outcomes," with all preceding epochs serving as intermediate training phases. Please see Appendix C for hyper-parameters and prompt details.

For the fine-tuned model fm used in our LHLP framework (Section 3), we adopt a smaller model, **Qwen2.5-1.5B-Instruction** (Team, 2024), to address three specific tasks: (1) validating actions, (2) extracting relations from observations, and (3) decomposing actions into verbs and objects. We begin by collecting game actions and observations from the LLM baseline. Next, we employ carefully crafted prompts for GPT-4, obtaining hundreds of annotated samples, which we subsequently refine manually to create the training datasets for each task. Details of the fine-tuned model fm are provided in Appendix A. While the experience summarizing model LLM_{es} is using **GPT-o3-mini**.

5 Result and Analyzes

This section assesses how the LPLH framework advances performance in IF games. First, we illustrate learning behaviors through the learning curves (Section 5.1), compare game scores across various baselines (Section 5.2), and conduct an ablation study (Section 5.3). These analyses highlight LPLH's adaptive, human-like language acquisition capacity and robust performance.

5.1 Learning curves

The learning curve in Figure 3 illustrates how a good LPLH framework progresses in *Zork1*, mimicking human learning behaviors. The max score (blue) follows a stepwise trajectory, reflecting human players' learning patterns, where breakthroughs occur after discovering key actions. Periods of stagnation suggest moments of trial-and-error exploration before achieving the next milestone. The learned actions (green) curve steadily increases, indicating continuous vocabulary acquisition. Initially, actions are learned rapidly, but the pace slows over time, mirroring human language acquisition, where easy-to-learn words are picked up early, and complex concepts require more experience. The visited rooms (orange) curve represents the agent's exploration behavior. A sharp rise at the beginning suggests an initial discovery phase, similar to human players actively navigating a new environment. As the curve flattens, it indicates fewer new rooms are available, paralleling how human players shift from exploration to problem-solving.

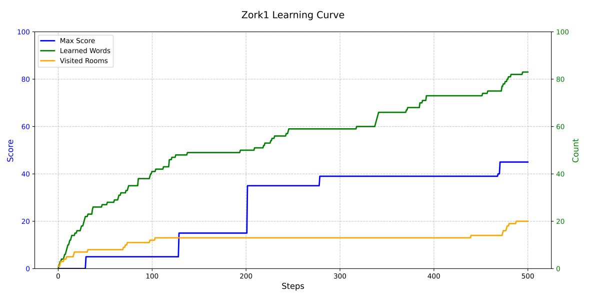


Figure 3: *Zork1* learning curve in scaled steps. For reference, human player's best trajectory gets 350 scores in 412 steps with 48 verbs, 57 objects (total 105 unique words), and 63 rooms.

5.2 Game Scores

Table 1 summarizes score performance of various agents on multiple IF games, comparing previous RL approaches with LLMs-based methods. LPLH framework markedly improves the performance of LLM-based agents by enabling dynamic adaptation and iterative learning during playing.

Across the board, we observe consistent and substantial gains when integrating LPLH into LLMs-based agents. For example, in *Detective*, Qwen-7B (LPLH) achieves a surprising 68/100 compared to the base model's 10/10, representing a 6.8× improvement on raw. Similarly, in *Spellbrkr*, Qwen-14B (LPLH) attains 41.7/60, outperforming both its base version and the RL agent DBERT-DRRN (D-D). These results highlight LPLH's capacity to leverage learned experiences akin to human players refining their strategies through trial and error.

Notably, LPLH agents match or exceed RL baselines in maximum achievable scores in some games. For instance, in *Omniquest* and *Balances*, GPT-o3-

[1]For all RL agents, they receive completed observation and inventory at each step. However, LLMs-based agent needs to decide when to call the command 'look' or 'i' to get such information by themselves.

	DRRN*	**KG-A2G***	**D-D***	**Qwen-7B**		**Qwen-14B**		**GPT-4o-mini**		**GPT-o3-mini**		**Max**
				base	*LPLH*	*base*	*LPLH*	*base*	*LPLH*	*base*	*LPLH*	
omniquest	5 / -	3 / -	4.9 / 5	1 / 5	5 / 5	1.5 / 5	5 / 5	2 / 5	5 / 5	4 / 5	5 / 5	**50**
detective	197.8 / -	207.9 / -	- / -	10 / 10	68 / 100	36 / 70	72 / 90	22 / 30	30 / 60	20 / 20	50 / 60	**360**
zork1	32.6 / -	34 / -	44.7 / 55	0 /0	9 / 15	9 / 35	39.7 / 45	6 / 10	10 / 15	30 / 35	33.8 / 45	**350**
zork3	0.7 / -	0.1 / -	0.2 / 4	0 / 0	0.6 / 1	2.0 / 3	2.6 / 3	1.8 / 3	2.8 / 3	3 / 3	3 / 3	**7**
ludicorp	13.8 / -	17.8 / -	12.5 / 18	1 / 1	1/ 1	10.5 / 12	11.7 / 13	1 / 1	2.6 / 3	4.4 / 7	8 / 11	**150**
balances	10 / -	10 / -	- / -	0 / 0	5 / 5	8.75 / 10	10 / 10	5 / 5	5 / 5	8.3 / 10	10 / 10	**51**
spellbrkr	37.8 / -	21.3 / -	38.2 / 40	0 / 0	25 / 25	25 / 40	41.7 / 60	18 / 40	38.3 / 50	31.3/ 50	47.5 / 60	**600**

Table 1: Game score results running on IF games. The DRRN*, KG-A2G*, and D-D* (DBERT-DRRNL) are RL agent; results are from their papers. *base* in LLMs-based agent generates action directly with some previous history, and *LPLH* is our approach. For the scores '-/-,' the first represents a **raw** score of the end, and the second represents the **max** score. In LLMs-based agents, the raw on *base* computes the average score in all runs, while the raw on *LPLH* computes the last three runs as "learning outcomes." Scores with blue mean the highest score in raw, and scores with underline are the highest score in max. The **Max** is the game's maximum score.

mini (LPLH) reaches the game's maximum score as same as RL agent, which both enter bottleneck stage (Ammanabrolu et al., 2020; Tuyls et al., 2022). For those LPLH agents beyond RL agents, it underscores LPLH's potential to discover optimal strategies without explicit reward engineering. Nonetheless, RL agents can achieve better scores when game engine's provided action candidates are tightly integrated with the game, enabling more precise decision-making in discrete action spaces—a stark contrast to the generative approach used by LLMs.

Our results suggest that LPLH framework enhances agents by fostering deeper contextual understanding and dynamic strategy adjustments beyond static priors. Moreover, LLMs-based agents with LPLH typically require fewer steps than RL methods to achieve comparable or higher scores (Guo et al., 2020), all without relying on additional external knowledge. As shown in Figure 1, LPLH agents exhibit human-like reasoning steps, providing a clear, self-explanatory rationale for their actions. Future work may focus on further optimizing LPLH to reinforce adaptive behavior in more complex IF settings, potentially bridging the gap between fine-grained RL solutions and the flexible, learned knowledge of LLMs-based approaches.

5.3 Ablation Study

In this section, we provide a more in-depth analysis of LPLH framework and evaluate contribution of each component. As shown in Table 2, we report both the raw and max scores, as well as their standard deviations (σ) to assess performance stability across different model variants. We take LPLH$_{14B}$[2] as our backbone model and observe that fine-tuning it with chain-of-thought reasoning (LPLH$_{14B}$-CoT[3]) boosts the raw score from 39.7 to 41.6, while maintaining a relatively small standard deviation, 2.4. However, this improvement comes at a higher computational cost.

Model	**raw**	**max**	σ
LPLH$_{14B}$	39.7	45.0	4.2
LPLH$_{14B}$ $-$ *CoT*	41.6	45.0	2.4
KG-map only	11.0	15.0	2.0
KG-map + exp	11.0	35.0	13.1
KG-map + as	27.8	35.0	6.8
exp only	25.6	34.0	9.0
exp + as	32.0	40.0	4.0
as-only	26.6	35.0	6.6
Qwen2.5-14B-Instruct	9.0	25.0	9.2
14B-*select one*	14.5	30.0	11.6

Table 2: Ablation results on '*Zork1*.' Where σ is the standard deviation; 'exp' represents the experience summarization; and 'as' represents the action space.

Next, we examine the effects of adding a knowledge graph mapping component (KG-map). Although this variant exhibits a slightly lower maximum score, its standard deviation is reduced, suggesting improved stability. The model achieves a higher maximum score when additional experiential data are introduced and exhibits a larger σ. Finally, incorporating an action-space mechanism provides further performance gain by reducing wasted actions. Combining this mechanism

[2]The backbone model is Qwen2.5-14B-Instrction.

[3]We use DeepSeek-R1-Distill-Qwen-14B (DeepSeek-AI et al., 2025) as the CoT distillation model for this task.

with experiential data leads to the most substantial overall results, demonstrating the effectiveness of a multi-faceted approach to enhancing LPLH.

We also compare against a Qwen2.5-14B-Instruct baseline (9 raw, 25 max, 9.17 σ) and observe further gains by incorporating a selective mechanism (14B-*select one*[4]). Each module independently boosts performance, and their synergy yields even stronger results. Future work includes exploring how transitioning from generating to selecting actions may further enhance reasoning.

6 Discussion on the significance of experiences in LPLH framework

This section discusses two examples in *Zork1* to show how experiences are essential for the LPLH framework to simulate humans. Both are how LPLH framework plays like humans with experience reflection to achieve higher scores and avoid failure, where we will use 'player' to represent the game character and 'agent' for LPLH framework.

6.1 Learning from Failure

In the case of how the LPLH framework learns from failure, the player meets the dark environment for the first time. The agent would not figure out what was going on and then try to take any new action, but a death followed anyway. After the player's death, the agent will automatically start summarizing this experience. During the summarization, agent will also focus on any partially missing events and suggestions for future reference. After the first attempt, the agent calms that *'You may need lights to avoid death in the dark...'*. However, the agent suggests finding lights since the player never finds light resources. After several attempts, the player finally finds a 'lantern' in the 'living room' and takes it. When a player goes to dark again, the experience will let the agent know that the light needs to be turned on first.

6.2 Learning from Success

Learning from success is more straightforward to analyze. Once the player solves a puzzle, the pertinent steps are captured and organized to guide future decision-making. Consequently, when the player encounters a similar situation, the agent retrieves these validated experiences and follows the proven trajectory of actions. Once the player obtains an "egg" at "Up a Tree," they earn five points during the initial exploration. In subsequent rounds, the agent consistently instructs the player to collect this "egg" at the start of the game. "egg" and "Up a Tree" are automatically stored in the kg-map, facilitating quick retrieval for future scenarios.

In LPLH framework, each interaction, success or failure, provides essential feedback that informs subsequent decisions (Schaul, 2015; Browne et al., 2012). By systematically archiving and reflecting on these experiences, the agent refines its understanding of the environment, thereby improving strategic behavior and adaptability over time (Bion and Hinshelwood, 2023; Boyd and Fales, 1983).

Additionally, as observed in our example (Section 5.2) and further confirmed through our manual playing, many IF games feature "special" commands or specific actions that are crucial for progressing or scoring points, which also has pointed in (Tuyls et al., 2022; Ammanabrolu et al., 2020). Such demands challenge RL and LLMs-based agents—even with long-term memory—since they require expert knowledge and creativity that come more naturally to humans. Addressing this gap remains a crucial direction for future work, underscoring the need to align agents with the intuition and imagination that guide human players.

7 Conclusion

In this work, we introduced **L**earning to **P**lay **L**ike **H**umans framework **LPLH**, a novel framework designed to guide LLMs to play IF games by simulating human player behaviors. To our knowledge, this is the first systemized approach leveraging LLMs to tackle well-known text-based IF games. This cognitively inspired approach for LLMs-based IF game agents integrates dynamic map building, action space learning, and experience-driven memory. LPLH framework balances narrative comprehension, exploration, and puzzle-solving by simulating human play processes without relying on external pre-training. Although our approach still falls short of specialized RL agents in certain games and cannot match human-level scores, it yields more interpretable, human-like behaviors and enables more context-aware decision-making in interactive fiction game domains. Furthermore, according to the performance of LLMs in IF games, we believe that IF games are a considerable challenge for LLMs in many aspects.

[4]The selective mechanism is choosing one action from game engines' action candidates same as RL approaches.

Limitation

To our knowledge, our proposed LPLH framework is the first attempt to enable LLMs to play IF games by simulating human players as a system works. Although we incorporate multiple modules for long-term memory, effectively managing and navigating that memory remains challenging. Currently, our approach uses a simple experience summarization that is only triggered when the agent loses or gains points. In contrast, human players naturally integrate relevant information into their memory at any point during gameplay, suggesting that a more dynamic summarization strategy could yield better results.

Furthermore, the framework relies on a JSON-structured kg-map as input. The consistency and clarity of this representation can influence the model's reasoning, indicating that further investigation is needed to determine the optimal representation method for LLM-based agents in IF tasks. We also attempted to evaluate the LPLH framework across different models; however, we could not perform extensive tests on a wide range of LLMs due to resource constraints. Future work should include more comprehensive experimentation and exploring adaptive memory-management techniques to address these limitations. Also, we only test a few IF games to show our framework performance, which may not be fully adopted for all IF game types. During the game, the learning process is still affected by many factors that could dramatically lead to score increases or decreases, which we have not found.

References

Espen Aarseth. 1995. *Cybertext: perspectives on ergodic literature*. University of Bergen.

Prithviraj Ammanabrolu and Matthew Hausknecht. 2020. Graph constrained reinforcement learning for natural language action spaces. *arXiv preprint arXiv:2001.08837*.

Prithviraj Ammanabrolu, Ethan Tien, Matthew Hausknecht, and Mark O Riedl. 2020. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. *arXiv preprint arXiv:2006.07409*.

Richard Bartle. 1996. Hearts, clubs, diamonds, spades: Players who suit muds. *Journal of MUD research*, 1(1):19.

Wilfred Bion and Robert Hinshelwood. 2023. *Learning from experience*. Routledge.

Evelyn M Boyd and Ann W Fales. 1983. Reflective learning: Key to learning from experience. *Journal of humanistic psychology*, 23(2):99–117.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Ioan Fazey, John A. Fazey, and Della M. A. Fazey. 2005. Learning more effectively from experience. *Ecology and Society*, 10(2).

Xiaoxiao Guo, Mo Yu, Yupeng Gao, Chuang Gan, Murray Campbell, and Shiyu Chang. 2020. Interactive fiction game playing as multi-paragraph reading comprehension with reinforcement learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7755–7765.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2020. Interactive fiction games: A colossal adventure. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7903–7910.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Guangming Huang, Yunfei Long, Cunjin Luo, Jiaxing Shen, and Xia Sun. 2024. Prompting explicit and implicit knowledge for multi-hop question answering based on human reading process. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13179–13189, Torino, Italia. ELRA and ICCL.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Raph Koster. 2013. *Theory of fun for game design*. " O'Reilly Media, Inc.".

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn LLM agents. In *ICLR 2024 Workshop on Large Language Model (LLM) Agents*.

OpenAI. 2025a. Gpt-4o mini: Advancing cost-efficient intelligence. Accessed: 2025-02-08.

OpenAI. 2025b. Openai o3 mini. Accessed: 2025-02-08.

OpenAI. November 30, 2022. Introducing ChatGPT. `https://openai.com/index/chatgpt/`. [Accessed 06-02-2025].

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex

Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Xiangyu Peng, Mark Riedl, and Prithviraj Ammanabrolu. 2022. Inherently explainable reinforcement learning in natural language. *Advances in Neural Information Processing Systems*, 35:16178–16190.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Tom Schaul. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Ishika Singh, Gargi Singh, and Ashutosh Modi. 2022. Pre-trained language models as prior knowledge for playing text-based games. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1729–1731.

Matthijs TJ Spaan. 2012. Partially observable markov decision processes. In *Reinforcement learning: State-of-the-art*, pages 387–414. Springer.

Dawn Spring. 2015. Gaming history: computer and video games as historical scholarship. *Rethinking History*, 19(2):207–221.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Katie Salen Tekinbas and Eric Zimmerman. 2003. *Rules of play: Game design fundamentals*. MIT press.

Chen Feng Tsai, Xiaochen Zhou, Sierra S Liu, Jing Li, Mo Yu, and Hongyuan Mei. 2023. Can large language models play text games well? current state-of-the-art and open questions. *arXiv preprint arXiv:2304.02868*.

Jens Tuyls, Shunyu Yao, Sham M. Kakade, and Karthik R Narasimhan. 2022. Multi-stage episodic control for strategic exploration in text games. In *International Conference on Learning Representations*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Shunyu Yao, Karthik Narasimhan, and Matthew Hausknecht. 2021. Reading and acting while blindfolded: The need for semantics in text game agents. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3097–3102, Online. Association for Computational Linguistics.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep calm and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754.

Thea Zander, Michael Öllinger, and Kirsten G Volz. 2016. Intuition and insight: Two processes that build on each other or fundamentally differ? *Frontiers in psychology*, 7:1395.

Jinming Zhang and Yunfei Long. 2025. MLD-EA: Check and complete narrative coherence by introducing emotions and actions. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 1892–1907, Abu Dhabi, UAE. Association for Computational Linguistics.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the*

62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), Bangkok, Thailand. Association for Computational Linguistics.

Yuqi Zhu, Xiaohan Wang, Jing Chen, Shuofei Qiao, Yixin Ou, Yunzhi Yao, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2024. Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities. *World Wide Web*, 27(5):58.

A Fine-tuned model fm

Before we use the fine-tuned model fm in LPLH framework. We collect the data through three different games (not in our test game): '*Dragon*,' '*Karn*', and '*Night*.'

In both games, we run with random pick action provided by the game engine and generate the action through LLMs. Then, we pair the action and sequenced observation as the basic training dataset. For this basic training dataset, we manually delete those repeat parts. Following that, we create different prompt templates for three task: (1) validating actions (Prompt in Table 4), (2) extracting relations from observations(Prompt in Table 5), and (3) decomposing actions into verbs and objects (Prompt in Table 6). After getting the results generated from GPT-4o, we manually selected the correct parts and then passed them to the train. We use LoRA (Hu et al., 2021) to train the model on LLaMA-Factory (Zheng et al., 2024). The training details are shown in Table 3.

We evaluate the model performance by running '*Zork1*' with 'walk-thought'[5]. For the task of validating actions, the accuracy is 90%. For relations extraction, the error rate is like 15%. And for splitting the actions, the accuracy is 98%.

B Baseline Details

DRRN (He et al., 2016) models the relevance between the state and possible actions to navigate large action spaces effectively.

KG-A2C (Ammanabrolu and Hausknecht, 2020) integrates dynamically constructed KG into the Advantage Actor-Critic framework to constrain the action space and improve decision-making.

DBERT-DRRNL (Singh et al., 2022) enhances the traditional DRRN architecture by incorporating DistilBERT (Sanh et al., 2020), a pre-trained language model, to provide richer text representations, thereby improving the agent's performance.

[5]the human player's best trajectory

Qwen2.5-7B-Instruct (Team, 2024; Yang et al., 2024): A 7B open-source model from Alibaba, designed for general-purpose natural language understanding and generation, optimized for efficiency and broad-domain applicability.

Qwen2.5-14B-Instruct (Team, 2024; Yang et al., 2024): A larger 14B version of Qwen, offering improved reasoning, generation quality, and contextual understanding.

GPT-4o-mini (OpenAI, 2025a): A lightweight version of GPT-4o, optimized for efficiency while maintaining strong reasoning capabilities, making it suitable for scalable applications.

GPT-o3-mini (OpenAI, 2025b): A compact version of OpenAI's third-generation model, designed for high-speed inference with reasonable performance in various NLP tasks, especially in constrained computational environments.

Parameter name	Value
`lora_rank`	16
`lora_alpha`	32
`lora_dropout`	0.1
`lora_target`	`all`
`learing rate`	$2e-5$
`epoches`	3

Table 3: hyper-parameters of fine-tuning

C Hyper-parameters and Prompts

C.1 Hyper-parameters

All experiences were running at 2 RTX4090 GPUs with torch type of *bf16*. ALL none-trained LLMs (LLM_{es} and LLM_a) run with the same temperature of 0.6. The fine-tuned model (fm uses a temperature of 0.1.

C.2 Prompt templates

Here, we show some essential prompt templates, which all models followed CoT reasoning. Table 7 shows how the baseline model generates the next command. Table 8 shows a prompt template of how to summarize the experience. The prompt template of action generation of the LPLH framework is showing Table 9. In our study, no any game name or specific game commands appear in all prompts. According to research done by (Tsai et al., 2023), the Chat-GPT knows IF game *Zork1*. When Chat-GPT knows the specific games, it will do well.

Instruction:
You are evaluating the outcome of a text-based game action based on the game's observation (feedback message) after the player's previous action. Your task is to determine if the action was successful or not.
<START OF INSTRUCTIONS>
- You will be given an observation text that follows the player's attempted action.
- If the observation indicates that the action was carried out successfully (e.g., it provides new information, describes the environment, or gives a positive confirmation), respond with:
<ais> True </ais>
- If the observation indicates that the action could not be performed (e.g., includes phrases like "You can't..." or "You cannot..."), respond with:
<ais> False </ais>
Note:
- An unsuccessful action usually explicitly states that the player cannot do something, or that the action fails.
<END OF INSTRUCTIONS>

Table 4: Prompt template: Action Validation.

Instruction:
<START OF INSTRUCTIONS>
You're going to extract triples in the format <subject, relation, object> from an input Observation along with previous actions you did, originating from a text-based game. Focus solely on where the character ('You') is located, what objects are in that location, and their immediate properties. The maximum length for any object name in the triples is three words, where length of location name has no limit.
Rules:
1. If the observation doesn't describe an environment or information is insufficient (e.g., "Opened", "Taken"), output |start| none |end| and skip other points.
2. Always use 'in' as the relation to represent the character's location. Convert any spatial descriptions (e.g., 'are facing', 'are standing', 'are behind') to the 'in' relation. If the input begins with a Room name (starts with a capital letter and does not end with a period), use it as the location.
Example:
Input: "Stairwell (First Floor) You're in the north stairwell."
Triple: <You, in, Stairwell (First Floor)>
3. If the observation doesn't include a precise location, do not provide any <You, in, *> triple.
4. Use 'have' as the relation to represent interactive objects present in the location. Focus only on the objects themselves as the 'obj' in the triple. Ignore decorative details unless they indicate an interactive object. Limit object names to a maximum of three words.
Example:
Input: "There is a small mailbox here."
Triple: <[Location], have, mailbox>
5. Do not include additional details or properties of objects. Only extract the objects themselves, ensuring object names are no longer than three words. But if a object have a relation to another object, such as 'in' and 'on', then extract that relation.
Example:
Input: "A buzzing water fountain has been moved."
Triple: "<[Location], have, water fountain>"
Input: "A sock is on th table."
Triple: "<[Location], have, sock>, <[Location], have, table>, <sock, on, table>"
6. If the input specifies a requirement or action needed to continue, use <location/object, need/require, something to action>.
Example:
Input: "Forest. You would need a machete to go further west."
Triple: <Forest, need, machete to go west>
7. For objects or locations mentioned with a direction (e.g., 'to the north', 'up to', 'down'), use <current location, direction, [new location]/to [direction]>.
Example:
Input: "Hall. To the southwest is the entrance to the Computer Site, and you can go east here as well as go up with a stair."
Triples: <Hall, southwest, Computer Site>, <Hall, east, to east>, <Hall, up, to up>
Note: Pay more attention to objects and directions than to objects' states or other decorative details.
Now, extract the relationships for the input step by step and merge all the results into a single output enclosed within |start| * |end|, where * represents the list of extracted triples.
<END OF INSTRUCTIONS>

Table 5: Prompt template: Relation Extraction.

Instruction:
<START OF INSTRUCTIONS>
You wil receive a previous input(step) from a text-based IF game, and please split the input into two parts, action and objs, as "<verb; [objs]>". Please follow these instructions to complete the task step by step.
Use the following rules:
1. If the action is a simple directional command (e.g., "north" or "n"), the object list should be empty.
For example:
Input: "west"
Response: "<act> <west; []> </act>"
2. If the action is "take all" or another "all" command (e.g., "take all"), treat "take all" as the verb and leave the object list empty.
For example:
Input: "drop all"
Response: "<act> <drop all; []> </act>"
3. If there are objects following the Verb (e.g., "eat", "take") or Verb phrase (e.g., "drop down", "go around"), list them. If prepositions (e.g., "on", "at", "with") are present, include them in the verb phrase using "&" as a placeholder, and list each noun object individually.
Final Output:
Use <act> <verb; [objs]> </act> format for final output where: "verb" represents the action phrase with placeholders "&" for objects. objs is a list of object nouns.
<END OF INSTRUCTIONS>

Table 6: Prompt template: Splitting Action.

Instruction:
You are playing the classic text-based interactive fiction game. Your goal is to explore, solve puzzles, collect treasures, and reach the winning end state. Throughout the game, you will:
1. Receive a history of the game's the action you performed, the new observation representing what you see or experience after your action. The action will help you understand what led to the new observation.
2. Have access only to the last 10 turns of conversation as your history. You must rely on these 10 turns and your internal reasoning to keep track of your location, inventory, puzzles, and progress.
3. Receive current new observation based on the last action and the current game states as input.
4. Produce all responses formatted between "|start|" and "|end|".

Your Task:
- At each turn, carefully read the provided new observation and the action you performed.
- Use your internal chain-of-thought to determine the best possible action to advance in the game, considering your inventory, location, known puzzles, and the limited historical context you have.
- Once you have reasoned through your options, produce exactly ONE game command.
- Always Format your command as this at the end of your response:
Final Command:
|start| [your chosen command] |end|
Guidelines:
- Avoid random or nonsensical actions. Each move should serve a purpose or help solve a puzzle.
- Try to use player (human) logic to guide your decision.
- You can Use 'look' command to examine the current location. And 'inventory' command to examine your inventory.
- Maintain continuity by leveraging the last 10 turns of conversation and the last action you performed as a guide. Use your internal chain-of-thought to continue.
- Always think first, then act. The chain-of-thought is mandatory before producing the final command.

Remember, you are playing a text-based game. Follow these instructions diligently, use your chain-of-thought to reason about your actions, and only format your final chosen command between "|start|" and "|end|".

Table 7: Prompt template: Baseline action generation.

Instruction:
<START OF INSTRUCTIONS>
You are a game engine summarizer. Your task is to read the current log of the game state and produce a concise, cohesive summary of the player's progress so far (This happens every time the player gets a score or loses a score). Do NOT reveal any hidden or undiscovered information. Focus only on details the player already knows or has directly experienced.
A list of "Step" will be provided. Each step includes:
- An observation (what the player sees),
- Info about moves and current score,
- The action taken just before the observation.
Summary Structure:
1. "location": where the player is (or what area is described) when the score changes. If the player has died, give the location name before death.
1.1 - One Location name Only.
1.2 - Description of situation.
2. "puzzle_status": what puzzles or obstacles have been solved to earn/lose the points.
2.1 - ONLY related steps to solve the puzzles directly. Any requirement for solving the puzzles, such as 'player need to <step>open door<step> at Room1 to enter <loc>Room2<loc>.
2.2 - Description of the puzzle.
3. "scoring": how the player earned/lost points for the last step. Any action leads to earning/losing points.
3.1 - Step done to earn/lose points.
3.2 - How many points are changed?
4. "important_experience": The experience can be used for the future. Only the most notable and valuable clues or items the player learned about for the global game experience or any warning must be recorded through all previous logs. Only Focus on confirmed information.
Earn Points - ONLY when player earn points, then we only need to know what leads to earn points and ingore other unchecked information.
- For example: 'player noticed there is a rabbit on the table (unchecked)' is not experienced. 'Room1. player open a locked door by a key (The key got from the roof)' is the experience as' player need to go to <loc>roof<loc> for a <obj>key<obj> to open the locked door in <loc>Room1<loc>.
Lose Points - ONLY when the player loses points (died usually or lost in the game, where 'lost' here means the player earned no points for a long time), you also need to give suggestions for the future.
- For example: 'player died in Room2. (Player saw a rabbit on table in Room1, but player did nothing with the rabbit)' you can now give the suggestion for next time that try to check <obj>rabbit<obj> before going <loc>Room2<loc>.
Remember:
- If no related puzzles (solve puzzles to show new location or new environment observations or earn points) are encountered, the whole 'puzzle_status' needs to be "No puzzles encountered yet."
- Please focus on how the player scored points with related puzzles and situations that occurred.
- Do not reveal hidden or undiscovered info.
- Keep it concise and factual based on the logs.
- When giving "important_experience", please reflect like an expert player (Always think about why this happened) as the payer's 'trace game experience'.
- If player has not died, the '*Lose Points*' in 'important_experience' should be 'none'. If player has died, the '*Earn Points*' in 'important_experience' should be 'none'.
- In your reasoning, if you find more than one earning or losing points, please ONLY summarize the last one based on previous steps.
Final Output Format:
- In the final output for any 'loc name', please use <loc> loc name <loc> to mark it, as well as 'step did before' (which steps solved the puzzles) by marking in <step> step did <step>, as well as 'interacted obj' (which player did valid action to obj) by marking in <obj> interacted obj <obj>; where the 'interacted obj' in step doesn't need this marking. And give a structured output based on points.
- At the end of the response, please outline TAGs (no more than 4) between <tag> * </tag> that are used for retrieval. put main location in <room> * </room> as one of the tag.
- After TAGs, please also give the difficulty for current puzzles in between <dif> * </dif>. You can combine the history steps with your expert player's experience to define the difficulty. - Please think about it first. Then, give your final completed player experience summary between '|start|' and '|end|'.
<END OF INSTRUCTIONS>

Table 8: Prompt template: experience summarization.

Instruction:
<START OF INSTRUCTIONS>
Instructions for Generating a Next Command in Text-Based Interactive Fiction
—
Objective Craft a single, context-aware **next command** with it’s motivation that propels the game forward, based on the current map, recent actions, and history of attempts. This command should represent one immediate player action.
—
Principles for Exploration, Puzzle-Solving, and Earning Points
1. **Analyze the Current Game State**
- **Room & Map Details**: Assess where you are, noting any exits, known layout, and significant objects.
- **Recent Attempts**: Reflect on the previous actions, the motivation of taking that action and observation after this attempt.
- **Inventory Check**: Identify items on hand (keys, tools, etc.) that might solve current puzzles or overcome obstacles.
- **Objects & Interactions**: Focus on confirmed items or directions. If uncertain leads might advance the game, consider them cautiously.
- **Action Selection**: Only choose to interact with an object (or perform an action) if you’re confident it will move the story forward.
2. **Use Retrieved Experiences and Past Attempts**
- **Relevance**: Apply past successes or observed clues that align with the current room or situation.
- **Avoid Repetition**: Do not repeat failing commands indefinitely. If a command fails, adjust strategy.
- **Focus on Gains**: Prioritize moves likely to unlock new paths, uncover essential items, or yield valuable information.
3. **Formulate a Single Effective Command**
- **One Action**: Provide exactly one executable command.
- **Purpose**: Briefly ensure it’s the most logical next step, considering both context and success likelihood.
- **Move command**: The full directions are [’north’, ’south’, ’east’, ’west’, ’southeast’, ’southwest’, ’northeast’, ’northwest’, ’up’, ’down’]
4. **Output Format**
- Present the final command and a short motivation in the following format without extra commentary:
“‘
You internal reasoning steps Here.
|start|
<com>[command]</com>
<rea>[short motivation for the decision-making reason]</rea>
|end|
“‘
—
Adaptation and Fallback Rules
1. **Priority Usage**
- **Highest Priority**: Items in ‘temp_have‘.
- **Next**: Options in ‘may_direction‘ or ‘may_have‘.
- **Then**: Verified directions (‘direction‘) or items (‘have‘).
2. **Conflict Resolution**
- Disregard prior attempts known to fail at this location or context.
- Validate uncertain (‘may_‘) directions or items before fully committing to them.
- After verify all the exits in one room then you can fully trust the map.
3. **Fallback Strategies**
- If uncertain, explore unvisited areas or re-examine (’look’) the current room.
- Look for overlooked clues or alternative ways forward.
4. **Exploratory Commands**
- If tools are available, think of how to use them on obstacles.
- In case an exploration fails, attempt a different angle—return to a previous room, look around again, or try another approach.
- **Explore the world**: It’s better to try all directions in each room to identify the exit and update the game map. For ‘may_direction‘, consider testing that path (e.g., “north”).
—
Remember: You are navigating a text-based world. Combine current observations with past knowledge to decide the best single move.
<END OF INSTRUCTIONS>

Table 9: Prompt template: LPLH action generation.