# How Far Can LLMs Improve from Experience? Measuring Test-Time Learning Ability in LLMs with Human Comparison

**Anonymous ACL submission**

## Abstract

As evaluation designs of large language models may shape our trajectory toward artificial general intelligence, comprehensive and forward-looking assessment is essential. Existing benchmarks primarily assess static knowledge, while intelligence also entails the ability to rapidly learn from experience. To this end, we advocate for the evaluation of **Test-time Learning**, the capacity to improve performance in *experience-based, reasoning-intensive tasks* during test time. In this work, we propose semantic games as effective testbeds for evaluating test-time learning, due to their resistance to saturation and inherent demand for strategic reasoning. We introduce an objective evaluation framework that compares model performance under both limited and cumulative experience settings, and contains four forms of experience representation. To provide a comparative baseline, we recruit eight human participants to complete the same task. Results show that LLMs exhibit measurable test-time learning capabilities; however, their improvements are less stable under cumulative experience and progress more slowly than those observed in humans. These findings underscore the potential of LLMs as general-purpose learning machines, while also revealing a substantial intellectual gap between models and humans, irrespective of how well LLMs perform on static benchmarks. Code and data are available[1].

*"Give a man a fish, and you feed him for a day; teach a man to fish, and you feed him for a lifetime."*

## 1 Introduction

As large language models continue to advance, the design of their evaluations becomes increasingly important, as it shapes the development priorities of the next generation of models and guides the broader trajectory toward artificial general intelligence (Chang et al., 2024). Current benchmarks predominantly focus on measuring the expertise of language models in performing specific tasks. However, intelligence is not solely defined by the possession of expert knowledge (Krathwohl, 2002; Minsky, 1988). For example, individuals without profound knowledge can still demonstrate intelligence through their speed to acquire new skills through experience (Silver and Sutton, 2025). This dimension of intelligence, the capacity for rapid learning, remains largely overlooked in existing evaluation frameworks.

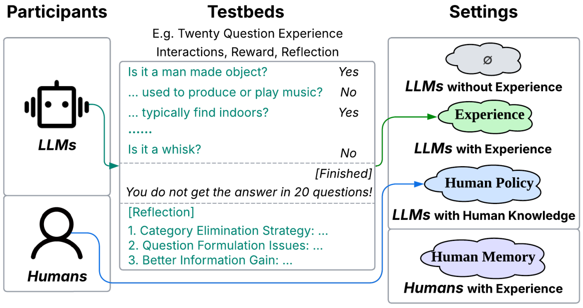
Participants
Testbeds
Settings
E.g. Twenty Question Experience
Interactions, Reward, Reflection
Is it a man made object?
Yes
... used to produce or play music?
No
... typically find indoors?
Yes
......
Is it a whisk?
No
[Finished]
You do not get the answer in 20 questions!
[Reflection]
1. Category Elimination Strategy: ...
2. Question Formulation Issues: ...
3. Better Information Gain: ...
LLMs
Humans
∅
LLMs without Experience
Experience
LLMs with Experience
Human Policy
LLMs with Human Knowledge
Human Memory
Humans with Experience

Figure 1: Test-time Learning Evaluation Pipeline.

Assessing the ability to learn quickly is challenging. Under the current LLM development paradigm, models undergo massive pre-training followed by domain-specific alignment (Achiam et al., 2023; Liu et al., 2024). Models are typically compared based on their final performance outcomes, without constraints on the amount of task-relevant training data they utilize. In this paper, we do not argue for altering this training paradigm, as it is the reason of success. Rather, we aim to design an evaluation framework that can directly measure a model's ability to improve rapidly at test time.

The characteristic we aim to evaluate aligns closely with the concept of "Test-time Learning", which refers to a model's ability to adapt and improve through its own test-time experience. The desirable end-state for artificial intelligence should have the ability to effectively improve its perfor-

---

[1] https://anonymous.4open.science/r/Test-time-Leaning

mance through a limited number of experiences by interacting with environments, reflecting on feedback and rewards, rapidly acquiring in-context or in-weight policies, and acting adaptively. Moreover, these test-time improvements should be capable of accumulating as experience grows, enabling continual adaptation and learning.

The concept of *test-time learning* shares conceptual similarities with *in-context reinforcement learning* (Laskin et al.; Lee et al., 2023; Grigsby et al.; Lu et al., 2023a) and *agent self-evolution* (Tao et al., 2024) to some extent. However, it is distinguished from these paradigms in several fundamental ways. (1) Generality of Environment: In-context reinforcement learning typically operates within classical RL domains such as adversarial bandits (Duan et al., 2016) or the dark room setting (Laskin et al.), which are characterized by constrained environments and limited action spaces. In contrast, test-time learning emphasizes generalization in open-ended environments, where the action space spans the full token space of a language model. (2) Beyond Memorization: Research on agent self-evolution has largely focused on tool-use or coding tasks that rely heavily on rote memorization or repeated exposure to similar instances (Qian et al., 2024). These setups often allow models to improve simply by recalling prior examples. Test-time learning is achieved in experience-based, reasoning-intensive tasks that require discovering latent patterns and executing self-proposed policies beyond surface-level recall.

In this work, we propose an **objective framework to evaluate the test-time learning ability** of current large language models. Rather than relying on static tasks like academic olympiads, we adopt competitive games, which are dynamic, resistant to saturation, and embed latent strategies, making them ideal for studying test-time learning.

Furthermore, we systematically evaluate performance across four test-time experience settings: (1) full experience with interactions, rewards, and model reflection; (2) model-derived policy based solely on game rules; (3) model-derived policy informed by both rules and accumulated test-time experience; and (4) human-authored policy. To compare LLM performance with human reasoning, we also recruit human annotators to perform the same task. The results reveal a clear gap between human and model test-time learning capabilities, highlighting promising directions for future research.

The experiment results show that LLMs demonstrate measurable test-time learning ability; however, these gains are not stable and consistent when experience accumulates. In contrast, human participants exhibit more stable and rapid learning. These findings highlight the need for further evaluation and improved training strategies to enhance the test-time learning of LLMs.

Importantly, our aim is not to build an elaborate framework centered on agentic workflows, but rather to propose a lightweight and objective pipeline for assessing whether models can benefit from test-time experience. We believe that systematic evaluation of test-time learning constitutes a key step toward advancing the capabilities of large language models.

## 2 Related Work

### 2.1 Test-time Learning

The concept "test-time learning" shares certain similarity with "test-time training", "in-context reinforcement learning" and "self-evolution" but also adopts key distinctions in its focus and formulation.

The first two concepts involve weight updates. Test-time training (Sun et al., 2020; Liu et al., 2021; Gandelsman et al., 2022; Sinha et al., 2023; Sun et al., 2019) primarily addresses distributional or domain shifts between training and test data by adapting model parameters at inference time. In-context reinforcement studies (Laskin et al.; Lee et al., 2023; Grigsby et al.; Lu et al., 2023a) involves training models from scratch to perform reinforcement learning tasks via in-context tokens. Self-evolution studies focus on performance improvements through in-context interactions without parameter updates (Tao et al., 2024; Lu et al., 2023b). For instance, Lange et al. (2024) proposed prompting strategies that enhance performance through structured interactions, and Yu and Feng (2025) introduced agentic workflows that integrate human knowledge to guide model behavior and maximize gains. Most prior work emphasizes engineering pipelines to improve in-context performance, often in application settings such as web navigation, tool use, or code generation—domains where improvements are frequently driven by retrieval or surface-level similarity to prior examples, rather than by the development of general strategies or deeper reasoning. As noted by Silver and Sutton (2025), “now is the time of experience,” highlighting the emerging view that future intelligent agents must learn through interaction to achieve higher-level of reasoning, rather than rely solely on static question answering-style evaluations.

In this work, we aim to objectively evaluate the extent to which LLMs can leverage experience at test time. Specifically, we quantify models' test-time gains and compare them against improvements guided by human-authored policies and human learning trajectories.

### 2.2 Evaluation Environments

For reasoning-intensive evaluation environments, in-context reinforcement learning studies have explored semantic and visual representations of reinforcement tasks such as the adversarial bandit (Laskin et al.; Lee et al., 2023), dark room (Laskin et al.; Lee et al., 2023) and Partially Observable Process Gym (Morad et al., 2023; Lu et al., 2023a), a set of simple environments designed to benchmark memory in deep RL. However, these tasks involve closed-ended environments with limited action spaces and are often easily solved by current large language models, as they may already encode effective policies, e.g., upper confidence bound(Garivier and Moulines, 2011). In contrast, we focus on open-ended experience-based reasoning-intensive tasks with token-level action spaces and moderate difficulty, where the optimal policy is not readily accessible or encoded in the model. Regarding the self-evaluation of LLMs, recent works have employed web-used (Yao et al., 2022), tool-assisted (Lu et al., 2023b), or static benchmarks, including math (Cobbe et al., 2021), code generation (Jiang et al., 2023; Luo et al., 2023), and general-purpose benchmarks (Chiang et al., 2023). However, these static evaluations are prone to saturation, and observed improvements may result from memorization or recall rather than from enhanced reasoning via learned policies.

In this work, we propose competitive game environments as effective testbeds for evaluating the test-time learning ability of LLMs. These environments are dynamic, resistant to saturation, open-ended, reasoning-intensive, and policy-driven, making them well-suited for assessing model ability to learn and adapt through experience.

## 3 Test-time Learning

### 3.1 Testbeds

The optimal environment for evaluating the test-time learning ability of large language model should satisfy the following requirements: 1) Moderate Difficulty: The environment should not admit a readily accessible optimal policy, either due to the nature of the task or the current limitations of large language models. 2) Structured Regularity: Tasks should contain underlying patterns that can be uncovered and leveraged through interaction and reasoning to enhance performance. 3) Beyond Memorization: Success should depend not on recalling previous answers, but on identifying generalizable rules or strategies that drive improvement.

These criteria highlight the importance of reasoning over purely knowledge-rich contexts. Classic reinforcement learning settings, such as adversarial bandit (Laskin et al.), have been rendered less meaningful for test-time learning evaluations, as many models have already internalized algorithms like Upper Confidence Bound in their knowledge. To address this, we adopt three diverse environments to evaluate test-time learning: a mathematics benchmark, a single-agent semantic game, and a multi-agent semantic game.

**AIME 2025** (MAA, 2025) refers to the American Invitational Mathematics Examination 2025, used to identify candidates for the U.S. team in the International Mathematical Olympiad (IMO). We leverage this most recent mathematics benchmark to examine the test-time learning capabilities of large language models in solving high-level mathematical problems.

**Twenty Question** (Abdulhai et al., 2023; Zhou et al., 2024) is a dialogue-based multi-turn single agent task in which a large language model attempts to identify a target word from a fixed set of 157 candidate words by asking up to twenty yes/no questions. The environment responds with "Yes", "No", or "Invalid" if the question is not a valid yes/no query. To ensure consistent understanding of the questions, the environment is simulated using the same LLM as the questioning model.

The 157 candidate words, adopted from prior work (Zhou et al., 2024), span diverse categories including animals, art, clothes, electronics, fruits, furniture, garden supplies, jewelry, kitchen tools, musical instruments, nature, office supplies, sports, tools, toys, vegetables, and vehicles. The candidate set remains fixed across the games, providing a controlled setting to evaluate whether the LLM can learn effective categorization and formulate increasingly informative dichotomous questions during test time. Performance is measured by NDCG@20 based on the rank of the correct guess.

**Who is undercover** (Xu et al., 2023) is a dialogue-based multi-turn multi-agent task. Each player is assigned a secret word: one player receives a distinct word as the undercover, while all others, civilians, share the same word. In each round, players provide verbal clues related to their secret words. By analyzing both their own and others' clues, players attempt to infer their roles. The objective for civilians is to identify the undercover, while the undercover aims to conceal their identity. Note we use the neutral word "difference" and "normal" instead of "undercover" and "civilian" in task instructions. This is motivated by the observation that large language models often refuse to acknowledge being the "undercover" due to value misalignment, as further discussed in Appendix C. The performance is evaluated based on the win rate.

### 3.2 Test-time Learning Setting

It is important to note that our objective is not to design an elaborate framework for maximizing task completion rates. Rather, we aim to provide a lightweight and objective evaluation framework that assesses a model's test-time learning, comparing its performance with and without prior experience, as well as against human-authored policies grounded in human reasoning. To this end, we adopt a vanilla evaluation setup consisting of two settings: a fixed number of experience setting (Laskin et al.) and an incremental experience setting (Suzgun et al., 2025).

#### 3.2.1 Evaluation with Experience

Table 1: Token Lengths of Context

| | Instruction | Experience | Policy |
|---|---|---|---|
| Twenty Question | 463 | 1011 | 243 |
| Who is Undercover | 341 | 2356 | 261 |

We aim to qualitatively assess whether current large language models exhibit test-time learning capabilities and the extent to which they improve. To this end, we encode historical experience and compare model performance with and without it.

We investigate efficient and objective methods to encode this historical experience. Table 1 reports the average context lengths for instruction, experience, and derived policy. To fully leverage past experience, the experience includes dialogue interactions, rewards, and model's self-reflections on interactions and rewards. The strategy is derived by the model itself based on all past experience.

In pilot studies, we experiment with two approaches: incorporating the full history experience directly, and self-derived policy from the full history. We fix the number of experience to five rounds, leading to context lengths of approximately 5k and 12k for Twenty Questions and Who is Undercover, respectively, while the derived policy contexts average 243 and 261 tokens. Although the first approach provides complete information, it incurs higher computational costs and underperforms compared to the second. Therefore, we adopt policy-based representations of past experience for further evaluation. This setup is illustrated in the left panel of Figure 2. To further isolate the influence of the model's self-derived policy pipeline, we include a rule-based policy as a baseline for comparison with the experience-based policy, in which strategies are derived from both rules and accumulated experience. This comparison helps ensure that observed improvements can be attributed to the incorporation of experience.

#### 3.2.2 Evaluation with Incremental Experience

The previous setting evaluates the test-time learning given limit amounts of prior experience. If a model demonstrates performance gains from such experience, it becomes essential to investigate whether these test-time improvements persist and accumulate as additional experience is acquired.

This motivates an incremental evaluation setting that requires efficient management of past experience. To support dynamic policy updates with growing experience, we adopt the memory management pipeline (Suzgun et al., 2025). As illustrated in the right panel of Figure 2, the agent without experience performs k independent test rounds, while the agent with experience conducts the same k rounds with a continuously updated policy pool based on accumulating experience. To ensure robust evaluation, we sample each setting (with and without experience) three times and compute the cumulative average reward. Let $r\_\mathrm{base}(t, i)$ denote the reward obtained by the agent without experience at test round $t$ in sample $i$, and $r\_\exp(t, i)$ denote the corresponding reward for the agent with experience. The cumulative average reward for the agent with experience up to round $t$ is denoted by $R\_\mathrm{his}(t)$. The computation of $R_{\mathrm{his}}(t)$ is provided below; $R_{\mathrm{base}}(t)$ is computed analogously.

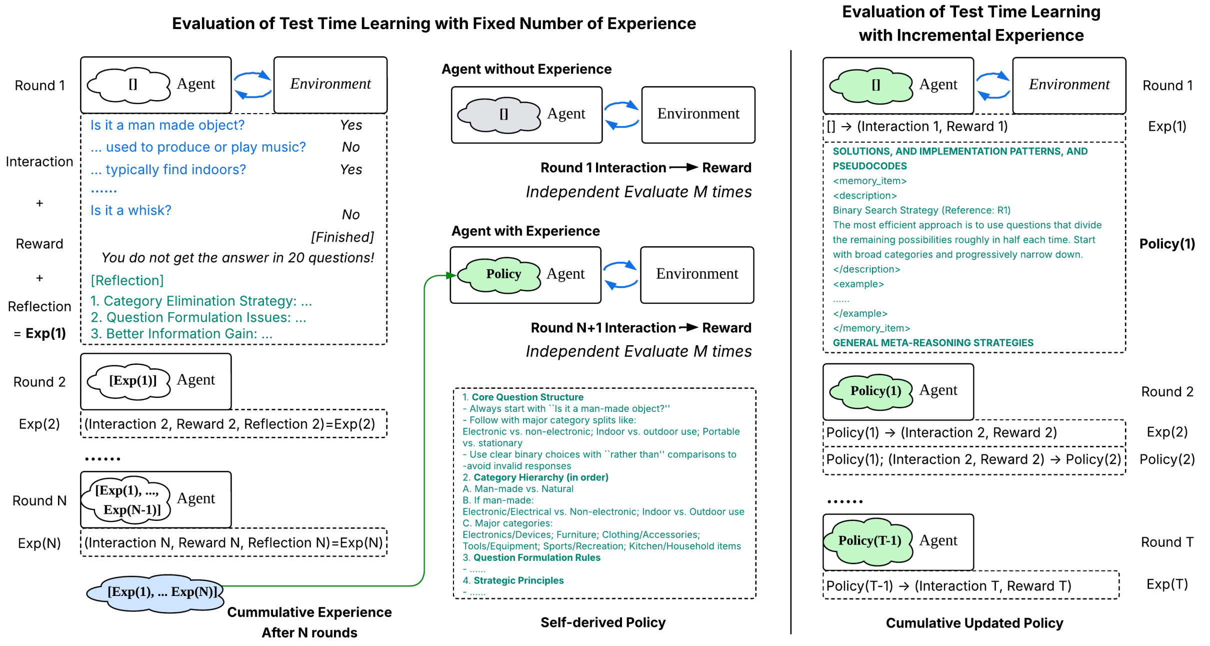
Evaluation of Test Time Learning with Fixed Number of Experience
Round 1
Interaction
+
Reward
+
Reflection
= Exp(1)
[]
Agent
Environment
Is it a man made object?
Yes
... used to produce or play music?
No
... typically find indoors?
Yes
......
Is it a whisk?
No
[Finished]
You do not get the answer in 20 questions!
[Reflection]
1. Category Elimination Strategy: ...
2. Question Formulation Issues: ...
3. Better Information Gain: ...
Round 2
[Exp(1)]
Exp(2)
(Interaction 2, Reward 2, Reflection 2)=Exp(2)
......
Round N
[Exp(1), ..., Exp(N-1)]
Exp(N)
(Interaction N, Reward N, Reflection N)=Exp(N)
[Exp(1), ... Exp(N)]
Cummulative Experience
After N rounds
Agent without Experience
Round 1 Interaction → Reward
Independent Evaluate M times
Agent with Experience
Policy
Round N+1 Interaction → Reward
Independent Evaluate M times
1. Core Question Structure
- Always start with ``Is it a man-made object?''
- Follow with major category splits like:
Electronic vs. non-electronic; Indoor vs. outdoor use; Portable vs. stationary
- Use clear binary choices with ``rather than'' comparisons to
-avoid invalid responses
2. Category Hierarchy (in order)
A. Man-made vs. Natural
B. If man-made:
Electronic/Electrical vs. Non-electronic; Indoor vs. Outdoor use
C. Major categories:
Electronics/Devices; Furniture; Clothing/Accessories; Tools/Equipment; Sports/Recreation; Kitchen/Household items
3. Question Formulation Rules
- ......
4. Strategic Principles
- ......
Self-derived Policy
Evaluation of Test Time Learning
with Incremental Experience
Round 1
Exp(1)
[] → (Interaction 1, Reward 1)
SOLUTIONS, AND IMPLEMENTATION PATTERNS, AND PSEUDOCODES
<memory_item>
<description>
Binary Search Strategy (Reference: R1)
The most efficient approach is to use questions that divide the remaining possibilities roughly in half each time. Start with broad categories and progressively narrow down.
</description>
<example>
......
</example>
</memory_item>
GENERAL META-REASONING STRATEGIES
Policy(1)
Round 2
Policy(1) → (Interaction 2, Reward 2)
Exp(2)
Policy(1); (Interaction 2, Reward 2) → Policy(2)
Policy(2)
......
Round T
Policy(T-1)
Policy(T-1) → (Interaction T, Reward T)
Exp(T)
Cumulative Updated Policy

Figure 2: Test-time Learning Evaluation Settings with Fixed Amount and Incremental Experience.

$$r_{\text{exp}}(t) = \begin{cases} \frac{\sum_i r_{\text{base}}(0,i)+\sum_i r_{\text{exp}}(0,i)}{|r_{\text{base}}(0,\cdot)|+|r_{\text{exp}}(0,\cdot)|}, & t=1 \\ \frac{\sum_i r_{\text{exp}}(t,i)}{|r_{\text{exp}}(t,\cdot)|}, & t>1 \end{cases} \qquad (1)$$

$$R_{\text{exp}}(t) = \frac{\sum_{1\le i\le t} r_{\text{exp}}(i)}{t} \qquad (2)$$

## 4 Experiments

In the experiments, we aim to answer the following questions:
(Q1) Do current large language models exhibit the ability to learn at test time?
(Q2) Can large language models achieve stable and consistent improvements when experience accumulates?
(Q3) How do humans adapt and improve their performance through experience?
(Q4) How do thinking models perform in test-time learning scenarios?

### 4.1 Experimental Setup

We aim to evaluate whether the current top-tier large language models have the ability to improve at the test time. Specifically, we evaluate gpt-4o (Hurst et al., 2024), Claude 3.5 Sonnet (Anthropic, 2024) and DeepSeek-V3 (Liu et al., 2024). We set the temperature to 1 to support the dynamic testbeds. For overall performance evaluations, we set prior interactions N=5, test cases M=32, which we find to yield stable results. In the cumulative setting, we extend the evaluation t=50 rounds.

During each interaction, the model is instructed to first perform explicit reasoning before generating its final output. The final response (a question, reflection, or policy in Twenty Questions; a speech, vote, reflection, or policy in Who is Undercover) is enclosed within <answer></answer> tags to ensure clarity and facilitate objective evaluation of both reasoning quality and task performance.

In the single-agent setting, the environment is simulated using the same model under evaluation to ensure alignment in question understanding and knowledge base. In the multi-agent setting, all other agents are instantiated with the same backbone LLM as the test agent to isolate test-time improvements from potential gains due to mere familiarity with another model's behavior. For each evaluation setting, the order of test rounds is fixed to ensure consistency across trials.

### 4.2 Overall Test-time Learning Performance (Q1)

We begin by investigating whether top-performing large language models exhibit measurable improvements at test time when provided with prior experience. Table 2 summarizes the overall performance across three environments under four evaluation settings: (1) without any policy, (2) with model-derived policy based solely on rules, (3) with model-derived policy based on both rules and

Table 2: Evaluation of Test-time Learning Ability of LLMs. "w/o Policy" denotes the baseline setting where the model is provided only with task rules. "w/ Rule Policy" indicates that the model receives both the rules and a test-time policy based only on rules. "w/ Exp. Policy" refers to having both rules and test-time policy from rules and model five rounds of experience containing interactions, rewards and reflections. "w/ Human Policy" indicates that the model is given rules along with a human-authored policy based on human understanding of the task. The best results are shown in **bold** and the second best are underlined.

| Task | Setting | GPT-4o | Claude 3.5 Sonnet | DeepSeek-V3 |
|---|---|---|---|---|
| **AIME 2025**<br>Single-Turn Math Problem | w/o Policy | 0.0000 | 0.0333 | **0.4333** |
| | w/ Exp. Policy | **0.0667** | **0.0667** | 0.3333 |
| | *Improve (%)* | *NA* | *100.00* | *-23.08* |
| **Twenty Question**<br>Multi-turn Single-agent | w/o Policy | 0.2422 | 0.2640 | 0.2641 |
| | w/ Rule Policy | 0.2199 | 0.1368 | 0.2033 |
| | *Improve (%)* | *-9.21* | *-48.18* | *-23.02* |
| | w/ Exp. Policy | 0.2563 | **0.2807** | 0.2746 |
| | *Improve (%)* | *5.80* | *6.33* | *3.97* |
| | w/ Human Policy | **0.2709** | 0.2624 | **0.2758** |
| | *Improve (%)* | *11.84* | *-0.61* | *4.41* |
| **Who is Undercover**<br>Multi-turn Multi-agent | w/o Policy | 0.1563 | 0.1250 | 0.2500 |
| | w/ Rule Policy | 0.0625 | 0.3125 | 0.1250 |
| | *Improve (%)* | *-60.01* | *150.00* | *-50.00* |
| | w/ Exp. Policy | 0.1719 | 0.1563 | 0.2813 |
| | *Improve (%)* | *10.02* | *25.04* | *12.50* |
| | w/ Human Policy | **0.1875** | **0.3438** | **0.4063** |
| | *Improve (%)* | *20.00* | *175.04* | *62.50* |

test-time experience, and (4) with human-authored policy. The inclusion of the human policy serves to assess the potentials of models.

In the Twenty Questions setting, we observe consistent performance gains when models are equipped with self-derived policies based on prior experience. In contrast, rule-based policies result in significant performance drops across all models, likely due to a misalignment between human-designed heuristics and model reasoning patterns, as further discussed in Section 4.6. Experience-based policies, however, lead to clear improvements, with Claude achieving the highest gain from its own test-time experience.

Interestingly, GPT-4o and DeepSeek-V3 both outperform their self-derived policies when provided with human-authored policies. This highlights a gap between the models' current test-time learning capabilities and their full potential, suggesting that either the quantity of experience or the quality of derived policies remains suboptimal. These limitations are further examined in Section 4.3 and Section 4.6. Claude performs marginally worse with human-authored policy, also indicating a possible misalignment between its internal reasoning and externally imposed guidance.

In Who is Undercover, test-time learning yields more substantial improvements. Claude again achieves the highest gain from experience-based policy, reinforcing its ability to leverage self-acquired strategies. Unlike other settings, the rule-based policy ranks as the second-best for some models, highlighting a divergent pattern in this multi-agent context. Additionally, human-authored policies consistently lead to the highest performance across all models, further underscoring the latent potential of test-time learning when guided by effective strategies. It is important to note that direct comparisons across models in this environment are not meaningful, as all agents in the multi-agent setting are instantiated using the same LLM that is being evaluated. This design ensures an objective assessment of test-time learning by isolating gains attributable to experience and strategic adaptation, rather than confounding effects such as familiarity with another model's behavior. Full instances of model-generated and human-authored policies are provided in Appendix B and analyzed in Section 4.6.

*Finding 1: Policies derived from past experience at test time yield measurable improvements across models and tasks.*

*Finding 2: The superior performance under human-authored policies reveals the untapped potential for enhancing models' test-time learning capabilities.*

### 4.3 Cumulative Improvement (Q2)

The above results demonstrate that large language models possess the ability to improve at test time. We next examine whether this improvement is consistent as experience accumulates. To this end, we adopt the cumulative evaluation setting described in Section 3.2.2. Figure 3 presents cumulative rewards over 50 rounds in the Twenty Questions task, comparing model performance with and without test-time policies derived from past experience.

Model performances vary. Claude successfully leverages cumulative experience, whereas other models struggle to maintain or improve performance as experience accumulates. For Claude, the experience-enabled setting consistently outperforms the baseline, particularly within the first five rounds, indicating effective strategy accumulation. However, the performance gap narrows in later rounds, suggesting diminishing returns from additional experience. GPT and DeepSeek show minimal gains from the accumulation of experience at test time. For GPT-4o, both curves overlap in the early rounds, with the experience-enabled setting beginning to slightly outperform the baseline around rounds 15–20. In contrast, DeepSeek-V3 shows a decline in performance after five rounds of accumulated experience, while the baseline remains stable. This suggests that its policy refinement process may introduce noise or compounding errors, limiting its ability to leverage experience effectively.

*Finding 3: Results reveal substantial differences in the consistency and effectiveness of test-time learning across models as experience grows.*

### 4.4 Human Study (Q3)

In the previous experiments, we demonstrate that certain large language models exhibit the ability to learn at test time through cumulative experience. To further understand the rate of improvement, we compare model learning speed with human.

We recruited eight human participants (undergraduate and PHD students) to perform the same Twenty Question task, playing 20 rounds cumulatively. Their results are summarized in Table 4, and their cumulative rewards are plotted alongside those of the best-performing model, Claude. Participants are divided into two groups based on performance variance across rounds.

The upper figure shows that all humans in this group achieve greater cumulative gains than Claude after 20 rounds, approaching near-optimal performance (represented by the black dotted line indicating the reward of perfect binary questioning). The lower figure includes participants with higher performance variability; nevertheless, their final cumulative rewards still exceed those of the LLM.

*Finding 4: Current top-tier large language models exhibit slower test-time learning speed compared to the learning efficiency of humans in the experience-based reasoning-intensive task.*

### 4.5 Performance of Thinking Models (Q4)

Table 3: Test-time Learning Performance of Thinking Models in Who is Undercover environment.

| **Setting** | **o1** | **DeepSeek-R1** |
|---|---|---|
| w/o Test-time Policy | 0.3000 | **0.2500** |
| w/ Test-time Policy | 0.2000 | 0.0000 |
| w/ Human Policy | **0.4000** | 0.2000 |

In pervious experiments, we evaluate large language models without explicit thinking mode. In this section, we examine the performance of thinking model: o1 (Jaech et al., 2024) and Deepseek-r1 (Guo et al., 2025), as reported in Table 3.

As shown in the table, test-time learning improvements are not observed for either thinking model when provided with self-derived policies. For o1, the test-time performance with self policy decrease and the test-time performance with human policy increase. For o1, performance decreases when incorporate test-time policy but improves when guided by a human policy, suggesting potential limitations in its ability to generate effective strategies autonomously, while still being capable of leveraging prior experience. For DeepSeek-R1, performance declines under both self-derived and human-authored policy conditions, compared to its baseline with no prior experience. This aligns with the findings reported in its original paper, which notes that few-shot prompting consistently degrades R1's performance. The authors explicitly recommend presenting tasks in a zero-shot format for optimal outcomes, suggesting that R1's internal reasoning is optimized for direct problem descriptions rather than for accumulating and adapting to test-time experience.

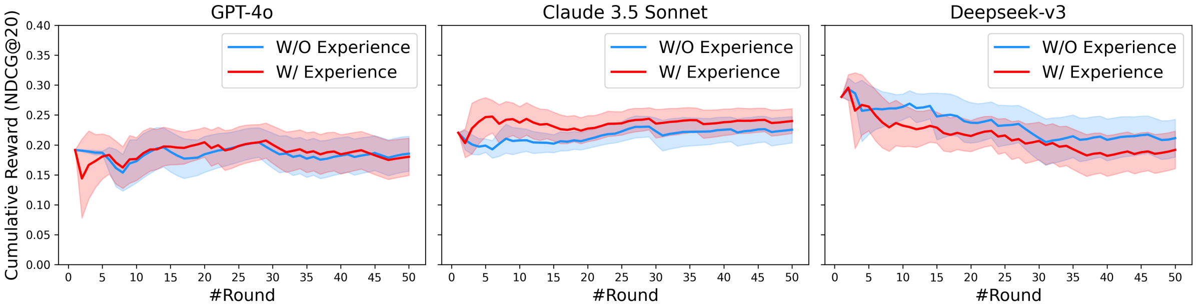

Figure 3: Cumulative Test-Time Learning Performance on Twenty Question.

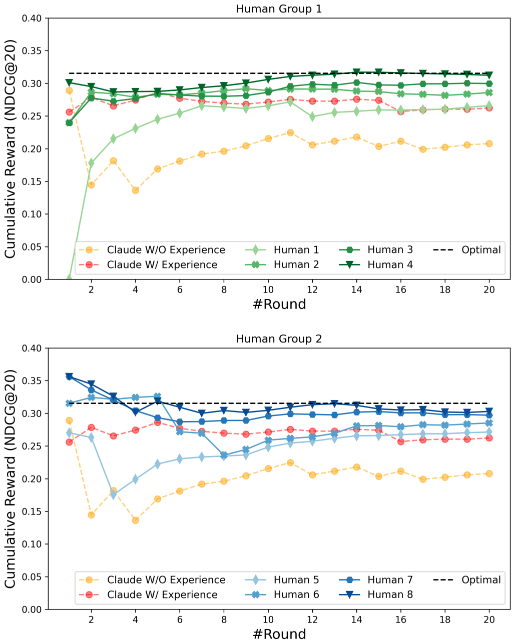

Figure 4: Human Performance on Twenty Question.

*Finding 5: Test-time learning is not observed in thinking models, consistent with the findings reported in R1's original paper that CoT in few-shot cases may degrade model performance.*

### 4.6 Further Analyses

In the Twenty Questions environment, test-time improvements (*w/ Exp. Policy* vs. *w/o Policy* in Table 2) primarily stem from earlier identification of item categories. Test-time policies such as "Begin with high-level distinctions" and "Identify the category of the answer word within the first five questions" help the model avoid overly specific guesses early on. We also analyze the failure of the *w/ Rule Policy* setting in this environment, which we attribute to a misalignment between model behavior and human preference. Model-generated questions often include specific examples (e.g., "Is it a living thing (animal, plant)?" or "Is it a ball (like basketball, baseball, football) rather than other sports equipment (like bats or rackets)?"), whereas questions in human-authored policies are general and abstract (e.g., "Is it a living thing?", "Does it use electricity?", "Is it commonly found indoors?"). These examples lead to the model adopt this format throughout the questioning process, which contributes to the performance decline. For humans, we observed more rapid test-time learning, with noticeable improvement after just a single game.

In the Who is Undercover environment, both DeepSeek and human demonstrate a key policy, "Deduce the opposing secret word". This is based on the observation that the undercover and normal players' words are typically semantically related. Recognizing this pattern allows participants to refine their clues and identities more effectively.

## 5 Conclusion

In this work, we advocate for evaluations of large language models' test-time learning ability, defined as the capacity to improve at test time in experience-based, reasoning-intensive tasks. Competitive games serve as effective testbeds due to their dynamic nature, resistance to saturation, and reliance on reasoning. We present an objective framework to access test-time learning under both static and cumulative experience settings, and evaluate models with different policies. We also compare the improvements of models to humans. Experimental results demonstrate that LLMs exhibit measurable test-time learning; however, these gains are often unstable and inconsistent across cumulative settings. In contrast, human participants show more stable and rapid learning. We highlight the need for greater emphasis on evaluation and training strategies to improve LLMs' test-time learning as a step toward artificial general intelligence.

## Limitations

This work aims to evaluate the test-time learning capabilities of large language models and compare their gains with humans. We employ one math benchmark and two semantic game environments as representative testbeds. While these settings provide meaningful insights, a broader range of evaluation environments is necessary for more comprehensive measurement of test-time learning.

Each experimental condition is run for thirty two to fifty rounds to ensure result stability. We examine five settings: no experience, rule-based policy without experience, full experience, experience-based policy, and human-authored policy, to provide a more thorough understanding of how LLMs benefit from test-time experience.

For the human studies, we recruit eight participants, each compensated $10 for completing 20 rounds of the Twenty Questions task, which typically takes 1 to 2 hours. While these participants offer a stable and useful baseline, involving more individuals across diverse environments would further strengthen the generalizability of our findings.

Finally, this paper currently does not incorporate test-time training, where a model's parameters are updated based on a small amount of test-time experience. This is due to two main reasons: the closed-source nature of the models evaluated, which precludes parameter access; and the current lack of effective in-parameter test-time learning methods tailored for reasoning-intensive tasks. Future work may explore this direction to further advance the understanding of test-time learning capabilities in LLMs.

## Ethical Considerations

In this study, we recruited eight human participants to complete twenty rounds of the Twenty Questions task. The task environment is non-sensitive and does not pose any potential risks or negative impacts to participants. All participants were fully informed that their interactions would be recorded and used for research purposes, and their consent was obtained prior to participation. In this paper, we used an AI assistant (GPT-4o) to check for grammatical errors. It was not used to directly generate any of the paper's content.

## References

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

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. Claude 3.5 sonnet.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, and 1 others. 2024. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

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

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. 2016. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. 2022. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385.

Aurélien Garivier and Eric Moulines. 2011. On upper-confidence bound policies for switching bandit problems. In *International conference on algorithmic learning theory*, pages 174–188. Springer.

Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. In *The Twelfth International Conference on Learning Representations*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, and 1 others. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Shuyang Jiang, Yuhao Wang, and Yu Wang. 2023. Selfevolve: A code evolution framework via large language models. *arXiv preprint arXiv:2306.02907*.

David R Krathwohl. 2002. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218.

Robert Lange, Yingtao Tian, and Yujin Tang. 2024. Large language models as evolution strategies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 579–582.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Stenberg Hansen, Angelos Filos, Ethan Brooks, and 1 others. In-context reinforcement learning with algorithm distillation. In *The Eleventh International Conference on Learning Representations*.

Jonathan Lee, Annie Xie, Aldo Pacchiano, Yash Chandak, Chelsea Finn, Ofir Nachum, and Emma Brunskill. 2023. Supervised pretraining can learn in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36:43057–43083.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Yuejiang Liu, Parth Kothari, Bastien Van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. 2021. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34:21808–21820.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. 2023a. Structured state space models for in-context reinforcement learning. *Advances in Neural Information Processing Systems*, 36:47016–47031.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Fei Mi, Baojun Wang, Weichao Wang, Lifeng Shang, and Qun Liu. 2023b. Self: Language-driven self-evolution for large language model.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

MAA. 2025. American invitational mathematics examination (aime).

Marvin Minsky. 1988. *Society of mind*. Simon and Schuster.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. 2023. Popgym: Benchmarking partially observable reinforcement learning. *arXiv preprint arXiv:2303.01859*.

Cheng Qian, Shihao Liang, Yujia Qin, Yining Ye, Xin Cong, Yankai Lin, Yesai Wu, Zhiyuan Liu, and Maosong Sun. 2024. Investigate-consolidate-exploit: A general strategy for inter-task agent self-evolution. *arXiv preprint arXiv:2401.13996*.

David Silver and Richard S Sutton. 2025. Welcome to the era of experience. *Google AI*.

Samarth Sinha, Peter Gehler, Francesco Locatello, and Bernt Schiele. 2023. Test: Test-time self-training under distribution shift. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2759–2769.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. 2020. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248. PMLR.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A Efros, and Moritz Hardt. 2019. Test-time training for out-of-distribution generalization.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. 2025. Dynamic cheatsheet: Test-time learning with adaptive memory. *arXiv preprint arXiv:2504.07952*.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. 2024. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*.

Lin Xu, Zhiyuan Hu, Daquan Zhou, Hongyu Ren, Zhen Dong, Kurt Keutzer, See Kiong Ng, and Jiashi Feng. 2023. Magic: Investigation of large language model powered multi-agent in cognition, adaptability, rationality and collaboration. *arXiv preprint arXiv:2311.08562*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Yajie Yu and Yue Feng. 2025. Policyevol-agent: Evolving policy via environment perception and self-awareness with theory of mind. *arXiv preprint arXiv:2504.15313*.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*.

# Appendix

## A Game Rule

### A.1 Twenty Question

You are playing the game Twenty Questions.
In this game, there are 157 candidate words: Airplane, Apple, Banana, Baseball, Baseball bat, Basketball, Battery, Bear, Bed, Belt, Blender, Boat, Bookcase, Boots, Bowl, Bracelet, Broccoli, Brooch, Bus, Bush, Cactus, Calculator, Calendar, Camera, Cantaloupe, Canvas, Car, Carrot, Cat, Celery, Chair, Chopstick, Clarinet, Computer, Computer keyboard, Cooking pot, Corn, Couch, Cow, Cucumber, Cup, Desk, Diary, Dog, Doll, Dress, Dresser, Drill, Drum, Earring, Elephant, Eraser, Flute, Football, Forest, Fork, Gloves, Glue, Golf ball, Grape, Guitar, Hairclip, Hammer, Harp, Hat, Headphone, Helicopter, Helmet, Horse, Jacket, Key, Kite, Knife, Lake, Lawn mower, Lego, Lion, Locket, Mango, Marker, Mattress, Meteorite, Microwave, Monitor, Motorcycle, Mountain, Necklace, Nightstand, Ocean, Onion, Orange, Paintbrush, Painting, Pan, Pants, Paper, Peach, Peas, Pen, Pencil, Pendant, Piano, Pillow, Pineapple, Plate, Pliers, Potato, Printer, Puzzle, Rabbit, Rake, Refrigerator, Ring, River, Rock, Saw, Saxophone, Scarf, Scissors, Scooter, Screwdriver, Sculpture, Sea, Sharpie, Sheep, Ship, Shirt, Shoes, Shovel, Skirt, Smartphone, Soccer ball, Socks, Spinach, Spoon, Stapler, Strawberry, Table, Television, Tennis ball, Tennis racket, Tiger, Tomato, Toothbrush, Toothpaste, Train, Tree, Trombone, Truck, Trumpet, Violin, Volleyball, Watch, Watering can, Watermelon, Whisk, Wrench.
One of the 157 words is the answer word. You can ask up to 20 yes/no questions to identify the answer word. You will get a Yes, No, or Invalid feedback for each question. Note that you cannot ask questions about the letters of the answer word. Questions can only be asked around semantics.

### A.2 Who is Undercover

Game Rules of Who is the Difference
1. Game Setup
Roles:
Among all players, only one player is the difference one. All others are normal players.
Secret Words:
All normal players receive the same secret word.
The difference receives a related but different word.
Initial Information:
Each player knows only their own word, not their role or others' words.

2. Game Phases
The game starts with three rounds of speaking phase, followed by one round of voting. After that, the game alternates between one round of speaking and one round of voting until it ends. Game ends when the difference is voted out, or the number of remaining players drops to three or fewer.
Speaking Phase
All players speak in turn, each giving one sentence as a clue about their word.
Forbidden actions include saying the word directly, hinting at letters, repeating clues from previous players. You do not have to describe your own word. The difference may pretend to match the regular players' word to avoid detection.
Voting Phase
All players vote for one player (cannot be yourself) at the same time. Regular players vote for who they think is the difference; the difference votes strategically.
The player with the most votes is eliminated. In case of a tie, no one is eliminated that round.
Victory Conditions
If the difference is voted out: regular players win
If the difference survives until only three players remain: difference wins

3. Strategy Tips
1. Identify your role early:
If more than one player's clue doesn't match your word, you are likely the difference. If only one clue differs, you may be a regular player or the difference. Do not jump to conclusions.

## B Learnt Policy

### B.1 Twenty Question

**GPT Test-time Policy from Experience**

To consistently improve performance in Twenty Questions, begin by asking broad, high-level questions that eliminate major categories (e.g., "Is it living?" or "Is it commonly found inside a house?"). Gradually refine with mid-game questions that target functional properties (purpose, interaction, context) rather than overly specific traits. Avoid semantic overload by keeping questions clear, concise, and focused on one characteristic at a time. If progress stalls, pivot to regroup broader ideas rather than fixating on narrow assumptions. Ultimately, efficient grouping and elimination of related possibilities will maximize the chances of identifying the answer within 20 questions.

### Claude Test-time Policy from Experience

**1. Core Question Structure**
- Always start with “Is it a man-made object?”
- Follow with major category splits like:
∗ Electronic vs. non-electronic
∗ Indoor vs. outdoor use
∗ Portable vs. stationary
- Use clear binary choices with “rather than” comparisons to avoid invalid responses

**2. Category Hierarchy (in order)**
A. Man-made vs. Natural
B. If man-made:
- Electronic/Electrical vs. Non-electronic
- Indoor vs. Outdoor use
- Major categories:
∗ Electronics/Devices
∗ Furniture
∗ Clothing/Accessories
∗ Tools/Equipment
∗ Sports/Recreation
∗ Kitchen/Household items

**3. Question Formulation Rules**
- Use clear yes/no format
- Avoid questions about letters or spelling
- Include comparison elements (e.g., “rather than”)
- Focus on physical characteristics and use cases
- Avoid ambiguous or overlapping categories

**4. Strategic Principles**
- Each question should aim to eliminate roughly half of remaining possibilities
- Track eliminated categories carefully
- Move from general to specific progressively
- Don’t get too specific too early
- Save direct object identification for final questions

### DeepSeek Test-time Policy from Experience

1. **Start Broad, Then Narrow**:
Begin with high-level categories (living/non-living, household/outdoor) to eliminate large groups quickly. Prioritize binary splits to maximize information gain early.
2. **Focus on Function and Context**:
For household items, drill down by room (kitchen/bathroom) or usage (cleaning, recreation, grooming). Ask about *how* or *where* the item is used (e.g., "Is it for cutting?" or "Found in the bathroom?").
3. **Avoid Invalid Questions**:
Use clear, single-focus questions (e.g., "Is it jewelry?" not "Is it worn on hands or neck?"). Split compound queries to prevent ambiguity.
4. **Guess Strategically in Small Pools**:
Once candidates are under 5, guess high-probability words (e.g., "spoon" after "kitchen utensil") instead of further elimination.
5. **Adapt and Pivot**:
If a category yields "No," immediately switch to the next most likely group (e.g., from "tools" to "decorative items"). Leverage feedback (e.g., "Invalid") to refine questioning.
**Key Principle**:
Each question should halve the remaining candidates, balancing systematic elimination with timely guessing.

### Human Policy

1. **Start Board**: Begin with high-level distinctions to quickly eliminate large groups of items. Useful splits include:
- Is it a living thing?
- Does it use electoricity?
- Is it commonly found indoors?
2. **Identify the category early**: Aim to determine the category of the answer word within the first 5 questions. Avoid overly specific guesses at this stage.
- Major categories among the 157 words include: fruit, vegetable, animal, transportation, sports equipment, musical instrument, electronic product, office equipment, furniture, kitchenware (for food preparing, eating), tools, entertainment items, clothing, and accessories, daily necessity, etc.
3. **Refine with category traits**: Once the category is identified, filter the candidate words accordingly. Then continue binary questioning using finer distinctions within that group.
4. **Guess when narrowed down**: If only 3 or fewer candidates remain, direct guessing becomes viable. Otherwise, continue using binary splits for efficiency.
5. **Your question should always be clear**: Avoid compound questions that can lead to invalid responses.

## B.2 Who is Undercover

### GPT test-time Policy

1. **Identify Your Role as Early as Possible**: If the descriptions given by more than one player do not match your secret word, then you are the "different one." If only one player’s statements conflict with yours, you might be an ordinary player or the "different one," but you cannot determine this immediately.
2. **Make Your Statements Ambiguous to Avoid Revealing Core Characteristics**: Regardless of whether you are an ordinary player or the "different one," try to use broad and ambiguous descriptions during the game. Avoid directly mentioning distinctive or obvious characteristics of the secret word to reduce the risk of being suspected.
3. **Spot Deviations and Pinpoint Suspicious Targets**: Over multiple rounds of statements, focus on characteristics that significantly deviate from the majority of players. Identify semantic inconsistencies. The "different one" usually struggles to fully match the clues provided by ordinary players, which may reveal their identity. Use these deviations to precisely

target your voting.
4. **Skillfully Imitate and Blend Into Group Clues**: If you are the "different one," observe the speaking style of the ordinary players and provide vague but relevant supplementary clues based on their descriptions. This will help align yourself with the group and make your statements appear "reasonable" and "credible," minimizing suspicion.
5. **Adapt Dynamically and Respond Flexibly**: Adjust your speaking strategy according to changes in the game's dynamics. If your statements are overly suspected, try to further obscure your clues and emphasize common characteristics to divert suspicion.

### Claude Test-time Policy

1. **Identify Identity Early**: If more than one player's description does not match your secret word, then you are likely the different one. If only one player's statement contradicts yours, you could be an ordinary player or the different one, and you cannot make an immediate judgment.
2. **Pay Attention to Consistency and Subtle Differences**: During the game, closely observe the statements of other players and note their subtle differences. If a player's description is significantly inconsistent with the majority, it may indicate they are the different one.
3. **Use Vague Descriptions**: Try to provide vague descriptions and avoid being too specific about your secret word. Choose characteristics that can apply to multiple items.
4. **Strategic Voting**: In critical voting rounds, do not always follow the majority. Sometimes strategically scattering votes can help you achieve your victory conditions, especially if you are the different one.
5. **Adjust Identity Judgments Timely**: Adjust your identity judgment based on other players' statements. If you find that other players' descriptions do not match your secret word, consider that you might be the different one and adjust your speaking strategy accordingly to avoid revealing your identity.
6. **Provide Unique Clues**: When speaking, try to provide unique clues that differ from those of other players. Avoid repeating characteristics that have already been mentioned to prevent exposing your secret word.

### DeepSeek Test-time Policy

1. **Identify your role as early as possible**: If more than one player's description does not match your secret word, then you are the differentiator. If only one player's statement contradicts yours, you could be either a regular player or the differentiator, and cannot determine your role immediately.
2. **Deduce the opposing secret word**: Regular players and differentiators have different secret words, but there is a certain connection between them. Both regular players and differentiators should deduce the opposing secret word based on their own word and previous players' statements as quickly as possible.
3. **Speaking strategy**: When uncertain about your role, provide relevant but broad clues to avoid statements that directly reveal your secret word. Ideally, these clues should align with the main characteristics of both your secret word and the opposing secret word. Once your role is confirmed, the differentiator should immediately begin describing the regular players' secret word while mimicking their speaking style to avoid suspicion. Regular players, on the other hand, should identify and highlight the most divergent player to pinpoint the sole differentiator.

### Human Policy

1. **Analyze comprehensively**: other people's statements may apply not only to their own secret word, but to other words as well, and they need to be analyzed and stated in a way that takes into account the scope of application of the description.
2. **Guessing and targeting the opponent's secret word**: The secret words of normal players and different people are related, but different. It is possible to guess the opponent's secret word for your own secret word, and to narrow down and target the opponent's secret word after someone else has spoken.
3. **Identify yourself as early as possible**: If >1 player's description does not match your own secret word, then you are a different person. If only 1 player's statement currently contradicts your own, you may be either a regular player or a dissenter and cannot be immediately recognized.
4. **Speaking with caution.** In the early stages of the game, when you cannot identify yourself, you need to speak very carefully. Do not reveal the unique clues to your secret word in your speech; choose a description that is broader and applies to a class of words (including your own and your opponent's secret word) to prevent being too specific and revealing your secret word in the first place, as well as avoiding speeches that might cause conflict.
Later in the game, after clarifying their identity, the regular player continues to describe their secret word and the Dissenter describes the regular player's secret word. Statements can be chosen for the popular attributes of the words to be described, avoiding describing overly niche clues that could trigger concern and suspicion.

## C Value Alignment

In Table 4, we present that the importance to use neutral expressions as the role names with negative connotations can create confusion and lead to denial when agent are assessing their identity.

Table 4: Deepseek-r1's thinking content about its gaming role identity. Using role names with negative connotations can create confusion and lead to denial when assessing whether it is "undercover". By using neutral role names, it can quickly and directly recognize its identity.

| Role Name | Thinking Context |
|---|---|
| Undercover | Based on the above analysis, the other four agents' statements are all more on the side of soymilk, and my word is milk, **so maybe I'm the undercover agent?** But according to the rules, there is only one undercover agent, while all the other civilians should be milk, but apparently their descriptions are more consistent with soy milk. **This suggests that I might be the undercover agent**, and it seems contradictory that my secret word is actually soymilk, but the system tells me my word is milk. **Or maybe there are other players who are undercover** and their descriptions are misleading. But according to the game settings, the word for civilians is the same and the word for undercover agents is different but related. If my word is milk, and the other civilians' word should be milk, but they speak like soy milk, **that means they might all be undercover?** That can't be true because there is only one undercover agent. So maybe my judgment is wrong, **or the undercover agent is among the others** and their descriptions are trying to confuse ...... **So as a civilian, I should point out their anomalies.** |
| Difference | Taken together, the descriptions of the other players seem to point to soymilk, whereas my secret word is milk, **so I may be the Different One**. As the Different One, I need to make the statement fit the secret word of the regular player (i.e. soy milk). |