2025-8-26

Multi-Turn Puzzles: Evaluating Interactive Reasoning and Strategic Dialogue in LLMs

Kartikeya Badola[1], **Jonathan Simon**[1], **Arian Hosseini**[1],
Sara Marie Mc Carthy[1], **Tsendsuren Munkhdalai**[1], **Abhimanyu Goyal**[1], **Tomáš Kočiský**[1], **Shyam Upadhyay**[1],
Bahare Fatemi[2] **and Mehran Kazemi**[1]
[1]Google DeepMind, [2]Google Research

arXiv:2508.10142v3 [cs.CL] 24 Aug 2025

Large language models (LLMs) excel at solving problems with clear and complete statements, but often struggle with nuanced environments or interactive tasks which are common in most real-world scenarios. This highlights the critical need for developing LLMs that can effectively engage in logically consistent multi-turn dialogue, seek information and reason with incomplete data. To this end, we introduce a novel benchmark comprising a suite of multi-turn tasks each designed to test specific reasoning, interactive dialogue, and information-seeking abilities. These tasks have deterministic scoring mechanisms, thus eliminating the need for human intervention. Evaluating frontier models on our benchmark reveals significant headroom. Our analysis shows that most errors emerge from poor instruction following, reasoning failures, and poor planning. This benchmark provides valuable insights into the strengths and weaknesses of current LLMs in handling complex, interactive scenarios and offers a robust platform for future research aimed at improving these critical capabilities.

Data available at huggingface.co/datasets/arianhosseini/mt_puzzles

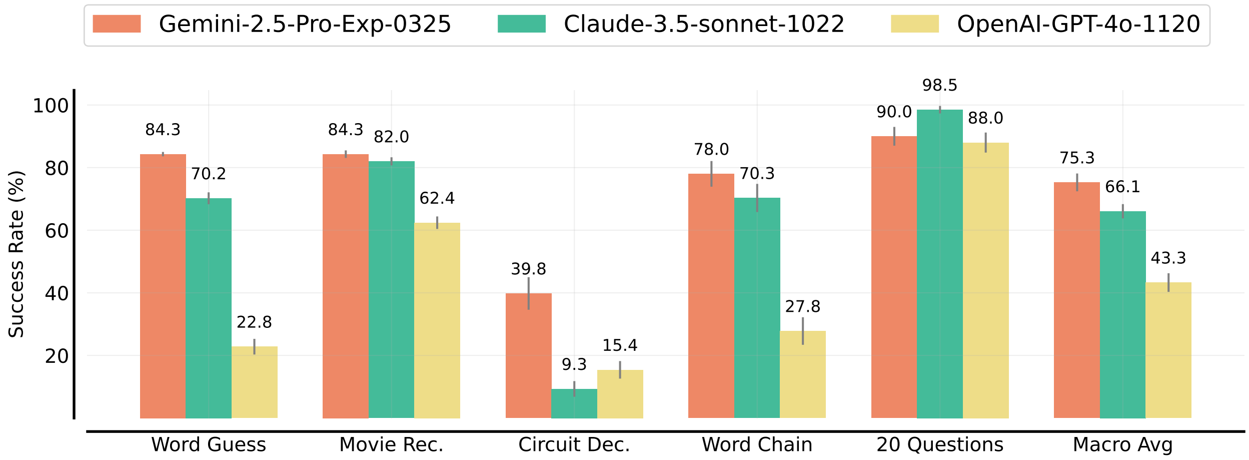


Figure 1 | **Multi-Turn Puzzles (MTP) Benchmark:** performance of Gemini 2.5-Pro-Exp, Claude-3.5-sonnet and OpenAI-GPT-4o across five tasks. While models perform well on Twenty Questions, significant headroom remains for the rest of the tasks.

1. Introduction

In situations where information is incomplete or hard to extract, large language model (LLM) agents must often take the initiative to proactively ask questions in a multi-turn interaction to gain more information, reduce uncertainty, and effectively solve problems. This requires elaborate, interactive, strategic planning and reasoning to decide the next move in a multi-turn conversation as well as a robust memory of previous interactions. This capability is crucial for LLMs to operate effectively in

---

Corresponding author(s): kbadola@google.com and mehrankazemi@google.com.
© 2025 Google DeepMind. All rights reserved

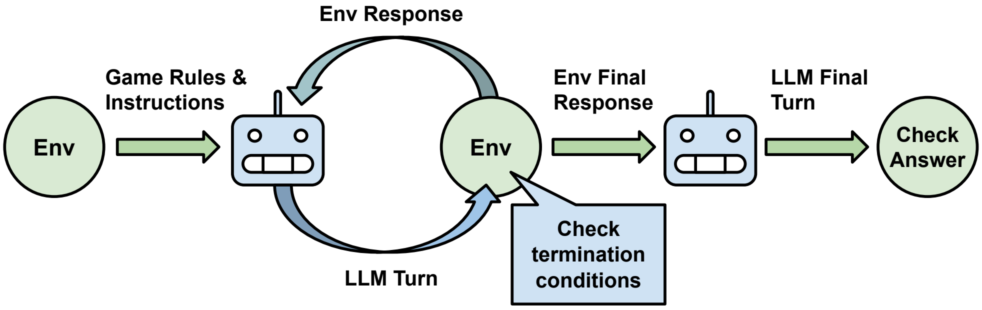


Figure 2 | **Multi-Turn Interaction:** LLM interacts with the environment in the Multi-Turn Puzzles benchmark and the final output is checked using rule-based methods.

complex, real-world environments where information is not always provided in a clear, single query such as virtual assistants and coding assistants (Zheng et al., 2024a). While LLM agents excel at single-turn problems (e.g., mathematical and coding problems (AI@Meta, 2024; DeepSeek-AI, 2025; Google, 2024; OpenAI, 2023) involving a single prompt), they often perform poorly when a user's instructions are unclear or insufficient and further information must be obtained from the user before providing a useful response. This is a fundamental problem requiring further study and development of LLMs for how to reason with incomplete information, stay consistent and logical across turns, and demonstrate information seeking behaviour to plan and reason efficiently with finite interactions.

To assess multi-turn reasoning and instruction-following capabilities of LLMs, we propose multi-turn puzzles, a novel benchmark consisting of five distinct tasks, each with unique challenges and verifiable metrics (Table 1):

- **Word Guess** tests the efficiency of deductive reasoning and the ability of models to engage in homogeneous turn taking behavior.
- **Movie Recommendation** assesses the model's capacity for information seeking in imperfect information scenarios and learning user preferences through interactive questioning.
- **Circuit Decoding** tests logical deduction and the ability of the models to learn about the inner-workings of a new tool through multiple interactions of feeding an input and observing the output.
- **Word Chain** probes the model's ability to retain and deterministically condition on information obtained through multiple rounds of symmetric interaction.
- **Twenty Questions** measures coherence and logical consistency in dialogue.

Together these tasks provide a comprehensive evaluation of multi-turn reasoning and interactive problem-solving of LLMs, offering insights into how autonomous agents can be trained to behave more intelligently in ambiguous situations. We benchmark frontier models, including OpenAI GPT-4o, Anthropic Claude 3.5 Sonnet and Gemini 2.5 Pro Exp on our multi-turn tasks. Figure 1 indicates that while significant progress has been made, there remains a need for further exploration into the intricacies of multi-turn reasoning and information seeking behaviours. We believe this represents a valuable opportunity for the community to collaboratively advance the state of the art.

2. Multi-Turn Puzzles (MTP) Benchmark

Navigating reasoning in domains where all the information is not readily present through multi-turn dialogue is a core capability of human intelligence, yet current benchmarks fail to adequately evaluate this ability. Most prevalent single-turn evaluations often fail to capture the complexities of real-world interactions, where problem-solving frequently involves iterative questioning, hypothesis refinement,

Name	Short Description	Metric
Word Guess	Guess the secret word in min attempts. Environment gives feedback on how close the guess is at each turn.	Normalized number of attempts to solve the task.
Movie Rec.	Probe the user to decode the user preference function for N turns. Pick a movie for the same user at N+1 turn.	Normalized rank of the final movie.
Circuit Dec.	Probe the C different boolean circuits for N turns. Predict the joint truth table of all the circuits at N+1 turn.	Normalized circuit-wise accuracy.
Word Chaining	Model and user take turns choosing allow-listed words that start with the last letter of the previously chosen word. The episode ends successfully if no words with the correct starting letter remain, or max_turns is hit.	Normalized % of trajectories with successful end state.
Twenty Questions	Model chooses a secret word. The user asks questions to determine what the word is. The model must answer these questions in a way that doesn't contradict previously-provided answers.	Normalized % of logically consistent trajectories.

Table 1 | Multi-turn Puzzle Task Descriptions and Metrics

and dynamic adaptation to evolving information. We aim to create a benchmark that specifically targets this gap. MTP consists of five diverse tasks, including Word Guess, Movie Recommendation, Circuit Decoding, Word Chaining and Twenty Question. These tasks are designed to assess a model's ability to (1) reason with partial information, (2) be logically consistent across turns, and (3) exhibit information-seeking behavior.

2.1. Data Creation and Metrics

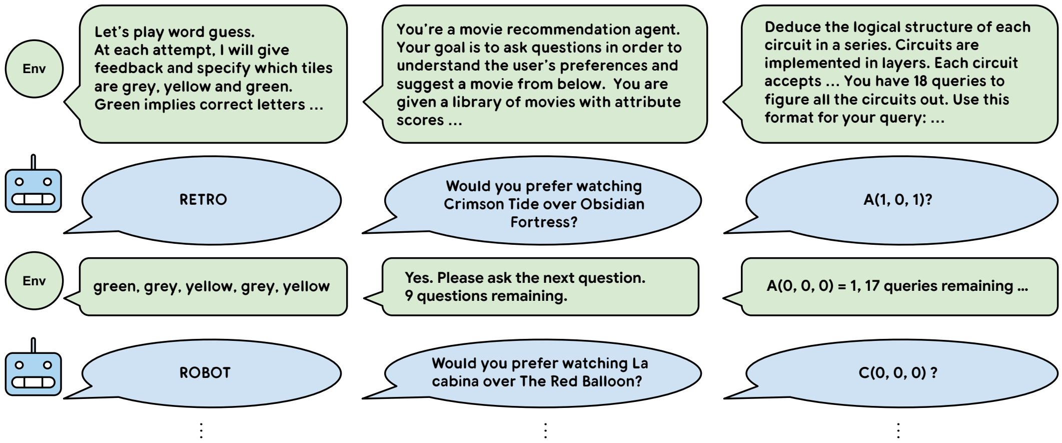
All of our tasks are synthetically created (see examples in Figure 3). They utilize rule-based environments and deterministic scorers, eliminating the need for human-in-the-loop or autorators, and ensuring tight confidence intervals. Figure 2 shows an overview of multi-turn interactions in our benchmark. This section presents a brief description of our task construction.

Word Guess The objective is to guess the secret word in the fewest attempts. We start with 10 unique vocabularies of 40 words. Each word has 5 letters. Since each word can serve as the secret word in a data point there are a total of 400 configurations. Section A.4 outlines the full instruction for this task.

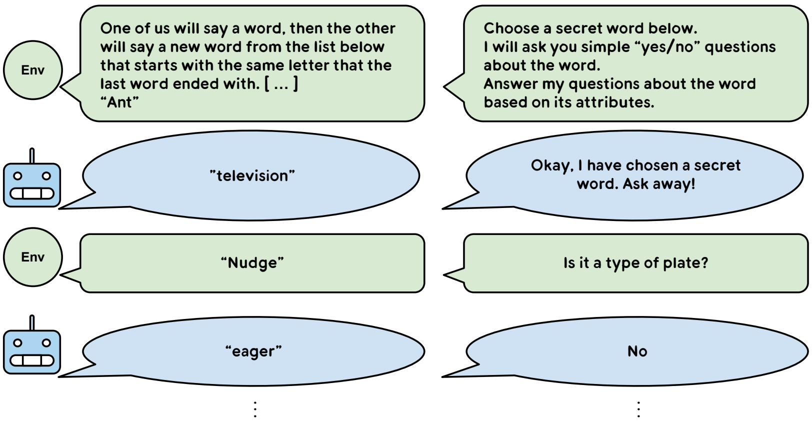
Movie Recommendation The goal is to discover the user's preference function over multiple turns and pick a movie for that user at the end. Each data point has a simulated user with a unique preference function, a seen set (for the questioning phase) and an unseen set (for the recommendation phase) of movies. There are 20 unique users, and 50 movie set variations, yielding a total of 1000 configurations for the task. Section A.1 describes how the users and the set of movies are initialized. Section A.4 outlines the instruction details for this task.

Circuit Decoding In this task, the objective is to determine the structure of a set of C unknown boolean circuits. Each circuit receives k boolean inputs and produces a single bit output. These C

circuits in total are made up of a fixed number of AND, OR and NOT gates. The model must deduce the functionality of all C circuits, ultimately producing their combined joint truth table. For the evaluation set, we keep $C \times 2^k = 24$ and 300 unique circuit configurations are constructed. Further details are discussed in section A.2.


Env
Let's play word guess. At each attempt, I will give feedback and specify which tiles are grey, yellow and green. Green implies correct letters ...
RETRO
Env
green, grey, yellow, grey, yellow
ROBOT
You're a movie recommendation agent. Your goal is to ask questions in order to understand the user's preferences and suggest a movie from below. You are given a library of movies with attribute scores ...
Would you prefer watching Crimson Tide over Obsidian Fortress?
Yes. Please ask the next question. 9 questions remaining.
Would you prefer watching La cabina over The Red Balloon?
Deduce the logical structure of each circuit in a series. Circuits are implemented in layers. Each circuit accepts ... You have 18 queries to figure all the circuits out. Use this format for your query: ...
A(1, 0, 1)?
A(0, 0, 0) = 1, 17 queries remaining ...
C(0, 0, 0) ?

(a) Word Guess (b) Movie Rec. (c) Circuit Dec.


Env
One of us will say a word, then the other will say a new word from the list below that starts with the same letter that the last word ended with. [...] "Ant"
"television"
Env
"Nudge"
"eager"
Choose a secret word below. I will ask you simple "yes/no" questions about the word. Answer my questions about the word based on its attributes.
Okay, I have chosen a secret word. Ask away!
Is it a type of plate?
No

(d) Word Chaining (e) Twenty Questions

Figure 3 | **Examples of Multi-turn Puzzle Tasks.** (a) **Word Guess**: The model attempts to guess a secret word, receiving feedback on letter correctness and position after each guess. (b) **Movie Recommendation**: The model interactively asks questions to understand user preferences before suggesting a movie. (c) **Circuit Decoding**: The model queries boolean circuits with different inputs to deduce their logical structure. (d) **Word Chaining:** The model and environment take turns saying words from a list, where each new word must start with the last letter of the previous word. (e) **Twenty Questions:** The model selects a secret word and must answer the user's yes/no questions in a logically consistent manner.

Word Chaining In each game, a lexicon is constructed by randomly sampling 500 words from SCOWL word list (Atkinson). The starting player is chosen randomly, and players alternate selecting words from the lexicon, adhering to: (i) the word must start with the previous word's final letter, and (ii) the word must not have been previously used. Rule violation leads to immediate loss. The game ends without loss if no valid moves remain or a turn limit is reached. The performance of an LLM is measured by its success rate (fraction of games completed without loss).

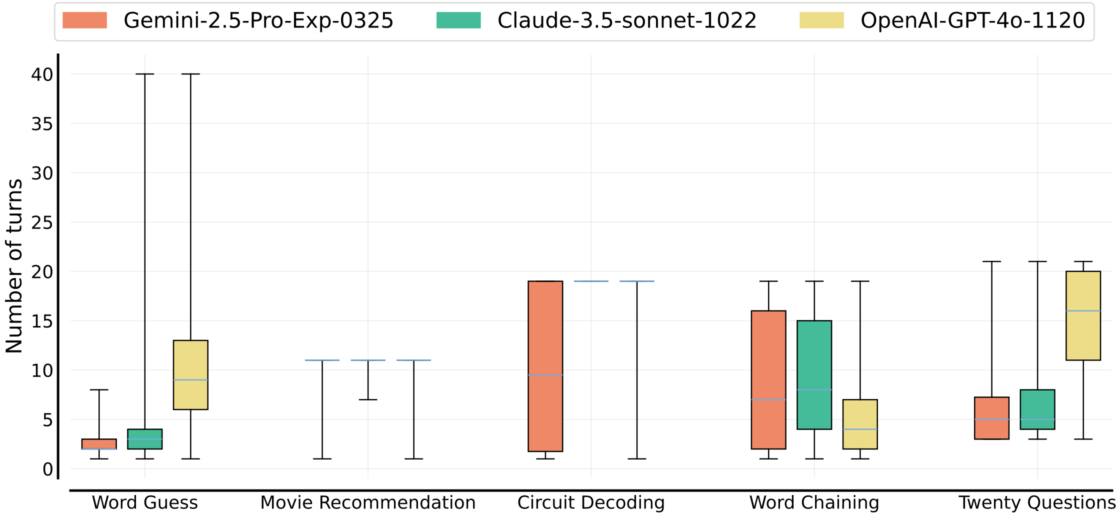


Figure 4 | **Distribution of turns taken by models.** The same game type can have a different number of turns depending upon whether early termination conditions determining a win/lose were reached. Task constraints (such as fixed interaction lengths) and model specific strategies together shape number of conversation turns.

Twenty Questions assesses an LLM's ability to maintain logical consistency during dialogue. The LLM selects a secret noun from a predefined WordNet lexicon (Fellbaum, 1998) (80-100 words with provided hypernyms). It must then answer up to 20 binary (yes/no) questions posed by the user regarding the secret word or its hypernyms (including transitive ones). A loss occurs if and only if the LLM provides an answer that is logically inconsistent with its previous responses within the same game instance.

2.2. Number of Turns and Episodes

Table 2 shows the maximum number of turns allowed per episode for each task, alongside the average number of turns executed by Gemini 2.5 Pro Exp, Claude 3.5 Sonnet and GPT-4o during evaluation. Puzzles' maximum turns range from 11 (10 interaction turns plus 1 final prediction turn) for Movie Recommendation to 40 for Word Guess. Puzzles have early-stopping conditions which makes it possible for the game to terminate before the maximum number of turns has been reached. This can occur in cases such as the model making an irrecoverable error (e.g. logical contradiction in Twenty Questions), or when no valid playable moves remain (e.g. running out of playable words in Word Chaining).

3. Experiments

We report the performance of Gemini-2.5-Pro-Exp-0325, Claude-3.5-sonnet-1022 and OpenAI-GPT-4o-1120 on our benchmark. For each experiment, we use the default sampling parameters provided by each API.

Figure 1 illustrates the performance of these models across five different tasks. The results indicate that while models perform well on the *Twenty Questions* task, there is significant room for improvement in the other tasks. To this end we probe model behavior through the lens of game duration, cost-dependent performance, inter-model turn taking, and information seeking strategies.

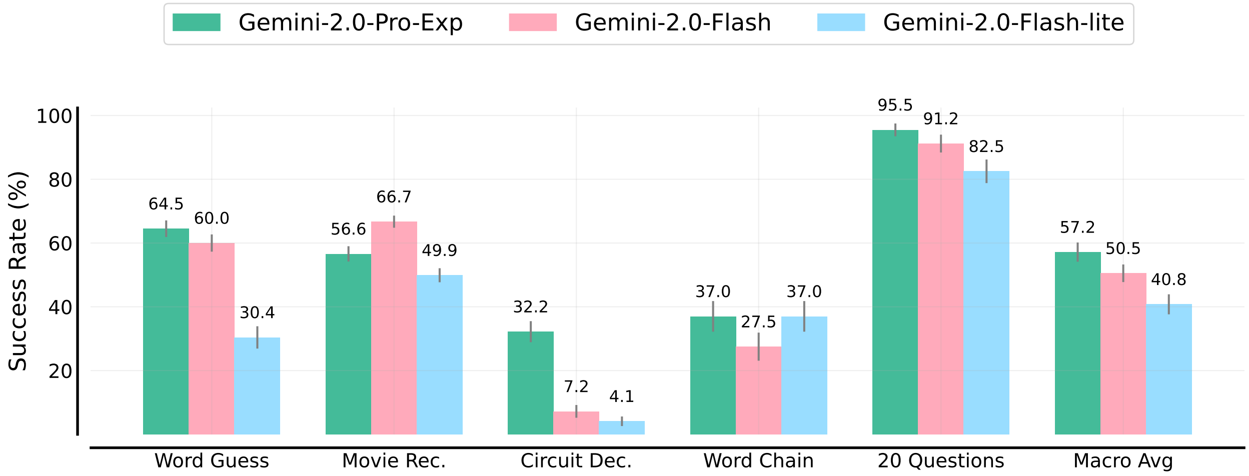


Figure 5 | Performance comparison of Gemini 2.0 Pro Exp, Gemini 2.0 Flash, and Gemini 2.0 Flash-lite models across the five multi-turn puzzle tasks. The results show a clear cost-performance trade-off, with the larger Pro model achieving higher success rates than the more cost-efficient Flash and Flash-lite variants across all tasks and the macro average.

3.1. How Many Turns Do Models Take?

Tasks such as Movie Recommendation, Circuit Decoding, and Twenty Questions have defined interaction limits followed by a final action, reflected in their maximum turn counts (e.g., 10+1, 18+1, 20+1 respectively). Notably, the average turns utilized by Gemini 2.5 indicate that some tasks (Table 2), like Circuit Decoding (avg. 18.85 turns), consistently require nearly the full interaction budget, while others like Word Guess (avg. 5.37 turns) are often completed more quickly.

Figure 4 shows the distribution of number of turns taken by each model across our tasks. In Movie Recommendation, the distribution for all models are tightly clustered around the maximum allowed 11 turns (10 interactions + 1 final), reflecting the task's design which necessitates utilizing the full interaction budget to gather user preferences before the final recommendation. Conversely, Word Guess shows significantly lower median turns across all models, particularly Gemini 2.5 and Claude 3.5, indicating that models often find the secret word efficiently, well before the generous 40-turn limit. Tasks like Circuit Decoding, Word Chaining, and Twenty Questions show greater variability. Circuit Decoding shows Claude 3.5 and GPT-4o consistently using the maximum 19 turns, while Gemini 2.5 displays a wider distribution, which in turn, is noticeable in their success rate on this task. Word Chaining shows considerable spread, reflecting difference in strategic depth. Different distributions in Twenty Questions show the duration for which logical consistency is maintained. Overall, the turn distributions effectively illustrate how task constraints (such as fixed interaction lengths) and model-specific strategies interact to determine conversation lengths.

3.2. Cost-Performance Trade-off

Recent work suggests that cost-effective models reason systematically differently compared to their larger counterparts (Hosseini et al., 2024). Building on this, we extend our analysis to evaluate how these more affordable models perform specifically on our suite of multi-turn interactive tasks. We evaluated different variants within the Gemini 2.0 family – Pro-Exp, Flash, and Flash-lite – which represent distinct tiers in terms of computational cost, with Pro being the most expensive and Flash-lite the cheapest.

Task	Num Problems	Max Num Turns	Gemini 2.5	Claude 3.5	GPT-4o
Word Guess	400	40	2.4	3.8	12.9
Movie Rec.	1000	11 (10 + 1)	11	11	10.9
Circuit Dec.	300	19 (18 + 1)	10.5	19	18.4
Word Chaining	400	20	9	9.3	5.1
Twenty Questions	400	21 (20 + 1)	6.3	6.2	15.1

Table 2 | **Multi-Turn Puzzles statistics and average number of turns played by frontier models.** The games include early-stopping conditions making it possible for the game to end before the maximum number of turns has been reached. This can occur in cases such as the model committing a logical contradiction in Twenty Questions, or running out of playable words in Word Chaining.

The results in Figure 5 reveal a clear trade-off between cost and performance. Gemini 2.0 Pro consistently achieved the highest success rates across all tasks, followed by Gemini Flash, and then Gemini Flash-lite. This trend is evident in the macro average scores. The performance gap varied depending on the task; for instance, the difference was particularly stark in Circuit Decoding (Pro: 32.2%, Flash: 7.2%, Lite: 4.1%), while relatively smaller, though still present, in tasks like Twenty Questions (Pro: 95.5%, Flash: 91.2%, Lite: 82.5%).

3.3. Can Weaker and Stronger Models Finish Each Other's Thoughts?

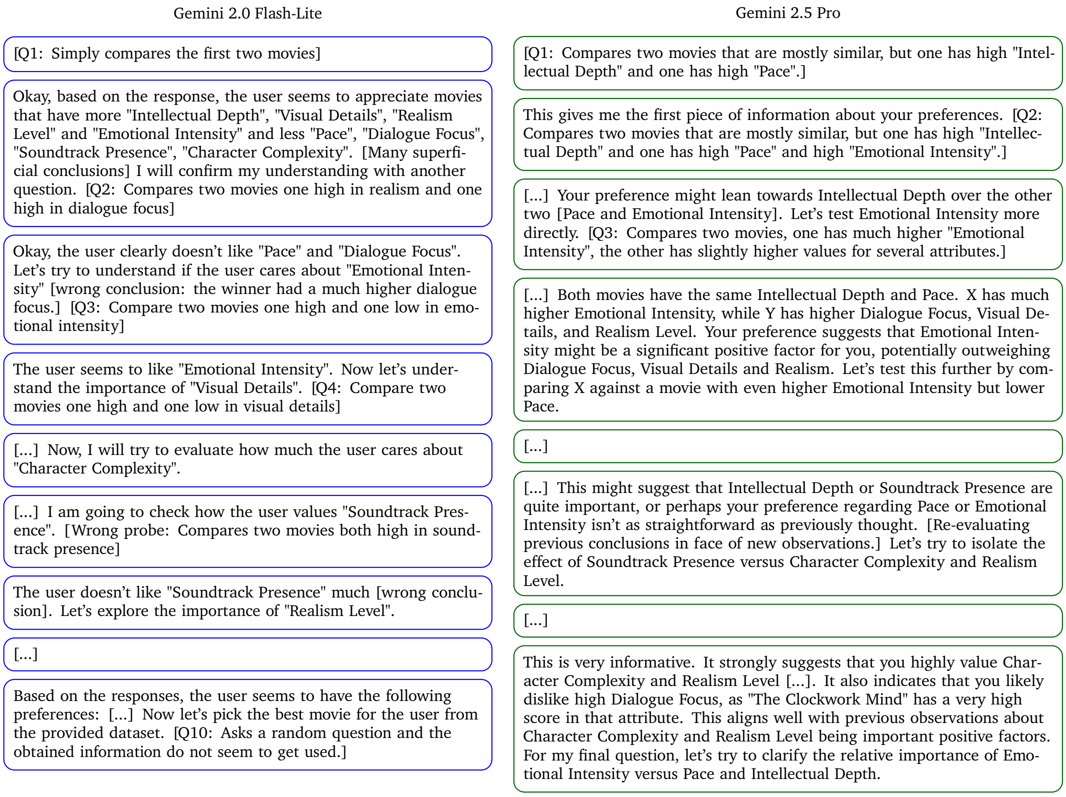
Often a model might be prompted with an inaccurate or incomplete context of a dialogue or reasoning. It is important for the model to be able to detect flaws in the provided history, extract useful information, and pick up on the strategy which was used. In this regard, we investigated whether a stronger model (Gemini 2.5 Pro) could effectively conclude a conversation initiated by a weaker one (Gemini 2.0 Flash), and vice versa. This experiment involves one model handling the initial interactions ($n - 1$), with the other model taking over for the final turn. We conduct this experiment on the Movie Recommendation task, which exhibits strong asymmetry between intermediate turns (preference probing) and the final turn (recommendation).

When the weaker 2.0 Flash model finished the 2.5 Pro model's interactions, the score reached 75.2%. Surprisingly, when the stronger 2.5 Pro model completed the conversation started by the 2.0 Flash model, the success rate was similar at 75.7%. Notably, substituting the weaker model for the final turn resulted in an 11% performance drop (relative to the stronger model's baseline), whereas, allowing the stronger model to finish yielded a 13% gain compared to the weaker' baseline. This underscores the impact of the final reasoning step in asymmetric tasks, indicating that a capable model executing the concluding turn can substantially mitigate deficiencies from earlier, less effective interaction turns.

3.4. Qualitative Analysis

We conducted a qualitative analysis comparing a more capable model (Gemini 2.5 Pro) and a less capable model (Gemini 2.0 Flash-Lite) on our Movie Recommendation task. Our goal was to understand the differences in their reasoning and planning approaches and identify the main reasons for performance gaps between weaker and stronger models. Our analysis revealed the following key distinctions, with a sample dialogue from both models illustrated in Figure 6.

Reasoning Strategy: The less capable model employs a simpler reasoning strategy, asking questions that focus on only one movie attribute at a time and drawing definitive conclusions about those attributes based on those individual responses. These questions often appear unrelated and do not



Gemini 2.0 Flash-Lite

[Q1: Simply compares the first two movies]

Okay, based on the response, the user seems to appreciate movies that have more "Intellectual Depth", "Visual Details", "Realism Level" and "Emotional Intensity" and less "Pace", "Dialogue Focus", "Soundtrack Presence", "Character Complexity". [Many superficial conclusions] I will confirm my understanding with another question. [Q2: Compares two movies one high in realism and one high in dialogue focus]

Okay, the user clearly doesn't like "Pace" and "Dialogue Focus". Let's try to understand if the user cares about "Emotional Intensity" [wrong conclusion: the winner had a much higher dialogue focus.] [Q3: Compare two movies one high and one low in emotional intensity]

The user seems to like "Emotional Intensity". Now let's understand the importance of "Visual Details". [Q4: Compare two movies one high and one low in visual details]

[...] Now, I will try to evaluate how much the user cares about "Character Complexity".

[...] I am going to check how the user values "Soundtrack Presence". [Wrong probe: Compares two movies both high in soundtrack presence]

The user doesn't like "Soundtrack Presence" much [wrong conclusion]. Let's explore the importance of "Realism Level".

[...]

Based on the responses, the user seems to have the following preferences: [...] Now let's pick the best movie for the user from the provided dataset. [Q10: Asks a random question and the obtained information do not seem to get used.]

Gemini 2.5 Pro

[Q1: Compares two movies that are mostly similar, but one has high "Intellectual Depth" and one has high "Pace".]

This gives me the first piece of information about your preferences. [Q2: Compares two movies that are mostly similar, but one has high "Intellectual Depth" and one has high "Pace" and high "Emotional Intensity".]

[...] Your preference might lean towards Intellectual Depth over the other two [Pace and Emotional Intensity]. Let's test Emotional Intensity more directly. [Q3: Compares two movies, one has much higher "Emotional Intensity", the other has slightly higher values for several attributes.]

[...] Both movies have the same Intellectual Depth and Pace. X has much higher Emotional Intensity, while Y has higher Dialogue Focus, Visual Details, and Realism Level. Your preference suggests that Emotional Intensity might be a significant positive factor for you, potentially outweighing Dialogue Focus, Visual Details and Realism. Let's test this further by comparing X against a movie with even higher Emotional Intensity but lower Pace.

[...]

[...] This might suggest that Intellectual Depth or Soundtrack Presence are quite important, or perhaps your preference regarding Pace or Emotional Intensity isn't as straightforward as previously thought. [Re-evaluating previous conclusions in face of new observations.] Let's try to isolate the effect of Soundtrack Presence versus Character Complexity and Realism Level.

[...]

This is very informative. It strongly suggests that you highly value Character Complexity and Realism Level [...]. It also indicates that you likely dislike high Dialogue Focus, as "The Clockwork Mind" has a very high score in that attribute. This aligns well with previous observations about Character Complexity and Realism Level being important positive factors. For my final question, let's try to clarify the relative importance of Emotional Intensity versus Pace and Intellectual Depth.

Figure 6 | Sample (substantially simplified) model responses for Gemini 2.0 Flash-Lite (left) and Gemini 2.5 Pro (right) on Movie Recommendation showing their reasoning and planning process (for brevity, only model responses are shown). The text provided inside [] are extra information and context provided by the authors.

contribute to a cohesive plan where subsequent questions build upon the information gained from previous ones. In contrast, the more capable model asks questions that consider multiple features simultaneously, allowing for richer insights from the user's responses. These questions are interconnected and form a more coherent plan, where the model designs its next questions based on observations from earlier interactions. This includes re-evaluating previous conclusions when new information contradicts them and investigating further. The samples provided in Figure 6 illustrate these differences in strategies.

Asking Effective Questions: When the models decide to seek specific information, we observed that the less capable model sometimes asks ineffective questions to obtain that information. For example, as shown in Figure 6, the model might aim to understand the user's preference for "Soundtrack Presence" but then ask a question comparing two movies that both have a high level of this feature. Conversely, the more capable model generally compares movies that are similar in most other aspects and primarily differ in the specific feature(s) it wants to explore.

Drawing Accurate Conclusions: We found that some of the conclusions drawn by the less capable model are inaccurate or lack depth. For instance, in Figure 6, we see that after the initial question, the model hastily makes numerous superficial assumptions about the user's likes and dislikes. Similarly, when comparing two movies that differ across multiple features, the model might draw conclusions about a single feature without considering the influence of the other differing features. However, the more capable model tends to derive more meaningful conclusions from the answers it receives and considers the interplay of various factors that could explain the user's feedback.

Strategic Use of Questions: Ideally, every question a model asks should contribute to gathering relevant information that aids in making a final recommendation. However, in the case of the less capable model, we observed instances where questions were not particularly useful. For example, the first question in Figure 6 simply compares the first two movies in the provided table, and the resulting information doesn't significantly contribute to the subsequent decision-making process. Additionally, the final question appears to be primarily asked to fulfill the task's requirements, as the model had already made a decision after nine questions. While definitively assessing the optimality of the more capable model's questions is challenging, we observed that it consistently probes for more relevant information that helps advance the task.

4. Related Work

Challenges in Multi-Turn Interaction. The capabilities of LLMs have spurred significant research interest, initially focusing heavily on single-turn tasks. Recent studies have highlighted the challenges LLMs face when transitioning from single-turn success to effective multi-turn interaction. Kwan et al. (2024) attribute this drop in conversational tasks (like recollection and expansion) to factors like the distance to relevant context and error propagation. Liang et al. (2024) echo these findings specifically within the domain of mathematical problem-solving, noting a performance decline in interactive settings requiring sustained reasoning compared to single-turn math problems. He et al. (2024) introduced the Multi-IF benchmark to assess multi-turn and multilingual instruction adherence, finding performance drops with each turn and particular difficulties with non-Latin scripts. Such studies underscore the need for better evaluation methodologies, hence we designed our tasks to only be solvable through logically and interactive dialogue.

Evaluating Multi-Turn Dialogue and Reasoning. Various benchmarks aim to evaluate multi-turn capabilities. Duan et al. (2024) evaluate the ability of LLMs to generate human-style multi-turn chat dialogues, using other powerful LLMs such as GPT-4 (OpenAI, 2023) as judges. Similarly, MT-Bench and Chatbot Arena both evaluate LLMs' multi-turn interaction quality (Zheng et al., 2024b). MT-Bench uses a multi-turn question set and LLM-based ratings, while Chatbot Arena relies on pairwise comparisons from real user interactions. Zhang et al. (2024) utilize an entity-deducing game where the model asks questions to probe conversational reasoning and planning. Our work complements this with tasks like Twenty Questions, where the model *answers* questions, testing its logical consistency. In contrast to these approaches focusing on general quality or instruction following, we evaluate LLMs specifically for multi-turn reasoning and information seeking abilities within puzzle scenarios using rule-based verifiable metrics.

Agentic Behavior and Interactive Environments. Platforms such as LMRL Gym (Abdulhai et al., 2023) and AgentBench (Liu et al., 2023) evaluate LLMs as agents in interactive settings. LMRL-Gym focuses on reinforcement learning (RL) capabilities (e.g. planning, credit assignment) in games like "guess my city", while AgentBench assesses instruction following, reasoning, and knowledge acquisition in simulated real-world environments like databases and web browsing (Deng et al., 2023; Yao et al., 2022). In contrast, our benchmark provides simpler controllable game environments, allowing for more targeted evaluation of multi-turn capabilities. Additionally, many of the aforementioned games require an LLM to simulate the game environments and user interactions while we provide rule based environments. Zheng et al. (2025) explores multi-turn prompting and fine-tuning for code generation, sharing our focus on multi-turn interaction. However, our puzzle environment uniquely requires logical deduction, strategic planning, and effective communication. Other relevant work includes text-based game engines, such as TextWorld (Côté et al., 2018) and games developed upon them, such as Coin Collector (Xu et al., 2020) and Jericho-QA (Ammanabrolu et al., 2020), primarily used for evaluating RL algorithms and agentic behavior. Nie et al. (2024); Tajwar et al. (2025) explore in-context RL, focusing heavily on exploration in partially observable environments. Hendrycks et al. (2021) evaluate other LLM capabilities such as scientific reasoning and morality specifically in multi turn settings. For a comprehensive survey and further details regarding the multi-turn interaction capabilities of LLMs refer to Zhang et al. (2025). Unlike benchmarks relying on LLM judges or focusing broadly on agent capabilities in complex simulations, MTP uses simpler, controllable game environments with deterministic, rule-based scoring. This allows for targeted evaluation of core interactive reasoning skills, distinct from general conversational quality or instruction following alone.

5. Conclusion

While LLMs demonstrate strong performance on single-turn tasks, real-world applications often necessitate sustained interaction, multi-turn reasoning and dialogue abilities. Our work introduces the Multi-Turn Puzzles (MTP) benchmark, contributing to this area by using interactive puzzle-solving as a specific testbed for these complex reasoning and dialogue planning skills. Evaluating current frontier models on MTP revealed significant performance disparities across tasks, highlighting substantial room for improvement, particularly in efficient information seeking (Movie Recommendation, Word Guess), complex deduction (Circuit Decoding), and strategic interaction (Word Chaining), although logical consistency (Twenty Questions) proved stronger. Qualitative analysis further pinpointed key failure modes related to reasoning strategies, question formulation, conclusion accuracy, and planning. The MTP benchmark, with its rule-based environments and deterministic metrics, offers a robust platform for diagnosing these weaknesses. Future work should leverage such insights to further study how models can be fine-tuned for more effective multi-turn reasoning and coherent, strategic dialogue, ultimately enhancing their ability to navigate the complexities inherent in real-world interactions.

References

M. Abdulhai, I. White, C. Snell, C. Sun, J. Hong, Y. Zhai, K. Xu, and S. Levine. Lmrl gym: Benchmarks for multi-turn reinforcement learning with language models, 2023. URL `https://arxiv.org/abs/2311.18232`.

AI@Meta. Llama 3 model card. 2024. URL `https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md`.

P. Ammanabrolu, E. Tien, M. J. Hausknecht, and M. O. Riedl. How to avoid being eaten by a grue: Structured exploration strategies for textual worlds. *CoRR*, abs/2006.07409, 2020. URL `https://arxiv.org/abs/2006.07409`.

K. Atkinson. Spell checking oriented word lists (scowl). `http://wordlist.aspell.net/`. Accessed: 2017-12-20.

M.-A. Côté, A. Kádár, X. Yuan, B. Kybartas, T. Barnes, E. Fine, J. Moore, R. Y. Tao, M. Hausknecht, L. E. Asri, M. Adada, W. Tay, and A. Trischler. Textworld: A learning environment for text-based games. *CoRR*, abs/1806.11532, 2018.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

X. Deng, Y. Gu, B. Zheng, S. Chen, S. Stevens, B. Wang, H. Sun, and Y. Su. Mind2web: Towards a generalist agent for the web, 2023. URL `https://arxiv.org/abs/2306.06070`.

H. Duan, J. Wei, C. Wang, H. Liu, Y. Fang, S. Zhang, D. Lin, and K. Chen. BotChat: Evaluating LLMs' capabilities of having multi-turn dialogues. In K. Duh, H. Gomez, and S. Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3184–3200, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.201. URL `https://aclanthology.org/2024.findings-naacl.201/`.

C. Fellbaum. Wordnet: An electronic lexical database. *MIT press Cambridge*, 1998.

G. T. Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv e-prints*, pages arXiv–2403, 2024.

Y. He, D. Jin, C. Wang, C. Bi, K. Mandyam, H. Zhang, C. Zhu, N. Li, T. Xu, H. Lv, S. Bhosale, C. Zhu, K. A. Sankararaman, E. Helenowski, M. Kambadur, A. Tayade, H. Ma, H. Fang, and S. Wang. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following, 2024. URL `https://arxiv.org/abs/2410.15553`.

D. Hendrycks, M. Mazeika, A. Zou, S. Patel, C. Zhu, J. Navarro, D. Song, B. Li, and J. Steinhardt. What would jiminy cricket do? towards agents that behave morally. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=G1muTb5zuO7`.

A. Hosseini, A. Sordoni, D. Toyama, A. Courville, and R. Agarwal. Not all llm reasoners are created equal, 2024. URL `https://arxiv.org/abs/2410.01748`.

W.-C. Kwan, X. Zeng, Y. Jiang, Y. Wang, L. Li, L. Shang, X. Jiang, Q. Liu, and K.-F. Wong. MT-eval: A multi-turn capabilities evaluation benchmark for large language models. In Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 20153–20177, Miami, Florida, USA, Nov. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1124. URL `https://aclanthology.org/2024.emnlp-main.1124/`.

Z. Liang, D. Yu, W. Yu, W. Yao, Z. Zhang, X. Zhang, and D. Yu. Mathchat: Benchmarking mathematical reasoning and instruction following in multi-turn interactions, 2024. URL `https://arxiv.org/abs/2405.19444`.

X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang, S. Zhang, X. Deng, A. Zeng, Z. Du, C. Zhang, S. Shen, T. Zhang, Y. Su, H. Sun, M. Huang, Y. Dong, and J. Tang. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv: 2308.03688*, 2023.

A. Nie, Y. Su, B. Chang, J. N. Lee, E. H. Chi, Q. V. Le, and M. Chen. Evolve: Evaluating and optimizing llms for exploration, 2024. URL `https://arxiv.org/abs/2410.06238`.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

F. Tajwar, Y. Jiang, A. Thankaraj, S. S. Rahman, J. Z. Kolter, J. Schneider, and R. Salakhutdinov. Training a generally curious agent, 2025. URL `https://arxiv.org/abs/2502.17543`.

Y. Xu, M. Fang, L. Chen, Y. Du, J. T. Zhou, and C. Zhang. Deep reinforcement learning with stacked hierarchical attention for text-based games, 2020. URL `https://arxiv.org/abs/2010.11655`.

S. Yao, H. Chen, J. Yang, and K. Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20744–20757. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/82ad13ec01f9fe44c01cb91814fd7b8c-Paper-Conference.pdf`.

C. Zhang, X. Dai, Y. Wu, Q. Yang, Y. Wang, R. Tang, and Y. Liu. A survey on multi-turn interaction capabilities of large language models, 2025. URL `https://arxiv.org/abs/2501.09959`.

Y. Zhang, J. Lu, and N. Jaitly. Probing the multi-turn planning capabilities of LLMs via 20 question games. In L.-W. Ku, A. Martins, and V. Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1495–1516, Bangkok, Thailand, Aug. 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.82. URL `https://aclanthology.org/2024.acl-long.82/`.

K. Zheng, J. Decugis, J. Gehring, T. Cohen, benjamin negrevergne, and G. Synnaeve. What makes large language models reason in (multi-turn) code generation? In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=Zk9guOl9NS`.

L. Zheng, W.-L. Chiang, Y. Sheng, T. Li, S. Zhuang, Z. Wu, Y. Zhuang, Z. Li, Z. Lin, E. Xing, J. E. Gonzalez, I. Stoica, and H. Zhang. LMSYS-chat-1m: A large-scale real-world LLM conversation dataset. In *The Twelfth International Conference on Learning Representations*, 2024a. URL `https://openreview.net/forum?id=BOfDKxfwt0`.

L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024b.

A. Task Creation Details

A.1. Movie Recommendation

1. **Users:**
   - **User preference function**: We create a population of users, each having their own unique preference function modelled as a linear function of movie attributes:

     $$\text{score} = w_1 \cdot \text{attribute}_1 + w_2 \cdot \text{attribute}_2 + \cdots + w_k \cdot \text{attribute}_k$$

     where w_i are weights and attribute_i are the movie attribute scores. Weights (w_i) are randomly generated for each user where $w_i \in [0, 1]$ and rounded to one decimal place
2. **Movies:**
   - **Budget Constraint**: To avoid outlier movies which would have high scores for all attributes, we assign a budget to each movie, limiting the total attribute scores:

     $$\text{movie\_budget} = \sum_{i=1}^{k} \text{attribute\_score}_i$$

     where k is the number of attributes. The movie_budget is randomly sampled between $(\text{base\_budget}, \text{base\_budget} + \text{budget\_delta})$.
   - **Minimum Credits:** Each attribute is assigned a minimum score (attribute_min_credit).
   - **Random Distribution:** The remaining budget (after assigning minimum credits) is distributed randomly across attributes, ensuring diversity in movie profiles.
   - **Seen vs. Unseen Sets:** Each episode has two sets of movies:
     - **Seen Set:** Used for the agent's questioning phase, n=20
       - Lower precision for attribute scores (fewer decimal places).
       - A pre-decided fraction of attributes may have their scores set to a min_value. This makes it easier for the agent to compare movies that only differ in a few attributes.
     - **Unseen Set:** Used for final recommendation, n=40
       - Higher precision for attribute scores.
       - Doesn't have any sparsity like the seen set

A.2. Circuit Decoding

- **Circuit Structure**: Each of the C circuits has k binary inputs and a single binary output. Each circuit can have a depth greater than 1, meaning gates can be connected in multiple layers.
- **Constraint on logical operators**: All C circuits, in aggregate, are composed of a fixed number of x AND gates, y OR gates, and z NOT gates. This constraint on the *total* number of available gates is what makes sure that all circuits can be decoded without the need of brute force querying of each input combination.
- **Circuit Construction**: The available logic gates (x AND, y OR, z NOT) are distributed among the C circuits. The distribution process ensures that all available gates are used and that each circuit has a sufficient number of AND/OR gates to combine the k inputs into a single output. Specifically, each circuit must have at least $\lceil \log_2(k) \rceil$ AND + OR gates. Once the gates are distributed to a circuit, the circuit structure is generated layer by layer. At each layer, any of the one operation can occur:
  - Two input signals (outputs from the previous layer) are randomly selected and combined using either an AND or an OR gate (if available).

  - Alternatively, a single input signal is randomly selected and inverted using a NOT gate (if available).
  - All remaining input signals from the previous layer are carried forward to the next layer.

This process continues until all allocated gates for that circuit are used, resulting in a single output signal.

A.3. Twenty Questions

- **Game Progression**: The model is told to pick a secret word from a fixed set of words and their corresponding Wordnet hypernyms (i.e. attributes). The user then asks the model a series of yes/no questions about the attributes of the chosen secret word (e.g. "Is it a type of mammal?"), to which the model responds. If at any point the model responds in a way that logically contradicts the previously provided responses, the game immediately ends in a loss for the model. Otherwise the game continues until the user makes a final guess about the identity of the secret word (e.g. "Is your word chihuahua?") to which the model responds. The model's response to this final question is again assessed for logical consistency, with the game ending in a loss if it is inconsistent, and ending in a win otherwise. The user attempts to guess the final word when either (1) there is only a single logically consistent word remaining, (2) the maximum number of turns has been reached, or (3) a stochastic early-guessing condition occurs.
- **Logical Contradictions**: A logical contradiction can occur in the following ways:
  - The user asks whether the secret word has a hypernym such that all words having that hypernym have already been excluded, and the model says "yes".
  - The user asks whether the secret word has a hypernym such that all remaining logically consistent words have that hypernym, and the model says "no".
  - There is one logically consistent word remaining, the user guesses it, and the model says "no".
  - The user guesses a word that has already been logically excluded, and the model says "yes".
  - No logically consistent words remain.
- **User Actions**: The user simulator does not attempt to play optimally to efficiently identify the secret word. Instead, it operates based on a probabilistic state machine, that sometimes makes guesses to which the answer is already known, in an attempt to catch the model in a logical contradiction. On each turn, the user simulator selects one of the following actions with the given probability:
  - Spontaneously attempt to guess the secret word, even if there are still multiple logically-consistent words remaining ($p = 0.02$).
  - When guessing the secret word, picking a word that has already been logically excluded ($p = 0.15$).
  - When guessing a hypernym, pick one that belongs only to words that have been logically excluded ($p = 0.1$).
  - When guessing a hypernym, pick one that belongs to all remaining logically consistent words ($p = 0.1$).
  - Guess a hypernym belonging to a proper subset of the non-excluded words (otherwise).
- **Word Set**: The words and associated hypernyms used in the games are pre-filtered according to the following criteria:
  - The words are nouns appearing in the SCOWL word list.
  - Each word has at least 7 hypernyms.

- Every word has the hypernym "physical entity".
- No word has the hypernym "abstraction".
- No two words have identical sets of hypernyms.
- No word is the hypernym of any other word in the set.

A.4. Examples

Movie Recommendation

```
user
You’re a movie recommendation agent who will interact with a user. Your goal is
to ask questions in order to understand the user’s preferences and suggest a
movie that they’ll enjoy the most. For this, you are given a library of movies
in the table below. Each movie has scores on {num_attributes} different
attributes: ...

The user’s preference can be modeled as a linear function of the {num_attributes
} scores and this function is hidden from you. The user will always act
consistently based on the predefined user preference function.
...

You’re only allowed to ask {num_questions} questions from the user, to uncover
the underlying user preference function. Note that each question can only be
formatted as follows:
"<question>Would you prefer watching $option_1 over $option_2?</question>".
...

The user will respond to your question. The response can only be "Yes", "No" or
"No Preference". ...

Only once you’re done interacting with the user after asking {num_questions}
questions one—by—one, you will be given a new set of movies and based on
whatever you’ve learnt about the user, your goal would be to pick the movie
which you think the user will enjoy the most.
...

The movie dataset which you can use to ask the user questions is provided in the
 table below.
...

You can go ahead with asking the first question. While making deductions, try to
 uncover the underlying user preference function.

model
<question>Would you prefer watching Beneath the Crimson Tide over The Obsidian
Fortress?</question>

user
Yes
Please ask the next question. 9 questions remaining. ...
...

user
No
Now apply whatever you learnt about the user and predict which movie will
satisfy the user the most from the following set:
...

Reason about the linear user preference function and then provide your final
answer. ...

model
<final_answer>A Dance with Time</final_answer>
```

Circuit Decoding

```
user
**Welcome to the Circuit Deduction Challenge!**

**Your Inventory:** You have a total of {components}. These are distributed
across {num_circuits} circuits, labeled {circuit_labels}.

**Your Goal:** Deduce the logical structure of each circuit.

**The Setup:**
* Each circuit accepts {num_inputs} binary inputs (0 or 1) and produces a single
 binary output (0 or 1).
* Circuits are implemented in layers. Each layer takes two inputs from the
previous layer and produces a single output.
* No circuit will always output 0 or always output 1, regardless of input.
* All circuit components are being used and are distributed without replacement
across {num_circuits} circuits.

**Your Tools:** You have {num_questions} queries to figure all the circuits out.
 Each query is a call to a circuit with specific inputs.
At each turn of the conversation with me, you must make only 1 query. Hence,
there will be {num_questions} turns of the conversation before you have to make
your final guess.

**How to Play:**
1. At each turn, you must make two queries
2. Use the following format for your query:
...
3. You must make only 1 query at each turn

**Think Strategically:**
* Reason about the behavior of each circuit based on the available components.
* Plan your {num_questions} queries carefully to maximize information gained.

**Final Submission:**
After your {num_questions} queries, provide the truth tables for all circuits ({
circuit_labels}) in this format:
...

Please start with your first query. Plan your steps before making the queries.

model
<query>A(0, 0, 0)</query><ctrl100>

user
A(0, 0, 0) = 1
Please send your next query. 17 queries remaining. ...
...

user
C(0, 0, 0) = 0
Please provide the truth tables for all circuits in the format requested. ...

model
<guess>101111110100010101011111111</guess>
```

Twenty Questions

```
user
Let’s play a game of 20 questions. I want you to choose a secret word (don’t
tell me what it is!) from the list below:

{word_list}

I will ask you simple yes/no questions about the word. You should answer these
questions with either ’yes’ or ’no’, according to the attributes listed below:

{word_to_attributes_list}

So for example if your word is ’dog’, and ’dog’ has the listed attribute ’object
’, and I ask ’Is it a type of object?’, you should say ’yes’, even though dogs
are not typically considered objects.
Conversely, if your word is ’dog’, and ’dog’ *does not* have the listed
attribute ’pet’, and I ask ’Is it a type of pet?’, then you should say ’no’,
even though dogs are typically considered pets.

When I think I know what the word is, I’ll ask you to confirm it, and you should
 again respond with either ’yes’ or ’no’.

Ok, let me know once you’ve picked your word (remember, don’t tell me what it is
), and we’ll get started!

model
Okay, I have chosen a secret word. Ask away!

user
Is it a type of plate?

model
no
...
```

Word Chaining

```
user
We’re going to play a game. One of us will say a word, then the other will say a
 new word that starts with the same letter that the last word ended with. (So
for example if I said ’ant’, you might say ’television’.) All words must come
from the list below. If you repeat a word, or select a word that isn’t on the
list, or break the first−letter rule, you lose. Please always wrap your chosen
word in single−quote marks.

Allowed words: {words}

model
Okay, I’ll start with ’encirclement’

user
Ok, my word is: ’topaz’.
...
```

Word Guess

```
user
Let’s play word guess. The secret word is from the following vocabulary: {vocab
}.
At each attempt, I will give feedback to you and specify which tiles are grey,
yellow and green. Green implies that the letter is in the word and in the
correct position, yellow implies that the letter is in the word but in the wrong
 position and grey implies that the letter is not in the word.

For e.g., if the secret word is SPARK and you guessed PROOF, then the feedback
would be:
yellow,yellow,grey,grey,grey.

Your goal is to correctly guess the secret word in the minimum number of
attempts and will be evaluated on the number of attempts you take to arrive at
the secret word. Think step by step to ensure that you’re taking the most
optimal actions to guess the secret word.

All of your guesses should be structured as follows:
<attempt>Your guess</attempt>

Make sure that all of your attempts are strictly from the vocabulary specified
above.
Start with your first attempt and make sure to follow the format specified above
.

model
<attempt>RETRO</attempt>

user
green,grey,grey,grey,yellow
...
```