A Good Plan is Hard to Find:[*] Aligning Models with Preferences is Misaligned with What Helps Users

Nishant Balepur[1] **Matthew Shu**[2] **Yoo Yeon Sung**[1] **Seraphina Goldfarb-Tarrant**[4]
Shi Feng[3] **Fumeng Yang**[1] **Rachel Rudinger**[1] **Jordan Boyd-Graber**[1]
[1]University of Maryland [2]Yale University [3]George Washington University [4]Cohere
nbalepur@umd.edu jbg@.umiacs.umd.edu

Abstract

To assist users in complex tasks, LLMs generate plans: step-by-step instructions towards a goal. While alignment methods aim to ensure LLM plans are helpful, they train (RLHF) or evaluate (ChatbotArena) on what users prefer, assuming this reflects what helps them. We test this with Planorama: an interface where 126 users answer 300 multi-step questions with LLM plans. We get 4388 plan executions and 5584 comparisons to measure plan helpfulness (QA success) and user preferences on plans, and recreate the setup in agents and reward models to see if they simulate or prefer what helps users. We expose: 1) user/model preferences and agent success do not accurately predict which plans help users, so common alignment feedback can misalign with helpfulness; 2) this gap is not due to user-specific preferences, as users are similarly successful when using plans they prefer/disprefer; 3) surface-level cues like brevity and question similarity strongly link to preferences, but such biases fail to predict helpfulness. In all, we argue aligning helpful LLMs needs feedback from real user interactions—not just preferences of what looks helpful—so we discuss the plan NLP researchers can execute to solve this problem.[1]

Step 1: Introduce the Paper's Plan[2]

Users increasingly rely on Large Language Models (LLMs) to assist with complex problems like coding (Wen et al., 2024), fact-checking (Min et al., 2023), and organizing day-to-day tasks (De Buyser, 2023). A common way an LLM supports these requests in practice—especially when it cannot do the task on its own—is with plans (Ouyang et al., 2023): step-by-step instructions for how to complete it (Newell et al., 1972). Plans improve task completion accuracy and efficiency (Roncone et al., 2017), teach problem-solving skills (Wood et al., 1976), and reduce cognitive load (Atkinson et al., 2000), making them a promising tool for human–AI collaboration.

LLM plans are widely used, but few study which plans let users solve tasks accurately and quickly—precluding their improvement. This broad goal is called helpfulness in LLM research: ensuring LLMs give outputs useful to humans (Askell et al., 2021). For this goal, developers first gather feedback to assess the helpfulness of LLM outputs (Ouyang et al., 2022), either using these signals to rank LLMs by helpfulness in leaderboards (Chiang et al., 2024), or tuning LLMs on the most helpful outputs via alignment methods like Reinforcement Learning with Human Feedback (Christiano et al., 2017, RLHF).

To align LLMs for plan generation, the feedback choice is key—defining what LLMs learn is helpful (Bansal et al., 2024). A de-facto protocol has users compare two LLM responses (e.g. plans) and pick the one they prefer (Stiennon et al., 2020). While standard in alignment (Tie et al., 2025), it assumes users accurately select what helps them. If this assumption fails, we may reward plans that look useful but do not truly help users solve tasks quickly or accurately. This failure case is often unnoticed, as developers align and evaluate LLMs on preferences.

This paper challenges the assumptions of alignment by building Planorama (Figure 1), an interface to study if users' preferred plans—the standard signal in alignment—help them solve problems—our real alignment target. We deploy LLM-created plans in our interface to help users solve multi-step math and trivia questions with calculator and web search tools—complex, verifiable problem-solving tasks. We find preferred plans via user votes in pairwise comparisons (§3.2.1) and helpful plans based on which let users solve questions quickly and accurately (§3.2.2)—unified into one metric with Item Response Theory (§5.1). In total, 126 users solve 300 distinct questions with 600 LLM plans, yielding 4388 plan executions and 5584 comparisons: a rich

[*]This refers to Flannery O'Connor's story "A Good Man is Hard to Find". We subtly reference it 6 times (Appendix A.1).

[1]Our code and data are available at: https://github.com/Pinafore/plan-helpfulness

[2]Given our paper's focus on plans, we structure our sections as step-by-step instructions.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 11568–11595
November 4-9, 2025

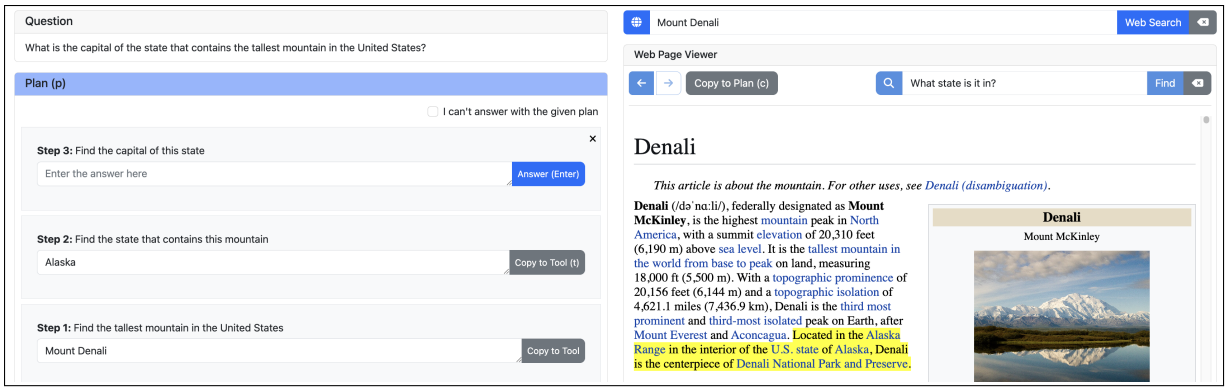

Figure 1: Overview of the `Planorama` interface. Users answer multi-step math or trivia questions (top left) with help from an LLM-generated plan (bottom left), seen one step at a time. We also provide built-in tools (right): calculators for math and web search for trivia. We collect 4388 execution traces on 600 question/plan pairs, measuring 126 users' accuracy and execution time.

testbed to check if preferred plans are also helpful.

While prior work has compared preferences and helpfulness to users (§8), plans are novel as models can use them (Wei et al., 2025); thus, we also see if models accurately predict what helps users by recreating our user feedback in models. We find plans models prefer via judgments from six reward models and GPT-4o (§4.2)—often used to score reasoning step quality (Cobbe et al., 2021)—and plans that help models via the accuracy/speed of a GPT-4o ReACT agent executing our plans (§4.1)—often used to solve tasks with tools (Yao et al., 2023).

After ensuring LLM plans help users and models more than no-assistance baselines (§6.1), we run a four-way comparison on which plans are preferred by and help users/models (§6.2). User/model preferences and agent outcomes barely beat random accuracy (< 0.63) at predicting which of two plans best helps users, so **standard user preferences can largely fail to capture what helps users**. Reward models also score preferred plans higher than helpful ones, so LLMs trained on such rewards may only look helpful. Lastly, users' accuracy and speed are mostly consistent when using plans they personally prefer or disprefer (§6.3), so disagreements in helpfulness and preferences are not individual noise, but inherent misalignment (Gilbert and Wilson, 2006).

To learn why preferences disagree with helpfulness, we qualitatively study plans. Simple features like brevity and question overlap often predict plan preferences but not helpfulness (§7.1), revealing shallow biases in user/model judgments uncorrelated with helpfulness. We then study all 129 cases when users prefer unhelpful plans, inferring they miss unexpected flaws, fall for steps with surface-level appeal, and disprefer unfamiliar solving methods (§7.2). Finally, we analyze 100 failed user and model executions on unhelpful plans, showing errors often occur not when steps are faulty, but when valid steps are executed poorly (§7.3); thus, training LLMs to be correct does not ensure they are helpful.

Preferences can diverge from what helps users, misaligning with our goals of helpfulness (Askell et al., 2021). This is alarming as preferences dominate LLM training like RLHF (Ouyang et al., 2022) and benchmarks like ChatbotArena (Chiang et al., 2024)—so we are pouring extensive resources and effort into a signal that might not help users at all. Thus, we urge more alignment work grounded in downstream user interactions and plan steps NLP researchers can help execute for this problem (§9).

Step 2: Define the Word "Plan"

Before discussing how we use plans (§3), we first define what a "plan" is. Our definition draws from education (Wood et al., 1976) and reinforcement learning (Fikes and Nilsson, 1971), which describe plans as a means to help students and models make better decisions. To adapt this for LLMs, we follow the definition from Valmeekam et al. (2023):

> The solution for a planning problem is a sequence of actions, or a plan, that when applied in the initial state will result in a state where the goal conditions are satisfied.

In our setting, the sequence of actions is high-level steps–drawn from best practices in educational theory for problem-solving (Wood et al., 1976). Our input question forms the initial state and "answering the question correctly/efficiently" is the goal condition—called "helpfulness" in this paper.

Plans are increasingly deployed to automate agentic tasks and support human decision-making

(§8), motivating our study of how they help different **players**—users and models—and how each perceives them. The following sections gather feedback from users (§3) and models (§4) on plans, then analyze disagreements in these signals (§5, §6).

Step 3: Deploy Plans to Help Users

To compare helpful versus preferred plans for users, we first need user feedback on LLM plans. We build **Planorama** (Figure 1): an interface to log users' success when solving multi-step questions assisted by LLM plans (helpfulness) and selections on plans users think help them (preferences). This section details Planorama, showing our source of questions and plans (§3.1), user preference and helpfulness collection (§3.2), and user recruitment (§3.3).

3.1 Generating Plans for Question Answering

Multi-step question answering (QA) is our testbed, as it is easier with plans (§6.1), objectively scored, and well-studied in NLP (Woods, 1973)—ideal for comparing preferred/helpful plans. We take 300 QA pairs (q, a) in two domains: GSM8k math (Cobbe et al., 2021)—multi-step equations; and MuSiQue (Trivedi et al., 2022) and MQuAKE (Zhong et al., 2023) multi-hop trivia—reasoning over many facts. We clean each q for correctness (Appendix A.2).

As our goal is to assess if the standard alignment protocol of pairwise preferences (Bai et al., 2022a) matches what helps users (§6.2), we need two plans $\mathcal{P} = (p_A, p_B)$ for each q. To create $\mathcal{P}$, we prompt LLMs for two stepwise plans $p = \{s_1, ..., s_n\}$ for q, where each $s_i \in p$ has a subtask for users to do, requesting a subanswer a_i;[3] the final subtask in s_n uses $a_i, ..., a_{n-1}$ to have users submit q's answer a.

To make plans $\mathcal{P}$ distinct for clearer user feedback (Lambert et al., 2024), we ask LLMs to vary plans in specificity, step count, and strategy. For more diversity, the 300 $\mathcal{P}$ are sampled evenly from GPT-4o (Hurst et al., 2024), Claude-3 Opus (Anthropic, 2023), Qwen-2 72B (Bai et al., 2023), and LLaMA-3 405B (Grattafiori et al., 2024). We ask LLMs to never reveal the answer a to q or any subanswer a_i for $s_i \in p$ so plans are high-level assistance rather than predictions; thus, helpfulness is based on how well plans guide users to the answer, not just prediction accuracy (Wen et al., 2025, §7.3). We manually verify plan quality (Appendix A.3).

[3]For example, for q ="*where was the first tsar born?*", step one could be s_1 = "Find the first tsar" with a_1 = "Ivan IV".

3.2 Answering Questions in Planorama

Having created plans (§3.1), we now deploy them in Planorama to elicit user preferences (§3.2.1) and how well they help users in QA (§3.2.2), letting us compare users' preferred and helpful plans (§6).

3.2.1 Preferences via Pairwise Comparisons

To get preferences, we use pairwise comparisons—the standard alignment feedback (Ji et al., 2023b). In a round of Planorama, users see a new question q and its plans $\mathcal{P}$ in a random order, and pick which $p \in \mathcal{P}$ they think will help them solve q more accurately/quickly (Figure 2). They can pick either p or mark a tie; the plan with more votes is "preferred". The user's choice does not alter which plan they use for q (§3.2.2) and such comparisons have little impact on QA success (Appendix A.5).

3.2.2 Helpfulness via User Plan Executions

After comparing plans (§3.2.1), users follow a plan so we can measure its helpfulness. For question q, users get one random plan $p \in \mathcal{P}$, only seeing its first step $s_1 \in p$ (Figure 1, left). Each s_i has high-level guidance leading to a subanswer a_i, and users are advised but not required to type a predicted subanswer $\hat{a}_i$, as it boosts problem-solving (Koretsky et al., 2016). After submitting $\hat{a}_i$, the next step s_{i+1} appears—one at a time for cognitive ease (Sweller, 1988). This repeats until the last step $s_n \in p$, where users submit the final answer $\hat{a}_n$ to q. If $\hat{a}_n$ matches the gold answer a—via the PEDANTS answer judge (Li et al., 2024b)—they can try another q; else, they keep trying until our 180-second time limit expires.

Some s_i need complex math/knowledge—hard to do alone—so we add a calculator with basic operations used in GSM8k ($+, -, \times, \div$) for math (Figure 7) and web search for trivia (Figure 1, right). In search, users can submit queries and view the most similar Wikipedia page[4] via Google's search API[5] and `ctrl+F` in pages via Cohere's Rerank API.[6]

To better ensure users follow p (beyond attention checks; §3.3), we also allow users to: 1) skip q; or 2) write their own plan for q. We omit data from (2) when later finding which p is helpful (§5.1). Users are also more accurate when using any p versus no plan (§6.1), so executing LLM plans is beneficial.

Lastly, we define plan helpfulness via education research (Sweller and Cooper, 1985): helpful plans

[4]Wikipedia is the source corpus for MuSiQue/MQuAKE. We ensure all q are solvable with Wikipedia (Appendix A.2).

[5]https://developers.google.com/custom-search/

[6]https://cohere.com/rerank

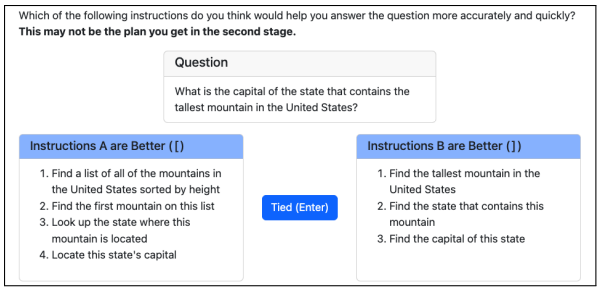

Figure 2: Users pick the plans they predict best help them (or mark `Tie`) in pairwise comparisons as their plan *preferences*.

let users solve q in less time for accurate, efficient problem-solving. Thus, we log users' accuracy—if they answer q correctly on their first try—and execution time—how many seconds they take—when using p; we later combine these into a single score to label which plan $p \in \mathcal{P}$ best helps users (§5.1).

3.3 Recruiting `Planorama` Problem-Solvers

We have 126 English-speaking users from university courses and online forums use `Planorama`, collecting 4388 execution traces and 5584 preferences on 600 question/plan pairs. Users can choose math and/or trivia, and the first question per task is a tutorial. For quality, we add two attention checks in comparisons (§3.2.1)—where one plan is clearly incorrect—and two in execution (§3.2.2)—where users retype text as steps (e.g. "type 144"); we omit the seven users failing these. Users receive coursework credit or $1/question (above minimum wage) and can attempt up to 300 questions. The top-12 users in accuracy and speed each receive an extra $50, gamifying `Planorama` (Hamari et al., 2014) to reward accurate and efficient problem-solving.

Step 4: Employ Plans to Help Models

So far, we curated user preferences to predict which plans help users (§3.2), but plans can also help or be preferred by models; agents can execute plans to capture helpfulness to models (§4.1) and reward models can score plans as model preferences (§4.2). This gives a four-way comparison (user/model preferred vs. user/model helpful plans) to test if users or models predict what helps users (§6). We now get model executions (§4.1) and preferences (§4.2).

4.1 Agent Implementation

To find which plans help models in QA, we use LLM agents: systems that use plans to solve multi-step tasks (Huang et al., 2024a). We adopt the standard agent framework ReACT (Yao et al., 2023),[7] which iteratively: 1) **reasons:** generates a chain-of-thought (Wei et al., 2022) to decide what to do next; 2) **acts:** calls a tool (e.g. calculator) to execute (1); and 3) **observes:** processes tool outputs from (2).

ReACT follows each plan $p = \{s_1, \ldots, s_n\}$ for a question q; it uses q and the first step s_1 as input and iteratively gives sub-answers $\hat{a}_i$ for each $s_i \in p$. ReACT has three tools: calculator, search, and SUBMIT for finalizing $\hat{a}_i$. Search mirrors §3.2.2, but to manage the context length, it returns just the first paragraph of the Wikipedia page and five sentences most similar to the search query. We prompt ReACT with exemplars from tutorial questions (§3.3), and have it execute step s_i until it calls SUBMIT to give $\hat{a}_i$. ReACT then moves to the next step s_{i+1}; we repeat this until submitting $\hat{a}_n$, taken as q's final answer. Following Nguyen et al. (2024b), we use GPT-4o as the base LLM (details in Appendix A.6).

Like with users, we log ReACT's accuracy and execution time on each plan $p \in \mathcal{P}$ for q; we later merge these (§5.1) to find which $p \in \mathcal{P}$ best helps ReACT solve q accurately and quickly. This mirrors user executions (§3.2.2), allowing us to compare plans that help users and models (§6.2, §7.3).

4.2 Reward Model Implementation

As model preferences (§3.2.1), we use reward models (Stiennon et al., 2020, RMs): trained to score response helpfulness $\hat{z}$ across domains. The RMs $r_\theta(p) \rightarrow \hat{z}$ score each plan $p \in \{p_A, p_B\}$. If $\hat{z}_A > \hat{z}_B$, the RM prefers p_A—predicting it as more helpful for users than p_B—and vice versa. We select six RMs with strong accuracy on RewardBench (Lambert et al., 2025): QRM (Dorka, 2024), GRM (Yang et al., 2024), Skywork-Reward (Liu et al., 2024), Nemotron (Wang et al., 2024c), InternLM2 (Cai et al., 2024), and ArmoRM (Wang et al., 2024a).

We also use GPT-4o as a generative RM (Zheng et al., 2023, LLM-as-a-judge), predicting which $p \in \mathcal{P}$ helps users answer q more accurately/quickly—mirroring user pairwise comparisons (§3.2.1). GPT-4o judges both orders of $\mathcal{P}$; a plan is "preferred" only if GPT-4o picks it both times, otherwise a tie.

Step 5: Locate the Most Helpful Plans

With our user/model feedback, we now find helpful and preferred plans in pair $\mathcal{P}$ for question q. Identifying preferred plans is simple—via majority vote (§3.2.1) or RMs (§4.2)—but our goal of helpfulness

[7] https://docs.cohere.com/v2/docs/tools-on-langchain

is multi-faceted—letting **players** (users or models) solve q quickly and accurately (Sweller, 1988)—so our helpfulness metric must balance both signals.

Averaging players' accuracy and time is a simple fix, but fails to control for player skill differences (Sung et al., 2025). Skilled users may thrive even on unhelpful plans and unskilled users may fail on helpful ones, so averages conflate a plan's helpfulness with the skill of who used it. We randomly assign plans in our study, so we cannot ensure equal-skill users execute both plans for each question.[8]

To control for player skill, we use Item Response Theory (Lord, 1952, IRT): an educational testing tool that models each test-taker's skill θ_j and exam item's difficulty β_i—skill needed to solve item i—inferred from test-taker responses. Similarly, we use question/plan pairs $(q, p)_i$ as items and player accuracy/execution time as responses to learn skill θ_j and difficulty β_i—a signal for "un-helpfulness". For plans (p_A, p_B) on q, if item (q, p_A) is less difficult than (q, p_B), players solved q more accurately/quickly with p_A than p_B, so p_A is more helpful.

IRT has been used to test if plans support learning (Ueno and Miyazawa, 2018), suggesting it can measure problem-solving helpfulness. Further, our claims are consistent even if helpfulness is defined via averages—preferences and helpfulness disagree (Appendix A.8)—but we still use IRT to rigorously control for skill. We now design IRT via Bayesian inference (§5.1), given its ease of implementation.

5.1 Item Response Theory Learns Helpfulness

IRT models each player's skill θ_j to learn two metrics for a question/plan item $(q, p)_i$: difficulty β_i—how hard it is—and discriminability γ_i—how well it discerns player skill. We use β_i for helpfulness, but interpret γ_i and θ_j in Appendix A.9. All random variables (RVs) use standard Normal priors:

$$\beta_i, \gamma_i, \theta_j \sim \text{Normal}(0, 1). \tag{1}$$

For an item/player (i, j), we observe two responses: player accuracy $a_{i,j} \in \{0, 1\}$ and execution time $t_{i,j} \in \mathbb{R}^+$. We model accuracy and time separately, transforming β_i/γ_i with slope m and intercept b:

$$b_\beta^{\text{acc}} \sim \text{Normal}(0, 1), \tag{2}$$

$$m_\beta^{\text{acc}} \sim \text{HalfNormal}(1), \tag{3}$$

$$\beta_i^{\text{acc}} = m_\beta^{\text{acc}} \cdot \beta_i + b_\beta^{\text{acc}}, \tag{4}$$

and same for γ_i^{acc}, β_i^{time}, and γ_i^{time}. Eq. 3 constrains slope $m > 0$ as otherwise, signs can flip—wrongly learning higher β_i^{acc} implies lower $a_{i,j}$ (Ghosh and Dunson, 2009). We first model $a_{i,j}$ via the standard 2-parameter logistic IRT model (Lord, 2012, 2PL):

$$a_{i,j} \sim \text{Bernoulli}\left(\text{sig}\left(\gamma_i^{\text{acc}}(\theta_j - \beta_i^{\text{acc}})\right)\right), \tag{5}$$

where $\text{sig}(x)$ is the sigmoid of x. Intuitively, Eq. 5 means players of skill θ_j exceeding item difficulty β_i^{acc} are likely accurate, while discriminability γ_i^{acc} alters how sharply the prediction changes with skill.

For time, we only model $t_{i,j}$ if player j *correctly* answers item i; failure speed does not meaningfully inform helpfulness.[9] Thus, only when $a_{i,j} = 1$, we model $\log(t_{i,j})$ as a Normal distribution based on IRT—a standard approach (Van der Linden, 2006):

$$\sigma_{\text{time}} \sim \text{HalfNormal}(0.5), \tag{6}$$

$$\mu_{\text{base}} \sim \text{Normal}(3.5, 1), \tag{7}$$

$$\mu_{\text{time}} = \mu_{\text{base}} + \gamma_i^{\text{time}}(-\theta_j + \beta_i^{\text{time}}), \tag{8}$$

$$\log(t_{i,j}), \sim \text{Normal}\left(\mu_{\text{time}}, \sigma_{\text{time}}\right). \tag{9}$$

Eq. 9 is interpreted like Eq. 5 but inverts the difference in θ_j and β_i^{time}, as if player skill exceeds item difficulty, $t_{i,j}$ should be *lower* to indicate efficient problem-solving, not higher as with $a_{i,j}$. The prior on μ_{base} (Eq. 7) maps the expected time to 0–180 seconds—the time limit users have (§3.2.2)—and improves IRT's fit of observed data (Appendix A.9).

Difficulty β_i of item $(q, p)_i$ captures helpfulness: lower β_i means players solved q more accurately and efficiently with p. While β_i also measures the difficulty of q, comparing β_i for items (q, p_A) and (q, p_B) controls q, isolating the plans and letting us compare the helpfulness of p_A and p_B for q (§6.2).

We learn RVs via NUTS (Hoffman et al., 2014) and use unique RVs to learn helpfulness for users (§3.2.2) and models (§4.1). RVs converge in 1000 epochs/5 chains (full evaluation in Appendix A.9).

Step 6: Compare Preferred/Helpful Plans

Equipped with metrics for helpfulness (§5), we now see if proxies—user-preferred plans in comparisons (§3.2.1); model-preferred plans via RMs and judges (§4.2); and model-helpful plans via agent outcomes (§4.1)—capture alignment's goal: what helps users (§3.2.2). After ensuring LLM plans help (§6.1), we show proxies fail to predict what helps users at aggregate (§6.2) and individual levels (§6.3), proving proxies in alignment can misalign with helpfulness.

[8] The same user cannot execute both plans, as they would already know the question's answer after using the first plan.

[9] For example, a user failing after a longer time could signal confusion (i.e. unhelpfulness) or motivation (i.e. helpfulness).

	Math Questions				*Trivia Questions*			
Proxy	User Prefer	User Helpful	GPT Prefer	GPT Helpful	User Prefer	User Helpful	GPT Prefer	GPT Helpful
User Prefer	—	52.000	67.333	55.333	—	55.667	68.667	49.000
User Helpful	52.000	—	58.667	57.333	55.667	—	56.333	62.667
GPT Prefer	67.333	58.667	—	52.667	68.667	56.333	—	54.333
GPT Helpful	55.333	57.333	52.667	—	49.000	62.667	54.333	—
QRM	60.000	56.000	72.000	42.667	65.667	51.333	56.333	36.667
GRM	53.333	54.667	66.000	38.667	64.333	51.333	57.000	40.667
Skywork	66.667	51.333	71.333	43.333	66.333	53.333	59.000	40.000
Nemotron	54.000	60.000	66.667	40.000	59.667	50.667	53.667	34.667
InternLM2	57.333	57.333	70.667	41.333	61.000	52.667	51.667	38.000
ArmoRM	56.667	56.667	68.000	39.333	59.667	52.000	53.000	42.667

Table 1: Agreement matrix on which of two plans users/GPT/RMs prefer and helps users/GPT (full matrix in Appendix A.8). No proxy accurately predicts what helps users (User Helpful column), so standard alignment feedback can misalign with helpfulness.

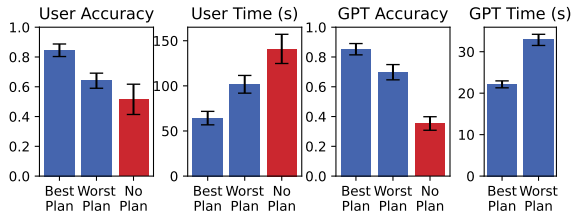

Figure 3: Users and agents (GPT) who opt-out of LLM plan assistance are slower/less accurate (macro-average) across the better/worse plans in each pair, so plans are generally helpful.

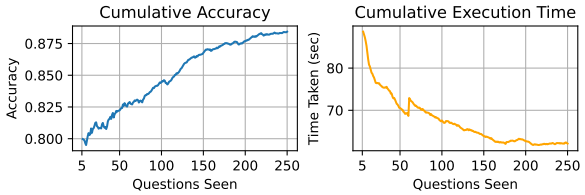

Figure 4: Users refine their QA problem-solving accuracy and execution time as they keep interacting with our LLM plans.

6.1 LLM Plans Drive Problem-Solving

Testing helpfulness is fruitless if plans do not help at all. To ensure this for users, we macro-average accuracy and time in users who executed LLM plans and who wrote their own plan (§3.2.2)—likely self-confident. Similarly, we ablate ReACT (§4.1), having GPT-4o perform QA with no plan. We group the better (higher mean accuracy/lower mean time) and worse plans in each pair; both often boost QA success versus no plan (Figure 3). Better/worse plans also yield different accuracy/time (95% confidence intervals; CIs), so plans exhibit discernible helpfulness. We then plot users' cumulative accuracy and speed as they solve more questions (Figure 4); both improve, so users refine their problem-solving with plans over time. Thus, LLM plans do help players.

6.2 User-Helpful Plans Escape Most Proxies

As plans help (§6.1), we now test if preferred plans for question q are helpful. We label plans in $\mathcal{P}$ as:

1. **User Helpful**: Which plan best improves user QA accuracy and speed, via IRT (§3.2.2, §5.1).
2. **User Preferred**: Which plan most users think help them, via pairwise comparisons (§3.2.1).
3. **GPT Helpful**: Which plan best helps the GPT-4o ReACT agent, mirroring (1) (§4.1, §5.1).
4. **GPT Preferred**: Which plan GPT-4o predicts best helps users, via LLM-as-a-judge (§4.2).
5. **RM Preferred**: Which plan our 6 reward models (§4.2) each score as most helpful for users.

We have 10 labels (six in (5)) on which plan $p \in \mathcal{P}$ is helpful/preferred for every question q. Helping users in (1) is the goal of alignment, but (2)–(5) can form proxies (Askell et al., 2021), so we now test how accurately they capture (1). If (2) or (4) deems plans tied, we assign a score of 0.5 (random guessing), as filtering ties precludes proxy comparisons.

Alignment signals may not always be helpful. No proxy accurately predicts which of two plans best helps users (Table 1, User Helpful column); accuracy is $< 63\%$—near random—so designing LLMs using preferences or agent outcomes can severely misalign them with what truly helps users. Interestingly, GPT slightly beats users in selecting helpful plans (User/Model Prefer vs User Help), so third-parties uninvolved in the task (external users, LLMs) may offer less biased helpfulness judgments (§7.2).

RMs can be adversarially helpful. Most RMs train on preferences (Stiennon et al., 2020) and our evaluation exposes this: RMs better predict plans players prefer (Table 1, User/Model Prefer) than what helps them. Notably, RMs score below random (< 0.5) at predicting what helps our GPT agent (§4.1), so they may be adversarially helpful (Ajwani et al., 2024)

	Math Regression				Trivia Regression			
Feature	User Prefer	User Help.	Model Prefer	Model Help.	User Prefer	User Help.	Model Prefer	Model Help.
# Steps	**-0.12 (0.00)**	-0.01 (0.92)	0.013 (0.68)	0.01 (0.91)	**-0.08 (0.00)**	**-0.20 (0.02)**	**0.17 (0.00)**	**-0.19 (0.02)**
μ_{words}	**-0.02 (0.00)**	0.00 (0.83)	-0.01 (0.15)	0.00 (0.99)	**-0.04 (0.00)**	**-0.07 (0.02)**	**-0.03 (0.00)**	-0.03 (0.33)
q-p Sim.	0.19 (0.79)	0.17 (0.81)	**0.66 (0.01)**	0.16 (0.78)	**0.30 (0.00)**	0.41 (0.36)	**0.50 (0.00)**	-0.26 (0.52)
Diverse.	**-0.60 (0.00)**	-0.54 (0.45)	**-1.03 (0.00)**	0.85 (0.12)	-0.23 (0.29)	-0.09 (0.90)	-0.18 (0.47)	-0.97 (0.15)
Read.	-0.00 (0.50)	-0.01 (0.44)	-0.00 (0.18)	0.00 (0.56)	0.00 (0.52)	**-0.01 (0.01)**	0.00 (0.76)	-0.00 (0.86)
Adj. R^2	0.123	-0.017	0.371	0.004	0.137	0.052	0.578	0.031

Table 2: Regression weights for how well features predict user/model preferred and helpful plans in math and trivia. Cells have feature weights, p-value in parentheses. The final row contains the adjusted R^2 of the regression (higher means easier to predict).

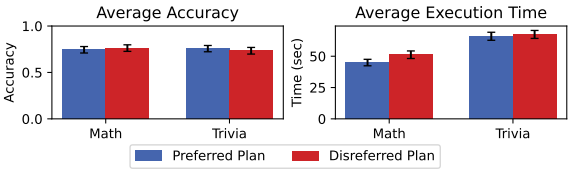

Figure 5: There are rarely differences (95% CIs) in accuracy and speed for users following their preferred vs dispreferred plan, so preference/helpfulness gaps are not just user variance.

if used to make plan for agents. This may happen as RMs learn preferences across domains (Gao et al., 2023), reinforcing biases linked to preferences but unrelated to helpfulness. We examine this in §7.1.

Takeaways. The core assumption of alignment—preferences reflect helpfulness—completely fails in our plans. Thus, we need more work studying ways to align LLMs with signals from downstream user interactions. In §9, we plan steps towards this goal.

6.3 Users Fail to Pick What Helps Themselves

As we aggregate helpfulness and preferences over users, their misalignment (§6.1) could stem from user variance (Kirk et al., 2024): users may fail to pick helpful plans on average but pick plans that help themselves. If so, we could use strategies that learn user-specific preferences to close this gap (Li et al., 2024a)—personalizing helpfulness per user.

We can test this: before solving question q, users pick a plan $\hat{p} \in \mathcal{P}$ as helpful, but follow a random plan $p \in \mathcal{P}$ (§3.2.2). By comparing mean accuracy/speed when users see their preferred ($p = \hat{p}$) or dispreferred ($p \neq \hat{p}$) plan, we can test if the choice impacts users' success. Individual variance is not the cause: users succeed regardless of the plan used (95% CIs; Figure 5). Thus, preferences do not capture helpfulness at aggregate and individual levels.

Step 7: Examine Features of Misfit Plans

To understand preference/helpfulness gaps (§6), we show why users may misjudge plans: shallow cues bias them (§7.1), some errors follow patterns (§7.2), and unhelpful plans are still valid (§7.3)—revealing upcoming challenges in aligning helpful LLMs (§9).

7.1 Users E.A.T. Up Surface-Level Features

To study plan biases, we see if question/plan pair features $f(q, p)$ predict preferences/helpfulness—1) step count; 2) mean words per step (μ_{words}); 3) word overlap of q and p; 4) diversity via type-token ratio in p (Richards, 1987); and 5) Flesch readability (Flesch, 1948)—thoroughly covering verbosity (Ye et al., 2025) in (1-2), relevance (Cool et al., 1993) in (3), and style (Schwarz, 2004) in (4-5).

To fix q, we use feature differences $f(q, p_A) - f(q, p_B)$ in plans (p_A, p_B) to predict differences in helpfulness (IRT; §5.1) or preferences (proportion picked; §3.2.1) via least squares (Fisher, 1922). If feature x's weight is positive and $p_A > p_B$ in x, p_A tends to be more helpful/preferred. We run linear regressions for users/models, merging GPT/RM outputs for model preferences (§4.2). Each regression gives an R^2 value for how well it fits its prediction.

Simple cues predict preferences ($R^2 \gg 0$) but not helpfulness ($R^2 \approx 0$) in math/trivia (Table 2). Users show an inverse verbosity bias—preferring short p as helpful—while models prefer more steps/lower μ_{step}. Users (trivia) and models (both) pick p with high word overlap in q; we speculate they tend to prefer outputs copying prompts (Chen and Goldfarb-Tarrant, 2025). In math, players pick p with low diversity, likely looking structured ("1) *find* x; 2) *find* y; ..."). Yet, these rarely predict helpfulness; most weights are insignificant. In trivia, short plans help, and for users, lower readability—likely more specific—but nothing predicts the helpfulness of plans in our math dataset.

We must curb biases to use preferences in alignment, or LLMs may perpetuate them (§9). Helpfulness escapes these heuristics, so it may be learnable with less risk of shortcuts (Gardner et al., 2021) or artifacts (Poliak et al., 2018; Balepur et al., 2024a).

7.2 Study: What Would of Been a Good Plan

To augment our regression (§7.1), we review (Bingham, 2023) all 129 cases when most users prefer the unhelpful plan, discussing patterns in these errs. **Plans are full of surprises.** Users prefer less support (§7.1), but knowing this is tough without doing the task. A trivia q requests the "*...country originating the sport played by the Auckland Aces*". Users pick p_A, where users must locate this in one query, but when one user tried this, the Google web search tool failed and sent them to the irrelevant page "Super Smash". In contrast, p_B has users design two smaller queries, yielding perfect accuracy. Since some flaws in plans only surface during execution—like errors in tool calls (Norman, 2014)—it is tough to predict helpfulness just by looking at responses. **Looks can be deceiving.** Users misjudge plans that *look* helpful. In a trivia q asking for "*Andrew Stanton's notable works*," p_A and p_B are similar, but p_A has users find his "major films" and in p_B, "biggest box office hit." Users prefer p_B, maybe due to its engaging phrasing, but it was not helpful (0.67 vs 1.0 acc.); five users with p_B were misled, searching "biggest box office hit by Andrew Stanton" and sent to the irrelevant page "John Carter".[10] In math, one p_A looks structured ("1) calculate x; calculate y;...") but yields an incorrect answer, while p_B is correct but with a redundant final step: "round the answer". p_B is more helpful, but **all** users pick p_A, maybe due to p_B's redundancy. Stylistic polish can mask flaws (Hosking et al., 2024), so LLMs aligned on preferences may trick users (Wen et al., 2025). **Users stick to what they know.** Users may misjudge plans with familiar strategies. In math q "*She scores 345 points in 15 games: 4 free throws and 5 2-pointers per game. How many 3-pointers did she average?*", p_A gets all 3-pointer points ($345 - 4(15) - 5(2)(15) = 45$) and divides by games ($\frac{45}{15} = \underline{3}$), while p_B reasons per game ($\frac{345}{15} = 23$), subtracts points ($23 - 4 - 2(5) = 9$), then divides ($\frac{9}{3} = \underline{3}$). Most users pick p_A, as common advice is to "sum before division",[11] but p_B is more helpful (0.3 vs 1.0 acc.). Thus, familiarity may blind users to more helpful strategies (Macaluso et al., 2022). **Takeaways.** Judging helpfulness sans execution is hard: plans fail suddenly and trick/bias users. To fix this, we encourage researchers to align models with feedback from downstream user interactions (§9).

[10] It is a box office *flop*, which is why Google search redirects there: https://en.wikipedia.org/wiki/John_Carter_(film)

[11] www.geeksforgeeks.org/practice-questions-on-average/

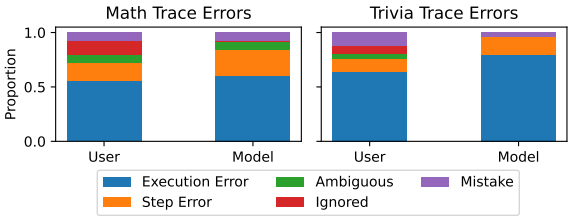

Figure 6: User and model trace errors on 100 unhelpful plans. Most errors occur when executing correct steps, so helpfulness goes beyond just correctness (examples in Appendix A.10).

7.3 There's no real Helpfulness in Correctness

If correctness ensured helpfulness, alignment with verifiable rewards (Lambert et al., 2024) could fix preference/helpfulness gaps; objective correctness metrics could capture helpfulness, avoiding flaws in subjective preferences (§7.1, §7.2). To test this, we study 100 failed execution traces on the 25 least helpful plans (via IRT; §5.1) for users/ReACT in math/trivia. We label each failure as: **1) step error** (incorrect step); **2) ambiguous** (unclear step); **3) execution error** (correct step but mis-executed); **4) ignored** (skipped step); or **5) mistake** (copying error), inferred from player tool calls/sub-answers.

For users and ReACT, most failures are not faulty steps, but poor executions of valid plans (Figure 6). Thus, as LLMs plans are often correct, helpfulness needs signals beyond correctness—like simplicity to limit execution errors, clarity to resolve ambiguity, and engagement to stop skipping—best learned from real downstream interactions with users (§9).

Step 8: Review Related Work

LLM Plans: Planning has long been considered a goal of AI (McCarthy, 1959), now studied in LLM reasoning: decomposing complex tasks (Khot et al., 2023; Zhou et al., 2023). It is often used in agents (Huang et al., 2024b), which iteratively plan steps and call tools (Yao et al., 2023; Schick et al., 2023), to solve multi-step math (Hendrycks et al., 2021), coding (Wang et al., 2024b), GUI (Nguyen et al., 2024a), and retrieval (Mialon et al., 2023) tasks.

While prior work studies plans in agents, we use them to help users. Many applications deploy LLM plans—literature search (Feng et al., 2024), teaching (Goslen et al., 2024), coding (Wen et al., 2024), fact-checking (Min et al., 2023), advice (Wester et al., 2024), long-form text generation (Balepur et al., 2023, 2025; Shao et al., 2024), and note-taking (De Buyser, 2023)—but few study what makes plans helpful for users. Conversely,

we build Planorama to locate which plans help users in multi-step QA, comparing preferences and helpfulness across users and models, and critically examining when this feedback disagrees.

Helpfulness: Helpfulness is a north star goal of alignment (Askell et al., 2021): making LLMs useful to users (Ouyang et al., 2022). It is now pursued by curating preferences on LLM outputs and tuning LLMs on those rated helpful (Stiennon et al., 2020). This data is used in methods like Direct Preference Optimization (Rafailov et al., 2023) and Reinforcement Learning with Human Feedback (Christiano et al., 2017, RLHF) for dialogue (Cui et al., 2024), QA (Ji et al., 2023a), and plan (Song et al., 2024) generation. Leaderboards like Chatbot Arena also use preferences to rank LLMs (Chiang et al., 2024).

While useful, recent work critiques preferences for alignment; they degrade safety (Ji et al., 2023a; Zhang et al., 2025) and personalization (Kirk et al., 2024; Sorensen et al., 2024), and can be ambiguous to elicit (Malaviya et al., 2024; Pitis et al., 2024). Similarly, we show standard proxies—preferences (Bai et al., 2022b) and agent simulations (Park et al., 2023)—can fail to capture helpfulness. While work compares preferences and helpfulness (Balepur et al., 2024b; Mozannar et al., 2025) and user/-model judgments (Bansal et al., 2024), we study all four in LLM plans and their qualitative differences.

Step 9: Submit the Final Conclusion

To aid users in complex tasks, we must rethink how we teach LLMs what helpfulness means. Standard feedback like preferences and agent outcomes can fail to capture what helps users **at all** (§6.2). This is not users' fault; it is tough to judge helpfulness, shaped by individuality (§6.3), stylistic cues (§7.1), unexpected execution errors (§7.2), and factors past correctness (§7.3). If we develop LLMs just via user preferences—as in RLHF (Ouyang et al., 2022) or ChatbotArena (Chiang et al., 2024)—versus downstream user interactions—as in Planorama—we will misalign LLMs: prioritizing what looks helpful, not what actually helps users (Saxon et al., 2024a).

While promising, practical issues remain. First, this feedback is costly and hard to define in subjective tasks (§10); we can remedy this by looping in experts to better judge helpfulness (Ley et al., 2010, e.g. educators for learning), teaching users to avoid shallow biases (§7.1), routing select cases for downstream signals (Miranda et al., 2024), or designing agents to better simulate humans (Liu et al., 2022). Second, while helpfulness is often our main goal (Bai et al., 2022a), user preferences still matter. Just optimizing on helpfulness risks a paternalistic "eat your veggies 🥦" effect; we can still "cook/season them 🧂" to a user's liking (Amershi et al., 2019). By executing these steps, we can move from LLMs that just look helpful, to LLMs that truly help users.

Acknowledgments

We would like to thank the CLIP lab at the University of Maryland and external collaborators for their help. Specifically, we thank Grace Chen, Yu Hou, Dayeon Ki, Dang Nguyen, Daniel Smolyak, and Navita Goyal for dogfooding our interface. We especially appreciate Connor Baumler for extensive discussions on the interface, analysis, and framing of this paper. We are also grateful for Vishakh Padmakumar and Aaron Gluck's feedback during earlier versions of this project. During the rebuttal period, we thank Connor Baumler, Abhilasha Ravichander, Vishakh Padmakumar, Dayeon Ki, Dang Nguyen, Yu Hou, Atrey Desai, Maharshi Gor, and Paiheng Xu for participating in our poll to convince a reviewer that we should keep the step-by-step format of our paper sections. Finally, we thank Ryan Zhang for brainstorming references to *"A Good Man is Hard to Find"* in this paper—our most creative (and favorite) feature of this work.

This material is based upon work supported by the National Science Foundation under Grant No. DGE-2236417 (Balepur), IIS-2339746 (Rudinger), and IIS-2403436 (Boyd-Graber). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Annotations and computing resources were made possible by a gift from Adobe Research. Access to Cohere's Reranker was possible through a Cohere for AI Research Grant.

10 Limitations

While our paper is the first to study how LLM plans help users and models in QA, we acknowledge we cannot comprehensively cover all tasks and models.

First, to compare preferences and helpfulness in users and models, we use multi-step math and trivia QA, as they are verifiable tasks well-researched in NLP (§3.1). Other tasks also have these qualities—like GUI navigation (Nguyen et al., 2024a), games (Samdarshi et al., 2024), and coding (Wen et al., 2024)—but we cannot study them due to resource

constraints; our user study cost $4000 for sufficient feedback. We encourage future work to extend our analysis to more verifiable domains (Lambert et al., 2025), and to develop protocols for measuring response helpfulness in harder-to-verify tasks such as writing (Chakrabarty et al., 2025). While past work has also found disagreements in preferences and helpfulness (Balepur et al., 2024b; Mozannar et al., 2025), examining this across more domains would confirm this is a general issue of alignment.

Next, our agents (§4.1) and reward models (§4.2) have large disagreements with helpfulness to users (§6.2), but other models we did not test could have higher agreement. In our experiments, we focus on standard, strong baselines: ReACT based on GPT-4o (Yao et al., 2023) and six of the highest-ranked RMs on RewardBench (Lambert et al., 2025). In future work, it would be interesting to examine if RMs fine-tuned on helpfulness can generalize across domains, and if persona-based prompting with LLM agents improves simulations for predicting which responses best helps users (Hu and Collier, 2024).

Lastly, while we primarily use our collected feedback to study plan helpfulness, our dataset is rich, containing 4388 full traces of human tool use, subanswers, and feedback on 600 multi-step plans and questions (§3.2). While further analysis is beyond this paper's scope, future work could use our data to test how users and agents call tools differently (He et al., 2022), which steps of plans mislead users (Ji et al., 2024), and how plans can be personalized to assist users with diverse needs (Ley et al., 2010).

11 Ethical Considerations

While unlikely in our setting, LLMs can generate harmful responses (Xu et al., 2024), so before deploying LLM-generated plans to users, we manually check all of them to ensure they are all harmless (Appendix A.3). Further, when releasing our dataset of user preferences and plan executions, all users will be referred to by numerical IDs to mitigate any privacy concerns. In our study, all users were compensated with extra credit coursework or monetary compensation, and it was made clear to users before signing up that they would be part of a research study. Our entire project was approved by an Institutional Review Board (IRB), allowing us to fully address any potential risks of our study.

References

Rohan Ajwani, Shashidhar Reddy Javaji, Frank Rudzicz, and Zining Zhu. 2024. Llm-generated black-box explanations can be adversarially helpful. *arXiv preprint arXiv:2405.06800*.

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N Bennett, Kori Inkpen, et al. 2019. Guidelines for human-ai interaction. In *Proceedings of the 2019 chi conference on human factors in computing systems*, pages 1–13.

Anthropic. 2023. Meet claude. https://www.anthropic.com/product. Accessed: 2024-09-10.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Robert K Atkinson, Sharon J Derry, Alexander Renkl, and Donald Wortham. 2000. Learning from examples: Instructional principles from the worked examples research. *Review of educational research*, 70(2):181–214.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

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

Nishant Balepur, Jie Huang, and Kevin Chang. 2023. Expository text generation: Imitate, retrieve, paraphrase. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11896–11919, Singapore. Association for Computational Linguistics.

Nishant Balepur, Abhilasha Ravichander, and Rachel Rudinger. 2024a. Artifacts or abduction: How do LLMs answer multiple-choice questions without the question? In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10308–10330, Bangkok, Thailand. Association for Computational Linguistics.

Nishant Balepur, Matthew Shu, Alexander Hoyle, Alison Robey, Shi Feng, Seraphina Goldfarb-Tarrant, and Jordan Lee Boyd-Graber. 2024b. A SMART mnemonic sounds like "glue tonic": Mixing LLMs with student feedback to make mnemonic learning stick. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14202–14225, Miami, Florida, USA. Association for Computational Linguistics.

Nishant Balepur, Alexa Siu, Nedim Lipka, Franck Dernoncourt, Tong Sun, Jordan Lee Boyd-Graber, and Puneet Mathur. 2025. MoDS: Moderating a mixture of document speakers to summarize debatable queries in document collections. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 465–491, Albuquerque, New Mexico. Association for Computational Linguistics.

Hritik Bansal, John Dang, and Aditya Grover. 2024. Peering through preferences: Unraveling feedback acquisition for aligning large language models. In *The Twelfth International Conference on Learning Representations*.

Andrea J Bingham. 2023. From data management to actionable findings: A five-phase process of qualitative data analysis. *International journal of qualitative methods*, 22:16094069231183620.

Zheng Cai, Maosong Cao, Haojiong Chen, Kai Chen, Keyu Chen, Xin Chen, Xun Chen, Zehui Chen, Zhi Chen, Pei Chu, et al. 2024. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*.

Tuhin Chakrabarty, Philippe Laban, and Chien-Sheng Wu. 2025. Can ai writing be salvaged? mitigating idiosyncrasies and improving human-ai alignment in the writing process through edits. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, CHI '25, New York, NY, USA. Association for Computing Machinery.

Hongyu Chen and Seraphina Goldfarb-Tarrant. 2025. Safer or luckier? llms as safety evaluators are not robust to artifacts. *arXiv preprint arXiv:2503.09347*.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, et al. 2024. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

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

Colleen Cool, Nicholas Belkin, Ophir Frieder, and Paul Kantor. 1993. Characteristics of text affecting relevance judgments. In *National online meeting*, volume 14, pages 77–77. Learned Information (Europe) Ltd.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: boosting language models with scaled ai feedback. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Bram De Buyser. 2023. Goblin.tools. `https://goblin.tools`. A collection of AI-assisted tools designed to help with executive functioning, created and maintained by Bram De Buyser. Accessed April 20, 2025.

Nicolai Dorka. 2024. Quantile regression for distributional reward models in rlhf. *arXiv preprint arXiv:2409.10164*.

KJ Feng, Kevin Pu, Matt Latzke, Tal August, Pao Siangliulue, Jonathan Bragg, Daniel S Weld, Amy X Zhang, and Joseph Chee Chang. 2024. Cocoa: Co-planning and co-execution with ai agents. *arXiv preprint arXiv:2412.10999*.

Richard E Fikes and Nils J Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3-4):189–208.

Ronald A Fisher. 1922. The goodness of fit of regression formulae, and the distribution of regression coefficients. *Journal of the Royal Statistical Society*, pages 597–612.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Leo Gao, John Schulman, and Jacob Hilton. 2023. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning*, pages 10835–10866. PMLR.

Matt Gardner, William Merrill, Jesse Dodge, Matthew Peters, Alexis Ross, Sameer Singh, and Noah A. Smith. 2021. Competency problems: On finding and removing artifacts in language data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1801–1813, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Joyee Ghosh and David B Dunson. 2009. Default prior distributions and efficient posterior computation in bayesian factor analysis. *Journal of Computational and Graphical Statistics*, 18(2):306–320.

Daniel T. Gilbert and Timothy D. Wilson. 2006. *Miswanting: Some Problems in the Forecasting of Future Affective States*, page 550–564. Cambridge University Press.

Alex Goslen, Yeo Jin Kim, Jonathan Rowe, and James Lester. 2024. Llm-based student plan generation for adaptive scaffolding in game-based learning environments. *International journal of artificial intelligence in education*, pages 1–26.

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

Juho Hamari, Jonna Koivisto, and Harri Sarsa. 2014. Does gamification work?–a literature review of empirical studies on gamification. In *2014 47th Hawaii international conference on system sciences*, pages 3025–3034. Ieee.

Wanrong He, Andrew Mao, and Jordan Boyd-Graber. 2022. Cheater's bowl: Human vs. computer search strategies for open-domain qa. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 3627–3639.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Matthew D Hoffman, Andrew Gelman, et al. 2014. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623.

Tom Hosking, Phil Blunsom, and Max Bartolo. 2024. Human feedback is not gold standard. In *The Twelfth International Conference on Learning Representations*.

Tiancheng Hu and Nigel Collier. 2024. Quantifying the persona effect in LLM simulations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10289–10307, Bangkok, Thailand. Association for Computational Linguistics.

Shijue Huang, Wanjun Zhong, Jianqiao Lu, Qi Zhu, Jiahui Gao, Weiwen Liu, Yutai Hou, Xingshan Zeng, Yasheng Wang, Lifeng Shang, Xin Jiang, Ruifeng Xu, and Qun Liu. 2024a. Planning, creation, usage: Benchmarking LLMs for comprehensive tool utilization in real-world complex scenarios. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 4363–4400, Bangkok, Thailand. Association for Computational Linguistics.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. 2024b. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*.

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

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Boyuan Chen, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. 2023a. Beavertails: Towards improved safety alignment of LLM via a human-preference dataset. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Jiaming Ji, Tianyi Qiu, Boyuan Chen, Borong Zhang, Hantao Lou, Kaile Wang, Yawen Duan, Zhonghao He, Jiayi Zhou, Zhaowei Zhang, et al. 2023b. Ai alignment: A comprehensive survey. *arXiv preprint arXiv:2310.19852*.

Zhenlan Ji, Daoyuan Wu, Pingchuan Ma, Zongjie Li, and Shuai Wang. 2024. Testing and understanding erroneous planning in llm agents through synthesized user inputs. *arXiv preprint arXiv:2404.17833*.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2023. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Hannah Rose Kirk, Alexander Whitefield, Paul Röttger, Andrew Michael Bean, Katerina Margatina, Rafael Mosquera, Juan Manuel Ciro, Max Bartolo, Adina Williams, He He, Bertie Vidgen, and Scott A. Hale. 2024. The PRISM alignment dataset: What participatory, representative and individualised human feedback reveals about the subjective and multicultural alignment of large language models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

M. Koretsky, B. Brooks, and A. Higgins. 2016. Written justifications to multiple-choice concept questions during active learning in class. *International Journal of Science Education*, 38:1747 – 1765.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James Validad Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi.

2025. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1755–1797, Albuquerque, New Mexico. Association for Computational Linguistics.

Tobias Ley, Barbara Kump, and Cornelia Gerdenitsch. 2010. Scaffolding self-directed learning with personalized learning goal recommendations. In *User Modeling, Adaptation, and Personalization: 18th International Conference, UMAP 2010, Big Island, HI, USA, June 20-24, 2010. Proceedings 18*, pages 75–86. Springer.

Xinyu Li, Ruiyang Zhou, Zachary C Lipton, and Liu Leqi. 2024a. Personalized language modeling from personalized human feedback. *arXiv preprint arXiv:2402.05133*.

Zongxia Li, Ishani Mondal, Huy Nghiem, Yijun Liang, and Jordan Lee Boyd-Graber. 2024b. PEDANTS: Cheap but effective and interpretable answer equivalence. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9373–9398, Miami, Florida, USA. Association for Computational Linguistics.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in llms. *arXiv preprint arXiv:2410.18451*.

Naiming Liu, Zichao Wang, Richard Baraniuk, and Andrew Lan. 2022. Open-ended knowledge tracing for computer science education. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*.

Frederic Lord. 1952. A theory of test scores. *Psychometric monographs*.

Frederic M Lord. 2012. *Applications of item response theory to practical testing problems*. Routledge.

Jessica A Macaluso, Ramya R Beuford, and Scott H Fraundorf. 2022. Familiar strategies feel fluent: The role of study strategy familiarity in the misinterpreted-effort model of self-regulated learning. *Journal of Intelligence*, 10(4):83.

Chaitanya Malaviya, Joseph Chee Chang, Dan Roth, Mohit Iyyer, Mark Yatskar, and Kyle Lo. 2024. Contextualized evaluations: Taking the guesswork out of language model evaluations. *arXiv preprint arXiv:2411.07237*.

John McCarthy. 1959. Programs with common sense.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. 2023. Gaia: a benchmark for general ai assistants. In *The Twelfth International Conference on Learning Representations*.

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained atomic evaluation of factual precision in long form text generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12076–12100, Singapore. Association for Computational Linguistics.

Lester James V Miranda, Yizhong Wang, Yanai Elazar, Sachin Kumar, Valentina Pyatkin, Faeze Brahman, Noah A Smith, Hannaneh Hajishirzi, and Pradeep Dasigi. 2024. Hybrid preferences: Learning to route instances for human vs. ai feedback. *arXiv preprint arXiv:2410.19133*.

Hussein Mozannar, Valerie Chen, Mohammed Alsobay, Subhro Das, Sebastian Zhao, Dennis Wei, Manish Nagireddy, Prasanna Sattigeri, Ameet Talwalkar, and David Sontag. 2025. The realhumaneval: Evaluating large language models' abilities to support programmers. *Transactions on Machine Learning Research*. Expert Certification.

Allen Newell, Herbert Alexander Simon, et al. 1972. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, et al. 2024a. Gui agents: A survey. *arXiv preprint arXiv:2412.13501*.

Dang Nguyen, Viet Dac Lai, Seunghyun Yoon, Ryan A Rossi, Handong Zhao, Ruiyi Zhang, Puneet Mathur, Nedim Lipka, Yu Wang, Trung Bui, et al. 2024b. Dynasaur: Large language agents beyond predefined actions. *arXiv preprint arXiv:2411.01747*.

Donald A Norman. 2014. Some observations on mental models. In *Mental models*, pages 7–14. Psychology Press.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Siru Ouyang, Shuohang Wang, Yang Liu, Ming Zhong, Yizhu Jiao, Dan Iter, Reid Pryzant, Chenguang Zhu, Heng Ji, and Jiawei Han. 2023. The shifted and the overlooked: A task-oriented investigation of user-GPT interactions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2375–2393, Singapore. Association for Computational Linguistics.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pages 1–22.

Silviu Pitis, Ziang Xiao, Nicolas Le Roux, and Alessandro Sordoni. 2024. Improving context-aware preference modeling for language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Adam Poliak, Jason Naradowsky, Aparajita Haldar, Rachel Rudinger, and Benjamin Van Durme. 2018. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 180–191, New Orleans, Louisiana. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741.

Roger Ratcliff. 1978. A theory of memory retrieval. *Psychological review*, 85(2):59.

Brian Richards. 1987. Type/token ratios: What do they really tell us? *Journal of child language*, 14(2):201–209.

A. Roncone, Olivier Mangin, and B. Scassellati. 2017. Transparent role assignment and task allocation in human robot collaboration. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1014–1021.

Prisha Samdarshi, Mariam Mustafa, Anushka Kulkarni, Raven Rothkopf, Tuhin Chakrabarty, and Smaranda Muresan. 2024. Connecting the dots: Evaluating abstract reasoning capabilities of LLMs using the New York Times connections word game. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21219–21236, Miami, Florida, USA. Association for Computational Linguistics.

Michael Saxon, Ari Holtzman, Peter West, William Yang Wang, and Naomi Saphra. 2024a. Benchmarks as microscopes: A call for model metrology. In *First Conference on Language Modeling*.

Michael Saxon, Fatima Jahara, Mahsa Khoshnoodi, Yujie Lu, Aditya Sharma, and William Yang Wang. 2024b. Who evaluates the evaluations? objectively scoring text-to-image prompt coherence metrics with t2IScorescore (TS2). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Sander Schulhoff, Michael Ilie, Nishant Balepur, Konstantine Kahadze, Amanda Liu, Chenglei Si, Yinheng Li, Aayush Gupta, H Han, Sevien Schulhoff, et al. 2024. The prompt report: A systematic survey of prompting techniques. *arXiv preprint arXiv:2406.06608*, 5.

Norbert Schwarz. 2004. Metacognitive experiences in consumer judgment and decision making. *Journal of consumer psychology*, 14(4):332–348.

Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawjo. 2024. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14.

Yijia Shao, Yucheng Jiang, Theodore Kanell, Peter Xu, Omar Khattab, and Monica Lam. 2024. Assisting in writing Wikipedia-like articles from scratch with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6252–6278, Mexico City, Mexico. Association for Computational Linguistics.

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. Trial and error: Exploration-based trajectory optimization of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, Bangkok, Thailand. Association for Computational Linguistics.

Taylor Sorensen, Jared Moore, Jillian Fisher, Mitchell Gordon, Niloofar Mireshghallah, Christopher Michael Rytting, Andre Ye, Liwei Jiang, Ximing Lu, Nouha Dziri, Tim Althoff, and Yejin Choi. 2024. Position: a roadmap to pluralistic alignment. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021.

Yoo Yeon Sung, Maharshi Gor, Eve Fleisig, Ishani Mondal, and Jordan Lee Boyd-Graber. 2025. Is your benchmark truly adversarial? AdvScore: Evaluating human-grounded adversarialness. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 623–642, Albuquerque, New Mexico. Association for Computational Linguistics.

John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285.

John Sweller and Graham A Cooper. 1985. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and instruction*, 2(1):59–89.

Guiyao Tie, Zeli Zhao, Dingjie Song, Fuyang Wei, Rong Zhou, Yurou Dai, Wen Yin, Zhejian Yang, Jiangyue Yan, Yao Su, et al. 2025. A survey on post-training of large language models. *arXiv preprint arXiv:2503.06072*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Maomi Ueno and Yoshimitsu Miyazawa. 2018. Irt-based adaptive hints to scaffold learning in programming. *IEEE Transactions on Learning Technologies*, 11(4):415–428.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36:75993–76005.

Wim J Van der Linden. 2006. A lognormal model for response times on test items. *Journal of Educational and Behavioral Statistics*, 31(2):181–204.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. 2024a. Interpretable preferences via multi-objective reward modeling and mixture-of-experts. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10582–10592, Miami, Florida, USA. Association for Computational Linguistics.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. 2024b. Openhands: An open platform for ai software developers as generalist agents. In *The Thirteenth International Conference on Learning Representations*.

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024c. Helpsteer 2: Open-source dataset for training top-performing reward models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Hui Wei, Zihao Zhang, Shenghua He, Tian Xia, Shijia Pan, and Fei Liu. 2025. Plangenllms: A modern survey of llm planning capabilities. *arXiv preprint arXiv:2502.11221*.

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

Jiaxin Wen, Ruiqi Zhong, Pei Ke, Zhihong Shao, Hongning Wang, and Minlie Huang. 2024. Learning task decomposition to assist humans in competitive programming. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11700–11723.

Jiaxin Wen, Ruiqi Zhong, Akbir Khan, Ethan Perez, Jacob Steinhardt, Minlie Huang, Samuel R. Bowman, He He, and Shi Feng. 2025. Language models learn to mislead humans via RLHF. In *The Thirteenth International Conference on Learning Representations*.

Joel Wester, Sander De Jong, Henning Pohl, and Niels Van Berkel. 2024. Exploring people's perceptions of llm-generated advice. *Computers in Human Behavior: Artificial Humans*, 2(2):100072.

David Wood, Jerome S Bruner, and Gail Ross. 1976. The role of tutoring in problem solving. *Journal of child psychology and psychiatry*, 17(2):89–100.

William A Woods. 1973. Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pages 441–450.

Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024. A comprehensive study of jailbreak attack versus defense for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7432–7449, Bangkok, Thailand. Association for Computational Linguistics.

Rui Yang, Ruomeng Ding, Yong Lin, Huan Zhang, and Tong Zhang. 2024. Regularizing hidden states enables learning generalizable reward model for LLMs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V Chawla, and Xiangliang Zhang. 2025. Justice or prejudice? quantifying biases in LLM-as-a-judge. In *The Thirteenth International Conference on Learning Representations*.

Wenxuan Zhang, Philip Torr, Mohamed Elhoseiny, and Adel Bibi. 2025. Bi-factorial preference optimization: Balancing safety-helpfulness in language models. In *The Thirteenth International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Zexuan Zhong, Zhengxuan Wu, Christopher Manning, Christopher Potts, and Danqi Chen. 2023. MQuAKE: Assessing knowledge editing in language models via multi-hop questions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15686–15702, Singapore. Association for Computational Linguistics.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, and Ed H. Chi. 2023. Least-to-most prompting enables complex reasoning in large language models. In *The Eleventh International Conference on Learning Representations*.

A Appendix

A.1 "A Good Man is Hard to Find" Trivia

Our title "A Good Plan is Hard to Find" is a reference to Flannery O'Connor's short story "A Good Man is Hard to Find".[12] To honor the story, we provide many references to it throughout the paper. For readers familiar with the story, we encourage you to find all six references; solutions are in Appendix A.12. We felt these references were fitting given our paper's use of trivia question answering.

We also hope that attentive readers recognize our section titles are organized as a step-by-step plan!

A.2 Dataset Collection

When collecting datasets, our goal was to find math and trivia questions that are complex to solve without assistance (i.e. LLM plans). While datasets like GSM8k (Cobbe et al., 2021), MuSiQue (Trivedi et al., 2022), and MQuAKE (Zhong et al., 2023) contain multi-step questions, they have existed for several years, so LLMs have likely been trained or optimized for such tasks (Saxon et al., 2024a).

To fix these issues, we first have GPT-4o answer questions without plans (§6.1) and only use a subset the model answers incorrectly, indicating they are nontrivial to solve. Next, after producing plans for these questions (Appendix A.3), we filter out those where either plan has less than two steps, meaning that no multi-step decomposition is needed.

Upon manual inspection, we discover that a large proportion of questions are simply difficult due to ambiguity errors or incorrect labels (Sung et al., 2025). Thus, we run two rounds of quality control: 1) reviewing all questions and correcting/rewriting faulty ones so there is no ambiguity and all answers are correct; and 2) repeating the process in (1) to ensure there are no remaining question errors.

In total, we collect 150 math and 150 trivia questions, detailed in Table 3. Most trivia questions are based on MQuAKE, as we discovered MuSiQue often contained errors in the questions GPT-4o answered incorrectly, making them subpar. All questions are in English, have no personal information, and are in the intended use of the dataset creators.

A.3 Plan Generation

We use zero-shot prompting with LLMs to generate two plans for each math and trivia question; Prompt A.1 for math and Prompt A.2 for trivia. We spend around three hours manually engineering the prompts based on best practices (Schulhoff et al., 2024)—following an iterative process of designing a prompt, manually checking a subset of plans for any issues (i.e. no difference between plans, plans revealing answers), and tweaking the prompts to remedy any issues. We use a temperature of 0.7 for diversity and distribute the generation of plan pairs across four LLMs for the 150 math/trivia questions: 38 for GPT-4o (Hurst et al., 2024) and Claude-Opus (Anthropic, 2023); and 37 for Qwen-72B (Bai et al., 2023) and LLaMA3-405B (Grattafiori et al., 2024). We access GPT-4o and Claude via their official APIs, and Qwen and LLaMA via DeepInfra.[13]

A.4 Full `Planorama` Interface

Figure 1 shows the `Planorama` interface for trivia questions, but we also show the interface for math in Figure 7; the interface is identical, but web search is replaced by a calculator. Upon acceptance, we will provide a video demo of the interface.

A.5 Impact of Pairwise Comparisons

When running our `Planorama` user study, two-thirds of our users are assigned to an experimental group where they complete pairwise comparisons and then execute plans. The final third are assigned to a group where they do not complete a pairwise comparison, but have the option to switch between plans during plan execution. Our hope was that this group could form another feedback signal for predicting helpfulness, but we found the feature was often unused—leaving it for future exploration. We omit all instances where users decided to swap plans during plan execution in our experiments.

However, when users are assigned to the swap group and do not swap between plans, we can measure their accuracy and execution time to study the impact of completing or not completing pairwise comparisons. We find no clear differences between the distribution of average accuracies and execution times between these users (Figure 8), suggesting completing pairwise comparisons has little impact on problem-solving success in `Planorama`.

A.6 ReACT Implementation

We base our implementation of ReACT on the original framework (Yao et al., 2023), which iteratively runs a three-step protocol of reasoning, acting, and observing. The prompts we use for ReACT are in

[12] https://www.sparknotes.com/short-stories/a-good-man-is-hard-to-find/summary/

[13] https://deepinfra.com/

Prompt A.6 for math and Prompt A.7 for trivia. We use GPT-4o (Hurst et al., 2024) with 0 temperature. The model was allocated ~ 8 hours to run on CPU only. All results are reported from three runs.

A.7 Reward Model Implementation

All reward models were implemented according to their official Huggingface code.[14] Each model was allocated ~ 3 hours. Nemotron was implemented with NVIDIA's official API (CPU-only).[15] Other reward models use one NVIDIA:RTXA6000. All reward model predictions are based on a single run. Our prompt for the GPT-4o judge is in Prompt A.4 and for all other reward models in Prompt A.5.

A.8 Further Helpfulness Agreement Analysis

We now provide further analysis on the agreement of helpfulness signals (§6.2). In Tables 4 and 5, we extend our results in Table 1 to form full 10×10 agreement matrices on math and trivia, respectively, to capture RM agreement; RMs have high agreement with each other—often above 80%—so they learn similar notions of helpfulness, likely because they have similar training data (Lambert et al., 2025).

Further, to ensure our IRT metric (§5.1) is not the only reason helpfulness conflicts with preferences, we replicate the agreement analysis in §6.2 but using accuracy and execution time alone to identify which plan is more helpful. We also implement a simple average, which first uses average accuracy to denote helpful plans, and then execution time as a tie-break. Our findings are consistent in math (Figure 12) and trivia (Figure 13): preferences do not accurately predict which plans actually help users, regardless of whether helpfulness is defined by IRT, accuracy, execution time, or averages.

A.9 IRT Analysis

Using best practices of evaluating metrics (Saxon et al., 2024b; Shankar et al., 2024), we validate our IRT model by studying its convergence, assessing its generalization, interpreting discriminability and skill, ablating our design, and verifying difficulty correlates with accuracy and time as we expect.

A.9.1 Convergence

To ensure we have trained IRT for sufficient epochs, we first study its convergence. We find the model quickly converges to modeling the observed data (Figure 9) and reaches low R-hat values and high Effective Sample Sizes (Figure 10), so the model converges across our five chains. Further, our five chains perfectly agree on which plan in a pair is more helpful (§6.2), with Fleiss's $\kappa = 1.0$ (Fleiss, 1971), so IRT consistently discerns helpfulness.

A.9.2 Generalization

While we primarily use IRT to capture helpfulness, we still test its generalization, seeing how much it overfits to our data. To do this, we train IRT on the first 80% of every user's execution history in `Planorama`, and check how well it models the observed, held-out accuracy and execution time. The model has only minor drops in log-likelihood on accuracy ($\sim 5\%$), showing it effectively generalizes to user accuracy on new items (Figure 11). Log-likelihood does drop more on execution time ($\sim 75\%$), but this is to be expected, as predicting response time is generally difficult (Ratcliff, 1978).

A.9.3 Parameter Interpretations

While we primarily study difficulty β_i in our IRT model, the other parameters—item discriminability γ_i and player skill θ_j—can also give insights into how users interact with plans in `Planorama`.

Discriminability γ_i captures how well plans discern between low-skill and high-skill players. In Tables 6 and 7, we show plans with the highest gap in discriminability for math and trivia, respectively. In math, plans with higher γ_i tend to be longer; it requires more skill to solve problems accurately and quickly with longer plans, likely because excess steps naturally slow players down. In trivia, plans with higher γ_i have unconventional and complex steps: asking users to search for timelines, use self-verification, and follow complex instructions like "Ascertain", so only stronger problem-solvers can handle this more difficult level of guidance.

To ensure player skill θ_j correlates with downstream task success in `Planorama`, we plot each θ_j for player j against their average accuracy, execution time, and number of questions seen. As expected, players with higher θ_j tend to be more accurate with lower execution time. Further, players with higher θ_j tend to answer more questions, aligning with our results in §6.1, confirming that users become more successful problem-solvers as they keep interacting with plans in `Planorama`.

[14] https://huggingface.co/spaces/allenai/reward-bench

[15] https://build.nvidia.com/nvidia/llama-3_1-nemotron-70b-reward

A.9.4 Ablations

We ablate two key parts of our model: 1) modeling player skills (Eq. 1); and 2) using base time μ (Eq. 7). Both steps improve the prediction of our observed data (Figure 15), with the removal of (1) having the largest drop—showing the need for capturing individual skill to measure helpfulness.

A.9.5 Helpfulness Interpretation

Next, we check IRT helpfulness scores (i.e. negative difficulty) behave as intended. As true helpfulness increases, the average accuracy of players on plans tends to rise and average log-time tends to drop (Figure 16), matching our expectation: helpful plans imply accurate, efficient problem-solving (Huang et al., 2024a). In Appendix A.9, we further test our model's convergence and generalization.

A.10 Qualitative Trace Analysis Details

To explain how we inferred each trace error type in §7.3, we show examples for each type in math:

- **Step Error:** A question asks for the average number of branches per foot in several trees. The plan tells the user to incorrectly divide the average number of branches by the average height, but this gives the ratio of averages, not the average ratio as the question intended.
- **Ambiguous:** One plan step says "Subtract the number of spots on Jean's upper torso from the result of Step 1 to find the spots on her sides" which is meant to convey two computations: "1) Subtract the spots"; and "2) find the spots on her sides", but users interpreted "to find" in the step as being the same computation.
- **Execution Error:** One plan asks users to find "the cost for Jessica's bracelets by multiplying the number of letters in her name by the cost per bracelet", but the user did $6(2) = 12$, instead of $7(2) = 14$, as "Jessica" has 7 letters.
- **Ignored:** One user left all subanswers blank and did not use the calculator (i.e. did calculations in their head), making it impossible to diagnose where they erred in the trace.
- **Mistake:** GSM8k answers must be rounded to the nearest integer. One user got the answer 81.78, but instead of rounding to 82 (the right answer), they submitted 81.78, leading to an incorrect final response. The user immediately fixed this mistake on their next attempt.

A.11 User Study Compensation Details

Part of our compensation for this study was in the form of university extra credit, which if not properly handled, could pose undue pressure on low-performing students. To address this, we discussed guidelines with the university professors to ensure students were not coerced into participating in our study. Concretely, we provided an alternative assignment of equal difficulty (a coding assignment for a CS class, a reading quiz in a visualization class) to ensure participants could still obtain extra credit if they did not want to participate in our study. Further, we note that the extra credit assignment only required students to answer 25 questions. For most students, this took $\sim$30 minutes; the assignment was also administered over five weeks, including a break period during the Spring term, providing students ample time to complete the study.

Regarding monetary compensation, we explicitly set the maximum time per question in our study to 180 seconds, so even if a user failed to answer every question in our study, they would still be compensated at a rate of 20 USD/hr—well above our region's minimum wage. The lowest average execution time of any user in our study was 168 seconds, so this user obtained 21.43 USD/hr.

A.12 "A Good Man is Hard to Find" Answers

If you have already looked for (or found) our references to "A Good Man is Hard to Find" by Flannery O'Connor, this section reveals all six of them.

The story concerns a family taking a road trip to Florida (subtly alluded to with the term "Drive" in §6.1) and after getting into a car accident, they meet an escaped criminal named The Misfit (thus, "Escape" in §6.2 and "Misfit Plans" in §7). This dialogue has many notable quotes, like "*She would of been a good woman ... if it had been somebody there to shoot her every minute of her life*" (mirrored in §7.2) and the last line of "*It's no real pleasure in life.*" (mirrored in §7.3). Lastly, as a silly piece of trivia,[16] O'Connor details a watermelon with the initials "E.A.T." carved into it (reflected in §7.1). We would be impressed if you got them all!

[16] https://www.naqt.com/

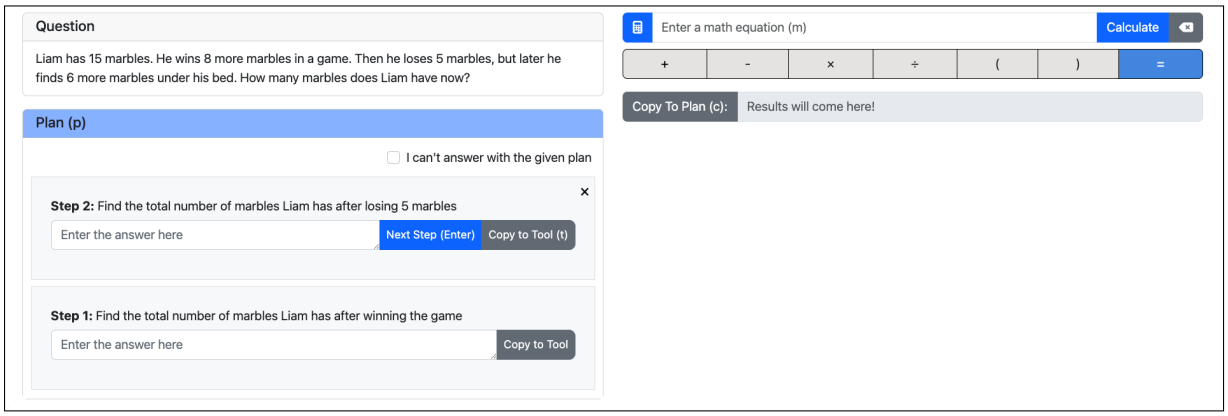

Figure 7: Overview of the Planorama interface for answering math questions. The interface mirrors Figure 1, but users have access to a calculator (right) rather than search.

	# q	Source(s)	Avg Words Per q	# Steps / p	Avg Executions / p	Avg Comparisons / p
Math	150	GSM8k (150)	50.48	3.11	8.99	9.91
Trivia	150	MuSiQue (10), MQuAKE (140)	18.39	2.85	7.86	9.91

Table 3: Summary of the Planorama dataset. We use 150 math and trivia questions mainly from GSM8k and MQuAKE. Our questions are supported by multi-step plans, typically 2-3 steps with high-level guidance for answering the question. We collect rich feedback from our users, with an average of 8.99 and 7.86 execution traces per plan, and 9.91 pairwise comparisons per plan.

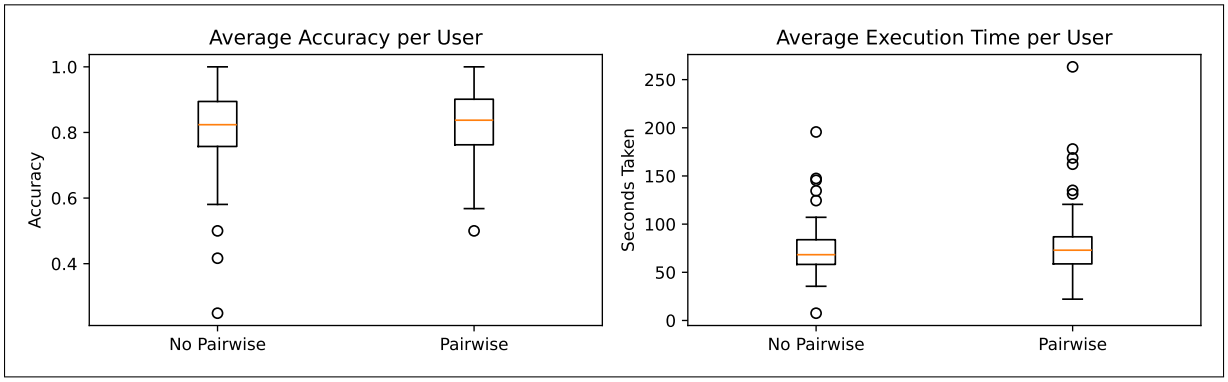

Figure 8: We find little differences in the distribution of average accuracies and execution times between users who complete and do not complete pairwise comparisons, suggesting it does not have much impact on problem-solving success.

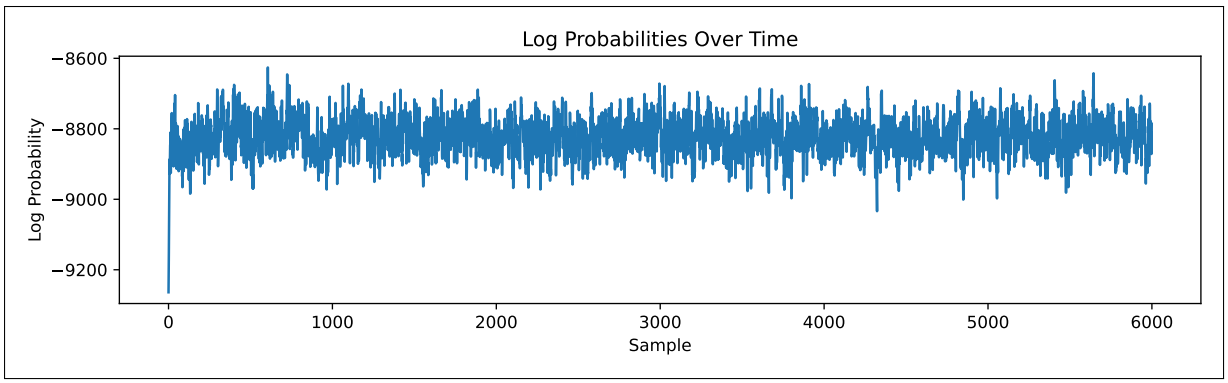

Figure 9: Plot of our IRT model's log-probability of predicting observed data. Our model quickly converges after a few samples, showing it adequately fits to our observed accuracy and execution time.

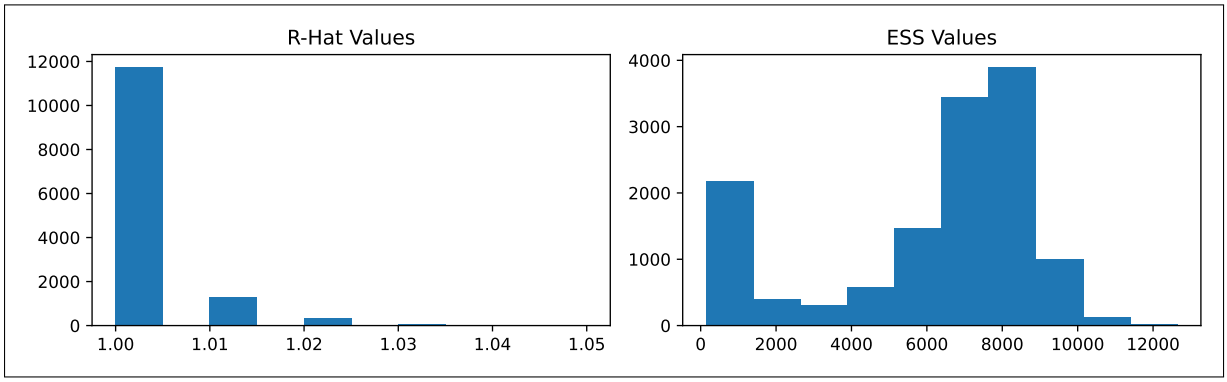

Figure 10: Distribution of R-hat and Effective Sample Size (ESS) values for our IRT model. R-Hat is always under 1.05 and the majority of ESS's are in the thousands, indicating strong convergence across our five chains.

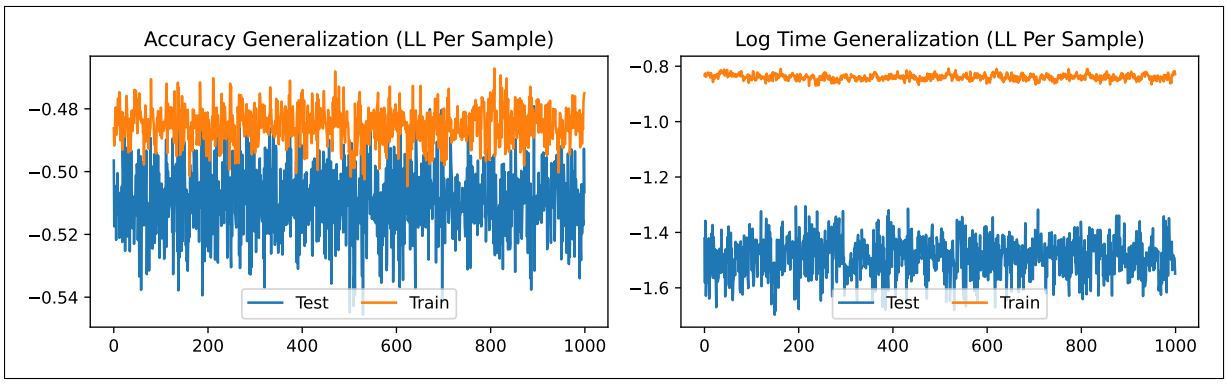

Figure 11: Generalization of IRT when predicting accuracy and execution time after being trained on the first 80% of the user's interactions in `Planorama`. The model effectively generalizes to predict accuracy, but struggles more with execution time.

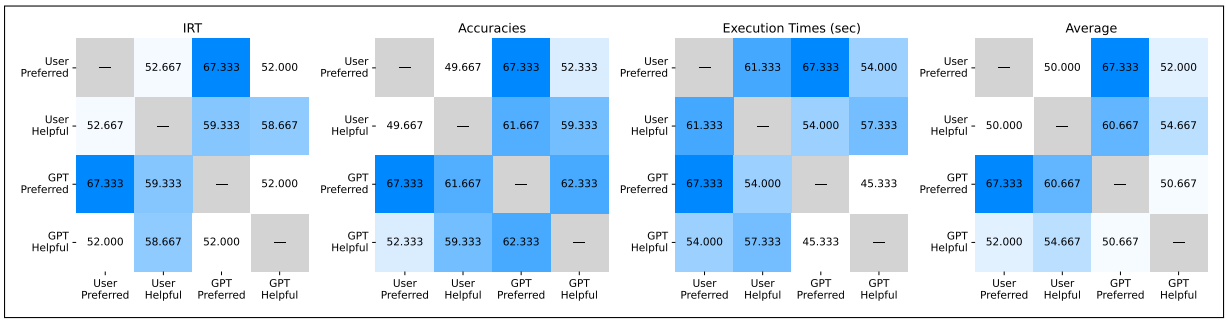

Figure 12: User/model perceived and true helpfulness agreement on **math** (Table 1, left) based on whether IRT, accuracy, time, or average accuracy/time dictates which plan is helpful. In every case, nothing accurately predicts what is truly helpful for users.

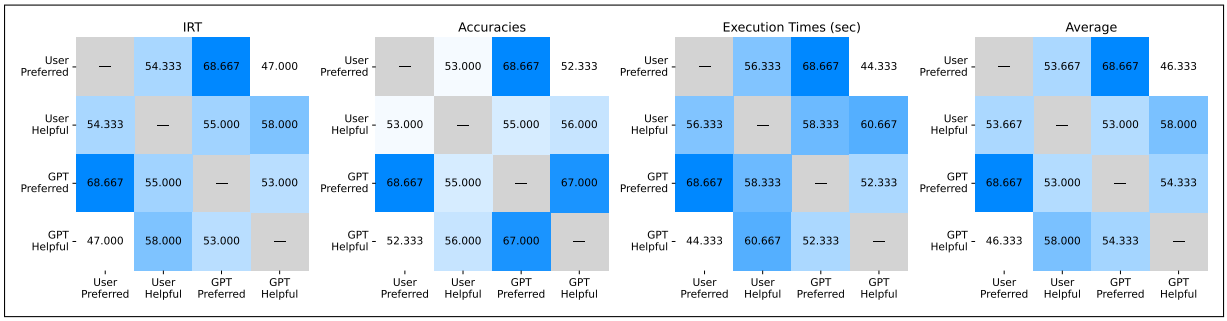

Figure 13: User/model perceived and true helpfulness agreement on **trivia** (Table 1, right) based on whether IRT, accuracy, time, or average accuracy/time dictates which plan is helpful. In every case, nothing accurately predicts what is truly helpful for users.

Figure 14: Correlation between player skill and `Planorama` interactions. Players with higher skill are typically more accurate, need less excecution time, and attempt more questions, aligning with our intuition.

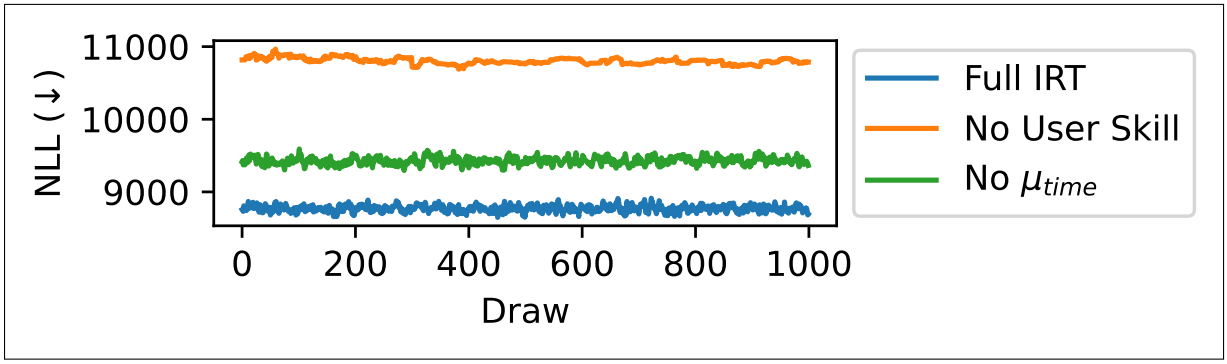

Figure 15: Ablating player skill (Eq. 1) and mean time priors (Eq. 7) degrade IRT's observed data predictions (negative log-likelihood), showing both better model true plan helpfulness.

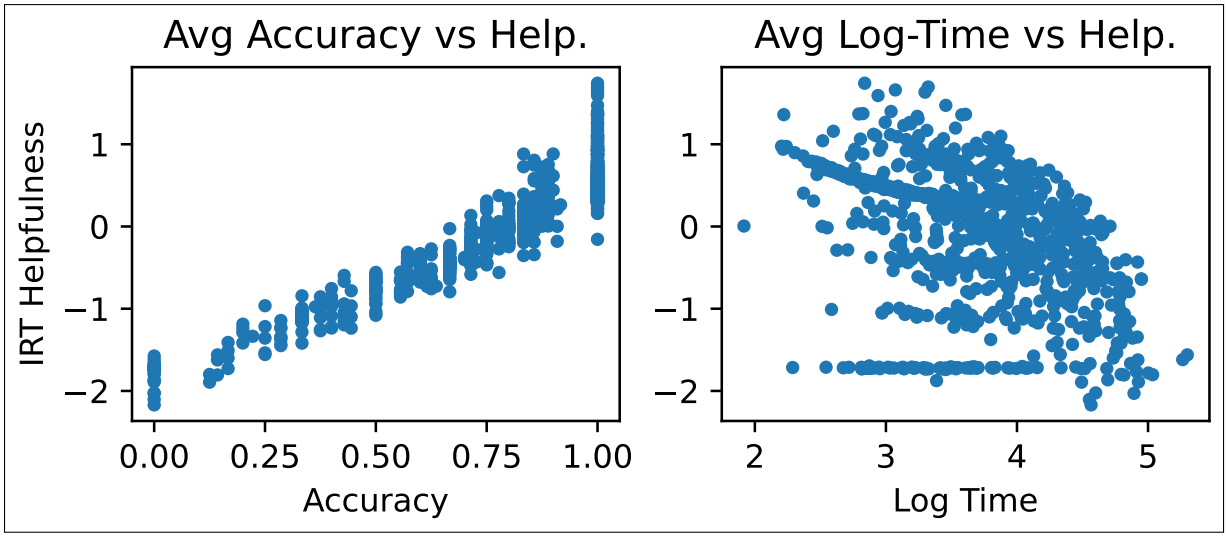

Figure 16: As IRT's true helpfulness metric rises, per-player accuracy rises and log-time drops, matching our expectations.

Proxy	User Judge	User Help	GPT Judge	GPT Help	QRM	GRM	Skywork	Nemotron	InternLM2	ArmoRM
User Judge	—	—	—	—	—	—	—	—	—	—
User Help	52.000	—	—	—	—	—	—	—	—	—
GPT Judge	67.333	58.667	—	—	—	—	—	—	—	—
GPT Help	55.333	57.333	52.667	—	—	—	—	—	—	—
QRM	60.000	56.000	72.000	42.667	—	—	—	—	—	—
GRM	53.333	54.667	66.000	38.667	81.333	—	—	—	—	—
Skywork	66.667	51.333	71.333	43.333	90.000	80.667	—	—	—	—
Nemotron	54.000	60.000	66.667	40.000	80.000	73.333	74.000	—	—	—
InternLM2	57.333	57.333	70.667	41.333	82.667	74.667	76.667	80.000	—	—
ArmoRM	56.667	56.667	68.000	39.333	78.000	83.333	77.333	80.667	79.333	—

Table 4: Full agreement analysis from Table 1 on math questions. As expected, RMs have high agreement with each other, showing that they all learn similar notions of helpfulness.

Proxy	User Judge	User Help	GPT Judge	GPT Help	QRM	GRM	Skywork	Nemotron	InternLM2	ArmoRM
User Judge	—	—	—	—	—	—	—	—	—	—
User Help	55.667	—	—	—	—	—	—	—	—	—
GPT Judge	68.667	56.333	—	—	—	—	—	—	—	—
GPT Help	49.000	62.667	54.333	—	—	—	—	—	—	—
QRM	65.667	51.333	56.333	36.667	—	—	—	—	—	—
GRM	64.333	51.333	57.000	40.667	84.000	—	—	—	—	—
Skywork	66.333	53.333	59.000	40.000	91.333	83.333	—	—	—	—
Nemotron	59.667	50.667	53.667	34.667	84.667	86.000	84.000	—	—	—
InternLM2	61.000	52.667	51.667	38.000	84.000	85.333	83.333	90.000	—	—
ArmoRM	59.667	52.000	53.000	42.667	76.667	84.667	77.333	78.667	82.000	—

Table 5: Full agreement analysis from Table 1 on trivia questions. As expected, RMs have high agreement with each other, showing that they all learn similar notions of helpfulness.

Question	High Discriminability	Low Discriminability
Carly is a pet groomer. Today, her task was trimming the four nails on each of the dogs' paws. She trimmed 164 nails, but three of the dogs had only three legs. How many dogs did Carly work on?	1. Calculate how many nails are trimmed for a dog with three legs. 2. Determine the total number of nails missing from all three-legged dogs. 3. Find out the total number of nails as if all dogs had four legs. 4. Calculate the total number of dogs by dividing total four-legged nails by nails per four-legged dog.	1. Estimate the number of four-legged dogs using the total number of nails. 2. Subtract the number of three-legged dogs to find four-legged dogs. 3. Add the three-legged dogs back to get the total number of dogs.
Two sisters, Elizabeth and Margareth, bought beads. Elizabeth bought 1 pack of red and 2 packs of clear beads, while Margareth bought 3 packs of blue and 4 packs of red beads. How many more beads does one sister have than the other, if each pack contains 20 beads?	1. Start by calculating the total number of beads in one pack of red beads. 2. Use that to calculate the total for Elizabeth's red and clear beads. 3. Calculate the total number of beads in Margareth's blue packs. 4. Calculate the total number of beads in Margareth's red packs. 5. Calculate the total beads for Margareth and subtract Elizabeth's total beads to find the difference.	1. Figure out how many beads Elizabeth bought by multiplying number of packs by beads per pack. 2. Do the same for Margareth. 3. Find the difference by subtracting the smaller total from the larger one.
Parker wants to find out what the average percentage of kernels that pop in a bag is. In the first bag he makes, 60 kernels pop and the bag has 75 kernels. In the second bag, 42 kernels pop and there are 50 in the bag. In the final bag, 82 kernels pop and the bag has 100 kernels. What is the average percentage?	1. Find the percentage popped for the first bag. 2. Repeat for the second and third bags. 3. Add the three percentages. 4. Divide by 3 to find the average.	1. Find the total number of popped kernels across all bags. 2. Find the total number of kernels across all bags. 3. Divide the popped kernels by the total kernels and multiply by 100.

Table 6: Comparison of the three plan pairs in math questions with the highest gap in discriminability.

Question	High Discriminability	Low Discriminability
How many stars are on the flag of the country where the spouse of the performer of Wrecking Ball is a citizen of?	1. Identify a song by a known artist that matches Wrecking Ball's release timeline. 2. Determine the country of citizenship for the known spouse of the artist from Step 1. 3. Find the number of stars on the flag of the country from Step 2.	1. Find the performer of the song Wrecking Ball. 2. Identify the spouse of the performer from Step 1. 3. Determine the country of citizenship for the spouse from Step 2. 4. Find the number of stars on the flag of the country from Step 3.
Who built the castle named after the city with an institution that educated the author of Species Plantarum?	1. Find the name of the author of Species Plantarum. 2. Determine the institution that educated the author identified in Step 1. 3. Identify the city where the institution found in Step 2 is located. 4. Ascertain the builder of the castle that carries the name of the city found in Step 3.	1. Identify the city that is home to an institution where the author of Species Plantarum was educated. 2. Find the name of the castle that shares its name with the city identified in Step 1. 3. Determine the builder of the castle identified in Step 2.
What is the city did the spouse of Nicolae Ceaușescu move to after receiving elementary education?	1. Identify Nicolae Ceaușescu's spouse. 2. Find the city the spouse moved to after elementary education. 3. Check if the city in Step 2 is the same as the city where the spouse received elementary education. 4. If the cities are different, find the city the spouse moved to after receiving elementary education.	1. Find the city where Nicolae Ceaușescu's spouse went to elementary school. 2. Determine the city the spouse moved to after Step 1.

Table 7: Comparison of the three plan pairs in trivia questions with the highest gap in discriminability.

Prompt A.1: Math Plan Generation Prompt (§3.1)

I will give you a math question, and your goal is to come up with two diverse plans with instructions that can help someone answer the question. The plans should lead to the correct answer, but more importantly should be highly optimized for speed. Each step of the plan should not reveal the answer, intermediate computations, any specific numbers in the input question, or how to exactly perform any calculation. Plans should focus on basic computations rather than setting up complex equations. Each plan should be between 2 and 10 steps. Steps should be high-level, brief, clear, and not include details about how to compute any intermediate values.

Please format your output as a JSON dictionary with key "plan1" for the first plan and "plan2" for the second plan. The value for each key should be a list of strings with steps on how to answer the question. Each step should be prefixed by its number (e.g. "Step 1:"). Steps should reference outputs form preceding steps using the number of the step.

Remember, plans should be accurate but highly optimized for speed which should be achieved by minimizing the number of steps, keeping descriptions brief, and using shortcuts that can skip the traditional route of answering the question. The two plans can differ in their overall strategy, specificity, number of steps, and what intermediate information to compute.

Produce plans for the question: q

Prompt A.2: Trivia Plan Generation Prompt (§3.1)

I will give you a trivia question, and your goal is to come up with two diverse plans with instructions that can help someone answer the question. Do not reference any tools or sources that should be used. The plans should lead to the correct answer, but more importantly should be highly optimized for speed. Each step of the plan should instruct the user to find an intermediate answer. Plans should not reveal the answer or intermediate answers, and can only contain information in the input question. Each plan should be between 2 and 10 steps. Steps should be high-level, brief, self-contained, and cannot include extra information, entities, and knowledge that is not present in the input question.

Please format your output as a JSON dictionary with key "plan1" for the first plan and "plan2" for the second plan. The value for each key should be a list of strings with steps on how to answer the question. Each step should be prefixed by its number (e.g. "Step 1:"). Steps should reference outputs form preceding steps using the number of the step.

Remember, plans should be accurate but highly optimized for speed which should be achieved by minimizing the number of steps, keeping descriptions brief, and using shortcuts that can skip the traditional route of answering the question. The two plans can differ in their overall strategy, specificity, number of steps, and what intermediate information to search for.

Produce plans for the question: q

Prompt A.3: GPT-4o Direct Answer Prompt (§6.1)

Answer the following question. Give just the answer and no explanation. Format your final answer as "Answer: [insert generated answer]"

Question: q

Prompt A.4: GPT-4o Judge Prompt (§4.2)

You will be given a question and two step-by-step plans that could help a human user answer the question (Plan A and Plan B). Your goal is to determine which plan would help a human user answer the question more accurately and quickly. Respond with just the letter of the plan.
Question: q
Plan A: p_A
Plan B: p_B
More Helpful Plan:

Prompt A.5: Reward Model Prompt (§4.2)

<user>Generate a plan to help me answer this question accurately and quickly: q<user>
<assistant>p<assistant>

Prompt A.6: ReACT Math Prompt (§4.1)

You will be a given a question and series of previous steps, actions, and thoughts, and your task is to generate the next Thought or Action that will lead you to the correct answer of the current step. Thoughts contain reasoning chains that help you decide which action you should call, while Actions are tool calls that can provide external information. The tools you have access to are:
- CALCULATE: Given an input equation, returns the result when evaluating the expression
- SUBMIT_STEP: Submit an answer to the step

All outputs from tool calls will be provided as Observations. You must call SUBMIT_STEP to answer each step, not to answer the final question. Below is an example of a full reasoning trace:
—
Question: Liam has 15 marbles. He wins 8 more marbles in a game. Then he loses 5 marbles, but later he finds 6 more marbles under his bed. How many marbles does Liam have now?
—
Step 1: Find the total number of marbles Liam has after winning the game
Thought: To find the total number of marbles Liam has after winning the game, we must add his initial 15 marbles with the 8 marbles he won after the game
Action: CALCULATE(15 + 8)
Observation: 23

Thought: We now have the number of marbles Liam has after winning the game, so we can submit 23 as the answer to this step
Action: SUBMIT_STEP(23)
Answer to Step 1: 23
—
Step 2: Find the total number of marbles Liam has after losing 5 marbles
Thought: To get the number of marbles Liam has after losing 5 marbles, we must subtract 5 from the 23 marbles in Step 1
Action: CALCULATE(23 - 5)
Observation: 18

Thought: We now have the number of marbles Liam has after he loses 5 of them, so we can submit 18 as the answer to this step
Action: SUBMIT_STEP(18)
Answer to Step 2: 18
—
Step 3: Find the final number of marbles Liam ends up with after finding more marbles under his bed
Thought: The final number of marbles Liam has is the 18 marbles from Step 2 plus the 6 more marbles Liam finds under his bed
Action: CALCULATE(18 + 6)
Observation: 24

Thought: We now have the final number of marbles Liam ends up with, so we can submit 24 as the answer to this step
Action: SUBMIT_STEP(24)
Answer to Step 3: 24

Now, generate the next Thought or Action for the following question: q
—

Prompt A.7: ReACT Trivia Prompt (§4.1)

You will be a given a question and series of previous steps, actions, and thoughts, and your task is to generate the next Thought or Action that will lead you to the correct answer of the current step. Thoughts contain reasoning chains that help you decide which action you should call, while Actions are tool calls that can provide external information. The tools you have access to are:
- SEARCH: Given an input search query, returns the title, first paragraph, and most similar context within a relevant Wikipedia page
- SUBMIT_STEP: Submit an answer to the step

All outputs from tool calls will be provided as Observations. You must call SUBMIT_STEP to answer each step, not to answer the final question. Below is an example of a full reasoning trace:
—
Question: What is the capital of the state that contains the tallest mountain in the United States?
—
Step 1: Find the tallest mountain in the United States
Thought: To find the tallest mountain in the United States, we can search for the right Wikipedia page with this query
Action: SEARCH(tallest mountain United States)
Observation: <Title>List of mountain peaks of the United States</Title><First Paragraph>This article comprises three sortable tables of major mountain peaks of the United States of America. </First Paragraph><Selected Content>The summit of a mountain or hill may be measured in three main ways: Download coordinates as: Of the 100 highest major summits of the United States, only Denali exceeds 6000 meters (19,685 feet) elevation...</Selected Content>
Action: SUBMIT_STEP(Mount Denali)
Answer to Step 1: Mount Denali
—
Step 2: Find the state that contains this mountain

Thought: To find the state with this mountain, we must find where Mount Denali is located
Action: SEARCH(Mount Denali location)
Observation: <Title>Denali</Title><First Paragraph>Denali (/de'na:li/), federally designated as Mount McKinley, is the highest mountain peak in North America, with a summit elevation of 20,310 feet (6,190m) above sea level. It is the tallest mountain in the world from base to peak on land, measuring 18,000ft (5,500m). With a topographic prominence of 20,194 feet (6,155m) and a topographic isolation of 4,621.1 miles (7,436.9km), Denali is the third most prominent and third-most isolated peak on Earth, after Mount Everest and Aconcagua. Located in the Alaska Range in the interior of the U.S. state of Alaska, Denali is the centerpiece of Denali National Park and Preserve. </First Paragraph><Selected Content>Denali (/de'na:li/), federally designated as Mount McKinley, is the highest mountain peak in North America, with a summit elevation of 20,310 feet (6,190m) above sea level...</Selected Content>

Thought: The last observation states that Denali is located in the U.S. state of Alaska, so we can submit Alaska as the answer to this step
Action: SUBMIT_STEP(Alaska)
Answer to Step 2: Alaska
—
Step 3: Find the capital of this state

Thought: To find the capital of this state, we must find the capital of Alaska, which can be found on its Wikipedia page
Action: SEARCH(Alaska capital)
Observation:<Title>Juneau, Alaska</Title><First Paragraph>Juneau ('dZu:noU i JOO-noh; Tlingit: Dzanti K'iheeni ['tsnthI k'I'hi:nI] transl. Base of the Flounder's River), officially the City and Borough of Juneau, is the capital of the U.S. state of Alaska, located along the Gastineau Channel and the Alaskan panhandle. Juneau was named the capital of Alaska in 1906, when the government of what was then the District of Alaska was moved from Sitka as dictated by the U.S. Congress in 1900. On July 1, 1970, the City of Juneau merged with the City of Douglas and the surrounding Greater Juneau Borough to form the current consolidated city-borough, which ranks as the second-largest municipality in the United States by area and is larger than either Rhode Island or Delaware. </First Paragraph><Selected Content>Atop the mountains is the Juneau Icefield, a large ice mass from which about 30 glaciers flow; two of them, the Mendenhall Glacier and the Lemon Creek Glacier, are visible from the local road system... </Selected Content>

Thought: The title of this page is Juneau, Alaska, so we can submit Juneau as the answer to this step
Action: SUBMIT_STEP(Juneau)
Answer to Step 3: Juneau
—
Now, generate the next Thought or Action for the following question: q
—