
How to Train Your Advisor: Steering Black-Box LLMs with ADVISOR MODELS

Parth Asawa*

UC Berkeley
pgasawa@berkeley.edu

Alan Zhu*

UC Berkeley
aczhu@berkeley.edu

Matei Zaharia
UC Berkeley

Alexandros G. Dimakis
UC Berkeley and Bespoke Labs

Joseph E. Gonzalez
UC Berkeley

Abstract

Foundation models are increasingly deployed as black-box services, where model weights cannot be modified and customization is limited to prompting. While static prompt optimization has shown promise, it produces a single fixed prompt that fails to adapt to different inputs, users, or environments. We introduce ADVISOR MODELS, lightweight parametric policies trained with reinforcement learning to reactively issue natural language steering instructions in-context to black-box models. The advisor is a second small model that sits between the input and the model, shaping behavior on a per-instance basis using reward signals from the environment. Across multiple domains involving reasoning and personalization, we show that ADVISOR MODELS outperform static prompt optimizers, discovering environment dynamics and improving downstream task performance. We also demonstrate the generalizability of advisors by transferring them across black-box models, as well as the framework’s ability to achieve specialization while retaining robustness to out-of-distribution inputs. Viewed more broadly, ADVISOR MODELS provide a learnable interface to black-box systems where the advisor acts as a parametric, environment-specific memory. We argue that dynamic optimization of black-box models via ADVISOR MODELS is a promising direction for enabling personalization and environment-adaptable AI with frontier-level capabilities.

1 Introduction

Frontier foundation models such as GPT-5 [OpenAI, 2025b] and Claude 4.1 [Anthropic, 2025] achieve state-of-the-art performance across a wide range of tasks, powered by strong reasoning capabilities. Yet, when customization or personalization is needed, options for models served via APIs remain limited. The primary tools are static prompt engineering or post-hoc filtering, but these methods are brittle because they produce a single, fixed instruction limited in size by the context length which cannot adapt to the nuances of different inputs, users, or contexts [Brown et al., 2020, Wang et al., 2023].

We propose ADVISOR MODELS, a framework for training a lightweight model to reactively steer a black-box model by generating input-specific advice (Figure 1). We challenge the assumption that advisor models must be stronger than the advisee but instead view advisors as having the ability to learn through experience and then transfer those lessons to a more powerful student model – that is unable (or unwilling) to learn directly. Crucially, advisor models can be optimized with reinforcement

*Equal contribution. Code available at: <https://github.com/az1326/advisor-models>

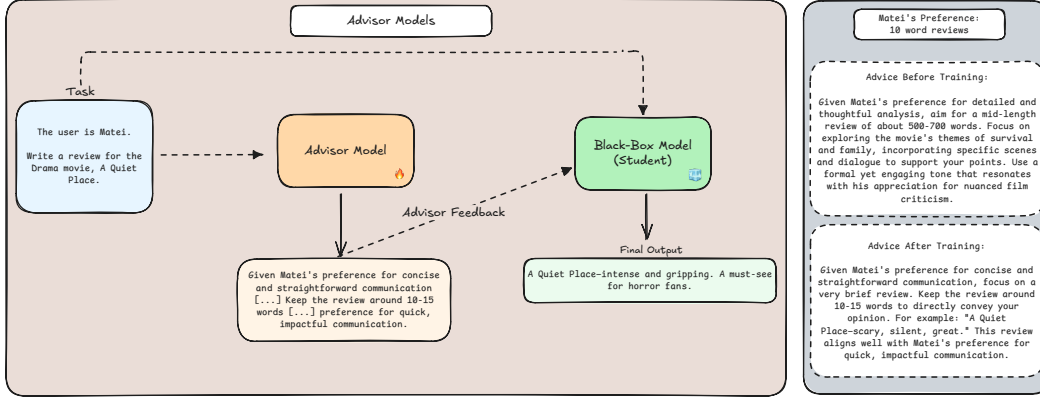


Figure 1: **Example of the ADVISOR MODELS workflow.** The input task is given first to the advisor model to generate advice. In this example, the task is personalization and the advisor feedback is specific to the user (i.e., Matei). The input task and the advice are then given to a strong black-box model (e.g. GPT-5) to generate the final output. The advisor learns via RL the preferences of the user. Crucially, advisor model RL training can be done with only API access to the black-box model.

learning, using task-specific rewards derived from the student model’s final outputs. This process enables the compound system to learn and adapt without any access to the student model’s parameters or gradients, making it an effective method for steering or personalizing black-box models. The *learning to advise* formulation re-frames prompting from a static search problem into learning a policy that generates custom advice for every instance.

We demonstrate ADVISOR MODELS in domains where latent preferences or rules vary across contexts—for example, user-specific writing or personalized learning material generation [Singh et al., 2025]. Our experiments show that ADVISOR MODELS efficiently adapt black-box models while current state-of-the-art static prompt optimizations fail to improve over unoptimized baselines. In particular, ADVISOR MODELS perfectly or near-perfectly learns user-specific rules (achieving 94-100% of possible reward), while static prompt optimizers fail to improve over the baseline (40-60% reward). On reasoning tasks requiring domain-specific knowledge or expertise, ADVISOR MODELS also demonstrates significant gains, improving performance on the MTOB low-resource translation task [Tanzer et al., 2024] from 28.1 to 43.7 and accuracy on the complex rule following RuleArena taxes task from 56% to 64%.

ADVISOR MODELS’s modular design also yields significant benefits in transferability and robustness. We show that trained advisors can be transferred between different black-box models with no loss in performance. Furthermore, because the black-box model remains unchanged, the system’s out-of-domain capabilities are fully preserved, which we confirm by showing no statistical difference in math accuracy when using advisors trained on entirely unrelated tasks, limiting problems such as catastrophic forgetting.

More broadly, we argue that ADVISOR MODELS offer a new paradigm for openly optimizing black-box models: learning to control them through interpretable, trainable decisions informed by environmental feedback, while preserving their powerful reasoning capabilities. This perspective also connects advisor models to discussions on memory in AI systems. Advisors can be viewed as a compact parametric memory that captures user or environment-specific latents and emits control signals at inference time, without storing histories or direct weight updates. We highlight these as important future directions.

In this paper, we explore the potential of ADVISOR MODELS as a new paradigm for controlling black-box systems. Our contributions are:

1. We introduce the **ADVISOR MODELS framework**, a novel method for reactively steering frozen, black-box models using trainable policies.
2. We demonstrate its effectiveness in settings requiring **domain adaptation or personalization**, where it can significantly outperforms static baselines.

3. We demonstrate the ability to **transfer advisors** across black-box models and show that the framework leads to **robustness to out-of-distribution inputs** even with a specialized model.

2 Related Work

Parameter-efficient Finetuning. ADVISOR MODELS may be viewed as a parameter-efficient finetuning technique, wherein the relatively small advisor model is tuned rather than the large student model. There is extensive literature on customizing large pre-trained models ranging from full finetuning to parameter-efficient methods such as adapters [Houlsby et al., 2019], LoRA [Hu et al., 2022], QLoRA [Dettmers et al., 2023]. However, a key distinction is that all these approaches require access to model weights, and so are inapplicable to the current generation of state-of-the-art frontier systems, which are only accessible through restricted APIs. As a result, the standard toolkit of fine-tuning and architectural augmentation cannot be used, motivating new research into API-only guiding techniques that rely instead on learned prompt policies [Zhou et al., 2022], or routing strategies around the black-box model [Chen et al., 2023]. Unlike previous techniques that learn solely in prompt space, our work trains an advisor with weight updates, allowing for more flexible and dynamic fine-tuning.

Static Prompt Optimization. Some prior work focused on automatically discovering fixed prompts for black-box models, using gradient-free search, evolutionary algorithms, or reinforcement learning to improve task performance across datasets [Khatab et al., 2022, 2024, Opsahl-Ong et al., 2024, Zhou et al., 2022, Yang et al., 2024b, Agarwal et al., 2024, Fernando et al., 2023, Agrawal et al., 2025]. While effective in some settings, such methods typically yield a single static prompt that must be reused for all instances of a task. In contrast, our work departs from this paradigm by training a lightweight advisor to reactively produce instance-specific natural language advice. This advice is then injected in-context into the black-box model, allowing adaptation to individual prompts and hidden latents rather than committing to one static prompt.

3 ADVISOR MODELS

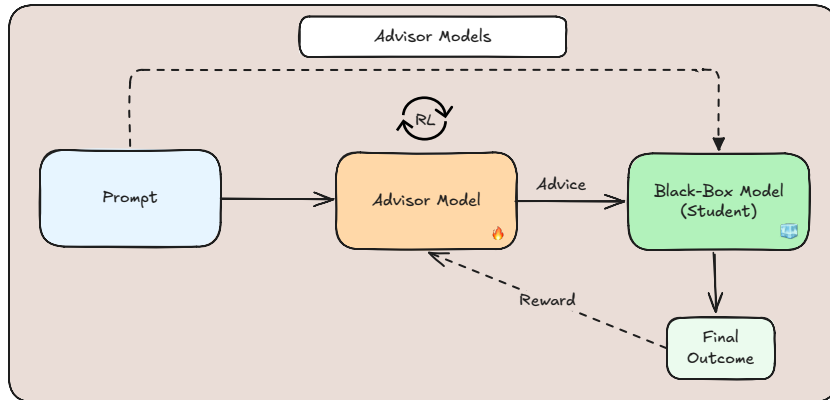


Figure 2: **ADVISOR MODELS combine open-source models with black box models.** ADVISOR MODELS generate instance-specific advice that is injected in-context to steer a black-box model. Rewards from the environment of the final output are used in GRPO to reinforce effective advice.

We introduce ADVISOR MODELS, a framework training a lightweight policy to steer black-box foundation models through natural language advice. The advisor sits between the user input and the black-box model, generating instance-specific guidance that is injected in-context for the black-box model (Figure 2).

Initialization consists of specifying prompts for the advisor and black-box model. The template to the advisor includes the input as well as any additional priors that might help guide the optimization process, if available. For example, if it is known that advice relevant to response length will be most helpful, the advisor prompt might contain instructions to generate advice related to length in order to guide the optimization. We discuss the effects of advisor prompt initialization in Section 4.1.2. The

template to the black-box model should simply instruct the model to answer the original input while considering the advisor’s output if applicable.

Training proceeds via reinforcement learning: the advisor samples candidate advice, the black-box model produces outputs conditioned on this advice, and a task-specific reward is computed from the output. We use Group Relative Policy Optimization (GRPO; Shao et al. [2024]) to update the advisor based only on observed rewards. Integrating ADVISOR MODELS into any RL training framework is as simple as modifying a step in the environment to make a generation call to the black-box model with the advisor’s output (policy’s action) before evaluating the reward on the final output.

The ADVISOR MODELS design turns prompt engineering into an RL problem, where the advisor learns which advice reliably elicits better performance. Unlike static prompt optimization, the advisor adapts the black-box model on a per-instance basis. No gradient access is needed from the black-box model, allowing for the full leveraging of frontier API-based models. As open-source models may also serve as the advised model, in this work we will use “black-box” and “student” model interchangeably.

The benefits of ADVISOR MODELS fall under two major axes that limited prior work explores:

1. **Reactive, instance-specific optimization.** A learned policy generates natural language advice conditioned to each input, in contrast to static prompting methods that provide a single fixed context.
2. **Leveraging black-box capabilities without modification.** The advisor is updated in gradient space while the black-box model remains unchanged. This produces a **robust system** that preserves black-box reasoning capabilities and avoids post-training risks such as catastrophic forgetting.

Beyond these core benefits, advisor models can also be viewed as a compact, trainable interface for parametrically capturing environment-specific latents. This perspective connects ADVISOR MODELS to ongoing discussions on personalization and memory in compound AI systems.

4 Evaluations

We evaluate ADVISOR MODELS in domains designed to highlight two desiderata: (1) the ability to leverage the strength of black-box models (e.g., reasoning, creative writing), and (2) the ability to adapt to latents that must be discovered through feedback. We believe that current static prompt optimization methods would struggle to steer models in different ways based on inputs in these settings. We then extend our experiments to complex reasoning settings to investigate the behavior of ADVISOR MODELS when improving capabilities is the primary objective.

Through our experiments, we aim to answer the following questions:

1. Can ADVISOR MODELS learn unstated environment latents where static optimizers fail?
2. How do ADVISOR MODELS behave on complex reasoning tasks where the primary objective is capability improvement?
3. Is the modular advisor-student architecture of ADVISOR MODELS robust to out-of-distribution inputs?

4.1 RQ1: Can ADVISOR MODELS Learn Unstated Latents?

Review Writing. Our first domain tests whether advisors can uncover user-specific hidden preferences in open-ended generation tasks. We use a review-writing setup where the system must produce book/movie/TV show reviews for named users. We reasonably assume a frontier black-box model’s ability to write high-quality reviews cannot be matched by the much smaller advisor model. Each user is associated with a latent preference that is not stated in the prompt, specifically along the axis of review length or reading level. Review prompts are drawn from the FSPO dataset [Singh et al., 2025], splitting the 500 prompts into 450 training and 50 evaluation.

We study two variants. In **review length**, each user has a preferred review length between 10 and 1000 words. The reward is continuous:

$$\text{Reward} = \frac{1}{1 + \frac{|\text{Review Length} - \text{Preferred Length}|}{\text{Preferred Length}}},$$

which equals 1 if the review exactly matches the preference and decays towards 0 as the magnitude of the deviation grows. In **review level**, users prefer reviews at five different reading levels (e.g., elementary school, high school, college professor, etc). In this setting, reward is binary, using GPT-5 mini [OpenAI, 2025b] as a judge to determine whether the review matches the specified level.

Math Solutions. A limitation of the review domains is that the quality of the generated reviews (independent of preference alignment) is subjective. A domain where quality is more objectively verifiable is mathematical problem solving, where correct solutions are of higher quality. We thus introduce the Math Solutions setting, where we ask the student model to produce teaching material for individual students in the form of worked solutions to math problems, sourced from the MATH-500 benchmark [Lightman et al., 2023]. Each student either prefers or dislikes questions posed to the reader and either prefers or dislikes presentation of multiple methods, making for 4 total distinct preferences. We split the MATH-500 benchmark into 400 training and 100 evaluation problems.

Compared to the review writing domains, math solutions both requires the advisor to learn multiple axes of preference at once and allows us to verify that the student model’s quality is not degraded by the ADVISOR MODELS system.

Environmental reward is discrete, with 0 reward given if no preference is met, 0.4 if only one is met, and 1 if both preferences are met. We use GPT-4.1 mini [OpenAI, 2025a] as a judge to determine whether or not a criteria is met.

Models and Baselines. We use GPT-4o mini [OpenAI, 2024] as the black-box model and Qwen2.5-7B-Instruct [Qwen Team, 2024] as the advisor model in all domains. Further training details are in Appendix A.

We compare the performance of our ADVISOR MODELS-trained pipelines against an un-optimized baseline and the state-of-the-art static prompt optimizer GEPA [Agrawal et al., 2025]. For fair comparison, GEPA was allowed to access the reward function an equivalent number of times to ADVISOR MODELS training.

4.1.1 Results

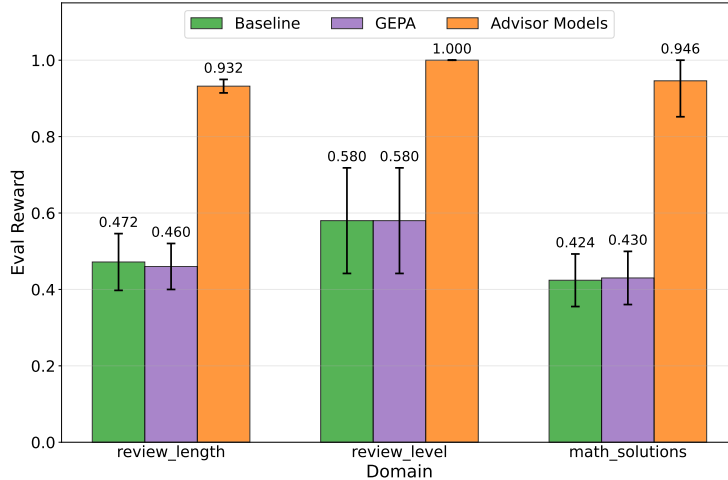


Figure 3: **ADVISOR MODELS outperform various baselines in hidden latent tasks.** ADVISOR MODELS can learn tasks almost perfectly in settings where state-of-the-art static prompt optimization methods cannot learn at all. 95% confidence intervals are provided.

We present the final performances of ADVISOR MODELS and GEPA on the three hidden latent domains in Figure 3. Across all three tasks, GEPA achieves a performance that is within statistical

uncertainty of the un-optimized baseline: about 0.47 reward for review length, 58% accuracy for review level, and 0.42 reward for math solutions.

On the other hand, ADVISOR MODELS achieves impressive results, perfectly solving the learning problem with 0.96 reward out of a maximum 1.0 for review length, 100% accuracy for review level, and 0.946 reward out of a maximum 1.0 for math solutions. This demonstrates an ability for ADVISOR MODELS to effectively learn in hidden latent settings where current static optimizers cannot.

Examples of generated advice from the start and end of training in the review length setting are presented in Appendix C. Qualitatively, early advisor outputs hallucinate user preferences, but by the end of training they converge toward the true latent. These findings suggest that dynamic, instance-specific advice allows black-box models to be effectively optimized in settings where static prompt optimization struggle.

4.1.2 Ablation: Advisor Prompt Initialization

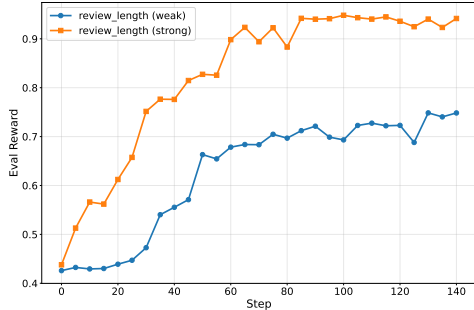


Figure 4: **Strong initialization leads to faster learning.** ADVISOR MODELS learning curve on the review length domain, under strong and weak initialization, 5 epochs. Generated advice improves in both settings but training with strong initialization leads to better final performance.

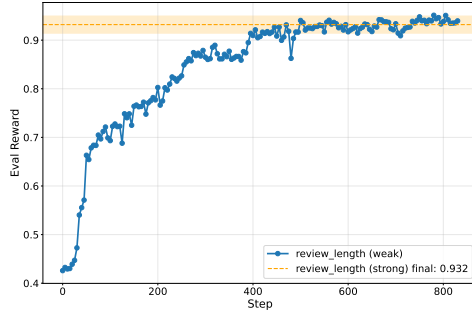


Figure 5: **Weak initialization can eventually learn.** ADVISOR MODELS learning curve on the review length domain with weak initialization for 30 epochs, with strong initialization performance after 5 epochs provided as reference. Performance eventually reaches the reference after extended training.

The advisor prompt initializations we used were relatively strong, including guidance to help the model sample relevant advice (e.g., “consider the review length preference of the target person”). We believe this initialization to be reasonable as practitioners will either have priors for desired axes of separation, or can provide a list of candidate axes. Nevertheless, we investigate the efficacy of ADVISOR MODELS under weak initialization, when no such guidance is given and the advisor is simply asked to provide advice. We thus re-train ADVISOR MODELS with a weaker advisor initialization prompt. The strong and weak initializations (as well as the student initialization) are presented in Appendix D.

We find that under weak initialization, ADVISOR MODELS still learn, achieving 0.749 reward (Figure 4). This falls below the near-perfect performance achieved by ADVISOR MODELS with strong initialization, but still out-performs the GEPA baseline (which was also provided with a strong initialization prompt). Notably, we observe that performance is still climbing at the end of 5 training epochs. We thus performed a longer weak initialization training run on the review length domain.

Figure 5 presents the evaluation curve of this run (continued from the initial 5 epoch run) and the performance of 5-epoch strong initialization training as a reference. With 25 additional epochs the system essentially achieve the performance of strong initialization training. This indicates that ADVISOR MODELS can learn hidden latents relying solely on the advisor’s ability to sample into learning signal, but that the process is greatly accelerated if the advisor is guided by strong priors. We believe that in practice, it is reasonable to assume that we have some idea of the target axes, and so training with strong initialization prompts is not unreasonable, if not the norm.

Table 1: **Performance on reasoning-intensive tasks.** While scores improve significantly with an advisor, qualitative analysis reveals the student model is often copying the advisor’s output verbatim. For math and MTOB, the advisor was a Qwen 2.5 7B model. For the more difficult RuleArena (Taxes) domain, a stronger Qwen 3 8B advisor was used.

TASK	METRIC	BASELINE	ADVISOR + STUDENT
MATH (MATH500)	ACCURACY	0.62	0.65
MTOB (KALAMANG→ENG)	CHRF SCORE	28.1	43.7
RULEARENA (TAXES)	ACCURACY	0.56	0.72

4.2 RQ2: How do ADVISOR MODELS Behave on Complex Reasoning Tasks?

As shown above, ADVISOR MODELS excel at adapting black-box models to specific styles and new knowledge. We now ask whether ADVISOR MODELS can *improve* the black-box models’ reasoning capabilities. We thus conducted a series of experiments on the complex reasoning domains to better understand their behavior.

Math Reasoning. We begin our reasoning evaluations by examining pure mathematical reasoning, training an advisor on a size 1000 subset of the Omni-Math dataset [Gao et al., 2024] with the objective of improving the black-box model’s accuracy on the of MATH-500 benchmark [Lightman et al., 2023]. Reward is binary, with 1 reward given for a correct solution and 0 otherwise.

To fully leverage the black-box model’s reasoning capabilities, we implement a 3-step variant of ADVISOR MODELS. Under this variant, the advisor is given an initial attempt by the student (step 1) to generate its advice (step 2), which the student uses to generate a final revised response (step 3). By starting from an initial black-box generation, we guard against the advisor providing harmful advice, and simplify the advisor’s role to that of a verifier. We found that this setup led to better outcomes in highly complex tasks. Further details are in Appendix B.1.

Low-Resource Translation. We additionally evaluate in domains where both reasoning and specialized task-specific knowledge are necessary. The first of these is the Machine Translation with One Book (MTOB) benchmark [Tanzer et al., 2024]. MTOB involves translating the extremely low-resource Kalamang language into English, requiring linguistic reasoning as well as specialized knowledge about Kalamang. Prompts consist of a passage to translate, as well as sample translations of words and sub-phrases in the passage drawn from a single book via string similarity. Models must understand when such samples are relevant to provide the best translations. Following the benchmark metric, we use chrF [Popović, 2015], a measure of string similarity, between the generated translation and ground-truth translation as reward. We sample 200 training and 50 test examples.

Complex Rule Following. Another domain involving both reasoning and task-specific knowledge is the RuleArena Taxes benchmark [Zhou et al., 2025], a collection of complex tax calculation scenarios requiring detailed understanding of applying the tax code. Prompts consist of an individuals profile (e.g., age, income, investments, filing status, etc.) and the complete tax filing instructions. Models must understand which parts of the tax code are relevant to the individual and produce a correct tax liability/refund value. Reward is binary, with 1 reward given for a /correct liability/refund value and 0 otherwise. We sample 75 train and 25 test examples. Due to the complexity of this task, we instead use GPT-4.1 mini as the black-box model and Qwen3-8B-Instruct [Qwen Team, 2025] as the advisor model, and follow the three-step setup used for math reasoning.

4.2.1 Results

We begin by discussing our results on the math reasoning domain, a task that is fully in-domain for the student model. We found that the advisor provided only limited improvements over the baseline performance (65% vs 62%). We attempted some variants of the ADVISOR MODELS framework to realize larger gains, which we discuss in Appendix B, but improvements remained limited. These outcomes suggest a boundary for advice in tasks that rely on precise, internal, step-by-step deduction. An advisor struggles to “teach” general reasoning in the same way it can convey a specific fact

unknown to the black-box model, pointing to the need for future work on more structured forms of guidance.

We thus turn our attention to MTOB and RuleArena, where both reasoning and specialized knowledge out-of-distribution to the student model are necessary. As shown in Table 1, here ADVISOR MODELS were able to provide significant improvements in performance over the baseline, improving chrF on MTOB from 28.1 to 43.7, and accuracy on RuleArena taxes from 56% to 64%. These results indicate that specialized domain knowledge that is advisable is a key component of domains conducive to ADVISOR MODELS.

Further investigation of traces revealed an interesting phenomenon. We found that in these complex, multi-step logical deduction tasks, the training dynamics of the ADVISOR MODELS framework shift. Instead of learning an advisor model that gives effective advice, the advisor is tuned into a problem-solving domain specialist. For example, in math accuracy and RuleArena often the advisor model would solve the problem fully in its advice which the student would repeat or lightly modify, while in MTOB the advisor model would give several possible full translations the student model would pick from. As this is a departure from the intended role of the advisor leaving all the intensive problem solving to the student, we term this behavior “over-advising.”

4.2.2 Addressing Over-Advising

Over-advising may be mitigated by strong prompting against the generation of full solutions by the advisor model or editing answers out of the generated advice, among other strategies. However, the occurrence of over-advising in reasoning tasks is not necessarily an undesirable property of ADVISOR MODELS.

RL has been shown to be an effective strategy for improving task-specific capabilities of open-source models, often beyond the frontier of closed models [DeepSeek-AI et al., 2025, Yang et al., 2024a, Liu et al., 2025]. Our results are no different, having trained an open advisor model to outperform the closed black-box model. A limitation of RL is a degradation in general capability as the model overfits to the target domain [Luo et al., 2025, Zhu et al., 2024, Zhang et al., 2023]. In ADVISOR MODELS, however, the presence of the general black-box model allows the full system to retain general capabilities (Section 4.3.2). Thus, ADVISOR MODELS has great potential in enabling the optimization of systems with strong target performance while retaining general capability. We examine this capability retention below.

4.3 RQ3: Are ADVISOR MODELS Robust?

The modular nature of ADVISOR MODELS allows us to view the advisor and student models separately. At the end of training, the advisor is specialized towards giving effective advice to the targeted task, while the student model is unchanged and retains all its initial capabilities. We thus approach the question of ADVISOR MODELS robustness from two perspectives. First, whether the advisor model is robust to transfer towards other student models. Second, whether the entire system with the unchanged student model is robust to out-of-domain inputs.

4.3.1 Cross-Student Advisor Transfer

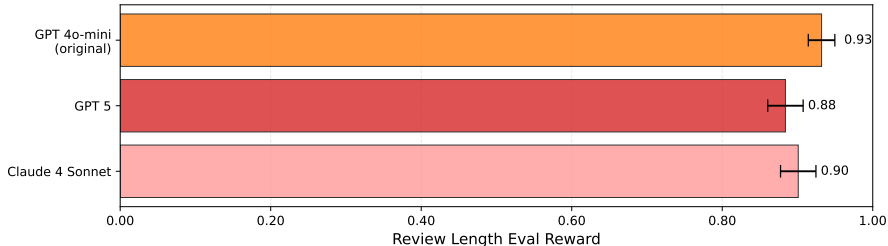


Figure 6: **ADVISOR MODELS transfer across different students.** Average reward on review level evaluation set of using advisor model trained with GPT-4o mini with different black-box models. The advisor model is able to provide effective advice to the unseen models, demonstrating the policy it learns is general.

An exciting consequence of advisor models operating in natural language is that it is theoretically possible for advisor models trained with one student model (e.g., GPT-4o mini) to effectively advise another (e.g., Claude 4 Sonnet). We test this insight by taking the advisor trained on the review length domain with GPT-4o mini as the student model and evaluating it with GPT-5 and Claude 4 Sonnet instead. We find that performance with all three student models are comparable, all achieving evaluation reward of about 0.90 out of 1, indicating that advisor transfer is feasible.

This property of advisor models can be leveraged to reduce training costs. The RL process for training the advisor model can incur significant monetary costs through many calls to the black-box model’s API. For our experiments with 400 training examples, training for 20 epochs involved approximately $400 \times 20 \times 8 = 64,000$ calls to the black-box model (only rollout process, not including calls potentially used in reward functions). Given the costs of frontier models today, this was on the order of \$10 for our experiments. However, cost could be magnitudes larger when working with more expensive black-box models. We have demonstrated that with advisor transfer, we can train advisors with cheaper student models and effectively deploy with larger, more expensive ones.

4.3.2 Frontier Capability Retention

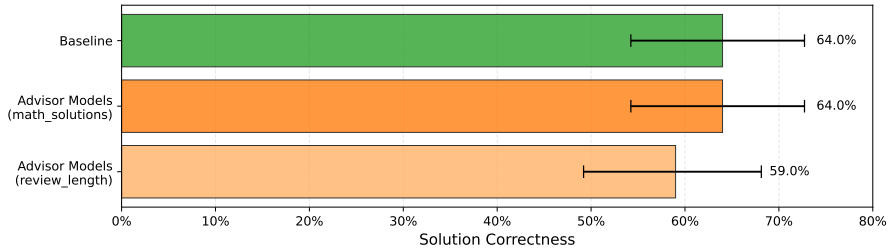


Figure 7: **Specializing an advisor does not degrade the student’s core capabilities.** We evaluate an ADVISOR MODELS system with advisors heavily optimized for stylistic preferences and unrelated preferences on the correctness of generated math solutions. The system’s performance is statistically indistinguishable from the un-optimized baseline, demonstrating the robustness of ADVISOR MODELS’s modular architecture. 95% confidence intervals are provided.

A key benefit of the modular ADVISOR MODELS architecture is its inherent robustness against catastrophic forgetting. Unlike direct fine-tuning, which can degrade a model’s general capabilities [Luo et al., 2025, Zhu et al., 2024, Zhang et al., 2023], our approach preserves the student model’s weights, leaving its core competencies unaltered.

We demonstrate this robustness in Figure 7. First, we take an ADVISOR MODELS system trained to optimize for stylistic preferences on math solutions and evaluate the correctness of its generated solutions. The resulting accuracy is statistically indistinguishable from the unadvised baseline, showing that optimizing for a preference within a domain does not harm performance on the primary objective. More strikingly, this robustness extends even to **out-of-domain advisors**. When we evaluate an ADVISOR MODELS system that was heavily optimized for an entirely unrelated task (review length), the system’s math solution accuracy remains statistically unchanged from the baseline. This demonstrates that the student model’s performance is not degraded even when receiving potentially irrelevant advice, highlighting the system’s resilience.

5 Conclusion & Future Work

We introduce ADVISOR MODELS, a framework that enables dynamic, instance-specific steering of powerful black-box models. We demonstrate that this approach can successfully learn environment latents such as user preferences, solving personalization tasks where state-of-the-art static prompt optimization methods completely fail, and improve performance in out-of-distribution reasoning tasks. These initial findings on controlled domains highlight the significant potential for ADVISOR MODELS to be scaled to more complex, real-world challenges.

Moreover, our experiments demonstrate that ADVISOR MODELS can learn a specialized, fine-tuned advisor while retaining a generalist student model, creating a system that is both transferable to other student models and robust to out-of-distribution inputs.

These results open up several exciting directions for future work, from designing advisor architectures that can provide more structured, multi-step guidance, studying parametric memory, and further exploring the dynamics of these hybrid advisor-student systems. We believe this research is a significant step towards creating more customizable, personalizable, and robust interactions with the most powerful AI models available.

Acknowledgments and Disclosure of Funding

We would like to thank Tyler Griggs for his many insightful discussions about working with SkyRL.

Sky Computing Lab is supported by gifts from Accenture, AMD, Anyscale, Cisco, Google, IBM, Intel, Intesa Sanpaolo, Lambda, Microsoft, NVIDIA, Samsung SDS, SAP, and VMware. We would additionally like to acknowledge Databricks for their generous compute support.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 2146752. 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.

References

- Yash Agarwal, Bidisha Bhattacharya, Yuxin Wang, Luyu Gao, Chris Callison-Burch, Uri Alon, Kelvin Guu, Karthik Narasimhan, Baptiste Rozière, Yann LeCun, et al. PromptWizard: Empowering large language models to design prompts. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=ZzHAD8m4fH>.
- Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziemis, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alexandros G. Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective prompt evolution can outperform reinforcement learning, 2025. URL <https://arxiv.org/abs/2507.19457>.
- Anthropic. Introducing Claude Opus 4.1. <https://www.anthropic.com/news/claude-opus-4-1>, August 2025. Accessed August 30, 2025.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL <https://arxiv.org/abs/2005.14165>.
- Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng

- Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025. URL <https://arxiv.org/abs/2501.12948>.
- Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL <https://arxiv.org/abs/2305.14314>.
- Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. *arXiv preprint arXiv:2309.16797*, 2023. URL <https://arxiv.org/abs/2309.16797>.
- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, Zhengyang Tang, Benyou Wang, Daoguang Zan, Shanghaoran Quan, Ge Zhang, Lei Sha, Yichang Zhang, Xuancheng Ren, Tianyu Liu, and Baobao Chang. Omnimath: A universal olympiad level mathematic benchmark for large language models, 2024. URL <https://arxiv.org/abs/2410.07985>.
- Tyler Griggs, Sumanth Hegde, Eric Tang, Shu Liu, Shiyi Cao, Dacheng Li, Charlie Ruan, Philipp Moritz, Kourosh Hakhmaneshi, Richard Liaw, Akshay Malik, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Evolving SkyRL into a highly-modular RL framework, 2025. Notion Blog.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International conference on machine learning*, pages 2790–2799. PMLR, 2019.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- Amirhossein Kazemnejad, Milad Aghajohari, Eva Portelance, Alessandro Sordani, Siva Reddy, Aaron Courville, and Nicolas Le Roux. Vineppo: Refining credit assignment in rl training of llms, 2025. URL <https://arxiv.org/abs/2410.01679>.
- Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive NLP. *arXiv preprint arXiv:2212.14024*, 2022. URL <https://arxiv.org/abs/2212.14024>.
- Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into state-of-the-art pipelines. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL <https://openreview.net/forum?id=sY5N0zY50d>.
- Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- Shu Liu, Sumanth Hegde, Shiyi Cao, Alan Zhu, Dacheng Li, Tyler Griggs, Eric Tang, Akshay Malik, Kourosh Hakhmaneshi, Richard Liaw, Philipp Moritz, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. SkyRL-SQL: Matching GPT-4o and o4-mini on Text2SQL with multi-turn RL, 2025. Notion Blog.

- Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 2025.
- OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. <https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/>, 2024.
- OpenAI. Introducing GPT-4.1 in the API. <https://openai.com/index/gpt-4-1/>, April 2025a. Accessed August 30, 2025.
- OpenAI. Introducing GPT-5. <https://openai.com/index/introducing-gpt-5/>, August 2025b. Accessed August 30, 2025.
- Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9340–9366, Miami, Florida, USA, nov 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.525. URL <https://aclanthology.org/2024.emnlp-main.525/>.
- Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, pages 392–395, 2015.
- Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL <https://qwenlm.github.io/blog/qwen2.5/>.
- Qwen Team. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models, 2024. URL <https://arxiv.org/abs/2402.03300>.
- Anikait Singh, Sheryl Hsu, Kyle Hsu, Eric Mitchell, Stefano Ermon, Tatsunori Hashimoto, Archit Sharma, and Chelsea Finn. FSPO: Few-shot preference optimization of synthetic preference data in LLMs elicits effective personalization to real users, 2025. URL <https://arxiv.org/abs/2502.19312>.
- Garrett Tanzer, Mirac Suzgun, Eline Visser, Dan Jurafsky, and Luke Melas-Kyriazi. A benchmark for learning to translate a new language from one grammar book, 2024. URL <https://arxiv.org/abs/2309.16575>.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023. URL <https://arxiv.org/abs/2203.11171>.
- An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024a.
- Greg Yang, Sifan Xie, Yao Chen, Xinjie Chen, Yuhuai Wang, Jian Chen, Jue Chen, and Azalia Mirhoseini. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024b. URL https://openreview.net/forum?id=s_E5WQeEaC.
- Zheng Zhang, Chen Zheng, Da Tang, Ke Sun, Yukun Ma, Yingdong Bu, Xun Zhou, and Liang Zhao. Balancing specialized and general skills in LLMs: The impact of modern tuning and data strategy. *arXiv preprint arXiv:2310.04945*, 2023.
- Ruiwen Zhou, Wenyue Hua, Liangming Pan, Sitao Cheng, Xiaobao Wu, En Yu, and William Yang Wang. RuleArena: A benchmark for rule-guided reasoning with LLMs in real-world scenarios, 2025. URL <https://arxiv.org/abs/2412.08972>.

Shuyan Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Chen, Yi Tay, Hyung Won Chung, William Fedus, et al. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022. URL <https://arxiv.org/abs/2211.01910>.

Banghua Zhu, Michael I Jordan, and Jiantao Jiao. Iterative data smoothing: Mitigating reward overfitting and overoptimization in RLHF. *arXiv preprint arXiv:2401.16335*, 2024.

A Training Details

We utilize a lightly modified fork of the SkyRL framework for all of our RL training runs [Griggs et al., 2025]. We run all training experiments unless otherwise noted on a single node of 8xH100s. For reproducibility, we provide key non-default training parameters used:

Table 2: Key training hyperparameters used for ADVISOR MODELS experiments.

HYPERPARAMETER	VALUE
TRAINING EPOCHS	10
TRAIN BATCH SIZE	16
POLICY MINI-BATCH SIZE	4
MICRO-BATCH SIZE (PER GPU)	2
LEARNING RATE	1.0×10^{-6}
MAX PROMPT LENGTH	8192
MAX GENERATION LENGTH	16384
TEMPERATURE	1.0

B ADVISOR MODELS Variants

In this section we discuss variations upon the core ADVISOR MODELS framework we tried. While these variants did not meaningfully improve upon the original ADVISOR MODELS formulation, we hope they can inspire future work.

B.1 3-step Advisor System

The second variation modifies the original ADVISOR MODELS formulation to reduce the difficulty of the advisor’s role. Rather than asking the advisor to provide advice given only the task prompt, it is also given an initial attempt by the student. There are thus 3 steps in this formulation rather than 2: first, the student model makes an initial attempt at the task; second, the advisor provides advice based on the attempt; third, the student model updates its attempt based on the advice.

Under this formulation, the advisor’s role is made simpler by the fact there is an initial attempt to evaluate. This shifts the role of the advisor from a pure generator to a verifier. In theory this should help training as the space for advice is more defined, but comes at the cost of twice as many calls to the student model.

B.2 Monte Carlo Reward Estimation

The first variation relates to the reward calculation. The reward we calculate should be the reward assigned to the advisor’s generations, i.e., a measure of how good the generated advice is. In the original ADVISOR MODELS formulation, we estimate the quality of the advice by measuring whether the student model is able to produce a good final output based on the advice. However, student model generations are non-deterministic, meaning the reward estimate has high variance.

One way to reduce this variance is through Monte Carlo reward estimation [Kazemnejad et al., 2025], where for a single piece of advice we take the average reward of multiple student responses rather than the reward of a single student response. This has the effect of giving a lower variance estimate of the reward for the advice. In theory, this should result in more stable training due to better credit attribution at the cost of a more expensive reward calculation.

C Advice Examples

In this appendix we present examples advice generated by ADVISOR MODELS before and after training for the same evaluation problems in the review length setting.

Review Length Eval Example #1

Prompt Info

Name: Matei

Latent Preference: 10-word long reviews

Prompt: Write a review for the Horror book, Frankenstein.

Advice at Step 0

Given Matei's preference for detailed analysis and a blend of personal critique and factual information, advise focusing on the book's complex themes, character development, and intriguing narrative structure. Matei benefits from a mix of objective observations and subjective experiences, so include both your personal reactions and insights backed by textual evidence. Since Matei enjoys detailed reviews, ensure your review is around 500-600 words, exploring the book's historical context, its impact on the genre, and your specific responses to the eerie atmosphere and psychological depth.

Advice after Training

Given Matei's preference for concise, straightforward reviews, focus on a brief, no-nonsense review. Keep your review to about 8-10 words, highlighting whether you enjoyed the book or not and why, if necessary. For example: "Frankenstein is a well-crafted horror book that is easy to read."

Review Length Eval Example #2

Prompt Info

Name: Alex

Latent Preference: 1000-word long reviews

Prompt: Write a review for the Drama movie, Titanic.

Advice at Step 0

For Alex, consider focusing on the emotional depth and historical significance of the film in your 3-4 sentence review. Since Alex appreciates detailed storytelling, mention how James Cameron masterfully combines romance with historical drama to create a moving narrative. Additionally, Alex might enjoy a touch on the visual and audio effects that enhance the Titanic's scale and the hardships faced by the characters.

Advice after Training

Given Alex's preference for detailed analysis, focus on discussing the plot, character development, and themes of the movie. Since Alex enjoys longer, more in-depth reviews, write a 1000-word review that explores not just the plot, but also the film's impact and its relevance today.

D Prompt Examples

In this appendix we present the advisor and student prompts we used to train and evaluate ADVISOR MODELS on the review length domain.

Review Length Advisor Prompt (strong)

System Prompt

You are a review writing advisor. Provide specific guidance for writing a review that matches the person’s preferences. Consider the length preferences and style that would work best for the target person.

Instruction

You need to provide advice for writing a review for {person}.
The task is: {prompt}
Provide specific advice about the review that would work best for {person}. Think carefully about the length of the review in your advice. Keep your advice to 3-4 sentences.

Review Length Advisor Prompt (weak)

System Prompt

You are a review writing advisor. Provide specific guidance for writing a review.

Instruction

You need to provide advice for writing a review for {person}.
The task is: {prompt}
Provide specific advice about the review for {person}. Keep your advice to 3-4 sentences.

Review Length Student Prompt

System Prompt

You are a review writer. Based on the prompt and advisor guidance, write a review that follows the guidance provided. Write a clear, well-structured review.

Instruction

Review Prompt: {prompt}
Advisor Guidance:
{advice}
Write a review following the advisor’s guidance.