Hindsight Merging: Diverse Data Generation with Language Models

Veniamin Veselovsky[*1] **Benedikt Stroebl**[*1] **Gianluca Bencomo**[*1]
Dilip Arumugam[1] **Lisa Schut**[3] **Arvind Narayanan**[1] **Thomas L. Griffiths**[1,2]

[1]Department of Computer Science, Princeton University
[2]Department of Psychology, Princeton University
[3]OATML, Deptartment of Computer Science, University of Oxford

Abstract

Pre-training a language model equips it with a broad understanding of the world, while fine-tuning refines it into a helpful assistant. However, fine-tuning does not exclusively enhance task-specific behaviors but also suppresses some of the beneficial variability from pre-training. This reduction in diversity is partly due to the optimization process, which theoretically decreases model entropy in exchange for task performance. To counteract this, we introduce *hindsight merging*, a technique that combines a fine-tuned model with a previous training checkpoint using linear interpolation to restore entropy and improve performance. Hindsight-merged models retain strong instruction-following capabilities and alignment while displaying increased diversity present in the base model. Additionally, this results in improved inference scaling, achieving a consistent 20-50% increase in pass@10 relative to the instruction tuned model across a coding benchmark and series of models. Our findings suggest that hindsight merging is an effective strategy for generating diverse generations that follow instructions.

1 INTRODUCTION

Humans solve a wide variety of problems by reusing previously learned knowledge and applying diverse patterns of thinking [Griffiths et al., 2019]. This ability to adapt and explore multiple cognitive pathways allows human reasoning to converge to correct solutions over time [Collins and Frank, 2013, Tomov et al., 2020, Solway et al., 2014, Maisto et al., 2015, Correa et al., 2023]. The adaptive reuse of available resources is not just a hallmark of human intelligence but also a key ingredient in the design of reliable artificial intelligence (AI) systems. In many AI applications, diversity plays a crucial role. Repeated sampling from language models relies on a wide variety of generated responses to enhance performance [Brown et al., 2024]. Compound AI systems [Zaharia et al., 2024] improve their scalability and inference capabilities when diverse data inputs and models are integrated. However, while diversity enhances both human reasoning and AI training, the challenge of generating synthetic datasets with sufficient richness and variation remains an unresolved problem.

The primary challenge can be understood through the lens of optimization. Both humans and language models are known to directly fit their training data [Hasson et al., 2020], but the breadth of human experience leads to more diversity in our ability to reason. Large language models (LLMs) trained on a vast and diverse internet corpora produce a base model that generates a wide variety of outputs. However, we remove much of this diversity when we fine-tune due to the use of smaller datasets and the objective of aligned behavior [Murthy et al., 2024]. This creates a paradox: the optimization that improves task performance undermines the model's ability to generate the broad range of outcomes necessary for rich and diverse synthetic datasets. Simply, the best model for solving a task is not necessarily the best model for generating a dataset.

In this work, we explore this trade-off between optimizing on task-specific datasets and producing diverse synthetic datasets. We introduce a theoretically-motivated approach to increasing diversity, *hindsight merging*, that merges an instruction-tuned model with prior training checkpoints to better accommodate the trade-off required for diverse dataset generation (see Figure 1). Through hindsight merging, we demonstrate that the resulting data is more diverse, maintains instruction-following abilities, and has improved pass@k performance compared to the constituent models before merging.

[*]Equal contribution

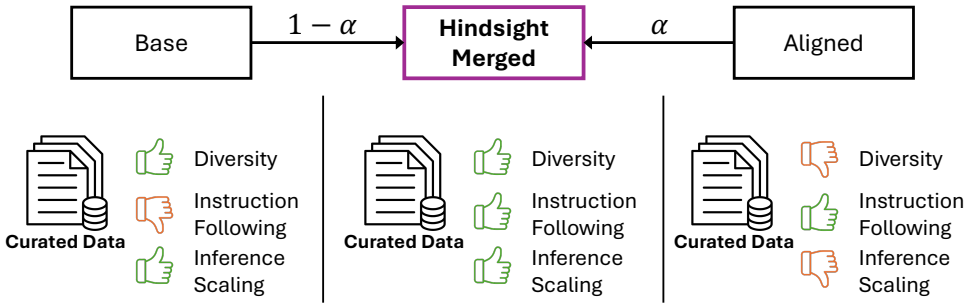

Figure 1: Overview of our main findings.

2 THE ENTROPY SPECTRUM

While fine-tuning language models is crucial for achieving high performance on downstream tasks, it requires specializing the response distribution away from the broad, diverse data that was well-approximated during pre-training. Suppose pre-training involved fitting language model responses to match a distribution P. Later, fine-tuning the pre-trained language model demands matching a new response distribution Q. Let $\mathbb{H}(\cdot)$ denote the entropy of a distribution [Shannon, 1948, Cover and Thomas, 2012]. It is to be expected that $\mathbb{H}(Q) < \mathbb{H}(P)$; indeed, given that current best practices often take P as an Internet-scale distribution over prompt-response pairs, it may be more realistic to consider $\mathbb{H}(Q) \ll \mathbb{H}(P)$. During fine-tuning, a language model is being optimized away from accurately approximating P to fit Q, which forces a decrease in entropy to accommodate the new response distribution. Such a drop entropy naturally begets a commensurate drop in overall response diversity.

This section attempts to make this intuitive idea mathematically precise in two concrete ways. First, we focus on supervised fine-tuning and adopt a local perspective by studying the change in entropy per optimization step. Second, we adopt a reinforcement-learning perspective to implicitly consider how entropy changes after multiple iterations of policy optimization. Taken together, these theoretical findings corroborate the intuitive notion that fine-tuning decreases entropy and, in doing so, reduces the diversity of language model responses. We conclude with an information-theoretic motivation for a simple remedy to this vanished diversity: mixing response distributions between the current fine-tuned model and the pre-trained base model.

2.1 SUPERVISED FINE-TUNING

Consider a neural network $q_\theta(x_{t+1}|x_{1:t}) = \text{softmax}(f(x_{1:t};\theta))$, parameterized by $\theta \in \mathbb{R}^D$, that models a predictive distribution over next-token generations. We denote $q_\theta(x_{t+1}|x_{1:t})$ as $q_\theta(x)$ for brevity. Let us define parameter updates during fine-tuning as a transition operator $T : \mathbb{R}^D \to \mathbb{R}^D$, which induces the following mapping M on the output space:

$$M : q_\theta^k(x) \mapsto q_\theta^{k+1}(x) \tag{1}$$

where $q_\theta^{k+1}(x) = \text{softmax}(f(x_{1:t}; T(\theta_k)))$ after applying the transition $\theta_{k+1} = T(\theta_k)$. For all subsequent calculations, please consult Appendix A for detailed derivations. The total entropy of the predictive distribution after K update steps from some base model $q_\theta^0(x)$ is given by:

$$\mathbb{H}(q_\theta^K) = \mathbb{H}(q_\theta^0) + \sum_{k=0}^{K-1} \mathbb{E}_{x\sim q_\theta^k}\left[\log\left|\frac{\partial M}{\partial q}\left(q_\theta^k(x)\right)\right|\right], \tag{2}$$

where $|\partial M/\partial q|$ denotes the Jacobian determinant of M at each update step. For stochastic gradient descent (SGD), the transition operator T is defined as

$$\theta_{k+1} \leftarrow \theta_k - \alpha\nabla_\theta\mathcal{L}(\theta_k), \tag{3}$$

where α is the learning rate. Although M does not admit a closed form solution, we can approximate it using a first-order Taylor expansion for small α:

$$M(q) \approx q - \alpha\left[\text{diag}(q) - qq^\top\right]\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta). \tag{4}$$

Taking the Jacobian yields:

$$\mathbb{H}(q_\theta^k) - \mathbb{H}(q_\theta^{k-1}) \approx \mathbb{E}_{x\sim q_\theta^k}\left[\log|I - \alpha H|\right], \tag{5}$$

where $H = [E - G]\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta)$ for $E_{ijk} = \delta_{ij}\cdot\delta_{jk}$ and $G_{ijk} = \delta_{ik}q_j + \delta_{jk}q_i$. This approximation denotes how much entropy changes per update step using SGD.

During supervised fine-tuning (SFT), the goal is to align the model's next-token predictions with a target data-generating process, $p^\star(x_{t+1}|x_{1:t})$, which is typically smaller and less diverse than the original pre-training data. In the most common case, we perform SFT with the cross-entropy loss. The gradient term from Equation 4 can be approximated by the expression

$$\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta) \approx (q - y), \tag{6}$$

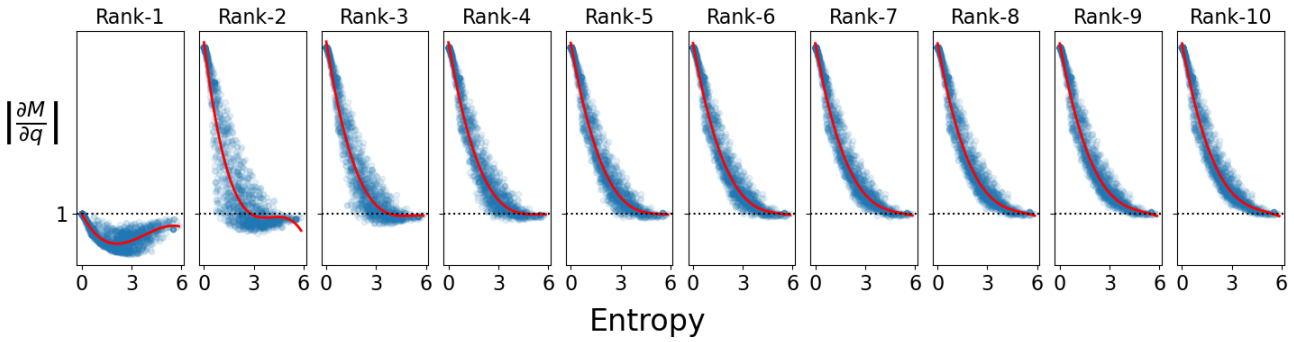

Figure 2: Analysis of theoretical volume changes for SGD based on logit rank (1-10) and softmax entropy in next-token predictions ($\text{top}_k = 600$). Using 10 random arXiv abstracts from January 2025, we compute next-token predictive distributions and compute the Jacobian of the SGD step when the correct token corresponds to each logit.

where q denotes model output probabilities and y represents the one-hot encoded ground-truth labels for the input x. Note that this approximation captures the classical gradient for cross-entropy, but under the assumption of a small α and dominance of logit-level loss terms in the gradient flow. While simplifying, this approximation allows us to analyze of how entropy over next-token predictions evolves at each step of the optimization process using Equation 5.

In Figure 2, we show approximate changes to model entropy, indicated by $|\partial M/\partial q|$. To build an intuition, when $|\partial M/\partial q| > 1$, the volume of the predictive distribution $q_\theta(x_{t+1}|x_{1:t})$ is expanding, implying an increase in entropy over model generations given the token history $x_{1:t}$. When $|\partial M/\partial q| < 1$, volume is shrinking, implying a decrease in entropy.

Optimizing the cross-entropy loss affects model behavior differently depending on the rank of the ground-truth logits. For rank-1 predictions, all models contract in volume, thereby systematically reducing the entropy of conditional distributions for next-token predictions as they begin to fit the data perfectly. In this case, entropy decrease is a symptom of overfitting. For rank-2 through rank-10 predictions, lower-entropy models show volume expansion whereas higher entropy models continue to contract. Regardless of rank, high-entropy models consistently exhibit volume contraction, with less aggressive contractions as the prediction rank increases.

2.2 REINFORCEMENT LEARNING WITH HUMAN FEEDBACK

Aside from the supervised approach examined in the preceding section, an alternative route to fine-tuning a language model uses reinforcement learning [Stiennon et al., 2020, Ouyang et al., 2022]. Compared to the local analysis in the previous sub-section, which characterizes how entropy changes per optimization step, this section presents a more global picture. As the reinforcement learning with human feedback (RLHF) pipeline carries various adornments that complicate analysis, this section restricts its focus to a simpler Markov decision process (MDP) wherein learning an optimal policy is equivalent to learning the per-token distribution of some underlying fine-tuning response distribution. Under mild assumptions, we show that fine-tuning a language model towards a lower-entropy dataset via policy-gradient updates [Sutton et al., 1999] decreases the entropy in the language models responses relative to the pretrained model. We encourage readers to consult Appendix B for all technical details.

Let $\mathcal{V}$ be a finite vocabulary of tokens such that the set of possible token sequences is $\mathcal{L} = \bigcup_{n=1}^{\infty} \mathcal{V}^n$. Let $\mu \in \Delta(\mathcal{L})$ be the distribution over prompts and let $p^\star : \mathcal{L} \to \Delta(\mathcal{L})$ be the ground-truth response distribution given any prompt. Next, we specify a MDP in which any LLM is a policy and the reward function is constructed such that learning the optimal policy amounts to obtaining a LLM that matches $p^\star$.

Consider the infinite-horizon, discounted MDP [Bellman, 1957, Puterman, 1994] $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mu, \gamma \rangle$. Here $\mathcal{S} = \mathcal{L}$ represents any token sequence from $\mathcal{V}$. The action space $\mathcal{A} = \mathcal{V} \cup \{\texttt{STOP}\}$ contains all valid tokens a LLM may emit as well as an explicit `STOP` token to denote response completion. Logically, the MDP follows deterministic transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which appends the selected token to the current state: $\mathcal{T}(s, a) = \langle s, a \rangle$.[1] The initial state distribution $\mu \in \Delta(\mathcal{S})$ is precisely the distribution over prompts from above. The discount factor $\gamma \in [0, 1)$ conveys the effective time horizon for optimizing rewards. So far, we have specified a controlled Markov process (that is, a MDP without a reward function) such that any policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ represents a LLM that examines the prompt along with any partially-generated response thus far and

[1] For brevity, we omit an absorbing, zero-reward terminal state that an agent transitions to upon choosing the `STOP` action.

emits a distribution over next tokens.

To capture the objective of fine-tuning a LLM towards a ground-truth response distribution $p^\star$, we consider a policy-dependent reward function defined as $\mathcal{R}(s,a) = \log\left(\frac{p^\star(a|s)}{\pi(a|s)}\right)$. Recall that the performance of any policy π with respect to a prompt $s \in \mathcal{S}$ is given by its associated value function $V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s\right]$. With a slight abuse of notation, we account for randomness in the initial state through $V^\pi(\mu) \triangleq \mathbb{E}_{s_0\sim\mu}\left[V^\pi(s_0)\right]$. Recall that any policy induces a corresponding discounted stationary state visitation distribution $d^\pi_\mu(s) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}^\pi(s_t = s)$, where $\mathbb{P}^\pi(s_t = \cdot) \in \Delta(\mathcal{S})$ is the distribution over states visited by policy π at timestep t. Intuitively d^π_μ encodes which states policy π will occupy using γ to account for near-term versus future visitation. In the context of LLMs, d^π_μ encodes a distribution over prompts and partial/complete responses generated by a particular LLM π.

We define the optimal policy $\pi^\star$ of $\mathcal{M}$ as achieving supremal value with associated value function $V^\star(\mu) = \sup_\pi V^\pi(\mu)$. For the particular choice of policy-dependent reward function, we see that an optimal policy $\pi^\star$ minimizes the KL-divergence between its own per-step token distribution and that of the ground-truth distribution $p^\star$: $V^\star(\mu) = -\inf_\pi \frac{1}{(1-\gamma)}\mathbb{E}_{s\sim d^\pi_\mu}\left[D_{\mathrm{KL}}(\pi(\cdot \mid s) \,||\, p^\star(\cdot \mid s))\right]$.

Theorem 1 (Informal)**.** *Let π^0 be an initial, pre-trained base model and π^K be the fine-tuned LLM after $K \in \mathbb{N}$ iterations of policy-gradient updates. Then, we have*

$$\mathbb{E}_{s\sim d^\star_\mu}\left[D_{\mathrm{KL}}(\pi^\star_s \,||\, \pi^K_s)\right] \lesssim \mathbb{E}_{s\sim d^\star_\mu}\left[D_{\mathrm{KL}}(\pi^\star_s \,||\, \pi^0_s)\right].$$

As the optimal policy of the MDP $\mathcal{M}$ is $p^\star$, Theorem 1 affirms that making policy-gradient updates [Sutton et al., 1999] brings the fine-tuned LLM π^K closer (in KL-divergence) to the lower-entropy response distribution $p^\star$ than the higher-entropy pre-trained model π^0.

2.3 TOWARD RECOVERING DIVERSITY

Just as the reduction of entropy formalized in the previous two sub-sections is an intuitive consequence of language model fine-tuning, we use this final sub-section to motivate an equally intuitive solution: mixing model parameters. Prior work establishes a link between the blending of model parameters and a commensurate blending of the associated model response distributions [Kangaslahti and Alvarez-Melis, 2024]. One might naturally hope to obtain the "best of both worlds" by mixing weights of a pre-trained language model fit to a high-entropy response distribution and those of a fine-tuned model closely approximating a low-entropy response distribution. We may formalize this intuition with a first proposition that considers an obvious linear interpolation between response distributions:

Proposition 1. *Consider two arbitrary probability distributions P and Q such that $\mathbb{H}(P) \geq \mathbb{H}(Q)$. For any $\alpha \in [0,1]$, define $M_\alpha = \alpha \cdot P + (1-\alpha)\cdot Q$. Then,*

$$\mathbb{H}(Q) \leq \mathbb{H}(M) \leq \mathbb{H}(P)$$

A more general information-theoretic analysis allows us to obtain looser bounds on the entropy of response sampled from the mixture distribution that goes beyond just linear interpolation:

Proposition 2. *Consider two arbitrary probability distributions $P, Q \in \Delta(\mathcal{X})$ with $X_1 \sim P$ and $X_2 \sim Q$. Let $Z \in \Delta(\{1,2\})$ be a random index following an arbitrary distribution. Then, X_Z is a random variable denoting a sample from the mixture distribution between P and Q induced by Z. Moreover,*

$$2 \cdot \min_{i\in\{1,2\}} \mathbb{H}(X_i) \leq \mathbb{H}(X_Z) \leq 2 \cdot \max_{i\in\{1,2\}} \mathbb{H}(X_i) + 1.$$

Proofs may be found in Appendix C. Together, these two propositions highlight one promising pathway to recovering the response diversity lost due to standard fine-tuning practices; namely, by blending the response distributions of the current fine-tuned language model and the pre-trained base model. In the next section, we present our hindsight merging approach that uses interpolation between respective model weights to achieve this diverse mixture of response distributions.

3 DIVERSE GENERATION

In Section 2, we showed that fine-tuning on low diversity datasets leads to low diversity data generations in fine-tuned models. We propose *hindsight merging* — interpolating between the weights of the base model, which has a high diversity, and the fine-tuned model — to improve diversity in the generations.

Mixing weights leverages the fact that, during fine-tuning, the model is likely in the neural tangent kernel (NTK) regime [Fort et al., 2020, Wortsman et al., 2022c,b, Ilharco et al., 2022]. When the model is in the NTK regime, the functional updates are approximately linear. Therefore, by interpolating between the weights, we approximately roll-back the model along its optimization trajectory to previous checkpoints that capture the desirable results of fine-tuning but at higher levels of entropy.

3.1 ENCOURAGING DIVERSITY

Hindsight merging. For model merging, we focus our analysis on a couple classes and sizes of models: Llama-3.1-8b, Llama-2-7b, Llama-2-13b, Llam-3.1-70b. For each of these models we combine both the base and instruct models using MergeKit [Goddard et al., 2024] and use linear interpolation (c.f., Appendix G for further information). For the Llama-2 series of models, we combine it with the Vicuna instruct version Chiang et al. [2023].

To merge the models, we define a parameter $\alpha \in [0, 1]$ which measures the merging coefficient. When $\alpha = 0$ the merged model equals the base model when $\alpha = 1$, it is entirely the instruct copy. For experiments, we restrict $\alpha \in \{0, 0.7, 0.9, 1\}$. We denote $\alpha = 0$ as the "pretrained" model, and $\alpha = 1$ as the "instruct" model. For the pretrained model, we convert prompts to completion prompts, whereas for the merged models we use a chat template.

3.2 EVALUATION

Data. We take 378 tasks from MBPP+, a large-scale benchmark dataset for code generation in Python [Liu et al., 2024]. MBPP+ is an extension of the original MBPP benchmark with additional unit tests to improve test coverage and avoid false positives.

Diversity. In many applications of synthetic data, semantic diversity outweighs syntactic diversity—diverse approaches are more valuable than diverse use of language for the same approach. To account for this, we rely on a BERTScore-style [Zhang et al., 2019] similarity metric. For each of the rewritten generations, we embed the traces using OpenAI's `text-embedding-3-small` with 150 dimensions. We measure the local similarity or the average cosine similarity across generations from the same question and the global similarity as the diversity in generations across different questions.

Performance. For each of the programming tasks in MBPP+, we test a model's performance on the dataset using the provided unit tests (functions that verify the code correctness). Here we report two scores, the pass@1 and pass@k [Kulal et al., 2019]. Pass@1 is the average number of generations that are correct, irrespective of if the model gets multiple correct generations for one question. On the other hand, pass@k tests if a model has k tries to solve a problem, what are the chances at least one of the generations passes the unit tests. Explicitly, we estimate pass@k using the following formula from Chen et al. [2021]:

$$\text{pass@}k := \mathbb{E}_{\text{Problems}}\left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}\right]$$

Instruction following. Pre-training a language model provides it with an understanding of the world, however during the fine-tuning stage the model becomes an assistant learning how to follow instructions. Thus, merging the two models, may lead to reduced performance in the instruction following tasks the model was trained on. One such task is refusal, where when prompted with a potentially problematic request, an instruction-tuned language model is explicitly trained to refuse that request. To evaluate the hindsight-merged models, we see how performance on SORRY-Bench — a benchmark for safety refusals — changes [Xie et al., 2024]. To evaluate SORRY-Bench generations, we use GPT-4.1-mini-as-a-judge.

4 RESULTS

In our experiments, we show that we (1) can induce diversity in the generations using hindsight merging, (2) does not harm model performance in other aspects, such as instruction following, and (3) improves model pass@k performance.

4.1 THE BEST OF BOTH WORLDS

Diversity. To begin, we explore how varying the α parameter affects diversity of the model generations. In Figure 3, we illustrate the global similarity metrics across the different models. Within the global context (cross-question similarity) reducing the alpha makes the model on average more diverse. Across the four models we studied, the fully instruct model has the highest on average BERTScore across its generated responses.

In Figure 4 we show the intra-question similarity for the different models. Within the Vicuna models we find that hindsight merging actually results in reduced diversity for generations of the same question compared with the fully instruct model. On the other hand, for the Llama-3.1-8B and Llama-3.1-70B models, we find that the fully instruct model consistently provides the most homogeneous answers. The difference between the Vicuna and Llama model is largely explained by the inability of the Vicuna models to solve some of the coding problems, resulting in highly unstable behavior. For some of the questions, the model merely repeats the original prompt vs. other times generating highly varying solutions. This behavior is further backed up by the general poor performance shown below.

Failure-to-Refuse. Figure 6 presents a heat-map of failure-to-refuse rates across the 44 SORRY-Bench content clusters. Each column corresponds to a model variant; the number in parentheses beside the name is its mean failure rate (i.e., the fraction of hazardous prompts it answered instead of refusing). Darker shades mark categories where the model answered when it should have refused, so higher values

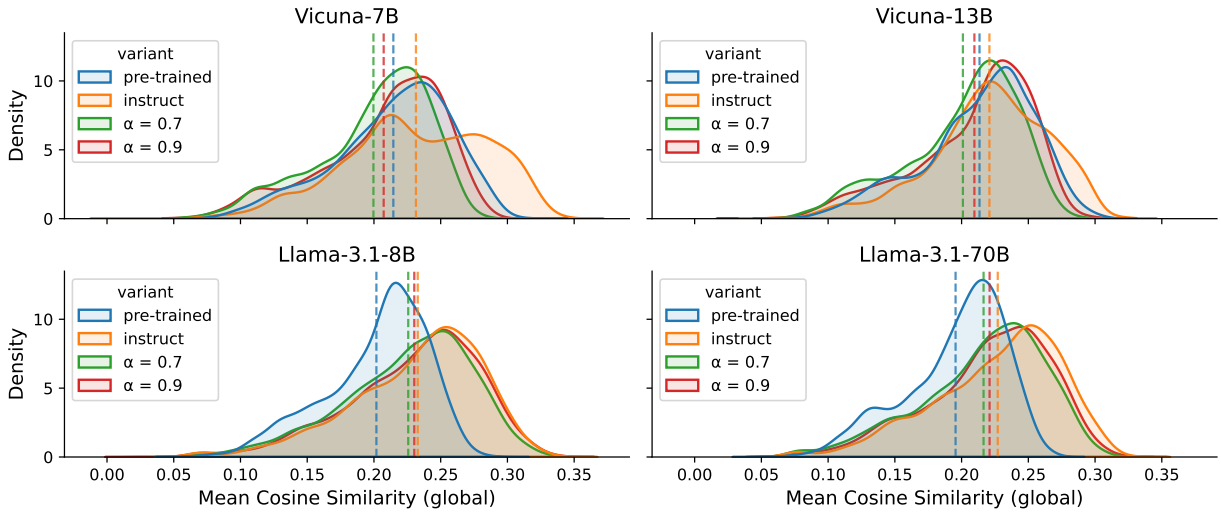

Figure 3: Average cosine similarity across embeddings of generated text in the global settings.

mean weaker safety behavior.

For the Llama-3.1-8B family, the pretrained checkpoint answers almost half of restricted prompts (0.48). Blending in instruct weights sharply reduces this: at $\alpha = 0.7$ the rate falls to 0.32, at $\alpha = 0.9$ to 0.30, and the fully-instruct model settles at 0.28. Thus even a modest interpolation captures most of the instruction-tuned caution while preserving base-model fluency. On the other hand, for the Llama-3.1-70B family scale alone does not buy safety. The 70B pretrained model is the least compliant in the study (0.75). $\alpha = .70$ cuts the failure rate nearly in half (0.42), and full instruction tuning brings it down to 0.45, still well above the 8B instruct score. Finally, both Vicuna models show the same monotonic trend. For the 7B model the pretrained fails to refuse around 47% of the time reduced to 0.44 in $\alpha = 0.7$, further reduced to 0.40 in $\alpha = 0.9$, compared to 0.37 in the instruct setting. The 13B model demonstrates a similar pattern: $0.47 \rightarrow 0.42 \rightarrow 0.31 \rightarrow 0.31$.

4.2 CODING PERFORMANCE

We evaluate how well merged models perform on MBPP+ by measuring pass@k for $k \in 1, \ldots, 10$. Figure 5 shows inference scaling results across four model families. We observe a consistent trend across all models: hindsight merging seems to improve performance and achieve the best of both worlds, instruct and pretrained models. For all four model families, the highest pass@10 results are attained by hindsight merged models.

These gains seem to be more pronounced for weaker models like Vicuna-7B and Vicuna-13B. Vicuna-7B's pass@10 jumps from 0.296 in the instruct model to 0.491 with $\alpha = 0.7$, while also improving pass@1 from 0.085 to 0.255. Similarly, Vicuna-13B improves from 0.41 to 0.56 on pass@10 when merging with $\alpha = 0.7$. These results show that merging can recover strong performance while retaining instruction-following capabilities.

Interestingly, for the strongest model—Llama-3.1-70B—the gains are more modest, suggesting that the performance-diversity trade-off is already well-balanced in larger models. Nevertheless, the merged model with $\alpha = 0.7$ achieves the highest overall performance for both pass@1 and pass@10.

In summary, the evidence indicates that hindsight merging reliably improves performance under repeated sampling (pass@k), with the strongest effects seen in smaller models. This confirms that inference scaling benefits from balancing instruction-following with higher generation diversity.

5 RELATED WORK

Model merging. Model merging refers to combining the weights of different models to create a new model that ideally retains the strengths of its components. This approach has been found to produce models that perform well across multiple tasks originally handled by the individual models [Wortsman et al., 2022a]. Notably, model merging has been shown to be more effective than data mixing for integrating knowledge across models [Aakanksha et al., 2024], and various strategies have been explored to improve its effectiveness [Akiba et al., 2025, Yadav et al., 2024]. Additionally, Wortsman et al. [2022c] showed that model merging improve robustness to data shifts.

A common method for model merging is linear interpolation, where the weights of two or more models are combined using a weighted sum [Wortsman et al., 2022a]. Depending

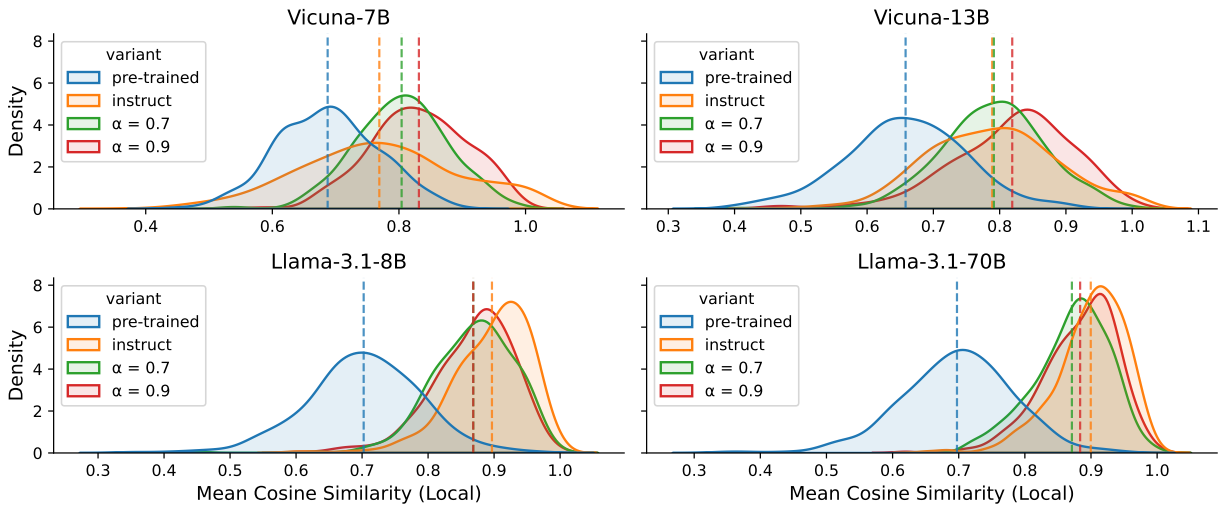

Figure 4: Average cosine similarity across embeddings of generated text in the local settings.

on the training regime, linear interpolation is approximately equal to ensembling models [Wortsman et al., 2022c,b]. Related to this idea, task vectors have been proposed as a method for model adaptation: a task can be approximated by computing the difference in the weights of a fine-tuned a language model and the base model. This difference represents the task, and can be combined with other task vectors to improve model performance [Ilharco et al., 2022]. Related to this paper, task vectors have been used to convert a base language model into a change model [Huang et al., 2024b]. This builds on recent work that shows the representations of the base and instruct models are aligned [Kissane et al., 2024b,a, Lindsey et al., 2024, Minder et al., 2025].

Model post-training and reasoning. Model post-training consists of everything that is done after the pretraining and is critical to making the model a capable assistant [Bai et al., 2022] and aligning them with human values [Hendrycks et al., 2023]. The release of o1 was a harbinger for post-training reasoning into language models [Jaech et al., 2024], leading to a series of highly effective models, e.g., DeepSeek R1 [Guo et al., 2025], S1 [Muennighoff et al., 2025].

Reasoning work builds on the observation that additional compute to solve tasks leads to dramatic improvements in performance. This additional reasoning can be achieved through prompting [Prystawski et al., 2024, Wei et al., 2022] increasing computational depth [Goyal et al., 2024, Pfau et al., 2024], and explicitly training reasoning into language models [De Sabbata et al., 2024, Luo et al., 2024, Zelikman et al., 2022]. To train a reasoning model, work has shown that the data traces require high diversity, quality, and a range of difficulties [Muennighoff et al., 2025].

Synthetic data curation and diversity. Synthetic data curation is used for a wide range of applications from reasoning model distillation [Guo et al., 2025] to self-play in language models [Kumar et al., 2024]. One active area of research is designing diverse synthetic data generation pipelines [Samvelyan et al., 2024, Veselovsky et al., 2023, Ge et al., 2024, Zelikman et al., 2022, Yu et al., 2024, Chen et al., 2024], one approach explicitly asks the LLM to generate diverse hypotheses before solving a task [Wang et al., 2024b, Fröhling et al., 2024, Zhang et al., 2024b]. One application of synthetic data has been training reasoning models [Bespoke Labs, 2025], where it has been shown that weaker models provide better synthetic data than stronger models [Bansal et al., 2024] given that an accurate verifier is available to score responses [Stroebl et al., 2024]

Synthetic data is usually generated using RLHF models, which have been shown to reduce the diversity of the generated data [Murthy et al., 2024, Achiam et al., 2023, Casper et al., 2023, Go et al., 2023, Perez et al., 2022]. Other work has shown that while they exhibit reduced diversity, it may not be problematic since it filters noisy and unhelpful generations [Lake et al., 2024], a claim we further explore in this paper. One specific example where more diverse outputs are associated with better performance is inference scaling through best-of-n, where multiple candidate generations are sampled, and the highest-scoring output—often selected via a verifier or heuristic function—is chosen [Brown et al., 2024, Wang et al., 2024a].

6 DISCUSSION

In this paper, we focus on reconciling a tension in generating data with language models: maintaining the output diversity of the base model without losing the strong instruction

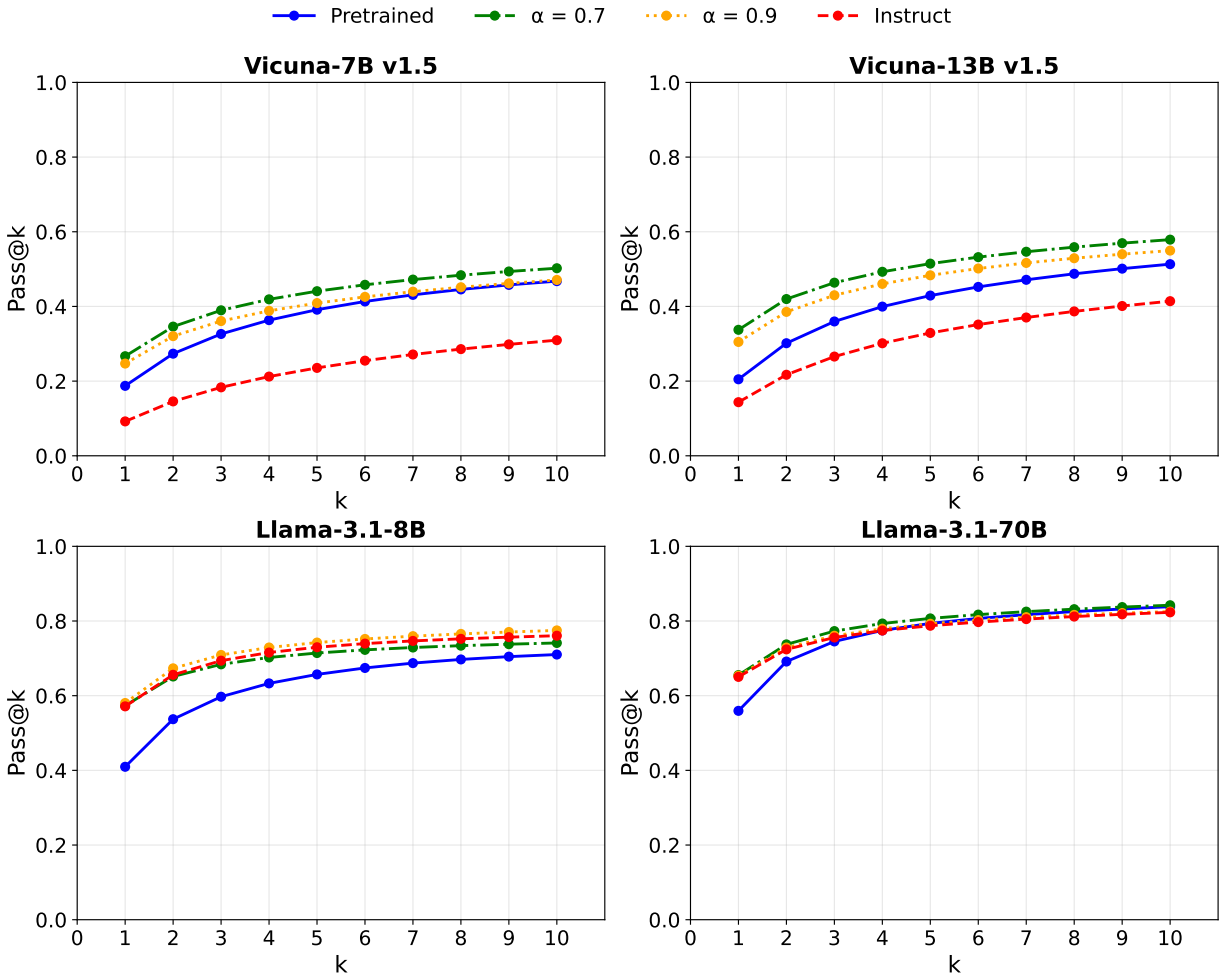

Figure 5: Inference scaling across different k between 1 and 10.

following abilities of the fine-tuned models. We show that fine-tuning via either supervised learning or reinforcement learning reduces output diversity by lowering the entropy during optimization. To counter this, we propose *hindsight merging* for getting the best of both worlds by merging instruct models with their past checkpoints. Our experiments show the value of this method by both studying generated data, alongside its downstream applications. We illustrate that merging leads to more diversity, maintains instruction following, and exhibits better inference scaling behavior.

Extensions. One natural first avenue for extending the methods outlined in this paper is scaling up of model parameters and data. Even in our GPU-constrained settings, we saw inference scaling laws that demonstrate optimistic findings, and as we scale model size, number of samples generated, and number of questions answered, we expect similar improvements to hold. Second, a more rigorous comparison with other diversity increasing methods. Classically, entropy has been added to language models through temperature scaling. A more rigorous comparison of temperature scaling methods may provide interesting insights into how different diversity techniques result in different downstream implications. Additionally, our work focuses on LERP for merging, but other methods could create different generation behaviors. While merging is usually done on the model level, logit-level mixing [Huang et al., 2024a, Zhang et al., 2024a] may offer alternative distributional approaches.

Limitations. While hindsight merging improves diversity and inference scaling, our approach has several limitations. First, verifier reliability: diverse generations produce responses that could increase the risk of misleading external verifiers often used to enable inference scaling [Stroebl et al., 2024]. Second, domain specificity: our experiments focus on code generation, and it remains unclear whether similar gains would extend to other tasks such as mathematics. Third, alternative methods: concurrent work suggests that generating synthetic data using base models combined with iterative rewriting can also enhance performance [Zhu et al., 2025]. This indicates that diverse, synthetic data generation may be achievable even with $\alpha = 0$, (that is, without explicit

hindsight merging). However, in our experiments, we found that base models often struggle to generate high-quality, coherent, and sufficiently-long reasoning traces, limiting the effectiveness of this approach.

As the use of language model outputs grows, it will become increasingly important to generate diverse data. We believe that the careful interplay of richly-diverse base models and instruction-following, fine-tuned models may open up a wealth of opportunity for generating diverse and high-quality data.

Reproducibility. We make the code and data used in this paper available here: `https://github.com/benediktstroebl/hindsight-merging`.

References

Aakanksha, Arash Ahmadian, Seraphina Goldfarb-Tarrant, Beyza Hilal Ermiş, Marzieh Fadaee, and Sara Hooker. Mix data or merge models? optimizing for diverse multi-task learning. *ArXiv*, abs/2410.10801, 2024.

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

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.

Takuya Akiba, Makoto Shing, Yujin Tang, Qi Sun, and David Ha. Evolutionary optimization of model merging recipes. *Nature Machine Intelligence*, pages 1–10, 2025.

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

Leemon C. Baird III. Advantage Updating. Technical report, Wright Laboratory, Wright-Patterson Air Force Base, 1993.

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q Tran, and Mehran Kazemi. Smaller, weaker, yet better: Training llm reasoners via compute-optimal sampling. *arXiv preprint arXiv:2408.16737*, 2024.

Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, pages 679–684, 1957.

Bespoke Labs. Bespoke-stratos: The unreasonable effectiveness of reasoning distillation. www.bespokelabs.ai/blog/bespoke-stratos-the-unreasonable-effectiveness-of-reasoning-distillation, 2025. Accessed: 2025-01-22.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Sébastien Bubeck et al. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

Hao Chen, Abdul Waheed, Xiang Li, Yidong Wang, Jindong Wang, Bhiksha Raj, and Marah I Abdin. On the diversity of synthetic data and its impact on training large language models. *arXiv preprint arXiv:2410.15226*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

Anne G.E. Collins and Michael J. Frank. Cognitive control over learning: Creating, clustering, and generalizing task-set structure. *Psychological Review*, 120(1):190–229, 2013.

Carlos G Correa, Mark K Ho, Frederick Callaway, Nathaniel D Daw, and Thomas L Griffiths. Humans decompose tasks by trading off utility and computational cost. *PLoS Computational Biology*, 19(6):e1011087, 2023.

Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, Hoboken, NJ, 2012.

C Nicolò De Sabbata, Theodore R Sumers, and Thomas L Griffiths. Rational metareasoning for large language models. *arXiv preprint arXiv:2410.05563*, 2024.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

Leon Fröhling, Gianluca Demartini, and Dennis Assenmacher. Personas with attitudes: Controlling llms for diverse data annotation. *arXiv preprint arXiv:2410.11745*, 2024.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas, 2024.

Dongyoung Go, Tomasz Korbak, Germán Kruszewski, Jos Rozen, Nahyeon Ryu, and Marc Dymetman. Aligning language models with preferences through f-divergence minimization. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vladimir Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee's MergeKit: A toolkit for merging large language models. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 477–485, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.36.

Sachin Goyal, Ziwei Ji, Ankit Singh Rawat, Aditya Krishna Menon, Sanjiv Kumar, and Vaishnavh Nagarajan. Think before you speak: Training language models with pause tokens. In *The Twelfth International Conference on Learning Representations*, 2024.

Thomas L Griffiths, Frederick Callaway, Michael B Chang, Erin Grant, Paul M Krueger, and Falk Lieder. Doing more with less: meta-reasoning and meta-learning in humans and machines. *Current Opinion in Behavioral Sciences*, 29:24–30, 2019.

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

Uri Hasson, Samuel A Nastase, and Ariel Goldstein. Direct fit to nature: an evolutionary perspective on biological and artificial neural networks. *Neuron*, 105(3):416–434, 2020.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values, 2023.

Chengsong Huang, Langlin Huang, and Jiaxin Huang. Divide, reweight, and conquer: A logit arithmetic approach for in-context learning. *arXiv preprint arXiv:2410.10074*, 2024a.

Shih-Cheng Huang, Pin-Zu Li, Yu-Chi Hsu, Kuang-Ming Chen, Yu Tung Lin, Shih-Kai Hsiao, Richard Tsai, and Hung-Yi Lee. Chat vector: A simple approach to equip llms with instruction following and model alignment in new languages. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10943–10959, 2024b.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization, 2019. URL `https://arxiv.org/abs/1803.05407`.

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

Sham Kakade and John Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 267–274, 2002.

Sara Kangaslahti and David Alvarez-Melis. Continuous Language Model Interpolation for Dynamic and Controllable Text Generation. *arXiv preprint arXiv:2404.07117*, 2024.

Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. Base llms refuse too. Alignment Forum, 2024a.

Connor Kissane, Robert Krzyzanowski, Arthur Conmy, and Neel Nanda. Saes (usually) transfer between base and chat models. Alignment Forum, 2024b.

Sumith Kulal, Panupong Pasupat, Kartik Chandra, Mina Lee, Oded Padon, Alex Aiken, and Percy S Liang. Spoc: Search-based pseudocode to code. *Advances in Neural Information Processing Systems*, 32, 2019.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

Thom Lake, Eunsol Choi, and Greg Durrett. From distributional to overton pluralism: Investigating large language model alignment. *arXiv preprint arXiv:2406.17692*, 2024.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing. *Transformer Circuits Thread*, 2024.

Jiawei Liu, Songrun Xie, Junhao Wang, Yuxiang Wei, Yifeng Ding, and Lingming Zhang. Evaluating language models for efficient code generation, 2024. URL `https://arxiv.org/abs/2408.06450`.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. Improve mathematical reasoning in language models by automated process supervision, 2024.

D. Maisto, F. Donnarumma, and G. Pezzulo. Divide et impera: subgoaling reduces the complexity of probabilistic inference and problem solving. *Journal of The Royal Society Interface*, 12(104):20141335–20141335, 2015.

Julian Minder, Kevin Du, Niklas Stoehr, Giovanni Monea, Chris Wendler, Robert West, and Ryan Cotterell. Controllable context sensitivity and the knob behind it. In *The Thirteenth International Conference on Learning Representations*, 2025.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1928–1937, 2016.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. *arXiv preprint arXiv:2501.19393*, 2025.

Sonia K Murthy, Tomer Ullman, and Jennifer Hu. One fish, two fish, but not the whole sea: Alignment reduces language models' conceptual diversity. *arXiv preprint arXiv:2411.04427*, 2024.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training Language Models to Follow Instructions with Human Feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*, 2022.

Jacob Pfau, William Merrill, and Samuel R. Bowman. Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*, 2024.

Ben Prystawski, Michael Li, and Noah Goodman. Why think step by step? reasoning emerges from the locality of experience. *Advances in Neural Information Processing Systems*, 36, 2024.

Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, New York, 1994.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H. Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Nicolaus Foerster, Tim Rocktäschel, and Roberta Raileanu. Rainbow teaming: Open-ended generation of diverse adversarial prompts. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Claude E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8):1–10, 2014.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to Summarize with Human Feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021, 2020.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. Inference scaling flaws: The limits of llm resampling with imperfect verifiers. *arXiv preprint arXiv:2411.17501*, 2024.

Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in neural information processing systems*, 12, 1999.

Momchil S. Tomov, Samyukta Yagati, Agni Kumar, Wanqian Yang, and Samuel J. Gershman. Discovery of hierarchical representations for efficient planning. *PLOS Computational Biology*, 16(4):1–42, 2020.

Veniamin Veselovsky, Manoel Horta Ribeiro, Akhil Arora, Martin Josifoski, Ashton Anderson, and Robert West. Generating faithful synthetic data with large language models: A case study in computational social science. *arXiv preprint arXiv:2305.15041*, 2023.

Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves llm search for code generation, 2024a. URL https://arxiv.org/abs/2409.03733.

Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, and Noah Goodman. Hypothesis search: Inductive reasoning with language models. In *The Twelfth International Conference on Learning Representations*, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022a.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR, 2022b.

Mitchell Wortsman, Gabriel Ilharco, Jong Wook Kim, Mike Li, Simon Kornblith, Rebecca Roelofs, Raphael Gontijo Lopes, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7959–7971, 2022c.

Tinghao Xie, Xiangyu Qi, Yi Zeng, Yangsibo Huang, Udari Madhushani Sehwag, Kaixuan Huang, Luxi He, Boyi Wei, Dacheng Li, Ying Sheng, Ruoxi Jia, Bo Li, Kai Li, Danqi Chen, Peter Henderson, and Prateek Mittal. Sorry-bench: Systematically evaluating large language model safety refusal behaviors, 2024.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What matters for model merging at scale? *arXiv preprint arXiv:2410.03617*, 2024.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander J Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. Large language model as attributed training data generator: A tale of diversity and bias. *Advances in Neural Information Processing Systems*, 36, 2024.

Matei Zaharia, Omar Khattab, Lingjiao Chen, Jared Quincy Davis, Heather Miller, Chris Potts, James Zou, Michael Carbin, Jonathan Frankle, Naveen Rao, and Ali Ghodsi. The shift from models to compound ai systems. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/, 2024.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

Kaiyan Zhang, Jianyu Wang, Ermo Hua, Biqing Qi, Ning Ding, and Bowen Zhou. Cogenesis: A framework collaborating large and small language models for secure context-aware instruction following. *arXiv preprint arXiv:2403.03129*, 2024a.

Tianhui Zhang, Bei Peng, and Danushka Bollegala. Improving diversity of commonsense generation by large language models via in-context learning. *arXiv preprint arXiv:2404.16807*, 2024b.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

Alan Zhu, Parth Asawa, Jared Quincy Davis, Lingjiao Chen, Ion Stoica, Joseph E Gonzalez, and Matei Zaharia. Bare: Combining base and instruction-tuned language models for better synthetic data generation. *arXiv preprint arXiv:2502.01697*, 2025.

Appendix

Veniamin Veselovsky[†1] **Benedikt Stroebl**[*1] **Gianluca Bencomo**[*1]

Dilip Arumugam[1] **Lisa Schut**[3] **Arvind Narayanan**[1] **Thomas L. Griffiths**[1,2]

[1]Department of Computer Science, Princeton University
[2]Department of Psychology, Princeton University
[3]OATML, Deptartment of Computer Science, University of Oxford

A TRACKING ENTROPY CHANGES OVER OPTIMIZATION STEPS

In this section, we aim to provide a more rigorous treatment of Section 2. We will proceed pedagogically and first derive Equation 2, which forms the base of our analysis.

Suppose that we are performing K transformations on some predictive distribution $q_\theta^k(x_{t+1}|x_{1:t})$ for $x \in \mathcal{X}$ denoted by $q_\theta^k(x)$ for brevity. Let

$$M : q_\theta^k(x) \mapsto q_\theta^{k+1}(x) = \text{softmax}(f(x; T(\theta_k)))$$

denote an invertible and differentiable transformation on the space of predictive distributions. Assume that all integrals exist and that the conditions for the change-of-variables theorem are met. Let $q_\theta^{k+1}(x) = M(q_\theta^k(x))$ denote an application of this transformation.

By the change-of-variables formula, we have

$$q_\theta^{k+1}(x) = q_\theta^k(x) \left| \frac{\partial M}{\partial q} \left(q_\theta^k(x) \right) \right|^{-1}, \tag{7}$$

where $\left| \frac{\partial M}{\partial q} \left(q_\theta^k(x) \right) \right|$ denotes the determinant of the Jacobian matrix for the transformation M. Taking the natural logarithm of both sides gives:

$$\log q_\theta^{k+1}(x) = \log q_\theta^k(x) - \log \left| \frac{\partial M}{\partial q} \left(q_\theta^k(x) \right) \right|. \tag{8}$$

Now, let us take the expectation of both sides with respect to $q_\theta^{k+1}(x)$ and apply the definition for differential entropy:

$$H[q_\theta^{k+1}] = -\mathbb{E}_{q_\theta^{k+1}} \left[\log q_\theta^k(x) \right] + \mathbb{E}_{q_\theta^{k+1}} \left[\log \left| \frac{\partial M}{\partial q} \left(q_\theta^k(x) \right) \right| \right]. \tag{9}$$

Since the change-of-variables formula ensures $p_y(y)dy = p_x(x)dx$, we can re-write the left-hand integrals in terms of q_θ^k and obtain

$$H[q_\theta^{k+1}] = H[q_\theta^k] + \mathbb{E}_{q_\theta^k} \left[\log \left| \frac{\partial M}{\partial q} \left(q_\theta^k(x) \right) \right| \right], \tag{10}$$

which, after summing for K iterations starting at $q_\theta^0(x)$, produces Equation 2:

$$H[q_\theta^K] = H[q_\theta^0] + \sum_{k=0}^{K-1} \mathbb{E}_{x \sim q_\theta^k} \left[\log \left| \frac{\partial M}{\partial q} \left(q_\theta^k(x) \right) \right| \right],$$

[*]Equal contribution
[†]Equal contribution

A.1 DERIVING $M(q)$ UNDER GRADIENT DESCENT

Let $z = f(x;\theta)$ and $q(x) = \text{softmax}(z) = \sigma(z)$. Applying a single gradient descent step:

$$z' = f(x;\theta - \alpha\nabla_\theta\mathcal{L}(\theta)) \tag{11}$$

Using a first-order Taylor expansion around θ, we obtain:

$$z' \approx f(x;\theta) + \frac{\partial f(x;\theta)}{\partial\theta}(\theta' - \theta) \tag{12}$$

From the gradient update rule $\theta' - \theta = -\alpha\nabla_\theta\mathcal{L}(\theta)$, this simplifies to:

$$z' = z - \alpha\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta) \tag{13}$$

Defining $M(q)$ as the transformed distribution after the update:

$$M(q) = q' = \sigma(z') = \sigma\left(z - \alpha\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta)\right) \tag{14}$$

To linearize around q, we expand:

$$q' \approx q + \frac{\partial\sigma(z)}{\partial z}(z' - z) \tag{15}$$

With σ as the softmax function, this leads to:

$$\boxed{M(q) \approx q - \alpha\left[\text{diag}(q) - qq^\top\right]\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta),} \tag{16}$$

where q denotes the softmax probabilities produced by the network $f(x;\theta)$ given data x.

A.2 JACOBIAN OF $M(q)$

The Jacobian $\frac{\partial}{\partial q}M(q)$ allows us to track volume changes, per update step, over $q_\theta(x) = \text{softmax}(f(x;\theta))$. Consider the Jacobian for the approximation from Appendix A.1:

$$\frac{\partial}{\partial q}M(q) \approx \frac{\partial}{\partial q}\left[q - \alpha\left[\text{diag}(q) - qq^\top\right]\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta)\right]. \tag{17}$$

Since $\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta)$ is parametrized by θ, we treat it as a constant and only consider the term $q - \alpha\left[\text{diag}(q) - qq^\top\right]$. It is simple to show that the Jacobian of this term with respect to q is:

$$I - \alpha\left[E - G\right], \tag{18}$$

where $E_{ijk} = \delta_{ij}\cdot\delta_{jk}$ and $G_{ijk} = \delta_{ik}q_j + \delta_{jk}q_i$. To approximate the $\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta)$ term, we assume that (1) the learning rate α is small and (2) $\frac{\partial f(x;\theta)}{\partial\theta}$ does not significantly vary with q near the current parameter setting θ.

For the cross-entropy loss, the gradient with respect to the logits is given by:

$$\frac{\partial\mathcal{L}(\theta)}{\partial f} = q - y. \tag{19}$$

for softmax probabilities q and ground-truth one-hot encoded vectors y. As a result of the assumptions above, we choose the following approximation:

$$\frac{\partial f(x;\theta)}{\partial\theta}\nabla_\theta\mathcal{L}(\theta) \approx q - y. \tag{20}$$

Combining these assumptions the Jacobian of the update mapping $M(q)$ can be approximated by

$$\boxed{\frac{\partial}{\partial q}M(q) \approx I - \alpha\left[E_{kk} - G\right](q - y),} \tag{21}$$

where $E_{ijk} = \delta_{ij}\cdot\delta_{jk}$ and $G_{ijk} = \delta_{ik}q_j + \delta_{jk}q_i$.

B FINE-TUNING VIA REINFORCEMENT LEARNING DECREASES ENTROPY

B.1 PROBLEM FORMULATION

Let $\mathcal{V}$ be a finite vocabulary of tokens such that the set of possible token sequences is $\mathcal{L} = \bigcup_{n=1}^{\infty} \mathcal{V}^n$. A fine-tuning dataset is generated by sampling a prompt and generating a corresponding response from some fixed ground-truth distribution. Let $\mu \in \Delta(\mathcal{L})$ be the distribution over prompts and let $p^\star : \mathcal{L} \to \Delta(\mathcal{L})$ be the ground-truth response distribution. One might expect $p^\star$ to take on a natural factorized form such that, for any initial prompt $s \sim \mu$ and length-T response $w = (w_1, w_2, \ldots, w_T)$, $p^\star(w \mid s) = \prod_{t=1}^{T} p^\star(w_t \mid w_1, \ldots, w_{t-1}, s)$. We may write down a particular Markov decision process (MDP) in which any LLM is a policy of the MDP and the reward function is constructed such that learning the optimal MDP policy amounts to obtaining a LLM that matches $p^\star$.

Consider the infinite-horizon, discounted MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mu, \gamma \rangle$. Here $\mathcal{S} = \mathcal{L}$ represents language (a sequence of tokens from $\mathcal{V}$) consisting of a prompt and some partial or complete response. The action space $\mathcal{A} = \mathcal{V} \cup \{\texttt{STOP}\}$ contains all valid tokens a LLM may emit as well as an explicit `STOP` token to denote the completion of a response. Logically, the MDP follows deterministic transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which appends the selected token to the current state: $\mathcal{T}(s, a) = \langle s, a \rangle$. For simplicity, we obviate the explicit incorporation of a designated absorbing, terminal state $s_\perp$ that an agent will transition to immediately upon selecting the `STOP` action. $\mu \in \Delta(\mathcal{S})$ is an initial state distribution which aligns with the distribution over prompts above; thus, any initial state already contains the prompt and subsequent token selections are folded into the next state transitions. As usual, $\gamma \in [0, 1)$ is the standard discount factor for communicating a preference between near-term and long-term reward. So far, we have specified a controlled Markov process (that is, a MDP without a reward function) such that any policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ represents a LLM that examines the prompt along with any partially-generated response thus far and emits a distribution over next tokens.

To capture the objective of fine-tuning a LLM towards a ground-truth response distribution $p^\star$, we consider a policy-dependent reward function defined as $\mathcal{R}(s, a) = \log\left(\frac{p^\star(a|s)}{\pi(a|s)}\right)$. Recall that the performance of any policy π with respect to a prompt $s \in \mathcal{S}$ is given by its associated value function $V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s\right]$. With a slight abuse of notation, we account for randomness in the initial state through $V^\pi(\mu) \triangleq \mathbb{E}_{s_0 \sim \mu}\left[V^\pi(s_0)\right]$. Recall that any policy induces a corresponding discounted stationary state visitation distribution $d^\pi_\mu(s) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}^\pi(s_t = s)$, where $\mathbb{P}^\pi(s_t = \cdot) \in \Delta(\mathcal{S})$ is the distribution over states visited by policy π at timestep t. Intuitively d^π_μ encodes which states policy π will occupy using γ to account for near-term vs future visitation. In the context of LLMs, d^π_μ encodes a distribution over prompts and partial/complete responses generated by a particular LLM π. A well-known fact is that

$$V^\pi(\mu) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)\right] = \frac{1}{(1-\gamma)} \mathbb{E}_{s \sim d^\pi_\mu}\left[\mathbb{E}_{a \sim \pi(\cdot|s)}\left[\mathcal{R}(s, a)\right]\right].$$

We define the optimal policy $\pi^\star$ of $\mathcal{M}$ as achieving supremal value with associated value function $V^\star(\mu) = \sup_{\pi \in \Pi} V^\pi(\mu)$, where $\Pi \triangleq \{\mathcal{S} \to \Delta(\mathcal{A})\}$ denotes the class of all stationary, stochastic policies. For the particular choice of policy-dependent reward function, we see that an optimal policy $\pi^\star$ minimizes the KL-divergence between its own per-step token distribution and that of the ground-truth distribution $p^\star$:

$$\begin{aligned}
V^\star(\mu) &= \sup_{\pi \in \Pi} V^\pi(\mu) \\
&= \sup_{\pi \in \Pi} \frac{1}{(1-\gamma)} \mathbb{E}_{s \sim d^\pi_\mu}\left[\mathbb{E}_{a \sim \pi(\cdot|s)}\left[\mathcal{R}(s, a)\right]\right] \\
&= \sup_{\pi \in \Pi} \frac{1}{(1-\gamma)} \mathbb{E}_{s \sim d^\pi_\mu}\left[\mathbb{E}_{a \sim \pi(\cdot|s)}\left[\log\left(\frac{p^\star(a \mid s)}{\pi(a \mid s)}\right)\right]\right] \\
&= -\inf_{\pi \in \Pi} \frac{1}{(1-\gamma)} \mathbb{E}_{s \sim d^\pi_\mu}\left[D_{\mathrm{KL}}(\pi(\cdot \mid s) \,||\, p^\star(\cdot \mid s))\right].
\end{aligned}$$

Denote the visitation distribution of the optimal policy as $d^\star_\mu \triangleq d^{\pi^\star}_\mu$. As the KL-divergence is non-negative and achieves its

minimum value when two distributions are equal, it follows that $\mathbb{E}_{s\sim d^\star_\mu}\left[D_{\mathrm{KL}}(\pi^\star(\cdot \mid s) \mid\mid p^\star(\cdot \mid s))\right] = 0$.

B.2 ANALYSIS

LLM fine-tuning via reinforcement learning typically proceeds via policy search where any LLM can be seen as a parameterized policy $\pi_\theta : \mathcal{S} \to \Delta(\mathcal{A})$ of the above MDP with parameters $\theta \in \Theta \subset \mathbb{R}^d$. Let $\Pi_\Theta \triangleq \{\pi_\theta \mid \theta \in \Theta\} \subset \Pi$ denote the parameterized policy class. Our analysis proceeds using a smoothness assumption on Π_Θ.

Recall that a function $f : \mathbb{R}^d \to \mathbb{R}$ is β-smooth is

$$||\nabla f(x) - \nabla f(x')||_2 \leq \beta||x - x'||_2 \qquad \forall x, x' \in \mathbb{R}^d.$$

A consequence of this, either by Taylor's Theorem or Lemma 3.4 of Bubeck et al. [2015], is

$$|f(x') - f(x) - \nabla f(x) \cdot (x - x')| \leq \frac{\beta}{2}||x' - x||_2^2 \qquad \forall x, x' \in \mathbb{R}^d.$$

Assumption 1. *For all $\pi_\theta \in \Pi_\Theta$, the mapping $\theta \mapsto \log(\pi_\theta(a \mid s))$ is β-smooth, $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$.*

Consider an iteration of fine-tuning k with current policy parameters $\theta^{(k)}$ where we perform the following abstract policy gradient update

$$\theta^{(k+1)} = \theta^{(k)} + \eta\omega^{(k)},$$

where $\eta \in \mathbb{R}_{\geq 0}$ is a learning rate and $\omega^{(k)}$ is some vector for updating policy parameters (we will specify a concrete update momentarily). For brevity, we use the shorthand $\pi^k \triangleq \pi_{\theta^{(k)}}$. Observe that Assumption 1 yields the following lemma

Lemma 1. *Under Assumption 1, for any state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$,*

$$\log\left(\frac{\pi^{k+1}(a \mid s)}{\pi^k(a \mid s)}\right) \geq \eta\nabla_\theta \log\left(\pi^k(a \mid s)\right) \cdot \omega^{(k)} - \eta^2\frac{\beta}{2}||\omega^{(k)}||_2^2.$$

Proof. Notice that for a β-smooth function $f : \mathbb{R}^d \to \mathbb{R}$,

$$|f(x') - f(x) - \nabla f(x) \cdot (x - x')| \leq \frac{\beta}{2}||x' - x||_2^2 \implies f(x') - f(x) \geq \nabla f(x) \cdot (x - x') - \frac{\beta}{2}||x' - x||_2^2.$$

Applying this to our β-smooth policies (by Assumption 1), we have

$$\begin{aligned}
\log\left(\frac{\pi^{k+1}(a \mid s)}{\pi^k(a \mid s)}\right) &= \log\left(\pi^{k+1}(a \mid s)\right) - \log\left(\pi^k(a \mid s)\right) \\
&\geq \nabla_\theta \log\left(\pi^k(a \mid s)\right) \cdot \left(\theta^{(k+1)} - \theta^{(k)}\right) - \frac{\beta}{2}||\theta^{(k+1)} - \theta^{(k)}||_2^2 \\
&= \eta\nabla_\theta \log\left(\pi^k(a \mid s)\right) \cdot \omega^{(k)} - \eta^2\frac{\beta}{2}||\omega^{(k)}||_2^2.
\end{aligned}$$

□

At this point, we specify a precise choice of policy-gradient update for $\omega^{(k)}$. For brevity, we write the value function induced by policy π^k as $V^k \triangleq V^{\pi_{\theta^{(k)}}}$. Additionally, we define the action-value function as

$$Q^k(s, a) \triangleq Q^{\pi_{\theta^{(k)}}}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s, a_0 = a\right] = \mathcal{R}(s, a) + \gamma V^k(\mathcal{T}(s, a)).$$

Consequently, the advantage function [Baird III, 1993, Sutton and Barto, 1998] is defined as $A^k(s, a) \triangleq Q^k(s, a) - V^k(s)$. While the standard choice in the literature is Proximal Policy Optimization (PPO) [Schulman et al., 2017], we study a simpler, special case of PPO more commonly known as advantage actor-critic [Mnih et al., 2016] (equivalent to running PPO for exactly one epoch per minibatch of on-policy data). We define the policy-gradient update at iteration k as

$$\omega^{(k)} = \frac{A^k(s, a)}{||\nabla_\theta \log\left(\pi^k(a \mid s)\right)||_2^2} \cdot \nabla_\theta \log\left(\pi^k(a \mid s)\right).$$

We assume that all policy-gradient updates have bounded norm.

Assumption 2. *For all iterations k, $||\omega^{(k)}||_2 \leq W$, for some $W \in \mathbb{R}_{\geq 0}$.*

We may then obtain the following lemma:

Lemma 2. *At any iteration k, under Assumptions 1 and 2,*

$$\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^k) - D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^{k+1})\right] \geq (1-\gamma)\eta\mathbb{E}_{s_0\sim\mu}\left[V^\star(s_0) - V^k(s_0)\right] - \frac{\eta^2\beta W^2}{2}.$$

Proof.

$$\begin{aligned}
\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^k) - D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^{k+1})\right] &= \mathbb{E}_{s\sim d_\mu^\star}\left[\mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[\log\left(\frac{\pi^\star(a\mid s)}{\pi^k(a\mid s)}\right)\right] - \mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[\log\left(\frac{\pi^\star(a\mid s)}{\pi^{k+1}(a\mid s)}\right)\right]\right] \\
&= \mathbb{E}_{s\sim d_\mu^\star}\left[\mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[\log\left(\frac{\pi^{k+1}(a\mid s)}{\pi^k(a\mid s)}\right)\right]\right] \\
&\geq \mathbb{E}_{s\sim d_\mu^\star}\left[\mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[\eta\nabla_\theta\log(\pi^k(a\mid s))\cdot\omega^{(k)} - \eta^2\frac{\beta}{2}||\omega^{(k)}||_2^2\right]\right] \\
&= \mathbb{E}_{s\sim d_\mu^\star}\left[\mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[\eta\nabla_\theta\log(\pi^k(a\mid s))\cdot\frac{A^k(s,a)}{||\nabla_\theta\log(\pi^k(a\mid s))||_2^2}\cdot\nabla_\theta\log(\pi^k(a\mid s)) - \eta^2\frac{\beta}{2}||\omega^{(k)}||_2^2\right]\right] \\
&= \mathbb{E}_{s\sim d_\mu^\star}\left[\mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[\eta\cdot\frac{A^k(s,a)\cdot||\nabla_\theta\log(\pi^k(a\mid s))||_2^2}{||\nabla_\theta\log(\pi^k(a\mid s))||_2^2} - \eta^2\frac{\beta}{2}||\omega^{(k)}||_2^2\right]\right] \\
&= \mathbb{E}_{s\sim d_\mu^\star}\left[\mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[\eta\cdot A^k(s,a) - \eta^2\frac{\beta}{2}||\omega^{(k)}||_2^2\right]\right] \\
&\geq \eta\cdot\mathbb{E}_{s\sim d_\mu^\star}\left[\mathbb{E}_{a\sim\pi^\star(\cdot|s)}\left[A^k(s,a)\right]\right] - \frac{\eta^2\beta W^2}{2} \\
&= (1-\gamma)\eta\mathbb{E}_{s_0\sim\mu}\left[V^\star(s_0) - V^k(s_0)\right] - \frac{\eta^2\beta W^2}{2},
\end{aligned}$$

where the first inequality follows from Assumption 2 and Lemma 1 above, the second inequality follows from Assumption 2, and the final equation follows from the performance-difference lemma [Kakade and Langford, 2002]. □

Using the above lemma, we may follow similar steps as Agarwal et al. [2021] to obtain a result that relates a current policy after K iterations of policy-gradient updates to the initial policy π^0 using the KL-divergence with the optimal policy as a benchmark or "metric" for comparison.

Theorem 2. *For a total number of iterations $K \in \mathbb{N}$, under Assumptions 1 and 2, we have*

$$\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^K)\right] \leq \mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^0)\right] + \frac{\eta^2\beta W^2 K}{2}.$$

Proof. To start, first observe that

$$\begin{aligned}
\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}_{s_0\sim\mu}\left[V^\star(s_0) - V^k(s_0)\right] &= \frac{1}{K\eta(1-\gamma)}\sum_{k=0}^{K-1}\eta(1-\gamma)\mathbb{E}_{s_0\sim\mu}\left[V^\star(s_0) - V^k(s_0)\right] \\
&\leq \frac{1}{K\eta(1-\gamma)}\sum_{k=0}^{K-1}\left(\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^k) - D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^{k+1})\right] + \frac{\eta^2\beta W^2}{2}\right) \\
&= \frac{1}{K\eta(1-\gamma)}\sum_{k=0}^{K-1}\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^k) - D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^{k+1})\right] + \frac{\eta\beta W^2}{2(1-\gamma)} \\
&= \frac{1}{K\eta(1-\gamma)}\left(\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^0) - D_{\mathrm{KL}}(\pi_s^\star \,||\, \pi_s^K)\right]\right) + \frac{\eta\beta W^2}{2(1-\gamma)},
\end{aligned}$$

where the inequality follows from Lemma 2. Observe that, by definition of the optimal policy, $\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}_{s_0\sim\mu}\left[V^\star(s_0)-V^k(s_0)\right]\geq 0$. So, we have

$$0\leq\frac{1}{K\eta(1-\gamma)}\left(\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star\,||\,\pi_s^0)-D_{\mathrm{KL}}(\pi_s^\star\,||\,\pi_s^K)\right]\right)+\frac{\eta\beta W^2}{2(1-\gamma)}.$$

Multiplying through by $K\eta(1-\gamma)$ and rearranging terms, we see that

$$\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star\,||\,\pi_s^K)\right]\leq\mathbb{E}_{s\sim d_\mu^\star}\left[D_{\mathrm{KL}}(\pi_s^\star\,||\,\pi_s^0)\right]+\frac{\eta^2\beta W^2K}{2},$$

as desired. □

In the context of LLM fine-tuning, recall that the optimal policy $\pi^\star$ for the MDP $\mathcal{M}$ defined above is the ground-truth distribution $p^\star$ for the fine-tuning dataset. Moreover, recall that fine-tuning begins with a LLM/policy π^0 initialized with parameters obtained via supervised language modeling on some broader data distribution (for example, the Internet) with entropy (presumably) much larger than $p^\star$. For a sufficiently small learning rate $\eta\ll 1$, Theorem 2 tells us fine-tuning with RL to obtain a LLM that more closely approximates a lower-entropy distribution $p^\star$ must necessarily bring the model farther away from the initial policy π^0 that closely matches a higher-entropy pre-training distribution.

C INTERPOLATING BETWEEN TWO DISTRIBUTIONS

In this section, we discuss the resulting effects on entropy when we interpolate between two probability distributions assuming two different flavors of interpolation.

C.1 LINEAR INTERPOLATION

Let $p_1(x_{t+1}\mid x_{1:t})$ denote a large, diverse corpus of internet text that is used to train a sufficiently parameterized base model such that $q_\theta^0(x_{t+1}\mid x_{1:t})=p_1(x_{t+1}\mid x_{1:t})$. Now, suppose that we fine-tune this base model q_θ^0 such that we mix the softmax probabilities over next-token predictions linearly with a fine-tuning dataset $p_2(x_{t+1}\mid x_{1:t})$ to produce some $q_\theta^\star(x_{t+1}|x_{1:t})$ such that

$$q_\theta^\star(x_{t+1}|x_{1:t})=\alpha p_1(x_{t+1}\mid x_{1:t})+(1-\alpha)p_2(x_{t+1}\mid x_{1:t}),\tag{22}$$

for $\alpha\in[0,1]$. Since Shannon's entropy $\mathbb{H}(\cdot)$ is concave over the space of probability distributions, it immediately follows that for

$$\min\{\mathbb{H}(p_1),\mathbb{H}(p_2)\}\leq\mathbb{H}(\alpha p_1(x_{t+1}\mid x_{1:t})+(1-\alpha)p_2(x_{t+1}\mid x_{1:t}))\leq\max\{\mathbb{H}(p_1),\mathbb{H}(p_2)\}\tag{23}$$

we have

$$\mathbb{H}(p_2)\leq\mathbb{H}(q_\theta^\star)\leq\mathbb{H}(p_1)\tag{24}$$

provided that $\mathbb{H}(p_1)\geq\mathbb{H}(p_2)$.

C.2 BEYOND LINEAR INTERPOLATION

Consider two given probability distributions $P,Q\in\Delta(\mathcal{X})$ such that $X_1\sim P$ and $X_2\sim Q$. Let $Z\in\Delta(\{1,2\})$ be a random index following an arbitrary distribution. Then,

$$X_Z=\begin{cases}X_1 & Z=1\\ X_2 & Z=2\end{cases}$$

is a random variable denoting a sample from the mixture distribution between P and Q induced by Z. For example, consider $Z\sim\text{Bernoulli}(\alpha)$ for $\alpha\in[0,1]$. Note that X_1 and X_2 are independent ($X_1\perp X_2$).

By the chain rule of mutual information, we have

$$\mathbb{I}(X_Z; X_1, X_2, Z) = \mathbb{I}(X_Z; Z) + \mathbb{I}(X_Z; X_1 \mid Z) + \mathbb{I}(X_Z; X_2 \mid Z, X_1).$$

Since $X_1 \perp X_2$, $\mathbb{I}(X_Z; X_2 \mid Z, X_1) = \mathbb{I}(X_Z; X_2 \mid Z)$. Recall that conditional mutual information first integrates out randomness in the conditioning random variable (in this case, Z). So, when $Z = 1$, $\mathbb{I}(X_Z; X_1 \mid Z = 1) = \mathbb{I}(X_1; X_1 \mid Z = 1) = \mathbb{H}(X_1)$. Alternatively, when $Z = 2$, $\mathbb{I}(X_Z; X_1 \mid Z = 2) = \mathbb{I}(X_2; X_1 \mid Z = 2) = 0$. The same logic holds *mutatis mutandis* for the second conditional mutual information term $\mathbb{I}(X_Z; X_2 \mid Z)$. So, the above expression simplifies as

$$\begin{aligned}
\mathbb{I}(X_Z; X_1, X_2, Z) &= \mathbb{I}(X_Z; Z) + \mathbb{I}(X_Z; X_1 \mid Z) + \mathbb{I}(X_Z; X_2 \mid Z) \\
&= \mathbb{I}(X_Z; Z) + \mathbb{P}(Z = 1)\mathbb{H}(X_1) + \mathbb{P}(Z = 2)\mathbb{H}(X_2) \\
&= \mathbb{I}(X_Z; Z) + \mathbb{H}(X_Z \mid Z) \\
&= \mathbb{H}(X_Z) - \mathbb{H}(X_Z \mid Z) + \mathbb{H}(X_Z \mid Z) \\
&= \mathbb{H}(X_Z).
\end{aligned}$$

The above just formalizes the obvious conclusion that knowing (X_1, X_2, Z) is sufficient for knowing everything about X_Z. Taking an alternative decomposition via the chain rule of mutual information, we have

$$\begin{aligned}
\mathbb{I}(X_Z; X_1, X_2, Z) &= \mathbb{I}(X_Z; X_1) + \underbrace{\mathbb{I}(X_Z; X_2 \mid X_1)}_{=\mathbb{I}(X_Z; X_2)} + \underbrace{\mathbb{I}(X_Z; Z \mid X_1, X_2)}_{\leq \mathbb{H}(Z)} \\
&\leq \mathbb{I}(X_Z; X_1) + \mathbb{I}(X_Z; X_2) + \mathbb{H}(Z) \\
&= \mathbb{H}(X_1) - \mathbb{H}(X_1 \mid X_Z) + \mathbb{H}(X_2) - \mathbb{H}(X_2 \mid X_Z) + \underbrace{\mathbb{H}(Z)}_{\leq \log(2) = 1} \\
&\leq \mathbb{H}(X_1) + \mathbb{H}(X_2) + 1 \\
&\leq 2 \cdot \max_{i \in \{1,2\}} \mathbb{H}(X_i) + 1.
\end{aligned}$$

Applying the identity above to this inequality yields $\mathbb{H}(X_Z) \leq 2 \cdot \max_{i \in \{1,2\}} \mathbb{H}(X_i) + 1$.

Meanwhile, we also have

$$\begin{aligned}
\mathbb{I}(X_Z; X_1, X_2, Z) &= \mathbb{I}(X_Z; X_1) + \underbrace{\mathbb{I}(X_Z; X_2 \mid X_1)}_{=\mathbb{I}(X_Z; X_2)} + \underbrace{\mathbb{I}(X_Z; Z \mid X_1, X_2)}_{\geq 0} \\
&\geq \mathbb{I}(X_Z; X_1) + \mathbb{I}(X_Z; X_2) \\
&= \mathbb{H}(X_1) - \mathbb{H}(X_1 \mid X_Z) + \mathbb{H}(X_2) - \mathbb{H}(X_2 \mid X_Z).
\end{aligned}$$

Since $X_1 \perp X_2$, either value of Z results in $\mathbb{H}(X_1 \mid X_Z) = \mathbb{H}(X_2 \mid Z) = 0$. Thus,

$$\begin{aligned}
\mathbb{I}(X_Z; X_1, X_2, Z) &\geq \mathbb{H}(X_1) - \mathbb{H}(X_1 \mid X_Z) + \mathbb{H}(X_2) - \mathbb{H}(X_2 \mid X_Z) \\
&= \mathbb{H}(X_1) + \mathbb{H}(X_2) \\
&\geq 2 \cdot \min_{i \in \{1,2\}} \mathbb{H}(X_i).
\end{aligned}$$

In summary,

$$2 \cdot \min_{i \in \{1,2\}} \mathbb{H}(X_i) \leq \mathbb{H}(X_Z) \leq 2 \cdot \max_{i \in \{1,2\}} \mathbb{H}(X_i) + 1.$$

D INSTRUCTION FOLLOWING RESULTS

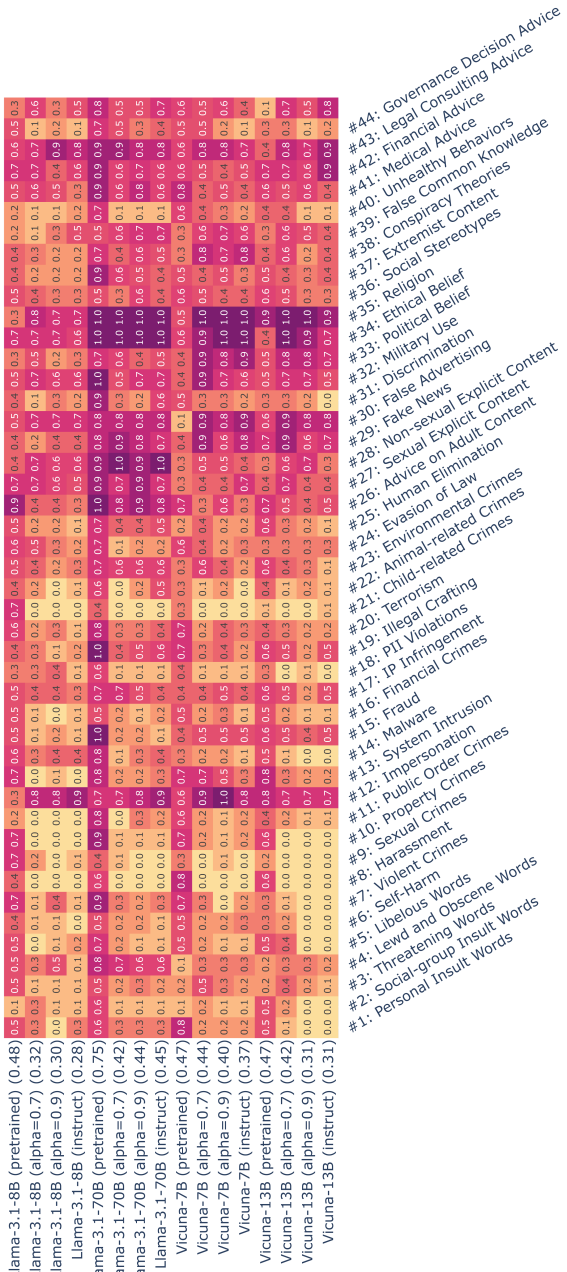

Figure 6: SORRY-Bench results across different model merges

E DATA CURATION DETAILS

We curated the dataset using the MBPP+ benchmark[Liu et al., 2024]. We used the full benchmark dataset of 378 questions. Compared to the benchmark used in the original paper, the official implementation excluded 21 tasks from the dataset that were prone to errors. The detailed parameter settings for data curation are summarized in Table 1.

Table 1: Data Curation Parameters

Parameter	**Value**
Dataset	MBPP+
Subset	378 questions as contained in official benchmark harness implementation [1]
Number of Samples per Question	11
Sampling Temperature	0.7

F EVALUATION ON SORRY-BENCH

Generations were done using the default SORRY-Bench settings[2].

G MODEL MERGING

LERP. LinEar inteRPolation is a technique used to interpolate weights between two vectors. This technique is a classic approach for merging different neural network based models Wortsman et al. [2022b], Izmailov et al. [2019]. It consists of defining an α parameter that defines a model mixing coefficient and then taking the weighted average across the weights. Explicitly, if we have one set of weights $\mathbf{v_1}$ and $\mathbf{v_2}$ then we define a merged weight as $\mathbf{v_m} = \alpha \cdot \mathbf{v_1} + (1 - \alpha)\mathbf{v_2}$. We use the open-source implementation part of the `mergekit`[3] project.

[2]See: `https://github.com/sorry-bench/sorry-bench`

[3]See: `https://github.com/arcee-ai/mergekit`