The Wisdom of Hindsight Makes Language Models Better Instruction Followers

Tianjun Zhang [*1] **Fangchen Liu** [*1] **Justin Wong** [1] **Pieter Abbeel** [1] **Joseph E. Gonzalez** [1]

Abstract

Reinforcement learning has seen wide success in finetuning large language models to better align with instructions via human feedback. The so-called algorithm, Reinforcement Learning with Human Feedback (RLHF) demonstrates impressive performance on the GPT series models. However, the underlying reinforcement learning algorithm is complex and requires additional training for reward and value networks. In this paper, we consider an alternative approach: converting feedback to instruction by relabeling the original one and training the model for better alignment in a supervised manner. Such an algorithm doesn't require any additional parameters except for the original language model and maximally reuses the pretraining pipeline. To achieve this, we formulate *instruction alignment* problem for language models as a *goal-reaching* problem in decision making. We propose Hindsight Instruction Relabeling (HIR), a novel algorithm for aligning language models with instructions. The resulting two-stage algorithm shed light to a family of reward-free approaches that utilize the hindsightly relabeled instructions based on feedback. We evaluate the performance of HIR extensively on 12 challenging BigBench reasoning tasks and show that HIR outperforms the baseline algorithms and is comparable to or even surpasses supervised finetuning. The implementation of HIR is available at `https://github.com/tianjunz/HIR`.

1. Introduction

Recent studies have shown that large language models could demonstrate unintended behavior when prompting it with an instruction (Bender et al., 2021; Bommasani et al., 2021; Weidinger et al., 2021). Such behavior is undesirable since the langue model could make up facts, generate toxic text or simply not be able to follow the intended behavior made by the instructions (Bender et al., 2021; Bommasani et al., 2021; Weidinger et al., 2021). As a result, a considerable amount of research effort has been put into designing better finetuning algorithms that can align the outputs of language models with human instructions (Leike et al., 2018; Askell et al., 2021). The most widely adopted approach is to deploy reinforcement learning (RL) algorithms to optimize for a manually defined or learned "alignment score" (Ouyang et al., 2022; Uesato et al., 2022). Impressive progress has been made in this direction, including the more recently released GPT series model (OpenAI, 2022).

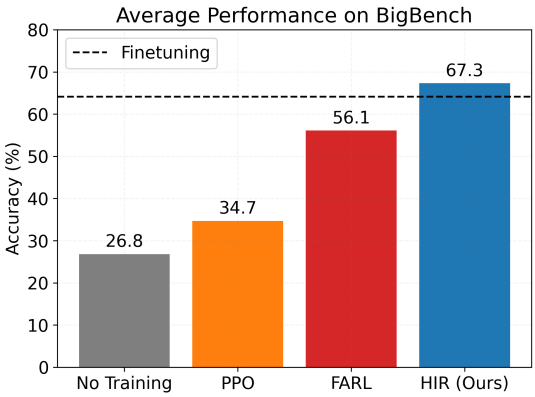

Figure 1. **Average Performance on BigBench.** HIR demonstrates a significant average performance gain over 12 tasks on BigBench compared to all baselines using FLAN-T5-Large.

Despite their good performance in the alignment, however, most of the prior work either uses Proximal Policy Optimization (PPO) (Schulman et al., 2017) to optimize for a trained alignment score module (Ouyang et al., 2022) or tries to apply imitation learning to a final-answer or reward-model filtered dataset (Uesato et al., 2022). The former approach is rather complex, sensitive to hyperparameters, and requires additional training in the reward model and value network. The latter one is less data-effective as it only makes use of the success instruction-output pairs, completely abandoning the ones that do not align.

[*]Equal contribution [1]University of California, Berkeley. Correspondence to: Tianjun Zhang <tianjunz@berkeley.edu>, Fangchen Liu <fangchen_liu@berkeley.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

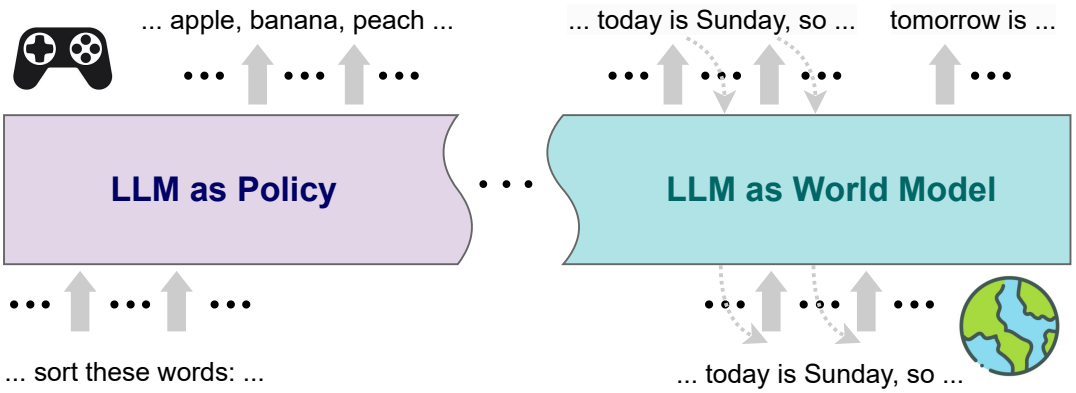

Figure 2. **Illustration of Large Language Model (LLM).** HIR views LLM as both a policy and a world model. Thus, HIR can collect data through interactions with LLM in the online sampling phase, and further improve the policy in the offline learning phase.

In this paper, we investigate whether we can design a simple finetuning algorithm that utilizes not only successful instruction-output pairs but also bootstrap from failed ones.

We first make the connection between the instruction alignment of language models and goal-reaching RL (Plappert et al., 2018), a special case of the general RL framework with an augmented goal space. This makes a straightforward correspondence, as we can view the instruction or task specification as the goal, and the language model as a goal-conditioned policy that can generate a sequence of word tokens to achieve the specified goal. To this end, a series of policy optimization algorithms (Andrychowicz et al., 2017; Eysenbach et al., 2022) tailored for goal-conditioned RL can be applied to the alignment problem of the language models.

The resulting algorithm we proposed, Hindsight Instruction Relabeling (HIR), adopts the central idea of relabeling the instructions in a hindsight fashion based on the generated outputs of the language model. HIR alternates between two phases: an online sampling phase to generate a dataset of instruction-output pairs, along with an offline learning phase that relabels the instructions of each pair and performs standard supervised learning. The algorithm does not require any additional parameters to train except the language model itself. We also adopt the relabeling strategy in HER (Andrychowicz et al., 2017) to make use of the failure data and use contrastive instruction labeling to improve the performance further.

We evaluate our algorithm extensively on 12 BigBench (Srivastava et al., 2022) language model reasoning tasks. The tasks we choose are very diverse, including logical deduction which requires logical understanding, object counting that involves math calculation, and geometric shapes that ask the model to understand the visual concept. We use the FLAN-T5 models (Chung et al., 2022) as the base model, comparing with the baselines of PPO (Schulman et al., 2017) and Final-Answer RL (Uesato et al., 2022). Results in Fig. 1 show that HIR significantly outperforms both baselines by 11.2% and 32.6% respectively. To summarize, our key contributions are:

- We propose a new perspective of learning from feedback via hindsight instruction relabeling, and connect the alignment problem of language model to goal-conditioned reinforcement learning.
- We propose a novel two-phase hindsight relabeling algorithm, which is more data-effective and doesn't require any additional RL training pipeline.
- Our method significantly outperforms baselines and is overall comparable to supervised fine-tuning (SFT) on 12 challenging BigBench reasoning tasks.

2. Related Work

Reinforcement Learning for Human Feedback Human feedback has been readily studied in the reinforcement learning setting(Ross et al., 2011; Kelly et al., 2019; Ibarz et al., 2018). Going as far back as inverse reinforcement learning to infer and model human preferences(Christiano et al., 2017; Wu et al., 2021; Lawrence & Riezler, 2018; Ziegler et al., 2019). More recent work starting with InstructGPT (Ouyang et al., 2022) has identified the benefits of RL for improving human alignment for open-vocabulary unstructured settings. In InstructGPT, humans wrote ground truth prompts on which GPT provided unsatisfactory responses, and a reward model was trained on this data to finetune GPT's responses. A similar line of work WebGPT utilizes human feedback from online data (Nakano et al., 2021). Although this approach requires expensive data collection, it as crucial to the successful release of ChatGPT (OpenAI, 2022), the first-of-its-kind general-purpose chatbots made available to the public. Our work focuses on

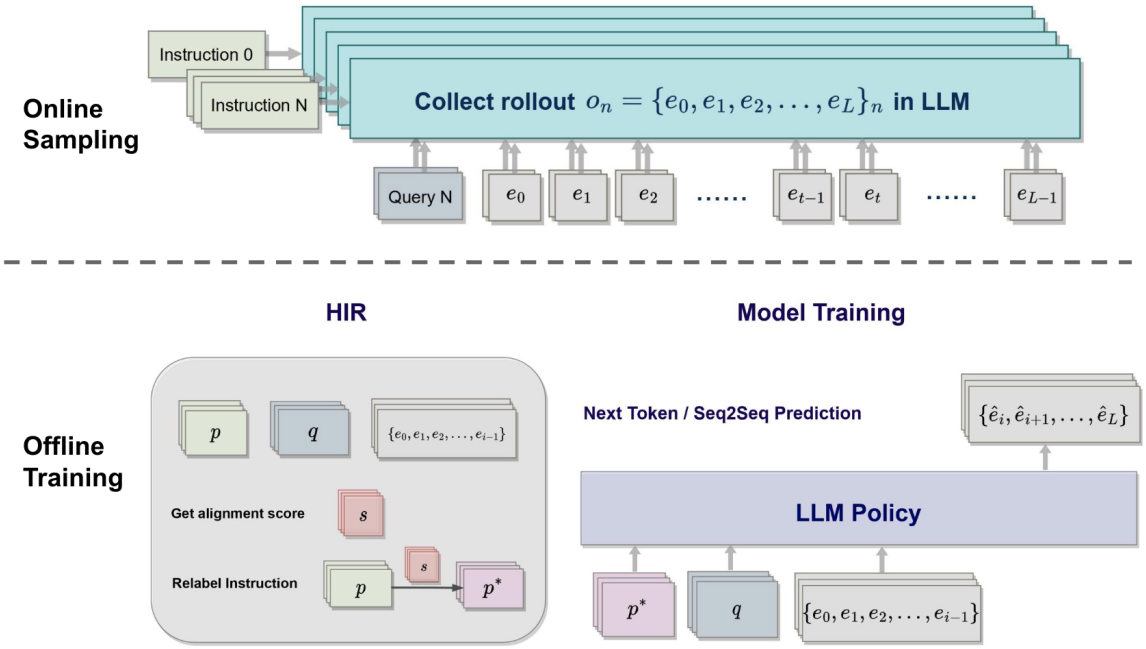

Figure 3. **Hindsight Instruction Relabeling.** HIR consists of two phases: online exploration phase and offline training phase. The algorithm alternates between the two phases until convergence.

the finetuning process for pretrained language models and offers a lighter-weight approach.

Prompt-Engineering Recent work has demonstrated that cleverly chosen prompts have the potential of dramatically improving pretrained LLM performance on specialized tasks from code generation to reasoning tasks (Wei et al., 2022; Zhou et al., 2022; Kojima et al., 2022). Another line of research lies in tuning prompt embedding vector through SGD (Lester et al., 2021; Liu et al., 2021b;a). There are also efforts on automatic prompt generation (Gao et al., 2020; Shin et al., 2020) or using RL for discrete prompt optimization (Zhang et al., 2022; Deng et al., 2022). In addition, multiple prior works have shown that combining finetuning and prompt engineering provides orthogonal benefits (Stiennon et al., 2020; Perez et al., 2021; Ouyang et al., 2022). Our approach avoids the manual effort required to prompt engineering for a specific task.

Two-stage Reinforcement Learning There have been numerous categories of work tackling offline reinforcement learning (Chen et al., 2021a; Janner et al., 2021; Jiang et al., 2022; Kumar et al., 2020). There have also been efforts to make transformers suitable for online exploration (Zheng et al., 2022). More recently, the Algorithm Distillation (AD) (Laskin et al., 2022) proposed a similar approach of alternating between online exploration and offline training. Note that HIR and AD tackles entirely different problems, while HIR focuses on improving language model alignment with RL, AD tackles the classical control problem. These ideas have been recently explored in finetuning language models as well (Huang et al., 2022; Li et al., 2022b; Zelikman et al., 2022).

Language Model with Reasoning Tasks. The tasks in our experiments require explicit reasoning steps for the language models. Solving math problem (Cobbe et al., 2021; Hendrycks et al., 2021; Ling et al., 2017) has long been an interesting application for this. More recently, a series of works have been focused on the multi-step reasoning part of the language models either by fine-tuning (Lewkowycz et al., 2022) or prompting (Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2022). These works have all along the effort to adopt language models for long-horizon reasoning tasks. Aside from these, there have also been works on trying to use language models for code generation (Li et al., 2022a; Chen et al., 2021b). This line of research also requires language to be capable of doing reasoning over the program tree structure.

3. Background

3.1. Reinforcement Learning Formulation

We can define a *Markov Decision Process* (MDP) by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$. $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space. $\mathcal{P}$ represents the transition probability $\mathcal{P}(s'|s,a)$, and $\mathcal{R}(s,a)$ is the reward function. The policy π is a mapping from $\mathcal{S}$ to $\mathcal{A}$. The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expectation of

the accumulated rewards $J(\pi) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)]$, where $a_t \sim \pi(a|s_t)$.

3.2. Goal-Conditioned Reinforcement Learning

Extending the previous RL setting to a multi-goal RL problem, we can augment standard MDP as $\langle \mathcal{G}, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \rangle$, where $\mathcal{G}$ represents the goal space. Meanwhile, both the reward function $\mathcal{R}(s, a, g)$ and policy $\pi(a|s, g)$ need to be goal-dependent. Thus, the objective is to find an optimal policy π^* that maximizes $J(\pi) = E_\pi[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, g_t)]$, where $a_t \sim \pi(a|s_t, g_t)$.

3.3. Align Language Models with Instruction

When dealing with instructions in language models, let $\mathcal{V}$ be the vocabulary (e.g., the set of predefined tokens) of a language model $\mathcal{M}$ and let $\mathbf{e}$ be the tokenized embedding of the model $\mathcal{M}$. For a simple example, an instruction (or prompt) may take the form: $\mathbf{p}$ = "Give the sentiment of the following sentence.", followed by the query $\mathbf{q}$ = "I like this movie." In this case, we want the language model to give its output $\mathbf{o}$ for the query $\mathbf{q}$ following the instruction $\mathbf{p}$.

How to align the model outputs with instructions remains an essential challenge. InstructGPT (Ouyang et al., 2022) proposes to first learn a reward model $\mathcal{R}(\mathbf{p}, \mathbf{q}, \mathbf{o})$, which can predict the alignment score based on human preference. Then it applies the standard RL pipeline to optimize the accumulated rewards.

4. Hindsight Instruction Relabeling

In this section, we will first discuss how one can formulate the language model alignment as a goal-conditioned RL problem in Sec. 4.1. Then we'll present an outline of our algorithm in Sec. 4.2. Finally, we will discuss the key concept of hindsight instruction relabeling in Sec. 4.3.

4.1. Instruction Following as Goal-conditioned RL

A language model $\mathcal{M}$ can take instructional prompt $\mathbf{p}$ and initial query token sequence $\mathbf{q} = \{\mathbf{q}_0, \ldots, \mathbf{q}_i\}$ as input, and autoregressively predict next token $\mathbf{e}_{i+1} = \mathcal{M}(\mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, \ldots, \mathbf{e}_i\})$. We can view standard prompt-conditioned language tasks (e.g. multi-step reasoning) as a goal-reaching problem, by formulating the MDP as follows:

- Goal space $\mathcal{G}$: space of instructional prompt $\mathbf{p}$
- State space $\mathcal{S}$: space of input token sequence $\mathbf{q} \cup \{\mathbf{e}_i\}$
- Action space $\mathcal{A}$: space of output token $\mathbf{e}_{i+1}$
- Transition probability $\mathcal{P}$: $\mathcal{M}(\mathbf{e}_{i+1}|\mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, \ldots, \mathbf{e}_i\})$
- Reward $\mathcal{R}$: alignment score of $\{\mathbf{e}_0, \ldots, \mathbf{e}_{i+1}\}$ with instruction $\mathbf{p}$ and query $\mathbf{q}$, can from human feedback or scripted feedback, which is not used in HIR.

Here all $\mathcal{G}$, $\mathcal{S}$ and $\mathcal{A}$ are space of token embeddings, but $\mathcal{G}$ corresponds to instructional prompts, while $\mathcal{S}$ and $\mathcal{A}$ corresponds to model inputs and outputs. In this way, we can also view the language model as a goal-conditioned policy:

$$\pi := \mathcal{M}(\mathbf{e}_{i+1}|\mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, .., \mathbf{e}_i\}) \tag{1}$$

Meanwhile, since the transition dynamics $\mathcal{P} = \mathcal{M}(\mathbf{e}_{i+1}|\mathbf{p}, \mathbf{q}, \{\mathbf{e}_0, \ldots, \mathbf{e}_i\})$ are also computed from the model outputs, we can also view this language model as a "world model" to interact with. Fig. 2 provides a pictorial illustration.

By observing this, there is a family of goal-conditioned RL algorithms, such as hindsight experience replay (HER) (Andrychowicz et al., 2017) that can potentially be applied for language model alignment.

4.2. Algorithm Overview

Inspired by the previous connection, we propose *Hindsight Instruction Relabeling*, a novel approach for instruction alignment. Similar to Algorithm Distillation (Laskin et al., 2022), HIR also consists of two phases: online sampling and offline relabeling. As shown in Fig. 2, the first stage is somewhat like the "exploration" stage. LLM can be viewed as a world model to interact and explore, where we can obtain data with reward (feedback). The second phase is to use the collected data to improve the policy. We discuss the two components respectively in the following sections.

Online Sampling. In the "online" sampling phase, we treat the model as both the environment and goal-conditioned policy. We want to mimic the exploration phase in the standard RL paradigm, where we often inject different noises into actions. In our case, we use a relatively large temperature τ for sampling. Specifically, given instruction $\mathbf{p}$ and query $\mathbf{q}$, we use $\tau = 1$ to get the output sequence $\mathbf{o} = \{\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_L\}$, which gives us the online replay dataset $\mathcal{D}_{\text{online}}$.

$$\mathcal{D}_{\text{online}} = \bigcup_{i=1}^{N} \left\{\mathbf{p}_i, \mathbf{q}_i, \mathbf{o}_i\right\} \tag{2}$$

Here each query $\mathbf{q}_i$ is sampled from the training dataset. Instruction prompt $\mathbf{p}_i$ is initialized to be a pre-defined sentence and will be corrected to align with the output $\mathbf{o}_i$ in the later stage.

Offline Relabeling. The key component of our algorithm is the offline relabeling part. In this part, for every instruction-output pair $(\mathbf{p}, \mathbf{q}, \mathbf{o})$ that are not necessarily

Algorithm 1 Two-Stage Hindsight Instruction Relabeling (HIR)

Input: Language Model $\mathcal{M}$, Initial Prompt $\mathbf{p}$, Training Set $\mathcal{D}_{\text{train}}$, Evaluation set $\mathcal{D}_{\text{eval}}$, Iteration N, Sampling Rounds T, Training Epochs K, Sampling Temperature τ, Empty RL dataset $\mathcal{D}_{\text{online}}$
for episode $n = 1, \cdots, N$ **do**
  for sampling rounds $i = 1, \cdots, T$ **do**
    Random sample batch of input queries $\mathcal{Q} \sim \mathcal{D}_{\text{train}}$
    Sample corresponding outputs $\mathbf{o_i} = \mathcal{M}(\mathcal{Q}, \mathbf{p}, \tau)$
    Appending the trajectory to RL Dataset $\mathcal{D}_{\text{online}} \leftarrow \mathcal{D}_{\text{online}} \cup (\mathcal{Q}, \mathbf{p}, \mathbf{o}_i)$
  end for
  for training rounds $t = 1, \cdots, K$ **do**
    Random sample batch of query-output pairs $(\mathcal{Q}, \mathcal{O}) \sim \mathcal{D}_{\text{online}}$
    Sample from $\mathcal{D}_{\text{online}}$ and apply relabeling as described in Sec. 4.3
    Train model $\mathcal{M}$ using loss in Eq. (6)
  end for
end for
Evaluate policy π_θ on evaluation dataset $\mathcal{D}_{\text{eval}}$

aligned, we relabel this pair with a new instruction that can align with the outcome of the model $(\mathbf{p}^*, \mathbf{q}, \mathbf{o})$. For the reasoning tasks we evaluate HIR on, we just use simple binary feedback and instruction based on answer correctness as introduced in Appendix. A.2.

The new instruction $\mathbf{p}^*$ is generated based on the feedback function $\mathcal{R}(\mathbf{p}, \mathbf{q}, \mathbf{o})$ and the instruction generation function $\phi(\mathbf{p}, \mathbf{q}, \mathbf{o}, \mathbf{r})$, which can either be learned or scripted. For example, in the framework of RLHF, if the learned reward model $\mathcal{R}(\mathbf{p}, \mathbf{q}, \mathbf{o})$ generates a score that ranks about 75% as in the training data, we can give additional scripted instructions to the model such as "give me an answer that ranks about 75% in training data". However, as most human-feedback data is hard to collect, we adopt a scripted feedback function, which is similar to Final-Answer RL (FARL) (Uesato et al., 2022). For simplicity, ϕ is also scripted based on the correctness of the reasoning outcome.

The central difference between HIR and FARL (Uesato et al., 2022) is whether to use hindsight experience. In FARL, the algorithm filters out the correct alignment instruction-output pairs and conducts imitation learning, while our relabeling procedure enables learning from failure cases as well.

After we got the relabeled instructions, we can perform standard supervised learning for these instruction-output pairs. We perform the standard seq2seq loss $\mathcal{L}_{\text{supervise}}$ to train our model.

Full Pipeline. Our full algorithm HIR is shown in Algorithm 1. The algorithm alternates between the online sampling phase to generate a dataset and the offline instruction relabeling phase for model improvement.

4.3. Instruction Relabeling

Performing offline instruction relabeling is crucial to the success of the algorithm. HER (Andrychowicz et al., 2017) relabels every *transition*[1] in order to improve the goal-conditioned policy at all times. Similar to HER, we conduct instruction relabeling at intermediate time steps on the generated sub-output.

In addition to hindsight relabeling, we also introduce a contrastive instruction labeling loss to push up the probability of a particular instruction-output pair but push down the other instruction-output pairs.

Sub-output Relabeling It is important to sample partial outputs and relabel the instruction. In this way, we could give more dense feedback through instruction relabeling. Note that one can flexibly control the granularity that we want the algorithm to provide this dense feedback. In another word, one could provide feedback at a sentence level or a paragraph level.

Consider we relabel the i-th time step. The input to the model is $\mathbf{q} \cup \{\mathbf{e}_0, ..., \mathbf{e}_{i-1}\}$. We can edit the instruction as a future goal based on the future alignment score:

$$\mathbf{p}^* = \phi\Big(\mathbf{p}, \mathbf{q}, \{\mathbf{e}_i, ..., \mathbf{e}_L\}, \mathcal{R}\big(\mathbf{p}, \mathbf{q}, \{\mathbf{e}_i, ..., \mathbf{e}_L\}\big)\Big)$$

where ϕ and $\mathcal{R}$ are the instruction generation function and feedback function as described in Sec. 4.2. The model takes new inputs $\mathcal{M}(\mathbf{p}^*, \mathbf{q}, \{\mathbf{e}_0, ..., \mathbf{e}_{i-1}\})$ and is trained to match the prediction target $\{\mathbf{e}_i, ..., \mathbf{e}_L\}$, and get the seq2seq loss $\mathcal{L}_{\text{supervise}}$ as in (Raffel et al., 2020). More details about relabeling can be found at Appendix. A.2.

We sample trajectories from the data collected during online

[1] (s, a, s') tuple with goal replacement g

Table 1. Examples of inputs and outputs for the BigBench tasks. For multiple-choice tasks, we provide the options that the language model can choose from as prompts.

	Tasks	Example Inputs	Outputs
Multiple Choice	Logical Deduction	"Q: The following paragraphs each describe a set of three objects arranged in a fixed order. The statements are logically consistent within each paragraph. In a golf tournament, there were three golfers: Amy, Eli, and Eve. Eve finished above Amy. Eli finished below Amy. Options: (A) Amy finished last (B) Eli finished last (C) Eve finished last"	"(B)"
	Date Understanding	"Q: Today is Christmas Eve of 1937. What is the date 10 days ago? Options: (A) 12/14/2026 (B) 12/14/2007 (C) 12/14/1937"	"(C)"
Direct Generation	Object Counting	"Q: I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have?"	"6"
	Word Sorting	"Sort the following words alphabetically: List: oven costume counterpart."	"costume counterpart oven"

interaction in D_{online} and then uniformly sample different timestep i using the relabeling process as above.

Contrastive Instruction Following. We also introduce the contrastive instruction labeling along with the standard fine-tuning loss in our offline instruction relabeling phase. Suppose $\mathbf{o}_i = \mathcal{M}(\mathbf{q}_i, \mathbf{p}_i)$. Given the log probability of $\mathbf{o}_i$ conditioned on $\mathbf{q}_k, \mathbf{p}_k$ as:

$$\mathcal{P}_{ik} = \log P_{\mathcal{M}}(\mathbf{o}_i | \mathbf{q}_k, \mathbf{p}_k) \tag{3}$$

We define the following contrastive loss:

$$\mathcal{L}_{\text{contastive}} = -\sum_{i=1}^{n} \log \frac{\exp(\mathcal{P}_{ii})}{\sum_{k=1}^{n} \exp(\mathcal{P}_{ik})} \tag{4}$$

This helps to avoid the model learning the behavior that maps the same output for different instructions, and also benefits the online phase as the loss pushes down the specific output for other instructions.

Entropy Regularization. As a common practice in RL, we apply entropy regularization to the output given a particular instruction. This negative entropy term ensures the sampling phase won't converge too early for better exploration.

$$\mathcal{L}_{\text{entropy}} = \sum_{i=1}^{n} \mathcal{P}_k \log \mathcal{P}_k \tag{5}$$

In practice, we add two coefficients α, β for the contrastive loss and entropy loss. So the final loss becomes:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{supervise}} + \alpha \mathcal{L}_{\text{contastive}} + \beta \mathcal{L}_{\text{entropy}} \tag{6}$$

5. Comparing to Previous Algorithms

HIR takes inspiration from HER and applies it to the language models. The resulting algorithm is simple (no extra parameter is required to train). We discuss the conceptual advantages of HIR comparing to several different previous algorithms (including RLHF, Algorithm Distillation (AD) (Laskin et al., 2022) and Final-Answer RL (FARL) (Uesato et al., 2022)) in this section.

Most closely, HIR takes a very similar approach comparing to AD. They both adopt the two-stage online sampling and offline training paradigm. However, they are inherently targeting at different domains: AD focuses on the control tasks while HIR is specifically tailored to language models. Moreover, as a goal-conditioned algorithm, HIR doesn't require any explicit modeling of reward or return. This significantly reduces the complexity of learning another reward or critic network, thus, yields a simple but elegant algorithm.

HIR is also related to the RLHF algorithm as they both try to learn from feedback to solve the instruction alignment problem. However, RLHF requires additional RL training. Since our dataset doesn't contain human feedback, we refer to it as PPO in the experiment sections. Compared with the standard PPO algorithm (Schulman et al., 2017), it exploits an additional KL penalty.

Compared to the FARL, HIR enables the algorithm to learn also from failure cases. Final-Answer RL only filters out the correct output from the sampling phase and uses them as the training data. With the capability of hindsight instruction relabeling, HIR handles failure data as well as successful ones. A more intuitive illustration can be found in Fig. 4.

6. Experiments

We conduct experiments with our method on the BigBench (Srivastava et al., 2022) tasks. Different from the traditional multiple-choice GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019) tasks, BigBench is a more challenging generation task that requires complex reasoning capabilities of the language models. We select a subset of the BigBench consisting of 12 complex tasks. The tasks we select are quite diverse, including reasoning the final results of a sequence of actions, understanding dates, and completing tasks that require simple arithmetic calculation. We compare against the standard reinforcement learning base-

Table 2. Performance of HIR on the 12 challenging BigBench reasoning tasks. Compared to all baselines including PPO and FARL, HIR achieves strong performance gain.

		Tracking Shuffled Objects (3)	Tracking Shuffled Objects (5)	Tracking Shuffled Objects (7)	Logical Deduction (3 Objects)
Pretrained	FLAN-T5-large	29.3	15.6	6.6	33.3
Supervised	Finetuning	**100.0**	17.0	13.4	90.0
Feedback	PPO	35.0	15.6	6.3	57.0
	FARL	90.0	15.6	10.0	86.7
	HIR (ours)	**100.0**	**61.2**	**42.6**	**91.7**

		Logical Deduction (5 Objects)	Logical Deduction (7 Objects)	Date Understading	Object Counting
Pretrained	FLAN-T5-large	44.0	49.3	35.1	31.0
Supervised	Finetuning	61.0	**64.0**	96.0	**70.0**
Feedback	PPO	44.0	43.0	90.5	33.0
	FARL	54.0	60.0	98.0	56.7
	HIR (ours)	**67.0**	62.0	**98.0**	65.0

		Geometric Shapes	Penguins in A Table	Reasoning about Colored Objects	Word Sorting
Pretrained	FLAN-T5-large	9.7	46.7	20.0	1.1
Supervised	Finetuning	90.0	53.0	**90.0**	**24.7**
Feedback	PPO	11.0	50.0	30.0	1.1
	FARL	66.7	**56.0**	77.0	3.4
	HIR (ours)	**90.3**	53.0	77.8	3.4

	No Additional Parameter	Utilize Failure Cases	Supervised Learning	No Additional KL Penalty*
PPO	✗	✓	✗	✗
FARL	✓	✗	✓	✓
HIR (Ours)	✓	✓	✓	✓

*: PPO adopts an KL penalty in addition to the alignment score to keep the new model close to the pretrained model

Figure 4. **Conceptual Comparison between HIR and baseline methods.** HIR is a simple supervised learning algorithm, does not require any additional parameter or KL penalty as an additional reward, and utilizes failure data.

lines: including RL with Human Feedback (PPO) (Ouyang et al., 2022) and Final-Answer Reinforcement Learning (FARL) (Uesato et al., 2022).

We first demonstrate the superior performance of HIR on the single-task fine-tuning with a base model FLAN-T5-large (Chung et al., 2022) in Sec. 6.1. We then validate such performance gain is consistent across different sizes of models (FLAN-T5-base and FLAN-T5-large) in Sec. 6.2. In addition to the performance gains, we also conduct thorough ablations studies on the entropy regularization coefficient, label smoothing factor, and sub-output sampling. All the experiment details including network architecture and hyperparameters can be found at the Appendix. A.1.

Evaluation Setup and Tasks. We introduce the evaluation setup and the tasks we used in our experiments. For the evaluation setup, instead of training a reward model and using a RL algorithm to optimize the objective, following (Uesato et al., 2022), we directly use the final answer in the training dataset to check the results generated by the language models as the feedback (e.g., correct answer or wrong answer). To be specific, we divide the task data into 80% for training and 20% for testing. At training time, we randomly sample a batch of questions as prompts from the training dataset, ask the language model to generate corresponding answers, and provide feedback via final answer checking.

For the BigBench tasks, we choose the 12 challenging tasks, including Tracking Shuffled Objects, Logical Deduction, Date Understanding, Object Counting, Geometric Shapes, Penguins in A Table, Reasoning about Colored Objects, and Word Sorting. These tasks include both multiple-choice tasks and direct-generation tasks. For both types of tasks, we formulate them as the generation task. Following (Chung et al., 2022), we provide options for the language model to choose from as prompts and ask it to generate the answer. There are some examples of this format in Tab. 1. We provide all the templates for the tasks in the Appendix. B.1. In this way, no additional parameter of the language model

Table 3. Performance of HIR on both FLAN-T5-large and FLAN-T5-base models. HIR shows significant improvements even with a much smaller model FLAN-T5-base.

	Tracking Shuffled Objects (3)	Tracking Shuffled Objects (5)	Tracking Shuffled Objects (7)	Logical Deduction (3 Objects)	Logical Deduction (5 Objects)	Logical Deduction (7 Objects)
FLAN-T5-base	34.7	18.4	7.4	36.7	30.0	32.9
HIR-T5-base	100.0 (**+65.3**)	36.8 (**+18.4**)	68.3 (**+60.9**)	73.3 (**+36.6**)	52.0 (**+22.0**)	57.1 (**+24.2**)
FLAN-T5-large	29.3	15.6	6.6	33.3	44.0	49.3
HIR-T5-large	100.0 (**+70.7**)	61.2 (**+45.6**)	42.6 (**+36.0**)	91.7 (**+58.4**)	67.0 (**+23.0**)	62.0 (**+12.7**)
	Date Understading	Object Counting	Geometric Shapes	Penguins in A Table	Reasoning about Colored Objects	Word Sorting
FLAN-T5-base	4.1	19.5	0.0	10.0	4.8	1.3
HIR-T5-base	98.0 (**+93.9**)	59.0 (**+39.5**)	43.1 (**+43.1**)	53.3 (**+43.3**)	73.3 (**+68.5**)	0.5 (-0.8)
FLAN-T5-large	35.1	31.0	9.7	46.7	20.0	1.1
HIR-T5-large	98.0 (**+62.9**)	65.0 (**+34.0**)	90.3 (**+80.6**)	53.0 (**+6.3**)	77.8 (**+57.8**)	3.4 (**+2.3**)

is needed for training (e.g., extra linear head layer).

Baselines. We compare HIR against the two popular RL baselines: PPO and Final-Answer RL. Instead of learning a reward module as in RLHF, we give the PPO algorithm a reward of 1 if the final answer checking is correct and 0 otherwise. Final-Answer RL first conducts the online sampling, then performs the final-answer checking to select only the correct results and use them to do imitation learning. For reference, we also report the number of performing standard fine-tuning. Note that the RL-based method is not directly comparable to fine-tuning, as the they only provides feedback on whether the answer is preferred or not; whereas in order to perform fine-tuning, the correct answer (potentially also the reasoning paths) is required. We also discuss the connections and advantages in Sec. 5.

6.1. HIR with FLAN-T5-large on BigBench

We evaluate HIR extensively using the BigBench tasks aforementioned. In Tab. 2, we compare the performance of HIR with PPO and Final-Answer RL, along with providing the reference performance of Fine-Tuning and the base model without any fine-tuning. From the results in Tab. 2, we see HIR outperforms almost all the baselines, even including fine-tuning by a good margin. Especially in hard tasks like Tracking Shuffled Objects (5) and (7), HIR surpasses the best baseline by 41.2% and 29.2%, respectively. Note that for PPO, we adopt the implementation of trlx by CarperAI[2] and heavily sweep the hyperparameters. However, its performance is still not quite satisfactory. We provide the details in Appendix. A.1.

In tasks that require direct generation, like Object Counting and Word Sorting, HIR is still being able to outperform all the baselines. However, its performance is not comparable to fine-tuning as fine-tuning directly provides the correct answer while HIR only performs final answer checking.

6.2. Effect of Base Model Sizes

We also conduct experiments to show that HIR can work well across different sizes of models. We compare FLAN-T5-base and FLAN-T5-large with the results shown in Tab. 3. We see that HIR can consistently improve the model performance regardless of its size, and achieve significant improvement of 40.5% and 43.0% of the models, respectively. These results also confirm that even though HIR is starting with a weaker model (which can bring challenges to the exploration phase during sampling), it can still gain significant improvements after rounds of training. This is particularly very important given that we don't have many strong language models to bootstrap.

6.3. Ablations

We conduct ablations on different aspects of the algorithm. We specifically study how the entropy coefficient, label smoothing parameters, and sub-output sampling can help with the performance. We present the results in Tab. 4. We can see that adding the entropy regularization term, label smoothing term, and sub-output sampling are all helpful to the final performance to some extent.

Table 4. Ablations on the different components of HIR. We see that each component of entropy regularization, label smoothing and sub-output sampling plays an important role in the algorithm.

	Geometric Shapes	Tracking Shuffled Objects (3)	Logical Deduction (3 Objects)
HIR	**90.3**	**100.0**	**91.7**
HIR (w.o. Sub-Sample)	86.1	100.0	75.0
HIR (w.o. Entropy)	47.2	100.0	48.3
HIR (w.o. Smooth)	84.7	100.0	23.3

[2]Implementation: https://github.com/CarperAI/trlx

7. Limitations and Future work

Limited feedback format and granularity HIR is mainly evaluated on reasoning tasks, where the feedback can be easily scripted based on answer correctness. However, this doesn't hold for a lot of open-ended scenarios (e.g. writing a story about a frog), where human feedback with different levels of granularity can be essential to get diverse and high-quality generation results from the model. However, studying the open-ended question with human feedback requires access to a pre-trained human preference reward model, or having a human labeler interface to provide feedback in an interactive manner for HIR. At the time HIR was invented, the community lacks an open-sourced powerful base model to perform RLHF on such challenging tasks.

Long-horizon generation tasks We show HIR's advantages over multiple-choice tasks, but "object counting" and "word sorting" tasks are two direct generation tasks without knowing any options, where HIR cannot outperform supervised fine-tuning. This might indicate that long-horizon generation tasks are too hard for our base model FLAN-T5 series without ground-truth supervision.

In general, generation tasks require more detailed feedback rather than binary. For example, in word sorting, knowing whether a result is right or wrong only provides very sparse information. By providing more informative feedback to the model, the performance got improved by a large margin (see Appendix.A.3).

8. Conclusion

In this paper, we proposed HIR that ties the connection between instruction alignment and goal-conditioned RL. This yields a simple two-stage hindsight relabeling algorithm, HIR that interacts with and improves language models. HIR utilizes both success data and failure data to train the language model effectively and doesn't require any additional training pipeline. HIR achieves impressive results on the BigBench tasks compared to the baselines.

As far as we know, HIR is the very first algorithm that applies hindsight relabeling to language models, which is a general approach to learn from feedback. We hope such work can inspire future research toward designing more efficient and scalable algorithms that can improve the alignment of large language models.

9. Acknowledgement

The author would like to thank Benjamin Eysenbach and Shane Gu for for helpful discussions throughout the project. This research is supported in part by NSF CISE Expeditions Award CCF-1730628, NSF NRI #2024675 and under the NSF AI4OPT Center. UC Berkeley research is also supported by gifts from Alibaba, Amazon Web Services, Ant Financial, CapitalOne, Ericsson, Facebook, Futurewei, Google, Intel, Microsoft, Nvidia, Sco- tiabank, Splunk and VMware.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., et al. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*, 2021.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623, 2021.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021a.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021b.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, E., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Deng, M., Wang, J., Hsieh, C.-P., Wang, Y., Guo, H., Shu, T., Song, M., Xing, E. P., and Hu, Z. Rlprompt: Optimizing

discrete text prompts with reinforcement learning. *arXiv preprint arXiv:2205.12548*, 2022.

Eysenbach, B., Zhang, T., Salakhutdinov, R., and Levine, S. Contrastive learning as goal-conditioned reinforcement learning. *arXiv preprint arXiv:2206.07568*, 2022.

Gao, T., Fisch, A., and Chen, D. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

Huang, J., Gu, S. S., Hou, L., Wu, Y., Wang, X., Yu, H., and Han, J. Large language models can self-improve. *arXiv preprint arXiv:2210.11610*, 2022.

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

Janner, M., Li, Q., and Levine, S. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34: 1273–1286, 2021.

Jiang, Z., Zhang, T., Janner, M., Li, Y., Rocktäschel, T., Grefenstette, E., and Tian, Y. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.

Kelly, M., Sidrane, C., Driggs-Campbell, K., and Kochenderfer, M. J. Hg-dagger: Interactive imitation learning with human experts. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8077–8083. IEEE, 2019.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*, 2022.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.

Laskin, M., Wang, L., Oh, J., Parisotto, E., Spencer, S., Steigerwald, R., Strouse, D., Hansen, S., Filos, A., Brooks, E., et al. In-context reinforcement learning with algorithm distillation. *arXiv preprint arXiv:2210.14215*, 2022.

Lawrence, C. and Riezler, S. Improving a neural semantic parser by counterfactual learning from human bandit feedback. *arXiv preprint arXiv:1805.01252*, 2018.

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., et al. Solving quantitative reasoning problems with language models. *arXiv preprint arXiv:2206.14858*, 2022.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022a.

Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*, 2022b.

Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*, 2017.

Liu, X., Ji, K., Fu, Y., Du, Z., Yang, Z., and Tang, J. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021a.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. Gpt understands, too. *arXiv preprint arXiv:2103.10385*, 2021b.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

OpenAI. Chatgpt: Optimizing language models for dialogue, Nov 2022. URL https://openai.com/blog/chatgpt/.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Perez, E., Kiela, D., and Cho, K. True few-shot learning with language models. *Advances in Neural Information Processing Systems*, 34:11054–11070, 2021.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL https://aclanthology.org/W18-5446.

Wang, A., Pruksachatkun, Y., Nangia, N., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. Superglue: A stickier benchmark for general-purpose language understanding systems. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., Cheng, M., Glaese, M., Balle, B., Kasirzadeh, A., et al. Ethical and social risks of harm from language models. *arXiv preprint arXiv:2112.04359*, 2021.

Wu, J., Ouyang, L., Ziegler, D. M., Stiennon, N., Lowe, R., Leike, J., and Christiano, P. Recursively summarizing books with human feedback. *arXiv preprint arXiv:2109.10862*, 2021.

Zelikman, E., Mu, J., Goodman, N. D., and Wu, Y. T. Star: Self-taught reasoner bootstrapping reasoning with reasoning. 2022.

Zhang, T., Wang, X., Zhou, D., Schuurmans, D., and Gonzalez, J. E. Tempera: Test-time prompting via reinforcement learning. *arXiv preprint arXiv:2211.11890*, 2022.

Zheng, Q., Zhang, A., and Grover, A. Online decision transformer. In *International Conference on Machine Learning*, pp. 27042–27059. PMLR, 2022.

Zhou, D., Schärli, N., Hou, L., Wei, J., Scales, N., Wang, X., Schuurmans, D., Bousquet, O., Le, Q., and Chi, E. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*, 2022.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.

A. Training and Implementation Details

A.1. Hyperparameters

We provide all the hyperparameters we used in our experiments. This includes all the experiment settings we used for the baselines and our method.

PPO For this baseline, we adopt the implementation of trxl from CarperAI. We directly use the GitHub repository and load the FLAN-T5-large as the base model. We perform hyperparameter sweeping over several key parameters in Tab. 5 as suggested in the original code base. We perform a grid search over 16 combinations on one task and select the best for all tasks. We also list all the hyperparameters we used after the grid search in Tab. 6.

Table 5. Hyperparameters used for sweeping RLHF baseline we tested on our tasks.

Hyperparameter	Value
Learning Rate (lr)	[0.0001, 0.001, 0.01, 0.1]
Initial KL Coefficient	[0, 0.01, 0.1, 0.5]

Table 6. Hyperparameters used for RLHF baseline we tested on our tasks.

Hyperparameter Value	
Learning Rate (lr)	0.0001
Initial KL Coefficient	0.1
Total Epochs	100
Number Layers Unfrozen	2
Optimizer	Adam
Weight Decay	1e-6
Learning Rate Scheduler	Cosine Annealing
Number Rollouts	512
PPO Epochs	4
Gamma	0.99
Clip Range	0.2
Clip Range Value	0.2
Value Loss Coefficient	1.0
Transformer Temperature	1.0
Transformer Top K	50
Transformer Top P	0.95

Final-Answer RL and HIR For Final-Answer RL, we directly use our codebase as its algorithm is very similar to ours. The only difference is that we filter the entire online sampling dataset with only the correct answers. So we keep the same hyperparameters for both and list it here.

A.2. Instruction Relabeling Strategy

Scripted Feedback on BigBench Reasoning Tasks We use a simple scripted binary function:

$$\mathcal{R}(\mathbf{o}, \mathbf{p}, \mathbf{q}) = \begin{cases} 1 & \mathbf{o} \text{ gives correct answer of } \mathbf{q} \text{ and } \mathbf{p} = p_{\text{correct}} \\ 1 & \mathbf{o} \text{ gives wrong answer of } \mathbf{q} \text{ and } \mathbf{p} = p_{\text{wrong}} \\ 0 & \text{otherwise} \end{cases}$$

where p_{correct} = "Generate a correct answer to this problem", and p_{wrong} = "Generate a wrong answer to this problem".

Table 7. Hyperparameters used for Final-Answer RL and HIR.

Hyperparameter Value	
Online Samples per Iteration	4
Sampling Temperature	1.0
Learning Rate (lr)	0.0005
Train Batch Size	64
Train Epochs per Iteration	10
Weight decay	0.0
Learning Rate Warmup Steps	0
Learning Rate Scheduler	constant
Label Smoothing	0.2
Entropy Regularization Coefficient	0.001
Contrastive Loss Coefficient	1

Scripted Instruction Relabeling We also relabel instruction based on

$$\phi(\mathbf{o}, \mathbf{p}, \mathbf{q}, \mathbf{r}) = \begin{cases} \mathbf{p} & \mathbf{r} = 1 \\ \neg\mathbf{p} & \text{otherwise} \end{cases}$$

$\mathbf{p}$ is initialized to be p_{correct} at the beginning of training. (let $\neg p_{\text{correct}} = p_{\text{wrong}}$ and the opposite also holds). Note that we never use those functions during evaluation, so they can only access the ground truth in the training set, which is a fair comparison with other baselines and SFT.

A.3. More Granular Feedback

To understand the effects of more granular feedback, we study the "object counting" task. This time we not only tell whether the model is right or wrong, but also how wrong it is. The instruction is edited to be "Give me a wrong answer that can be corrected by [plus/minus] [a number]". We found this largely improve the performance on object counting from 65% to 92%.

B. Dataset

B.1. Dataset Examples

Here in the section, we provide all the templates we used to train our model for all 12 tasks. The tasks consist of 10 multiple choice tasks and 2 direct generation tasks. In Tab. 8, we list all the training template for our pipeline.

Table 8. Examples of inputs and outputs for the BigBench tasks. For multiple-choice tasks, we provide the options that the language model can choose from as prompts.

	Tasks	Example Inputs	Outputs
Multiple Choice	Logical Deduction (3)	"Q: On a shelf, there are three books: a black book, an orange book, and a blue book. The blue book is to the right of the orange book. The orange book is to the right of the black book. Options: (A) The black book is the leftmost. (B) The orange book is the leftmost. (C) The blue book is the leftmost."	"(A)"
	Logical Deduction (5)	"Q: On a shelf, there are five books: a gray book, a red book, a purple book, a blue book, and a black book. The red book is to the right of the gray book. The black book is to the left of the blue book. The blue book is to the left of the gray book. The purple book is the second from the right. Options: (A) The gray book is the leftmost. (B) The red book is the leftmost. (C) The purple book is the leftmost. (D) The blue book is the leftmost. (E) The black book is the leftmost."	"(E)"
	Logical Deduction (7)	"Q: The following paragraphs each describe a set of three objects arranged in a fixed order. The statements are logically consistent within each paragraph. In a golf tournament, there were three golfers: Amy, Eli, and Eve. Eve finished above Amy. Eli finished below Amy. Options: (A) The black book is the leftmost. (B) The yellow book is the leftmost. (C) The white book is the leftmost. (D) The gray book is the leftmost. (E) The purple book is the leftmost. (F) The orange book is the leftmost. (G) The green book is the leftmost."	"(B)"
	Tracking Shuffled Objects (3)	"Q: Alice, Bob, and Claire are playing a game. At the start of the game, they are each holding a ball: Alice has a orange ball, Bob has a white ball, and Claire has a blue ball. As the game progresses, pairs of players trade balls. First, Alice and Bob swap balls. Then, Bob and Claire swap balls. Finally, Alice and Bob swap balls. At the end of the game, Alice has the Options: (A) orange ball. (B) white ball. (C) blue ball."	"(C)"

	Tasks	Example Inputs	Outputs
Multiple Choice	Tracking Shuffled Objects (5)	"Q: Alice, Bob, Claire, Dave, and Eve are playing a game. At the start of the game, they are each holding a ball: Alice has a pink ball, Bob has a white ball, Claire has a red ball, Dave has a purple ball, and Eve has a yellow ball. As the game progresses, pairs of players trade balls. First, Alice and Dave swap balls. Then, Claire and Eve swap balls. Then, Alice and Bob swap balls. Then, Dave and Claire swap balls. Finally, Alice and Claire swap balls. At the end of the game, Alice has the Options: (A) pink ball. (B) white ball. (C) red ball. (D) purple ball. (E) yellow ball."	"(A)"
	Tracking Shuffled Objects (7)	"Q: Alice, Bob, Claire, Dave, Eve, Fred, and Gertrude are playing a game. At the start of the game, they are each holding a ball: Alice has a green ball, Bob has a white ball, Claire has a yellow ball, Dave has a pink ball, Eve has a orange ball, Fred has a black ball, and Gertrude has a brown ball. As the game progresses, pairs of players trade balls. First, Bob and Gertrude swap balls. Then, Fred and Claire swap balls. Then, Dave and Gertrude swap balls. Then, Bob and Gertrude swap balls. Then, Alice and Claire swap balls. Then, Gertrude and Claire swap balls. Finally, Eve and Claire swap balls. At the end of the game, Alice has the Options: (A) green ball. (B) white ball. (C) yellow ball. (D) pink ball. (E) orange ball. (F) black ball. (G) brown ball."	"(F)"
	Date Understanding	"Q: Yesterday was April 30, 2021. What is the date today in MM/DD/YYYY? Options: (A) "05/01/2021" (B) "02/23/2021" (C) "03/11/2021" (D) "05/09/2021" (E) "04/29/2021" "	"(A)"
	Geometric Shapes	"Q: This SVG path element ¡path d=M̈ 59.43,52.76 L 75.49,27.45 L 54.92,4.40 M 54.92,4.40 L 23.70,7.77 L 15.15,42.15 L 34.51,57.44 L 59.43,52.76ï¿ draws a Options: (A) circle (B) heptagon (C) hexagon (D) kite (E) line (F) octagon (G) pentagon (H) rectangle (I) sector (J) triangle"	"(C)"
	Penguins in a Table	"Q: Here is a table where the first line is a header and each subsequent line is a penguin: name, age, height (cm), weight (kg) Louis, 7, 50, 11 Bernard, 5, 80, 13 Vincent, 9, 60, 11 Gwen, 8, 70, 15 For example: the age of Louis is 7, the weight of Gwen is 15 kg, the height of Bernard is 80 cm. What animals are listed in the table? Options: (A) bears (B) crocodiles (C) elephants (D) giraffes (E) penguins"	"(E)"
	Reasoning about Colored Objects	"Q: On the nightstand, you see a mauve stress ball and a purple booklet. What color is the booklet? Options: (A) red (B) orange (C) yellow (D) green (E) blue (F) brown (G) magenta (H) fuchsia (I) mauve (J) teal (K) turquoise (L) burgundy (M) silver (N) gold (O) black (P) grey (Q) purple (R) pink"	"(Q)"
Direct Generation	Object Counting	"Q: I have a blackberry, a clarinet, a nectarine, a plum, a strawberry, a banana, a flute, an orange, and a violin. How many fruits do I have?"	"6"
	Word Sorting	"Sort the following words alphabetically: List: oven costume counterpart."	"costume counterpart oven"