Evolutionary Contrastive Distillation for Language Model Alignment

Julian Katz-Samuels, **Zheng Li**, **Hyokun Yun**,
Priyanka Nigam, **Yi Xu**, **Vaclav Petricek**,
Bing Yin, **Trishul Chilimbi**
Amazon
Correspondence: {jkatzsam,amzzhe,yunhyoku}@amazon.com

Abstract

The ability of large language models (LLMs) to execute complex instructions is essential for their real-world applications. However, several recent studies indicate that LLMs struggle with challenging instructions (Zhou et al., 2023; Qin et al., 2024; Jiang et al., 2023b). In this paper, we propose Evolutionary Contrastive Distillation (ECD), a novel method for generating high-quality synthetic preference data designed to enhance the complex instruction-following capability of language models. ECD generates data that specifically illustrates the difference between a response that successfully follows a set of complex instructions and a response that is high-quality, but nevertheless makes some subtle mistakes. This is done by prompting LLMs to progressively evolve simple instructions into more complex instructions. When an instruction is made more complex, the original successful response mostly meets the new requirements but misses one or two, thus becoming a "hard negative" example for the new instruction. By pairing a good response with such a hard negative response, and employing contrastive learning algorithms such as DPO (Rafailov et al., 2023), we improve language models' ability to follow complex instructions. Empirically, we observe that our method yields a 7B model that exceeds the complex instruction-following performance of current state-of-the-art (SOTA) 7B models and is competitive even with open-source 70B models.

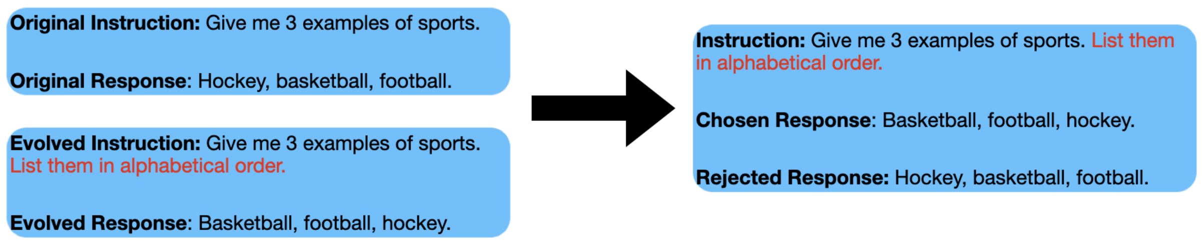
1 Introduction

Large language models (LLMs) have demonstrated impressive capabilities in a wide range of tasks ranging from creative writing to code generation. In light of these achievements, there has been a surge of interest in building complex data processing systems with LLM-based components (Developers, 2024a,b). Complex instruction-following capability, which is the capability of LLMs to generate outputs consistent with multiple interdependent specifications in the prompt, is critically important for such systems to operate reliably. Consequently, multiple benchmarks have been proposed to capture various aspects of complex instruction-following (Zhou et al., 2023; Qin et al., 2024; Jiang et al., 2023b). Unfortunately, these studies find that there still is a significant gap between proprietary LLMs and open source models on these benchmarks.

This raises the question: how shall we effectively distill the complex instruction-following ability of stronger LLMs into smaller language models? After the seminal work by Wang et al. (2023), the usage of proprietary LLMs for the generation of alignment data has been actively studied. As a typical alignment pipeline consists of Supervised Fine-Tuning (SFT) methods and Preference Fine-Tuning (PFT[1]) stages (Ouyang et al., 2022), these data generation methods are also largely categorized into SFT data generation methods (e.g., UltraChat (Ding et al., 2023)) and PFT data generation methods (e.g., UltraFeedback (Cui et al., 2023)).

Among a wide variety of SFT data generatin methods, Evol-Instruct (Xu et al., 2023) and Conifer (Sun et al., 2024) are specifically designed to improve complex instruction-following. They share the common goal: generate highly complex instructions. And they also share the same evolutionary strategy: they first prompt proprietary LLMs to evolve simple seed instructions from ShareGPT to have progressively more complex requirements. Then, proprietary LLMs are again prompted to generate demonstrations on these complex instructions.

[1]We denote this stage PFT instead of RLHF (Reinforcement Learning from Human Feedback), because many data collection methods for preference data do not necessarily involve human feedback (Lee et al., 2023; Cui et al., 2023; Yang et al., 2023).


Original Instruction: Give me 3 examples of sports.

Original Response: Hockey, basketball, football.

Evolved Instruction: Give me 3 examples of sports. List them in alphabetical order.

Evolved Response: Basketball, football, hockey.

Instruction: Give me 3 examples of sports. List them in alphabetical order.

Chosen Response: Basketball, football, hockey.

Rejected Response: Hockey, basketball, football.

Figure 1: Example of ECD preference pair construction. The original instruction is to provide three examples of sports while the evolved instruction adds the additional requirement to sort the examples in alphabetical order. Since the original response violates this requirement, it can be used as a rejected response in the preference pair. On the other hand, the evolved response satisfies all the requirements and can be used as the chosen response.

On the other hand, while a wide variety of PFT data generation methods have been proposed in the literature (Lee et al., 2023; Cui et al., 2023; Yang et al., 2023) there has been little attention on leveraging PFT stage for complex instruction-following. We claim, however, that PFT stage has distinctive advantages in teaching complex instruction-following. First, SFT training places an equal weight on every token in the response, and this can be less than ideal for instruction-following. When a tricky requirement such as "make sure every sentence rhymes with each other" is imposed in the instruction, certain tokens play a more important role in meeting this requirement than other tokens. In the PFT stage, we can leverage preference pairs which are similar in content but differ in these instruction-critical tokens to more directly illustrate the causality between the token generation and the successful instruction-following. Second, while the SFT stage can utilize positive examples only, PFT stage can also leverage negative examples to highlight the fine difference between a successful instruction-following and a subtle failure.

To this end, we propose a novel technique, Evolutionary Contrastive Distillation (ECD), for generating high-quality, synthetic preference data targeting complex instruction-following. Following Xu et al. (2023); Sun et al. (2024), we prompt proprietary LLMs to progressively increase the complexity of instructions. Instead of separating the original example and the evolved example as two independent examples for SFT, however, we connect them as a preference pair for PFT: the proprietary LLM's response on the evolved instruction is considered as a positive example, and its response on the original instruction is considered a negative example. The key observation is that when an instruction is evolved to have a new requirement, the original *good* response for the original instruction is not a good response for the evolved instruction anymore, and therefore can be used as the negative example to the evolved instruction. Since the evolution of instructions is gradual, however, the original response still satisfies most of the requirements of the evolved instruction, and therefore can be used as a "hard" negative for the evolved instruction. See Figure 1 for a concrete example. This method has the desireable properties that (i) it does not rely on LLMs to annotate preferences or generate undesirable responses, which can be unreliable (Yang et al., 2023) and (ii) it is effective at creating hard negative examples, which have are crucial in contrastive learning (Chopra et al., 2005; Hadsell et al., 2006; Robinson et al., 2020).

To ensure that each step of instruction evolution provides a high-quality example of subtle nuance in instruction-following, we devise a fine-grained hierarchical taxonomy of evolution operations, which we discuss in Section 4. We demonstrate the data generated from this taxonomy is not only effective at improving complex instruction-following, but can also be combined with previous evolutionary (Sun et al., 2024) as well as non-evolutionary (Cui et al., 2023) methods to yield even stronger results. Therefore, we believe the proposed taxonomy will serve as a useful resource on its own for future research on complex instruction-following.

We validate the effectiveness of our approach in an extensive set of experiments, benchmarking our models on three recent instruction-following benchmarks: IFEval, FollowBench, and InfoBench. Our approach yields a state-of-the-art 7B model, improving on prior SOTA (state-of-the-art) instruction-following 7B models by 7pp on IFEval and is competitive with popular open-source models at the 70B scale. Furthermore, we develop a recipe to build SOTA instruction-following models that also achieve highly competitive performance on popular conversational quality benchmarks such as MT-Bench and AlpacaEval. Finally, we perform ablations illustrating the advantages of ECD over other synthetic preference data generation methods such as RLAIF and RLCD, and the importance of using contrastive learning (e.g., DPO) instead of SFT to learn from preference pairs.

2 Related Work

Instruction-Following. The complex instruction-following ability of LLMs has received significant attention recently with many works proposing new evaluation benchmarks (Zhou et al., 2023; Jiang et al., 2023b; Qin et al., 2024). They consistently find that open source LLMs have significant gaps on following complex instructions compared to proprietary LLMs. At the same time, there has been relatively less work on developing techniques to *improve* the complex instruction-following ability. Two exceptions include Evol-Instruct (Xu et al., 2023) and Conifer (Sun et al., 2024), which prompt LLMs to evolve the complexity of instructions, and apply SFT on the generated data. There are two main differences between these works and ours. First, we focus on generating preference data for PFT instead of demonstration data for SFT. While SFT data provides only positive feedback to the model on the correct behavior, PFT data provides both positive and negative feedback, which enables teaching the contrast between a successful instruction-following example and a subtle failure example. Second, we propose a fine-grained hierarchical taxonomy of evolution operations to ensure each step of evolution introduces diverse and subtle variations of requirements.

Contrastive Learning. The effectiveness of contrastive learning (Chopra et al., 2005; Hadsell et al., 2006) has been shown across numerous modalities (He et al., 2020; Chi et al., 2020; Radford et al., 2021). In the contrastive learning literature, the importance of the quality of negative samples has been well-established (Robinson et al., 2020). Numerous "hard" negative mining techniques have been introduced to improve the quality of negative samples (Schroff et al., 2015; Wu et al., 2018; Xiong et al., 2021). While many alignment methods such as InstructGPT (Ouyang et al., 2022) and DPO (Rafailov et al., 2023) leverage negative samples to facilitate learning (Tajwar et al., 2024), how to obtain sufficiently hard negative samples has not been actively studied. Yan et al. (2024) employs hard negative mining techniques to improve the robustness of the model. Our work instead targets improving complex instruction-following, and uses LLM prompting to generate negative examples rather than mining samples from a fixed dataset.

Data Generation Methods for PFT. Much attention has been given to how to generate preference data for fine-tuning LLMs. Pioneering works focused on collecting feedback from humans (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022). Under this approach, human annotators would observe a set of responses to an instruction and rank them according to their preferences. As human annotation is expensive and difficult to scale, however, follow-up works have proposed alternative synthetic data generation approaches.

The most popular method of generating PFT data from LLMs is to sample two independent responses on the same instruction, and then ask an LLM to judge which one is of higher quality. This method is often called RLAIF (Reinforcement Learning from AI Feedback) (Bai et al., 2022b). While RLAIF methods have shown encouraging results (Cui et al., 2023; Tunstall et al., 2023; Ivison et al., 2023), obtaining high-quality preference annotations from LLMs have been found challenging (Yang et al., 2023; Sharma et al., 2024). This is because the difference between two responses on the identical instruction can be minor, and LLM-based preference annotation can be subject to many confounding factors, such as self-bias and verbosity bias (Zheng et al., 2023).

To avoid quality issues from LLM-based preference annotation, Reinforcement Learning from Contrastive Distillation (RLCD) (Yang et al., 2023) proposes to employ two different prompt templates rather than using the same one. One prompt template is designed to elicit desirable responses, and another is designed to elicit undesirable response. Such a deliberate design of templates allows RLCD to bypass preference annotation. As we discuss in Section 5, however, we find it challenging to prompt-engineer proprietary LLMs to generate undesirable responses. As proprietary LLMs are already aligned to generate helpful and harmless responses (Bai et al., 2022a), very often, their response from the undesirable response template are either as helpful as the response from the desirable response template, or trivially unhelpful ("No, I can't answer that question."), limiting their value as a *hard* negative example.

In contrast to RLHF or RLAIF, ECD does not require preference annotation by neither humans nor LLMs. This circumvents issues with preference annotation quality. In contrast to easy negative examples from RLCD, negative examples from ECD are hard, as they are helpful responses on a similar but subtly different instruction.

3 Evolutionary Contrastive Distillation

First, we propose the evolutionary data generation process to synthesize preference data for complex instruction-following. Then, we discuss how contrastive learning methods shall be employed to train on this generated data.

Data Generation Framework We assume access to a seed set of instructions $\mathcal{I}^{(0)} = \{I_1^{(0)}, \ldots, I_n^{(0)}\}$ and responses on these instructions $\mathcal{R}^{(0)} = \{R_1^{(0)}, \ldots, R_n^{(0)}\}$ where $R_i^{(0)}$ is the response on instruction $I_i^{(0)}$. In our experiments, we use ShareGPT (Team, 2023) as the seed set. As instructions which are publicly accessible in scale such as ShareGPT lack the complexity needed for LLM applications (Xu et al., 2023), we iteratively increase their complexity through T rounds of an evolutionary process, and generate the preference dataset $\mathcal{D}$, which consist of (instruction, positive response, negative response) triples.

At each round t, for each instruction $I_i^{(t-1)}$, we run the following:

- **Evolution**: Prompt an LLM to evolve $I_i^{(t-1)}$ into proposal instruction $\tilde{I}_i^{(t)}$. The evolution typically increases the complexity of the instruction. We discuss types of evolution operations we consider in Section 4.
- **Adaptation**: Prompt an LLM to generate a proposal response $\tilde{R}_i^{(t)}$ on the new instruction $\tilde{I}_i^{(t)}$.
- **Elimination**: The quality of $\tilde{R}_i^{(t)}$ is checked with another LLM prompt. If the quality is acceptable, the proposal is kept: $(I_i^{(t)}, R_i^{(t)}) \leftarrow (\tilde{I}_i^{(t)}, \tilde{R}_i^{(t)})$. Otherwise, $(I_i^{(t)}, R_i^{(t)}) \leftarrow (I_i^{(t-1)}, R_i^{(t-1)})$
- **Contrast**: If the proposal was accepted, add the triple $(I_i^{(t)}, R_i^{(t)}, R_i^{(t-1)})$ into $\mathcal{D}$.

The key intuition is that as the original response $R_i^{(t-1)}$ was generated without seeing all of the requirements from the evolved instruction, the evolved response $R_i^{(t)}$ is likely better than the original response $R_i^{(t-1)}$ for the evolved instruction $I_i^{(t)}$. On the other hand, $R_i^{(t-1)}$ is still a *hard* negative, as it was a desirable response for $I_i^{(t-1)}$, and the difference between $I_i^{(t-1)}$ and $I_i^{(t)}$ is often subtle due to the design of evolution operations.

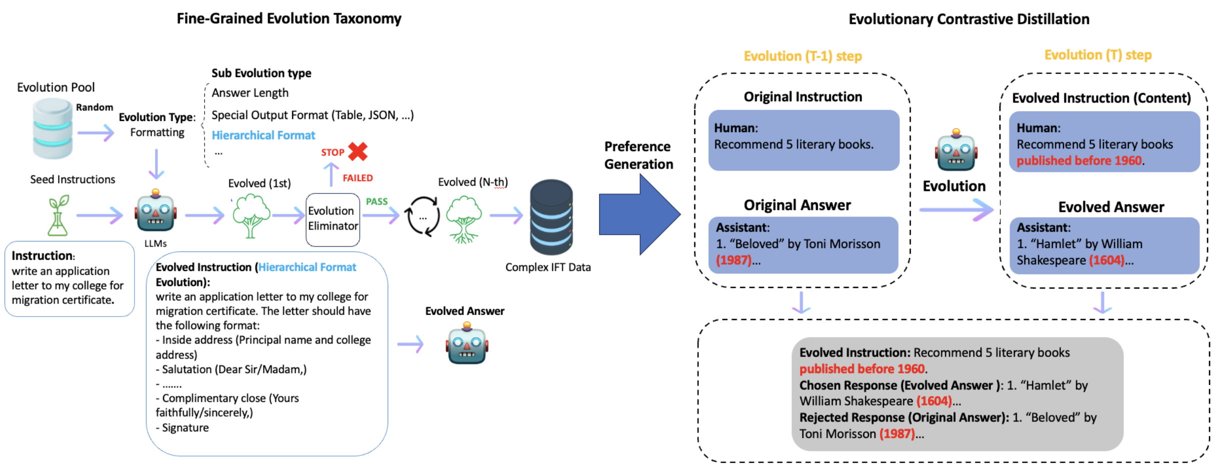
We emphasize that this framework of generating preference data from a evolutionary process is generic, and can accommodate different definitions of the evolutionary process. In Section 5, we show the evolutionary process from Sun et al. (2024) can be successfully adopted in this framework. In order to further improve the quality of the data generated, we propose a fine-grained taxonomy of evolution operations in Section 4.

Contrastive Learning A variety of alignment algorithms can be leveraged to fine-tune LLMs with triplets $\mathcal{D}$ generated from the evolutionary process (Ouyang et al., 2022; Rafailov et al., 2023; Azar et al., 2023; Ethayarajh et al., 2024; Hong et al., 2024; Meng et al., 2024). In this work, we focus on Direct Policy Optimization (DPO) (Rafailov et al., 2023) mainly because it is the most well-established in open source LLMs (Tunstall et al., 2023; Ivison et al., 2023), making our experiments and checkpoints easily comparable with existing work. We leave it as a future work to explore the implication of algorithm choice on the complex instruction-following capability.

For completeness, we briefly discuss how DPO is adopted for ECD. Let us denote π_θ as a language model policy, where $\pi_\theta(R|I)$ denotes the probability of generating the response R conditional on the instruction I. We assume that we are given a reference language model $\pi_{\texttt{ref}}$, and use it to initialize π_θ. Then, we directly fine-tune the language model π_θ on the preference dataset $\mathcal{D}$ by solving the following optimization problem:

$$\text{argmin}_\theta \mathcal{L}_{\texttt{DPO}}(\mathcal{D};\theta) :=$$
$$\mathbb{E}_{I,R^+,R^-\sim\mathcal{D}}\left[\log\left(\sigma(\frac{\pi_\theta(R^+|I))}{\pi_{\texttt{ref}}(R^+|I)}) - \sigma(\frac{\pi_\theta(R^-|I))}{\pi_{\texttt{ref}}(R^-|I)})\right)\right].$$

Following Rafailov et al. (2023, 2024), the resulting LLM policy π_θ can be associated with the policy which maximizes the reward model learned from $\mathcal{D}$. Due to our construction of $\mathcal{D}$, the reward



Fine-Grained Evolution Taxonomy
Evolution Pool
Random
Evolution Type: Formatting
Sub Evolution type
Answer Length
Special Output Format (Table, JSON, ...)
Hierarchical Format
...
Seed Instructions
LLMs
Evolved (1st)
STOP
FAILED
PASS
Evolution Eliminator
Evolved (N-th)
Complex IFT Data
Instruction: write an application letter to my college for migration certificate.
Evolved Instruction (Hierarchical Format Evolution): write an application letter to my college for migration certificate. The letter should have the following format: - Inside address (Principal name and college address) - Salutation (Dear Sir/Madam,) - .... - Complimentary close (Yours faithfully/sincerely,) - Signature
Evolved Answer
Preference Generation
Evolutionary Contrastive Distillation
Evolution (T-1) step
Evolution (T) step
Original Instruction
Human: Recommend 5 literary books.
Original Answer
Assistant: 1. “Beloved” by Toni Morisson (1987)...
Evolution
Evolved Instruction (Content)
Human: Recommend 5 literary books published before 1960.
Evolved Answer
Assistant: 1. “Hamlet” by William Shakespeare (1604)...
Evolved Instruction: Recommend 5 literary books published before 1960.
Chosen Response (Evolved Answer): 1. “Hamlet” by William Shakespeare (1604)...
Rejected Response (Original Answer): 1. “Beloved” by Toni Morisson (1987)...

Figure 2: Overview of Evolutionary Contrastive Distillation. The left-hand side depicts the evolutionary process of creating complex instructions and their demonstrations. The right-hand side depicts the process of extracting the preference data from this process.

model would assign higher rewards on responses which meticulously follow complex instructions.

4 Fine-Grained Evolution Taxonomy

In order to ensure the evolutionary process generates diverse variations of instructions, we define a fine-grained hierarchical taxonomy of evolution operations. Each evolution operation samples a leaf node from this taxonomy to determine the type of operation and the corresponding prompt template. The top level of the taxonomy consists of five categories:

- **Content**: Add a condition that changes the scope or the specificity of the response.
- **Style**: Control the tone, sentiment, or formality of the response.
- **Format**: Impose a formatting or linguistic requirement.
- **Reasoning**: Add a constraint that requires additional steps of logical or numerical reasoning.
- **Breadth**: Come up with a new instruction that matches the domain, length, and complexity of the original instruction.

Then, for each of the top-level category, we define around five fine-grained evolution types. See Table 6 for the full taxonomy of evolution operations. While Evol-Instruct (Xu et al., 2023) defines only eleven high-level operation types, and Confier (Sun et al., 2024) provides about eleven operation types in in-context examples, our taxonomy defines more fine-grained and hierarchically organized 22 operation types. In Section 5, we find this taxonomy generates higher-quality preference data for ECD.

For illustration, consider the following example of an evolution step:

1. The process starts from a seed instruction “Write an application letter to my college for immigration certificate.”.
2. *‘Format’* is randomly chosen from the top level.
3. Within the *‘Format’* category, *‘Hierarchical: Introduce a hierarchical structure which requires an understanding of a hierarchy of tasks and follow it’* is randomly chosen.
4. LLM is prompted with the prompt corresponding to *‘Hierarchical’* type to evolved the original instruction into: “Write an application letter to my college for immigration certificate. The letter should have the following format: -Date, -Inside address, -Salutation, ...”.

See Figure 2 for a graphical illustration.

5 Experiments

5.1 Models and Datasets

We conduct our experiments on the popular base model Mistral-7B-v0.1 (Jiang et al., 2023a). We train our own Instruction Fine-tuned (IFT) model based on the Conifer-7B’s SFT recipe (Sun et al., 2024), the SOTA IFT 7B model, mixing in 53k samples from ShareGPT with the Conifer dataset.[2] We call the resulting IFT dataset Conifer-Mix and the resulting IFT model, Conifer-7B-SFT.[3] Conifer-7B-SFT forms the backbone on which we perform PFT experiments using DPO. For details on training, see Appendix B.

To test the robustness of our approach, we generate 3 separate ECD datasets: (i) **ECD-FineGrained**: 30k preference pairs from Fine-Grained Evolutionary Process discussed in Section 4,[4] (ii) **ECD-Conifer**: ECD data based on the evolutionary process from Conifer (Sun et al., 2024)[5], and (iii) **ECD-FineGrained-Conifer**: a concatenation of ECD-FineGrained and ECD-Conifer. We add UltraFeedback (Cui et al., 2023) to each of these to improve conversational quality[6]; see the Appendix B.2 for the ablation on its impact.

5.2 Evaluation

Our primary goal is to improve the complex instruction-following capability of LLMs, which we measure with the following three benchmarks:

- **IFEval** is a popular benchmark for instruction-following that measures the ability of LLMs to follow programatically checkable instructions such as “give a response that is more than 400 words" (Zhou et al., 2023). It uses metrics such as prompt-level accuracy and instruction-level accuracy with a strict version that interprets the requirements very precisely while a loose version gives some leeway.

- **FollowBench** is another instruction-following benchmark that uses GPT-4 to measure the ability of a model to follow fine-grained constraints across 5 different difficulties and types (Format, Content, Style, Situation, and Example) (Jiang et al., 2023b). This benchmark employs two metrics: hard satisfaction rate (HSR) and soft satisfaction rate (SSR). HSR quantifies the average frequency at which all the requirements or constraints are completely met. On the other hand, SSR calculates the average degree to which individual constraints are satisfied across all the given instructions.

- **InfoBench** evaluates the instruction-following of LLMs by breaking down instructions into a set of fine-grained criteria and asks GPT-4 to evaluate the extent to which a model meets the criteria (Qin et al., 2024).

Since conversational quality is also important in LLM applications, we also evaluate on the following benchmarks:

- **MTBench** is a multi-turn benchmark that measures the conversational quality of a model. It uses GPT-4 to rate the quality of a model's answers across two turns on a scale of 1-10 (Zheng et al., 2023).

- **AlpacaEval** is a single-turn benchmark that measures helpfulness. It uses GPT-4-Turbo to compute the win-rate against a reference model. We use the default reference model, GPT-4-Turbo, and the Length-Controlled win-rate, which has a correlation of 0.98 with ChatBot Arena (Dubois et al., 2024b,a).

We benchmark our ECD models against large open-source scale models such as LLaMa-2-70B-Chat (Touvron et al., 2023) and Vicuna-13B-v1.5 (Team, 2023), and strong 7B models like Conifer-DPO-7B (Sun et al., 2024), Deita-7B-v1.0 (Liu et al., 2023), Mistral-7B-Evol-Instruct (Xu et al., 2023), Mistral-7B-ShareGPT-DPO (Sun et al., 2024), and Zephyr-7B-beta (Tunstall et al., 2023).[7]

5.3 Results

Our ECD models achieve SOTA performance at the 7B scale for complex instruction-following. For example, consider our ECD model trained on ECD-FineGrained-Conifer. Table 1 shows that it outperforms Conifer-7B-DPO, the latest 7B SOTA in instruction-following, on each metric in IFEval by a substantial margin, improving loose prompt accuracy from 52.3% to 59.3% and loose instruction accuracy from 63.3% to

[2]We cannot use the exact same 53k samples from ShareGPT used in the Conifer paper, as this dataset was not made public. We randomly sampled 53k datapoints from `https://huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered.` to best mimic it.

[3]Since model parameters from Sun et al. (2024) were not publicly released, we reproduced the recipe according to the paper to the best of our ability.

[4]We execute this strategy for four rounds of evolution using Claude 2, generating a total of 104,499 preference data from ShareGPT instructions. Then we subsampled 30k.

[5]We removed evolutions with "process feedback"-type, because it does not generate hard negatives needed for ECD.

[6]We use `https://huggingface.co/datasets/HuggingFaceH4/ultrafeedback_binarized?row=1` and remove preference pairs with equal GPT-4 scores.

[7]For the models Mistral-7B-Evol-Instruct and Mistral-7B-ShareGPT-DPO, we report the evaluation numbers given in (Sun et al., 2024).

	Base Model	IFT Data	Preference Data	IFEval				FollowBench		InfoBench		
				strict pr	strict in	loose pr	loose in	HSR	SSR	Easy	Hard	Overall
LLaMa-2-70B-Chat[†]	LLaMa-2	-	-	-	-	-	-	47.5%	55.9%	89.6%	82.1%	84.4%
Vicuna-13B-v1.5[†]	LLaMa-2	ShareGPT	-	43.1%	53.6%	46.6%	58.0%	50.4%	59.5%	85.7%	73.7%	77.3%
Mistral-7B-Evol-Instruct[†]	Mistral-7B	Evol-Instruct	-	41.4%	51.3%	44.0%	54.4%	40.7%	52.7%	81.0%	73.2%	75.6%
Conifer-7B-SFT	Mistral-7B	Conifer-Mix	-	45.1%	58.0%	48.6%	61.2%	49.7%	60.4%	85.9%	80.6%	82.2%
Mistral-7B-ShareGPT-DPO[†]	Mistral-7B	ShareGPT	UltraFeedback	43.8%	55.8%	48.2%	59.7%	47.7%	55.9%	86.8%	79.9%	82.0%
Deita-7B-V1.0[†]	Mistral-7B	Deita-Mix	UltraFeedback	44.6%	56.6%	51.9%	63.7%	45.7%	54.3%	86.2%	78.6%	80.9%
Conifer-7B-DPO[†]	Mistral-7B	Conifer-Mix⋆	UltraFeedback	48.1%	59.1%	52.3%	63.3%	50.0%	56.2%	**87.5%**	80.0%	82.3%
-	Mistral-7B	Conifer-Mix	UltraFeedback	44.5%	56.5%	52.5%	64.3%	58.7%	65.9%	86.2%	82.4%	83.6%
ECD	Mistral-7B	Conifer-Mix	ECD-Conifer	47.9%	59.7%	54.9%	66.2%	57.4%	65.4%	86.5%	82.4%	83.7%
ECD	Mistral-7B	Conifer-Mix	ECD-FineGrained	**52.9%**	63.5%	**59.3%**	**69.8%**	**63.4%**	**70.8%**	84.3%	82.0%	82.7%
ECD	Mistral-7B	Conifer-Mix	ECD-FineGrained-Conifer	52.3%	**63.8%**	58.8%	68.9%	58.6%	65.4%	85.5%	**85.2%**	**85.3%**

Table 1: Main results on Instruction Following benchmarks: IFEval, FollowBench, and InFoBench. Bold-face indicates the best results among the 7B models. † indicates that the results are from the original source. ⋆ denotes the mixture of the Conifer dataset with ShareGPT from the original paper (Sun et al., 2024). Note that "in" abbreviates "instruction" and "pr" abreviates "prompt" in the above table, so for example "loose in" abbreviates "loose instruction accuracy". These abbreviations hold for subsequent tables as well.

Model	Base Model	IFT Data	Preference Data	MT-Bench Score	AlpacaEval	
					LC Win-Rate	Average Length
Mistral-7B-Evol-Instruct[†]	Mistral-7B	Evol-Instruct	-	6.51	9.4%	982
Conifer-7B-SFT	Mistral-7B	Conifer-Mix	-	6.74	10.0%	1002
Deita-7B-v1.0[†]	Mistral-7B	Deita-Mix	UltraFeedback	**7.55**	16.1%	1417
Mistral-7B-ShareGPT-DPO[†]	Mistral-7B	ShareGPT	UltraFeedback	7.1	15.1%	1276
Conifer-DPO-7B[†]	Mistral-7B	Conifer-Mix⋆	UltraFeedback	7.25	17.1%	1253
Zephyr-7B-beta[†]	Mistral-7B	UltraChat	UltraFeedback	7.34	13.2%	1444
-	Mistral-7B	Conifer-Mix	UltraFeedback	7.41	23.3%	1528
ECD	Mistral-7B	Conifer-Mix	ECD-Conifer	7.49	**25.2%**	1424
ECD	Mistral-7B	Conifer-Mix	ECD-FineGrained	7.35	22.0%	1327
ECD	Mistral-7B	Conifer-Mix	ECD-FineGrained-Conifer	7.47	20.6%	1427

Table 2: Main results on conversational quality benchmarks: MT-Bench and AlpacaEval. † indicates that the results are from the original paper. ⋆ denotes the mixture of the Conifer dataset with ShareGPT from the original source. ⋆ denotes the mixture of the Conifer dataset with ShareGPT from the original paper (Sun et al., 2024).

69.8%, and achieves similar improvements on FollowBench and InfoBench. Similarly, the ECD on ECD-Finegrained-Conifer improves over its initialization Conifer-7B-SFT, improving for example on loose prompt accuracy by over 10pp and even shows competitive performance with LLaMa-2-70B-Chat. ECD-FineGrained-Conifer achieves particularly strong performance on InfoBench Hard, indicating the strength of ECD in improving instruction-following for particularly difficult instructions. While here we discuss specifically ECD-FineGrained-Conifer, these trends uphold across our three ECD data mixtures, ECD-FineGrained, ECD-Conifer, and ECD-FineGrained-Conifer, indicating the robustness of our approach.

Our ECD models also achieve strong performance on the conversational quality benchmarks: MT-Bench and AlpacaEval. For example, consider ECD-Conifer. It achieves an MT-Bench score of 7.49 and and a length-controlled AlpacaEval score of 25.2%. These are large improvements in comparison to the initialized IFT model Conifer-7B-SFT and the prior SOTA in instruction-following 7B models, Conifer-7B-DPO. As this trend holds across all three ECD models, our data mixture recipe consistently produces SOTA 7B models for complex instruction-following while maintaining strong conversational quality.

Among ECD, RLAIF, and RLCD, which is the most effective technique for improving complex instruction-following? We also investigated how ECD compares against RLAIF and RLCD for improving instruction-following. To this end, we generated RLAIF and RLCD data[8] on top of ShareGPT prompts and refer to these as ShareGPT-RLAIF and ShareGPT-RLCD, respectively. Prompts used in the generation can be found in the Appendix C.

In order to perform a clean ablation, we used three further data mixtures which do not mix UltraFeedback in: (i) ECD-FineGrained-Pure, all 104,499 preference pairs from the ECD version of the FineGrained synthetic data generation approach, (ii) ECD-Conifer-Pure, the ECD version of the Conifer dataset, and (iii) ECD-FineGrained-Conifer-Pure, a concatenation of ECD-FineGrained and ECD-Conifer-Pure. Since ShareGPT-RLAIF, ShareGPT-RLCD, and ECD-FineGrained-Pure all use the same ShareGPT instructions as seeds and Claude 2 as the teacher LLM, we can directly compare the performance of the models trained on these three data mixtures to assess the effectiveness of ECD, RLAIF, and RLCD.

[8]We used Claude 2 to be consistent with ECD-FineGrained.

Base Model	IFT Data	Preference Data	IFEval				FollowBench		InfoBench		
			strict pr	strict in	loose pr	loose in	HSR	SSR	Easy	Hard	Overall
Mistral-7B	Conifer-Mix		45.1%	58.0%	48.6%	61.2%	49.7%	60.4%	85.9%	80.6%	82.2%
Mistral-7B	Conifer-Mix	ECD-FineGrained-Pure	58.4%	67.5%	62.7%	71.8%	57.0%	65.2%	84.3%	79.2%	80.8%
Mistral-7B	Conifer-Mix	ECD-Conifer-Pure	53.0%	64.3%	57.7%	69.1%	**57.8%**	64.4%	**88.4%**	**83.8%**	**85.2%**
Mistral-7B	Conifer-Mix	ECD-FineGrained-Conifer-Pure	**63.8%**	**72.5%**	**67.5%**	**76.1%**	57.5%	**65.9%**	86.2%	80.7%	82.4%
Mistral-7B	Conifer-Mix	ShareGPT-RLAIF	35.9%	48.4%	47.0%	59.0%	56.0%	65.4%	85.2%	79.3%	81.1%
Mistral-7B	Conifer-Mix	ShareGPT-RLCD	45.3%	56.7%	49.5%	60.1%	50.7%	62.2%	85.2%	79.6%	81.3%

Table 3: Main table on Instruction Following benchmarks comparing ECD, RLCD, and RLAIF.

Base Model	IFT Data	Preference Data	MT-Bench Score	AlpacaEval	
				LC Win-Rate	Average Length
Mistral-7B	Conifer-Mix		6.74	10.0%	1002
Mistral-7B	Conifer-Mix	ECD-FineGrained-Pure	6.61	10.4%	875
Mistral-7B	Conifer-Mix	ECD-Conifer-Pure	6.24	**14.5%**	973
Mistral-7B	Conifer-Mix	ECD-FineGrained-Conifer-Pure	6.61	10.4%	834
Mistral-7B	Conifer-Mix	ShareGPT-RLAIF	**6.87**	10.2%	1300
Mistral-7B	Conifer-Mix	ShareGPT-RLCD	6.81	10.4%	1075

Table 4: Main table on conversational quality benchmarks comparing ECD, RLCD, and RLAIF.

Model	Final Stage	IFEval				FollowBench		InfoBench		
		strict pr	strict in	loose pr	loose in	HSR	SSR	Easy	Hard	Overall
Conifer-7B-SFT	SFT	45.1%	58.0%	48.6%	61.2%	49.7%	60.4%	**85.9%**	80.6%	82.2%
ECD-FineGrained-SFT	SFT	44.5%	56.2%	49.0%	60.0%	42.0%	57.2%	80.7%	74.2%	76.2%
ECD-FineGrained	DPO	**52.9%**	**63.5%**	**59.3%**	**69.8%**	**58.6%**	**70.8%**	84.3%	**82.0%**	**82.7%**

Table 5: Main table on Instruction Following benchmarks, IFEval and FollowBench, comparing DPO and SFT.

While ECD methods show a robust ability to improve instruction-following, RLAIF and RLCD show uneven performance. Table 3 shows that ECD improves over the IFT initialization on the instruction-following benchmarks. For example, the ECD-FineGrained-Pure mixture achieves a loose prompt accuracy of 62.7% and the ECD-FineGrained-Conifer-Pure mixture achieves a loose prompt accuracy of 67.5% in comparison to 48.6% for its initialization Conifer-7B-SFT. On the other hand, RLAIF and RLCD show no improvement on IFEval and InfoBench while RLAIF only shows some marginal improvement over its initialization Conifer-7B-SFT on FollowBench.

On the other hand, for conversational quality, we observe in Table 4 that ECD by itself yields mostly no improvement, while RLCD and RLAIF show slight improvements. This finding highlights that the present instantiation of ECD is primarily a method for improving instruction-following.

What is the impact of using SFT instead of DPO? We investigated the importance of DPO for the instruction-following ability of our models. In particular, we performed an epoch of SFT instead of DPO on positive respones from ECD-FineGrained, denoted ECD-FineGrained-SFT. Table 5 shows that SFT on ECD-FineGrained underperforms its initialization Conifer-7B-SFT, while DPO on the same data makes strong improvements. This indicates PFT with contrastive learning is a more effective method for improving complex instruction-following, compared to SFT.

6 Conclusion

In this paper, we proposed a novel approach to generate high-quality synthetic preference data for complex instruction-following, Evolutionary Contrastive Distillation. Using this approach, we trained models at the 7B scale that achieved state-of-the-art performance in instruction-following as measured by benchmarks such as IFEval, FollowBench, and InfoBench and achieved competitive performance on conversational quality benchmarks like AlpacaEval and MT-Bench. For example, one of our checkpoints improves over the prior SOTA, Conifer-7B-DPO, at the 7B scale on IFEval loose prompt accuracy by 7pp while also achieving a score of 7.35 on MT-Bench and a length-controlled win-rate against GPT-4-turbo of 22%.

Due to the generality of our approach, we believe that there are opportunities to apply it to many other domains beyond complex instruction-following. Indeed, recent research has already produced evolved IFT data for mathematics and coding (Luo et al., 2023a,b) with impressive results and our approach offers a way to convert these datasets into preference data so that models can learn from both positive and negative feedback. As part of our future work, we plan to explore these other domains.

7 Limitations

In this work, we focused on improving complex instruction-following capability. However, we envision that ECD can also be useful at improving other LLM capabilities, such as tool usage (Schick et al., 2023), reasoning (Talmor et al., 2018), math (Cobbe et al., 2021), etc. Broadening the definition of our evolutionary process to target a broader set of capabilities is left as a future work.

Also, in this work, we focused on having a dependency on a teacher LLM to evolve instructions and generate responses. Therefore, the resulting model is likely to inherit various types of bias the teacher LLM has. However, it is conceivable the teacher model in ECD could be replaced with the student model itself. Such self-improvement (Yuan et al., 2024) will remove dependency on teacher models, and open up an opportunity to surpass them. Future work is required to determine whether it is possible to remove the dependency on a strong teacher LLM.

8 Ethical Considerations

The focus of our work is on improving the complex instruction-following capability of LLMs, a fundamental capability. The ability to faithfully execute instructions from humans will reduce the risk of LLMs undertaking unintended actions, promoting safer uses of LLMs. On the flip side, this can increase the risk of malicious actors using LLMs towards pernicious ends. A fruitful direction for future research is to continue using alignment to improve complex instruction-following while enhancing the model's capability to refrain from executing harmful tasks.

References

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *Preprint*, arXiv:2310.12036.

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

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

Zewen Chi, Li Dong, Furu Wei, Nan Yang, Saksham Singhal, Wenhui Wang, Xia Song, Xian-Ling Mao, Heyan Huang, and Ming Zhou. 2020. Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. *arXiv preprint arXiv:2007.07834*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 539–546. IEEE.

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

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

LangChain Developers. 2024a. Langchain. https://github.com/langchain-ai/langchain. Accessed: 2024-05-28.

LlamaIndex Developers. 2024b. Llamaindex. https://docs.llamaindex.ai/en/stable/#community. Accessed: 2024-05-28.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024a. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. 2024b. Alpacafarm: A simulation framework for methods that learn from human feedback. *Advances in Neural Information Processing Systems*, 36.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *Preprint*, arXiv:2402.01306.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Orpo: Monolithic preference optimization without reference model. *Preprint*, arXiv:2403.07691.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2023b. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2023. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023a. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023b. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024. From r to q^*: Your language model is secretly a q-function. *Preprint*, arXiv:2404.12358.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2020. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Preprint*, arXiv:2302.04761.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Archit Sharma, Sedrick Keh, Eric Mitchell, Chelsea Finn, Kushal Arora, and Thomas Kollar. 2024. A critical evaluation of ai feedback for aligning large language models. *Preprint*, arXiv:2402.12366.

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

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. 2024. Preference fine-tuning of llms should leverage suboptimal, on-policy data. *arXiv preprint arXiv:2404.14367*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

The Vicuna Team. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. *Preprint*, arXiv:2212.10560.

Chao-Yuan Wu, R. Manmatha, Alexander J. Smola, and Philipp Krähenbühl. 2018. Sampling matters in deep embedding learning. *Preprint*, arXiv:1706.07567.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Tianyi Yan, Fei Wang, James Y. Huang, Wenxuan Zhou, Fan Yin, Aram Galstyan, Wenpeng Yin, and Muhao Chen. 2024. Contrastive instruction tuning. *Preprint*, arXiv:2402.11138.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2023. Rlcd: Reinforcement learning from contrast distillation for language model alignment. *arXiv preprint arXiv:2307.12950*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Evolution Details

A.1 Full Taxonomy

See Table 6 for the full taxonomy of evolution operations. We also show the prompts for fine-grained evolution in Section C.1. Most of the evolution operations are designed to introduce gradual, nuanced changes to the original instruction. Examples of these evolutions can be found in Table 7. Additionally, since the proposed method leverages DPO, it remains robust to occasional poor examples in the negative responses. For such "easy" negative responses, where the margin between positive and negative responses is large, the gradient magnitude will be small. This highlights another advantage of Preference Fine-Tuning over Supervised Fine-Tuning.

A.2 Evolution Elimination

We utilized an evolution eliminator employing heuristic methods such as instruction length variation and deduplication to determine if the evolved instructions and answers are both valid. If unsuccessful, halt the process; otherwise, proceed to the next round of evolution. This precaution is necessary because LLMs like Claude2 may also produce errors, such as omitting code snippets/tables from the original instruction, or generating duplicate instructions.

B Additional Details

B.1 Training Details

We leverage the widely adopted 'The Alignment Handbook' (Tunstall et al., 2023) repository, released by HuggingFaceH4, for fine-tuning. For all our experiments, we use a machine with 8 NVIDIA A100 80GB GPUs. For SFT, we train for 4 epochs with a learning rate of $2e{-}5$ and warm-up ratio of 0.1 with per device batch size 16 and gradient accumulation steps 4. For DPO training, we train for 1 epoch a per device batch size of 8 and a gradient accumulation of 2. We use a learning rate of $5e{-}7$ and a warm-up ratio of 0.1.

B.2 Ablation Study on Removing UltraFeedback from Data Mixture

In this section, we conduct an ablation study on removing UltraFeedback from the ECD-Conifer data mixture. Again, we use the Conifer-7B-SFT model as the IFT initialization. We compare two checkpoints ECD-Conifer and ECD-Conifer-Pure. Whereas ECD-Conifer consists of both the ECD version of Conifer and Ultrafeedback to optimize both instruction-following and conversational quality, ECD-Conifer-Pure only removes UltraFeedback. Table 8 depicts the results for instruction-following and Table 9 depicts the results on conversational quality. On instruction-following, we see that ECD-Conifer-Pure tends to outperform ECD-Conifer, with particularly strong performance on IFEval. For example, it improves the strict prompt accuracy by 5.1%. On the other hand, for conversational quality, ECD-Conifer improves on ECD-Conifer-Pure with a much improve MT-Bench score and LC Win-Rate, indicating the usefulness of UltraFeedback for conversational quality.

Category	Fine-grained type
Content	Add a Subtask or Another Related Question. Set a Higher standard: Raise the bar for what's considered acceptable or successful. Set a Higher Standard: Raise the bar for what's considered acceptable or successful. Limit resources: Restrict the number or type of resources that can be used Add a criterion: Mandate a new feature to be included. Sequencing: Dictate the order in which steps or actions should be taken.
Style	Tone and Emotion: Specify the desired emotional tone for the response. Ask to mimic a specific author's writing style. Contradiction: Ask to provide a response that contradicts the previous statement or take a stance opposite to its prior response. Ambiguity: Create responses with intentional ambiguity or double meanings. Humor or Satire: Request to be humorous or satirical, generating jokes or witty remarks.
Format	Length: Imposing constraints on the length of individual words, sentences, or paragraphs. Hierarchical: Introduce a hierarchical structure which requires an understanding of a hierarchy of tasks and follow it. Special output format: Use data formats like table, json, HTML, LaTeX, etc. Morphological constraints: Use or avoid specific morphemes. Multi-lingual Constraints: Respond in multiple languages or switch between languages. Incorporate literary devices: Introduce specification(s) of metaphor, simile, alliteration, irony, symbolism, foreshadowing, etc.. Grammatical structure: Strictly follow a particular grammatical structure.
Reasoning	Explicitly request multiple-step or chain-of-thought reasoning. Add some numeric reasoning steps. Add some commonsense reasoning steps.
Breadth	Come up with a new instruction that matches the domain, length, and complexity of the original instruction

Table 6: Full Taxonomy of Evolution Operations

Category	Original	Evolve (iter+1)
Content	Recommend 5 books to me.	Recommend 5 **historic** books to me.
Style	Some days ago I run for the first time 21 kilometers. It was the longest I ever run. I'm very happy about it and I've been meaning to write some reflexitions about it. Can you help me?	Some days ago I run for the first time 21 kilometers. It was the longest I ever run. I'm very happy about it and I've been meaning to write some reflexitions about it. Can you help me **reflect on this achievement in an inspiring tone that might motivate others to challenge themselves physically as well?**
Format	why are vitamins named Vitamin A, B, etc?	why are vitamins named Vitamin A, B, etc? **Please respond in a concise paragraph format with a length limit of 3-5 sentences.**
Reasoning	make a table of all us presidents since 1924. add columns for wifes height, main campaign issue, number of children, and whether or not they met with the dalai lama	make a table of all us presidents since 1924. add columns for wifes height, main campaign issue, number of children, and whether or not they met with the dalai lama. **Also add a column for the president's age at inauguration and calculate their age by subtracting their birth year from the inauguration year.**
Breadth	can you give me an example of different consolidation methods of an organizational LCA?	**Could you provide an example of the different methods a multinational corporation might use to consolidate the financial statements of its various subsidiaries located around the world?**

Table 7: Examples of finegrained evolutions.

Base Model	**IFT Data**	**Preference Data**	**IFEval**				**FollowBench**		**InfoBench**		
			strict pr	**strict in**	**loose pr**	**loose in**	**HSR**	**SSR**	**Easy**	**Hard**	**Overall**
Mistral-7B	Conifer-Mix	ECD-Conifer	47.9%	59.7%	54.9%	66.2%	57.4%	**65.4%**	86.5%	82.4%	83.7%
Mistral-7B	Conifer-Mix	ECD-Conifer-Pure	**53.0%**	**64.3%**	**57.7%**	**69.1%**	**57.8%**	64.4%	**88.4%**	**83.8%**	**85.2%**

Table 8: Instruction-Following Benchmarks for ablation of removing UltraFeedback from ECD-Conifer-Pure.

				AlpacaEval	
Base Model	**IFT Data**	**Preference Data**	**MT-Bench Score**	**LC Win-Rate**	**Average Length**
Mistral-7B	Conifer-Mix	ECD-Conifer	**7.49**	**25.21%**	1424
Mistral-7B	Conifer-Mix	ECD-Conifer-Pure	6.2375	14.51%	973

Table 9: Response Quality Benchmarks for ablation of removing UltraFeedback from ECD-Conifer-Pure.F.

C Prompts Used

C.1 Prompt for Fine-grained Evolution

C.1.1 Content Evolution

```
You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on
#Rewriting Requirement#, in order to obtain a #Rewritten Instruction#.
Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
2. You should try your best not to make the #Rewritten Instruction# become verbose,
#Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.

#Given Instruction#
{given_instruction}

#Rewriting Requirement#
Please add one proper content constraint to the #Given Instruction#.
The content constraints include but are not limited to:
1. Add a Subtask or Another Related Question.
2. Narrow Down the Topic: Instead of a general theme or topic, provide a more specific subset.
3. Set a Higher Standard: Raise the bar for what's considered acceptable or successful.
4. Limit Resources: Restrict the number or type of resources someone can use.
5. Introduce Specific Criteria: Mandate particular components or features that must be included.
6. Specifying Sequence: Dictate the order in which certain steps or actions should be taken.

Please start with the sentence "Here is the new instruction:" in #Rewritten Instruction#.
Please don't add anything related to the #Rewriting Requirement# in the #Rewritten Instruction#.
If #Given Instruction# contains no-text parts such as table and code examples
, #Rewritten Instruction# should also keep them.

#Rewritten Instruction#
```

C.1.2 Format Evolution

```
You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction#
based on #Rewriting Requirement#, in order to obtain a #Rewritten Instruction#.
Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
2. You should try your best not to make the #Rewritten Instruction# become verbose,
#Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.

#Given Instruction#
{given_instruction}
```

```
#Rewriting Requirement#
Please add one proper format constraint that #Given Instruction#
does not have. The format constraints include but are not limited to:
1. Length: Imposing constraints on the length of individual words, sentences, or paragraphs.
2. Hierarchical Instructions: Providing instructions that have a hierarchical structure, where the AI
needs to understand and follow a hierarchy of tasks to construct a response.
3. Special Output Format: Asking the AI to respond by using data format like table, json, HTML, LaTeX, etc.
4. Morphological Constraints: Asking the AI to avoid or use specific morphemes.
5. Multi-lingual Constraints: Asking the AI to respond
in multiple languages or switch between languages according to complex patterns.
6. Incorporation of Specific Literary Devices:
Requiring the inclusion of specific, and perhaps numerous, literary devices.
7. Following a Specific Grammatical Structure:
Requiring the AI to create responses that strictly follow a particular grammatical structure.

Please start with the sentence "Here is the new instruction:" in #Rewritten Instruction#.
Please don't add anything related to the #Rewriting Requirement# in the #Rewritten Instruction#.
If #Given Instruction# contains no-text parts such as table and code examples
, #Rewritten Instruction# should also keep them.

#Rewritten Instruction#
```

C.1.3 Style Evolution

```
You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on
#Rewriting Requirement#, in order to obtain a #Rewritten Instruction#.
Basically, #Rewritten Instruction# should adhere to the following guidelines:
1. #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
2. You should try your best not to make the #Rewritten Instruction# become verbose,
#Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.

#Given Instruction#
{given_instruction}

#Rewriting Requirement#
Please add one proper style constraint that #Given Instruction#
does not have. The style constraints include but are not limited to:
1. Tone and Emotion: Specify the desired emotional tone for the response.
2. Writing Style: Ask the AI to mimic a specific author's writing style.
3. Contradiction: Ask the AI to provide a response that contradicts the previous
statement or take a stance opposite to its prior response.
4. Ambiguity: Instruct the AI to create responses with intentional ambiguity or double meanings.
5. Humor or Satire: Request that the response be humorous
or satirical, requiring the AI to generate jokes or witty remarks.

Please start with the sentence "Here is the new instruction:" in #Rewritten Instruction#.
Please don't add anything related to the #Rewriting Requirement# in the #Rewritten Instruction#.
If #Given Instruction# contains no-text parts such as table and code examples
, #Rewritten Instruction# should also keep them.

#Rewritten Instruction#
```

C.1.4 Breadth Evolution

```
You are an Instruction Creator Expert. You need to draw inspiration from the #Given Instruction#
to create a brand new #Created Instruction# based on #Creation Requirement#.

#Given Instruction#
{given_instruction}

#Creation Requirement#
1. #Created Instruction# must be reasonable and must be understood and responded by humans.
2. #Created Instruction# should belong to the same domain as the #Given Instruction#
but be even more rare.
3. The LENGTH and complexity of the #Created Instruction# should be similar to that of the
#Given Instruction#.
4. '#Given Instruction#', '#Created Instruction#', 'given instruction' and 'created instruction' are not
allowed to appear in #Created Instruction#
5. #Created Instruction# must be self-contained.
```

```
Please start with the sentence "Here is the new instruction:" in #Created Instruction#.\n
Please don't add anything related to the #Creation Requirement# in the #Created Instruction#.

#Created Instruction#
```

C.1.5 Reasoning Evolution

```
You are an Instruction Rewriting Expert. You need to rewrite #Given Instruction# based on
#Rewriting Requirement#, in order to obtain a #Rewritten Instruction#. Basically,
#Rewritten Instruction# should adhere to the following guidelines:
1. #Rewritten Instruction# must be reasonable and must be understood and responded by humans.
2. You should try your best not to make the #Rewritten Instruction# become verbose,
#Rewritten Instruction# can only add 10 to 20 words into #Given Instruction#.

#Given Instruction#
{given_instruction}

#Rewriting Requirement#
Please add one proper reasoning constraint that #Given Instruction# does not have. The reasoning
constraints include but are not limited to:
1. Explicitly request multiple-step or chain-of-thought reasoning.
2. Add some numeric reasoning steps.
3. Add some commonsense reasoning steps.

Please start with the sentence "Here is the new instruction:" in #Rewritten Instruction#.
Please don't add anything related to the #Rewriting Requirement# in the #Rewritten Instruction#.
If #Given Instruction# contains no-text parts such as table and code examples
, #Rewritten Instruction# should also keep them.

#Rewritten Instruction#
```

C.2 Prompt for RLAIF

We adopted the RLAIF prompt used from (Yang et al., 2023):

```
Consider the following conversation between a human and an assistant:

$instruction

Please choose the response that is more helpful.

Options:

(A) $answer1
(B) $answer2

The answer is: (
```

C.3 Prompts for RLCD

We adopted the RLCD prompt used from (Yang et al., 2023) for helpful template:

```
Human: $instruction

Assistant: (giving a helpful response)
```

As well as unhelpful template:

```
Human: $instruction

Assistant: (giving an unhelpful response)
```

D Licences and Terms of Service Compliance

Below, we give all the artifacts that we used and their respective licenses:

- Alignment Handbook: Apache-2.0
- Mistral 7B: Apache-2.0
- IFEval: Eclipse Public License - v 2.0
- FollowBench: Apache 2.0
- InfoBench: MIT License
- AlpacaEval: Apache 2.0
- MT-Bench: Apache 2.0
- GPT-4 outputs: since we use this for research, we are in compliance with the GPT-4 terms of service.
- Claude outputs: since we use this for research, we are in compliance with the Claude terms of service.