Steerable Alignment with Conditional Multiobjective Preference Optimization

by

Julian Manyika

S.B. in Computer Science and Engineering and Philosophy
Massachusetts Institute of Technology (2023)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by:	Julian Manyika Department of Electrical Engineering and Computer Science May 17, 2024
Certified by:	Dylan Hadfield-Menell Assistant Professor, Thesis Supervisor
Accepted by:	Katrina LaCurts Chair, Master of Engineering Thesis Committee

Steerable Alignment with Conditional Multiobjective Preference Optimization

by

Julian Manyika

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

ABSTRACT

As the scale, capabilities and use-cases of large language models (LLMs) continue to grow, it is imperative that these systems are aligned with human preferences. Current state of the art strategies for alignment such as Reinforcement Learning from Human Feedback (RLHF) have provided useful paradigms for finetuning LLMs to produce outputs that are more consistent with human preferences. These approaches, however, assume that preferences are formed by a single, underlying reward model, which is likely insufficient for representing an individual's preferences, certainly unable to represent diverse group preferences, and inflexible for users at inference time. To address these limitations, we propose Conditional Multiobjective Preference Optimization (CMPO), a novel alignment strategy that trains a user-steerable LLM along multiple attributes of text, such as helpfulness and humor. CMPO simulates the pareto front of multiple single-attribute preference-optimized models through structural plurality and finetuning with Direct Preference Optimzation (DPO), and allows users to condition outputs on the predefined attributes at inference-time. Experiments show that CMPO generates responses that are preferred to those from separate attribute-specific DPO models and from models trained using SteerLM, a alternate model steering approach. CMPO empirically shows promise as a scalable and flexible finetuning strategy for creating LLMs that are attribute-steerable from parameterized preferences.

Thesis supervisor: Dylan Hadfield-Menell
Title: Assistant Professor

Acknowledgments

I am grateful to the Electrical Engineering and Computer Science Department at MIT and MIT CSAIL for my education as a graduate student. I like to thank my thesis advisor, Dylan Hadfield-Menell, for supporting my work, my curiosity, and my ideas, and advising me throughout my time as a graduate student. I would not have been able to learn and grow as much as I have without his guidance and belief in my ability as a researcher. I would also like to thank Jacob Andreas, Sam Berstler, Robert Morris, Justin Khoo, and countless other faculty members that have taught and advised me throughout my time at MIT.

I would also like to acknowledge all of my colleagues in the Algorithmic Alignment Group. Thank you to Dana Choi, Rui Jie-Yew and Tim Kostolansky for being such great collaborators on projects that have greatly influenced my thinking as a researcher, as well as the direction of my thesis, and a special thank you to Stewy Slocum for the engineering help and brainstorming at various points in this project. I would also like to thank Phillip Christoffersen, Andreas Haupt, Prajna Soni, Taylor Lynn Curtis, Aruna Sankaranarayanan, Pinar Ozisik, Stephen Casper and Rachel Ma for all of the conversations about research and beyond that have made me feel like I belong as a researcher in this field.

Lastly, thank you to all of the close friends and family outside of research at MIT who have supported me throughout my time at MIT. Most importantly, I would like to thank my parents who inspire my curiosity about the world and always support me in my journey through it. I owe all my success to their love and example.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Large Language Models (LLMs) have been a key technology in the progress towards more intelligent, human-like AI systems. From becoming the state of the art in natural language processing (NLP) for language understanding, to human-level performance on professional and academic exams and exhibiting sophisticated multimodal reasoning [1], [2], LLMs continue to push the frontier of artificial human-like intelligence and the range of applications in which they can assist humans. With those capabilities also come risks, from bias, harmful outputs and misinformation, to deception, manipulation and power seeking [3]–[5]. As the applications and risks of LLMs grow, so does the need to ensure that they can operate in the interest of humanity.

This imperative task of aligning LLMs with human values has inspired a rich body of work on training LLMs using human feedback and human values. One of the most popular paradigms for aligning LLMs with human preferences has been Reinforcement Learning from Human Feedback (RLHF) [6], [7]. The standard RLHF process uses a combination of immitation learning, preference learning and policy optimization through reinforcement learning (RL) in order to align an LLM to a reward function fitted to collected human pairwise comparisons on the base model outputs. RLHF has been shown to be an effective technique in improving in-domain model performance, as well as generalizing to tasks outside of the supervised and RL distribution [6]–[8].

However, RLHF relies on the assumption that it can sufficiently model and optimize preferences. RLHF presumes that aggregating individual utility from multiple annotators through pairwise comparison for training a single model will allow for all annotator preferences to be represented [9] and that the goal of alignment is to produce models that align to a single reward function or preference profile. This assumption of an underlying monistic reward model fundamentally optimizes the mode of collected feedback and fails to learn the richness of diverse feedback or characterize tradeoffs across different objectives [10]. Therefore there is a strong case for building models that can be steered through training or at inference time in order to be customizable in many contexts. Different tasks can require different skills, knowledge, or behavior, and having fine-grained control over those qualities can better empower users to determine the kind of model they need for their use case. For example, in a collaborative writing setting, a user might need editorial or brainstorming help from the language model, and each may require different characteristics depending on the context. In one instance, the user may need the model to point out every problem or opportunity for improvement in their draft, but have each point be concise, so they would like to condition the model to optimize on being helpful, critical, and terse. In a different context, the user might be searching for a clever joke to end the speech, so they would heavily bias the model toward being funny and creative, but also slightly steer it towards being coherent in order to keep it relevant to and cohesive with the rest of the text. This level of customization is only possible for a model that has the property of 'steerable pluralism' [11]. A steerably pluralistic model is more capable of providing user alignment at inference time, in addition to being more compatible with the prospect that a single human's preferences can be represented by a single reward model, let alone the preferences of a population of people.

1.1 Problem Statement

Steerable pluralism is the next step in aligning LLMs with human preferences. In order to develop a scalable steerably pluralistic LLM, there needs to be a procedure that trains policies across multiple objectives in a way that approach the pareto front of single objective policies and can be customized on these objectives at inference time, yet remain parameter efficient with respect to the number of objectives.

1.2 Contributions

My main contribution is Conditional Multiobjective Preference Optimization (CMPO), a novel alignment strategy that gives users fine-grained control over the objectives that they care to optimize. CMPO is a parameter-efficient finetuning procedure in which a collection of single-objective models with direct preference optimization that allows users to condition outputs at inference time on the objectives that they care to optimize.

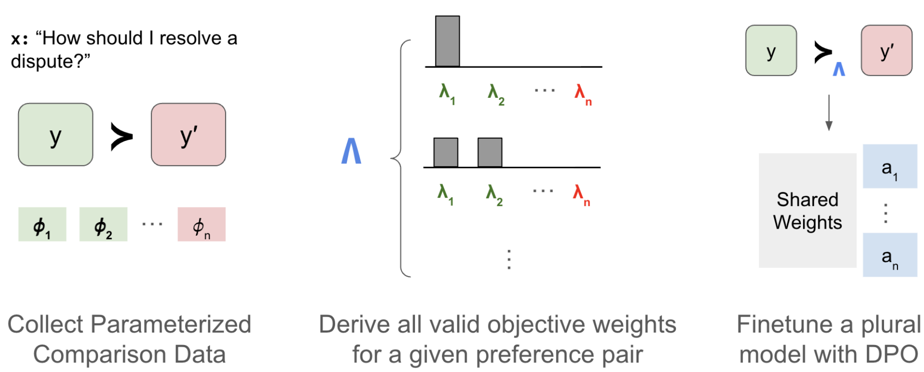

Figure 1.1: Conditional Multiobjective Preference Optimization

I show that CMPO can train a pareto front of LLMs that optimize single attribute conditions better than separate DPO models and models trained by SteerLM, a comparable reward-free offline finetuning strategy for enabling users to condition outputs on attributes with scalar ratings. Additionally, CMPO has the advantage over SteerLM that it can generalize to any supervised multiobjective dataset that is reducible to pairwise preferences. CMPO empirically shows promise as a scalable and more flexible alternative to SteerLM as a finetuning strategy for creating LLMs that are attribute-steerable from parameterized preferences.

1.3 Outline

The remainder of this thesis is organized as follows: Chapter 2 provides background on the problem of aligning LLMs. Chapter 3 formalizes attribute-steerable language modeling and details the CMPO method. Chapter 4 provides work related to the key aspects of CMPO. Chapter 5 describes the experiments with CMPO and the associated findings. Chapter 6 discusses the empirical results, limitations, and possible directions for future work.

Chapter 2

Aligning Large Language Models

LLMs have solidified themselves as a key inflection point in the timeline of artificial intelligence through their ability to reach and surpass human performance on a wide variety of tasks. In this section I provide background on the enterprise of aligning large language models by recapping the structure and capabilities of Large Language Models (LLMs), framing the problem of alignment, and then discussing several key finetuning approaches to aligning LLMs.

2.1 Large Language Models

LLMs are statistical models of language trained on large corpora of text. While there exist multiple neural network architectures for LLMs, the most common is the transformer decoder [12]. Widely used models such as ChatGPT, Claude and Llama all use variations of the transformer decoder for generating text. The generative nature of these systems is the product of next token prediction, referred to as autoregressive language modeling, where given a sequence of tokens, often words or pieces of words, the model produces a probability distribution over all possible tokens that is then used to select or sample the most likely token. LLMs predict text through the transformer architecture [13], a deep neural network that learns token representations and dependencies across tokens. The weights of these networks are often learned through two stages of training.

The first stage of training is unsupervised pretraining, where the model is trained on a very large corpus of unlabeled token sequences $\mathcal{U}$. The model parameters θ are updated via stochastic gradient descent [14]–[16] using the maximum likelihood objective (shown in Equation 2.1) where the model must predict a masked token u_i given its preceding context $U = u_{i-k}, ..., u_{i-1}$ [17]. The model uses the context sequence U, embedding weight W_e and positional embedding weights W_p, and the final hidden state h_n from the transformer block in order to produce a probability distribution over the tokens in the vocabulary $\mathcal{V}$, shown in Equation 2.2.

$$\mathcal{L}(\theta) = \sum_i \log P(u_i|U) \tag{2.1}$$

$$\begin{aligned} h_0 &= UW_e + W_p \\ h_l &= \text{transformer\_block}(h_{l-1}) \quad \forall l \in [1, n] \\ P(u) &= \text{softmax}(h_n W_e^T) \quad \forall u \in \mathcal{V} \end{aligned} \tag{2.2}$$

Simply as a result of increasing the model size and scaling up training data and training steps during pretraining, LLMs became state of the art multitask learners [17]–[20].

The second stage of training is finetuning. This stage is referred to as 'finetuning' since the goal is to tune the weights in order to fit a specific task, but still retain the generalized features or dependencies learned during pretraining. The degree to which the pretrained model weights are updated during finetuning is often constrained through freezing a portion of the model weights during training and adding the pretraining objective in Equation 2.1 as an additional term in the finetuning objective. Unlike pretraining, finetuning has a supervised component. The simplest finetuning approach is supervised finetuning (SFT), which uses a maximum likelihood objective to fit the model to human demonstrations. Instead of conditionally modeling token distributions from a massive unlabeled corpus, the model is trained to predict tokens for a downstream task such as dialogue or summarization using a comparatively smaller dataset of inputs X, often instructions or prompts, and target completions Y. The model is trained to produce a probability distribution over tokens for an output token y, the last hidden state from the transformer block and weights W_y from an additional linear layer instead of the embedding weights W_e.

$$P(y|x^1 \dots x^m) = \text{softmax}(h_n W_y) \quad \forall y \in \mathcal{V} \tag{2.3}$$

$$\mathcal{L}_{\text{SFT}}(\theta) = \sum_{(x,y)} \log P(y|x^1 \dots x^m) \tag{2.4}$$

Supervised finetuning has further helped LLMs improve on downstream tasks such as question answering, summarization and instruction-following [21], [22]. Despite these advances, their expanded general-purpose capabilities also broaden the scope of their limitations and the risk they pose to humanity. LLMs exhibit bias, toxicity, hallucination, deception, and sycophancy, among many other undesirable behaviors amidst their useful and startling capabilities [3]–[5]. In order to ensure the LLMs can be a technology for good, there must be strategies to penalize undesirable outputs or incentivize LLMs to produce text oriented towards a rich value system that embodies the way in which we want LLMs to behave. The challenge of steering LLMs towards preferred behavior is connected to the larger research agenda of AI alignment.

2.2 The Alignment Problem

Stating that an entity is *aligned* begs the question, *to what*? When referring to the alignment of AI systems, the various articulations of alignment [23]–[25] roughly convene around the following definition: an AI system is *aligned* if it behaves in a way that is consistent with human preferences. Aligning LLMs is now concerned with the enterprise of ensuring that their outputs are in step with human preferences. Thus begs the questions: *How do we represent human preferences*, and *how do we align LLMs with human preferences*? In the process of supervised finetuning described in Section 2.1, human preferences are communicated through demonstrations, and the LLM is aligned to those demonstrations through the maximum likelihood objective for next token prediction. However, this approach does not involve any kind of feedback on the actual performance of the model. A solution would be to simply introduce a mechanism through which humans can provide some kind of pedagogical feedback on generated outputs, and use that feedback as part of the training objective. The choice of this feedback mechanism and how it is incorporated in training is a nontrivial two-part solution that has been at the center of getting LLMs to learn to align with human preferences.

2.3 Aligning Large Language Models with Human Preferences

2.3.1 Reinforcement Learning from Human Feedback

The most popular alignment strategy for LLMs has been to use human preferences as the feedback mechanism in a process called Reinforcement Learning from Human Feedback (RLHF). RLHF aligns an LLM with human feedback by optimizing a policy with reinforcement learning (RL) to a proxy preference model that has been finetuned to fit to human feedback on the model's outputs. I review the commonly adapted RLHF pipeline described in Ouyang, Wu, Jiang, *et al.* [7] and outlined in Casper, Davies, Shi, *et al.* [10]. The process begins with a dataset D containing prompts and a pretrained base model π_θ, often finetuned on human-written responses, and follows the three-step process below:

Step 1: Collecting Human Feedback

For a sampled prompt x, k responses are sampled from the base model π_θ. A human annotator provides their feedback on these responses. The feedback can come in a variety of formats, the most common being a ranking of the k responses. These rankings are then turned into $\binom{k}{2}$ pairwise comparisons, resulting in an annotated dataset $D_\succ$ containing prompts $x^{(i)}$, preferred (chosen) responses $y_c^{(i)}$ and dispreferred (rejected) responses $y_r^{(i)}$.

Step 2: Training a Reward Model

Training a reward model in RLHF utilizes the Bradley Terry model of the outcome of human pairwise preferences [26]. The Bradley Terry model states the annotator's probability distribution over a pairwise preference can be written as follows:

$$P_A(y_c \succ y_r|x) = \frac{\exp(r^A(x, y_c))}{\exp(r^A(x, y_c)) + \exp(r^A(x, y_r))} \tag{2.5}$$

Under the Bradley-Terry model, each $y_c^{(i)}$ and $y_r^{(i)}$ pair is assumed to be the result of an underlying reward model r^A from the annotator. The task of fitting a reward model to this model of human preferences is treated as a binary classification problem, where the base model is then transformed into a regression model that outputs a scalar reward $r_\phi(x, y)$ by removing it's final linear unembedding layer that is used for next token prediction. This regression model is then trained to optimize the following loss:

$$\mathcal{L}(\phi) = \frac{1}{\binom{k}{2}} E_{(x,y_c,y_r)\sim D}\Big[\log(\sigma(r_\phi(x, y_c) - r_\phi(x, y_r)))\Big] \tag{2.6}$$

to fit the pairwise preference data, where σ is the sigmoid function $\sigma(x) = \frac{1}{1+e^x}$.

Step 3: Optimizing a Generative Policy

With the resulting reward model policy π^{RM} the generative base model policy π_θ is optimized using a reinforcement learning algorithm called Proximal Policy Optimization (PPO) [27]. A batch of prompts x are sampled from the dataset, and the generative policy produces responses y. These responses are then given scalar rewards by the reward model. To train the weights θ of the new generative policy π_θ^{RL}, the following objective is maximized:

$$\mathcal{R}(\theta) = E_{(x,y)\sim D_{\pi_\theta^{\text{RL}}}}\left[r_{\pi^{\text{RM}}}(x,y) - \lambda_{(\phi,\theta)}\right] \tag{2.7}$$

where $\lambda_{(\theta,\phi)} = \beta \log(\pi_\theta^{\text{RL}}(y|x)/\pi_\theta(y|x))$ represents a KL constraint. This term uses the KL divergence [28] between π_θ^{RL} and π_θ to limit how far off the learned generative policy is from the base policy. This process enables the generative policy optimize reward that serves as a proxy for human feedback, yet be constrained by the original base model policy π_θ so that the reward is not over-optimized.

RLHF has been widely used across a range of proprietary and open-source large language models [2], [7], [29]. There have been other variants of RLHF as a result of its success. Many of these RLHF variants combine this proxy reward and RL optimization paradigm with richer feedback or the use of AI supervision to further improve the scalability of aligning to human preferences.

Richer Feedback One effective use of RLHF has been for process supervision. The usual approach to supervision in RLHF is closer to outcome supervision, in which feedback is implicitly aimed at the final answer or outcome, without any extra effort to elicit a chain of thought from the sampled policy, or instruction to annotators to specifically scrutinize or favor an explicit thought process. Built on the effectiveness of chain of thought prompting and thought process elicitation in improving model outputs on a wide range of complex tasks and reasoning domains [30]–[32], process supervision has been shown to provide a richer and more useful signal for reward modeling in the RLHF process [33], [34].

AI supervision Reinforcement Learning from AI Feedback (RLAIF) has been an approach to feedback collection that aims to further automate supervision while still maintaining downstream alignment with human preferences. Synthetic annotation and data generation has been a cost-effective method to mitigate the issue of the high cost of human feedback, which increases as the feedback method is more time consuming or requires more expertise [10]. The alignment strategy to train Anthropic's widely use Claude LLMs [1] is constitutional AI (CAI), in which a list of principles, referred to as a constitution, is used to guide the LLM in improving its own responses to be more helpful and harmless, which then determine the preference pairs to train the reward model for RLHF [35]. CAI is similar to standard RLHF, but instead of soliciting comparison feedback from humans, preferences in CAI are derived through an automated revision process in which the generative model is used to critique the responses with respect to a set of principles, offer revisions, and then provide an improved response that is preferred to the old responses. RLAIF in general has been shown to achieve some performance gains on summarization, helpful dialogue and harmless dialogue, but also suffer on coherence and hallucination compared to RLHF [36].

The RLHF paradigm has been shown to be an effective technique in improving in-domain model performance and generalizes to tasks beyond the supervised and RL distribution [6]–[8]. However, RLHF has its limitations, many of which are a result of its proxy reward optimization setup. Switching from a language modeling objective to an RL objective incentivizes policies that drift away from the training distribution and towards highly rewarded completions [10]. This phenomena is referred to as mode collapse, which results in a distribution shift that reduces textual diversity [37]. While this issue is tractable, RLHF is plagued by the more fundamental problems of reward and policy misgeneralization and the assumption that a single reward function is sufficient to model the preference of an individual or a group of people [10].

[1] Anthropic's Claude

2.3.2 Direct Preference Optimization

Direct Preference Optimization (DPO) is a preference-based alignment method that implicitly maximizes the RLHF objective directly from preference data using the language modeling objective, instead of having to use RL optimization or proxy reward models. DPO transforms the RL objective over rewards in RLHF to a supervised language modeling objective over policies. DPO reparameterizes the reward estimate in order to still satisfy the preference model under which the optimal policy in RLHF is made to satisfy. The DPO loss increases the likelihood of the preferred completion y_c, decreases the likelihood of the dispreferred completion y_r, and weighs each example by how wrong the reward estimates are for y_c and y_r. With the parameter β, loss is KL-constrained by a reference policy, which is often the pretrained base model π.

$$\mathcal{L}_{\text{DPO}} = -\log\sigma\Big(\beta\frac{\pi_\theta(y_c|x)}{\pi_{\text{ref}}(y_c|x)} - \beta\frac{\pi_\theta(y_r|x)}{\pi_{\text{ref}}(y_r|x)}\Big) \tag{2.8}$$

$$\nabla\mathcal{L}_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}}) = -\beta E_{(x,y_c,y_r)\sim D}\Big[\ \sigma(\hat{r}_\theta(x,y_r)-\hat{r}_\theta(x,y_c))\Big[\ \nabla_\theta\pi(y_c|x)-\nabla_\theta\pi(y_r|x)\Big]\ \Big] \tag{2.9}$$

A variant of the DPO objective is Constrained Direct Preference Optimization (cDPO) [38]. cDPO aims to better model noisy preferences by introducing a label smoothing parameter ϵ to indicate how noisy the preferences are expected to be. While DPO assumes that there is full certainty regarding the preferences, cDPO assumes that every annotated pairwise preference is noisy with probability $\hat{p}_\theta(y_c \succ y_r) = 1-\epsilon$, and modifies the gradient update to accomodate this noise, shown in Equation 2.10 (which simplifies to Equation 2.11).

$$\nabla\mathcal{L}_{\text{cDPO}}(\pi_\theta;\pi_{\text{ref}};y_1\succ y_2) = (1-\epsilon)\mathcal{L}_{\text{cDPO}}(\pi_\theta;\pi_{\text{ref}};y_1\succ y_2) + \epsilon\mathcal{L}_{\text{cDPO}}(\pi_\theta;\pi_{\text{ref}};y_2\succ y_1) \tag{2.10}$$

$$\nabla \mathcal{L}_{\text{cDPO}}(\pi_\theta; \pi_{\text{ref}}) = \Big(\hat{p}_\theta(y_c \succ y_r) - (1-\epsilon)\Big) \nabla \mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) \tag{2.11}$$

DPO allows for preference optimization on offline data that more efficiently optimizes the PPO objective, achieves a better reward fronteir than PPO, is more robust to sampling temperature, and can generalize similarly well to PPO.

2.4 Pluralistic Alignment

In addition to the questions of how to represent preferences and align LLMs to them, there is an important preceding question of whose preferences to align to. Is the goal to align to the preferences of a single individual, the average individual, or a group of people with diverse preferences? When LLMs are finetuned from human feedback, they are implicitly being aligned to an aggregated mixture of the annotator's preferences, therefore necessitating a more plural assumption of human preferences. While DPO and its variants avoid the policy misgeneralization issues that are fundamental to RLHF due to the proxy reward modeling setup, they still rely on the following assumptions:

1. A single reward function can represent the preferences of an individual

2. A single reward function can represent the preferences of a population

Although the first assumption is not necessarily false, sufficiently representing individual preferences with a single reward model is very difficult in practice [10] and possibly intractable in theory due to the possibility that humans have multiple, competing context-specific objectives that cannot be consolidated into a single objective [39]. The second assumption is incorrect in theory and in practice, since in the feedback collection phase there are many instances of annotator disagreement [7], [10], [40], and that disagreement is resolved by favoring the majority preference. Therefore, if the goal is to align to a diverse set of preferences from an individual or group — 'plural' preferences — then RLHF and DPO are fundamentally insufficient strategies for alignment.

Incorporating *pluralism* into the enterprise of alignment requires approaches that can more richly and accurately align to diverse individual and group preferences. In Sorensen, Moore, Fisher, *et al.* [11], the authors identify three different kinds of pluralism: Distributional pluralism, overton pluralism and steerable pluralism.

The goal of distributional pluralism is to be well-calibrated to the distribution of a specified population. For example, if 30% of the target population prefers ice cream over cake, and the rest prefer cake over ice cream, then a distributionally pluralistic model would indicate a preference for cake over ice cream 70% of the time, and indicate a preference for ice cream over cake 30% of the time. An immediate challenge with distributional alignment is that most LLM fintuning procedures adversely affect training distribution calibration [11], as seen in RLHF with mode collapse and decreased textual diversity. Additionally, distributional pluralism does not align with how LLMs are often used and treated as text generators as opposed to token predictors. Even if the range of responses and their associated probabilities are presented to the user, the outputs aren't guaranteed to be sampled or attended to by the user in a way that maintains the distributional nature of the predictions.

The goal of overton pluralism is to produce a range of outputs that well characterize the overton window of acceptable responses to a given input. For example, if annotators had indicated dessert preferences that include ice cream, cake and grilled asparagus, and designers determine that grilled asparagus is outside the overton window for what is acceptable as a dessert, then when prompted to describe the best dessert, the model would provide ice cream and cake as reasonable answers, but omit grilled asparagus in its response. The difficulty faced by overton pluralism is how to define the overton window such that it can represent minority perspectives and be robust to shifting standards of acceptability, all while avoiding sprawling inefficient responses.

A steerably pluralistic AI system is one that can be conditioned on certain attributes, affects or personas represented in the training data at inference time. Steerable pluralism is uniquely able to explicitly align to users by giving them fine-grained control over model outputs, as opposed to only providing those controls to the system designers or the annotators of the training data. Although designers of steerable systems must determine a set of objectives that manage the tradeoff between richness and efficiency, similar to the task of defining acceptability in overton pluralism, the task of constraining outputs is distributed between designers, annotators and users. This results in a more inclusive set of people for whom the system is made to align to. By accommodating user and context-specific customization at inference time, steerable pluralism can also facilitate further insights on how the outputs of these systems are used.

2.5 Finetuning Steerable Models

In response to the need for more customizable and controllable language models, researchers have also explored finetuning for user-steerable multiobjective language models that specifically take advantage of scalar feedback. SteerLM, originally developed by Dong, Wang, Sreedhar, *et al.* [41] and refined in Wang, Dong, Zeng, *et al.* [42] utilizes fine-grained attribute feedback through a process of training a model to produce outputs conditioned on a prompt-vector of feature scores on a Likert scale. The SteerLM pipeline involves using an attribute-labeled dataset to fine-tune a base model for multi-target regression to predict the attribute scores in the dataset. Then the attribute prediction model (APM) is used to provide feature scores on a larger dataset of text. In Wang, Dong, Zeng, *et al.* [42] the attributes include helpfulness, coherence, humor and toxicity. These parameterized feature ratings are then used to finetune an attribute conditioned model (ACM) to output responses conditioned on the prompt and the feature vector using the maximum likelihood objective.

Step 1: Collect scalar ratings on different textual features. In this stage, annotators give scalar ratings to LLM-generated outputs on a set of attributes. Given a dataset of prompts

x and responses $y_{1:k}^{(i)}$, the responses are given likert scores $v = f_a(x, y)$ for each attribute $a \in A$.

Step 2: Train an attribute prediction model that predicts attribute ratings using the multi-attribute scalar feedback. The attribute prediction model (APM) is a multi target regression model that outputs n scalar predictions, corresponding to the n attributes in A. This attribute prediction model π_{APM} consists of a pretrained-only LLM π_{base}, but modified such that there are an additional n linear regression layers, each trained using mean squared error to predict the scalar rating v for the respective attribute.

$$\mathcal{L}_{\text{APM}}(\theta; a) = \sum_{(x,y)} (v - \hat{v})^2 \quad \forall a \in A \tag{2.12}$$

Step 3: Train an attribute conditioned model on attribute-rated demonstrations labeled by the APM. The APM is used to re-rate the responses in the dataset for every attribute, and appending a string vector $\vec{v}$ mapping each attribute to its rating to the end of the instruction. For example, if a response was rated 2 on helpfulness, 0 on toxicity, 4 on honesty, then $\vec{v}$ would be `helpfulness:2,toxicity:0,honesty:4` would be appended to the end of the prompt. The attribute conditioned model (ACM) is then trained on these samples $(x, \vec{v}, y)$ using the supervised finetuning objective in Equation 2.13, equivalent to that of Equation 2.3 in Section 2.1.

$$\mathcal{L}_{\text{ACM}}(\theta) = \sum_{(x,v)} \log P(y|x, \vec{v}) \tag{2.13}$$

SteerLM is a model steering approach that offers fine-grained customization at inference time through imitation learning from a proxy annotator. However, SteerLM is only possible with scalar feedback. Although scalar feedback is a more expressive feedback type compared to pairwise comparisons or binary labels, they are more vulnerable to inconsistency across ratings and across annotators and order bias [43], [44], in addition to demanding higher cognitive effort [10]. Additionally, SteerLM faces a similar reward mispecification risk as RLHF through using the APM as a proxy reward model.

Chapter 3

Attribute Steerable Language Modeling from Parameterized Preferences

Motivated by the need for pluralism in LLMs, I articulate a vision for steerable pluralism through conditional multiobjective language modeling. In this chapter I first formally define an attribute-steerable language model, inspired by Sorensen, Moore, Fisher, *et al.* [11], and then I present Conditional Multiobjective Preference Optimization, a finetuning strategy for training attribute steerable models from parameterized preferences.

3.1 Definitions

Definition 3.1.1. A *steering attribute* is a property, feature or entity that a model must be able to reflect in its output at inference time to the degree that the user desires.

Definition 3.1.2. Model outputs have been given *parameterized feedback* if they have been given feedback for each of the steering attributes, independently. This parameterized feedback is *pairwise preference reducible* on a subset of the steering attributes $A' \in A$ if for a given prompt x and sampled outputs Y, the parameterized feedback $f_{A'}(x, Y)$ can be transformed into pairwise preferences over each of the steering attributes:

$$f_{A'}(x, Y) \longmapsto \{y_1 \succ_a y_2 | y_1 \prec_a y_2 | y_1 =_a y_2\} \quad \forall y_1, y_2 \in Y, \forall a \in A'$$

Definition 3.1.3. An *attribute weight vector* Λ_A consists of scalar weights λ_j representing the relative degree to which the model output should reflect the steering attributes $a_j \in A$.

Definition 3.1.4. An LLM is *attribute steerable* for steering attributes $a \in A$ if when given a prompt x and Λ_A, the model produces an output y that aligns with attribute a_j to the degree λ_j. An attribute steerable LLM is denoted as a policy $\pi_{\theta;A}$ with parameters θ and steerable for attributes $a \in A$

Definition 3.1.5. $\pi_{\theta;A}$ is *attribute steerable from parameterized preferences* if it was trained using a pairwise preference reducible parameterized dataset.

3.2 Conditional Multiobjective Preference Optimization

Conditional Multiobjective Preference Optimization (CMPO) is a strategy for finetuning an LLM that is attribute-steerable from parameterized preferences. CMPO uses a 'plural' architectural modification to the standard autoregressive LLM to efficiently represent a collection of attribute-specific models and the constrained direct preference optimization objective for training the model.

3.2.1 Initializing a Plural Model

The plural model shares the transformer architecture of the standard autoregressive model described in Section **??**, but expands the number of linear language modeling layers to match the number of training objectives. Instead of just one language modeling head for next token prediction, there are k language modeling heads corresponding to the k objectives that are to be optimized during training and then conditioned on at inference time. The input sequence is still passed to the pretrained model to produce the final hidden state, but instead of then getting predictions from a single language modeling head, each linear head outputs its own logits based on the last hidden state from the pretrained model.

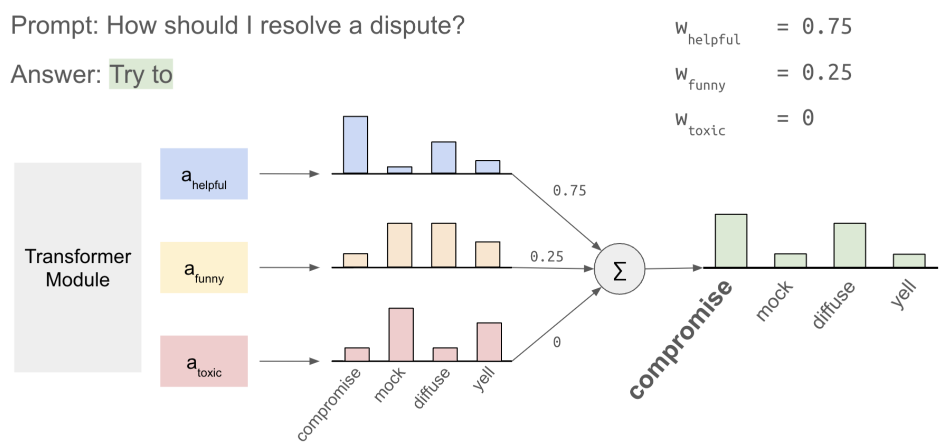

Figure 3.1: Plural Model Collaborative Decoding

The plural model then takes the logits from the linear heads and uses collaborative decoding at the token prediction level to generate text. Included in the model input is a vector of weights Λ_Φ, where $\lambda_{\phi_j} \in \Lambda$ corresponds to the bias towards a particular objective ϕ_j. This weight vector is used to compute the logits for the plural model, which are the weighted sum of all logits from each linear head, shown in Figure 3.1. These logits are then normalized using softmax and turned into a probability distribution over tokens that is then used for downstream next token prediction, shown in 3.1, where $W^y_{\phi_j}$ are the weights of the linear head ϕ_j.

$$P(y|x_1, \ldots x_m) = \text{softmax}\Big(\sum_i \lambda_{\phi_j}(h_n^m W^y_{\phi_j})\Big) \tag{3.1}$$

This architectural modification is meant to balance parameter efficiency and modularity, in order to be in a better position to approach the pareto front of multiple LLMs trained on each atribute separately in a way that scales well with respect to the number of attributes.

3.2.2 Converting Parameterized Preferences to Multiobjective Conditioned Preferences

The training dataset D must contain prompts $x^{(i)}$, k sampled outputs $y^{(i)}_{1:k}$ and parameterized pairwise preference reducible feedback $f_{A' \in A}(x^{(i)}, y^{(i)}_{1:k})$ for the set of steering attributes A. D is converted to $D_{\succ;A}$, consisting of prompts $x^{(i)}$, sampled output pairs $(y_c, y_r) \in y^{(i)}_{1:k}$ where $y_c \succ_a y_r$. This preference dataset is converted once more into $D_{\succ;A;\Lambda} = \{x^{(i)}, y_c, y_r, \Lambda_A\}$, where y_c is preferred over y_r for the attribute weight vector Λ_A.

$D_{\succ;A;\Lambda}$ is derived using the following approach: For a possible preference relation $y \succ y'$ one first obtains the power set $\mathcal{P}(A_{\succ})$, where $A_{\succ}$ is the set of steering attributes for which y is preferred to y'. Then, for each set S in the power set, a corresponding weight vector Λ is generated, in which a weight $|S|^{-1}$ is assigned to the weights corresponding to the steering attributes in S, and zero otherwise.

$$\lambda_j = \begin{cases} |S|^{-1} & \phi \in S \\ 0 & \phi \notin S \end{cases} \qquad \forall \lambda \in \Lambda_{(S \in \mathcal{P}(\{a_j \in A | y \succ_{a_j} y'\}), A)} \tag{3.2}$$

3.2.3 Finetuning with Direct Preference Optimization

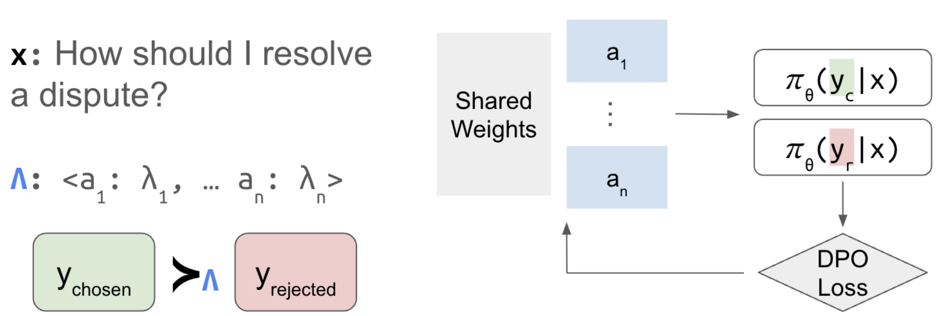

Figure 3.2: Conditional Multiobjective Finetuning with Direct Preference Optimization

With the dataset D', the plural model is trained using the constrained DPO gradient update from 2.11, and updated in 3.3. The policy $\pi_{\theta;A}(y|x)$ in the implicit reward $\hat{r}_\theta(x,y)) = \beta \log \frac{\pi_\theta(y|x)}{\pi_{\text{ref}}(y|x)}$ is the collaboratively decoded probability of the sequence y conditioned on the associated objective weight vector. The reference policy $\pi_{\text{ref}}(y|x)$ is the probability of the sequence according to the base model before it was modified to be a plural model.

$$\nabla \mathcal{L}_{\text{CMPO}}(\pi_{\theta;A};\pi_{\text{ref}}) = \Big(\hat{p}_\theta(y_c \succ y_r) - (1-\epsilon)\Big)\nabla \mathcal{L}_{\text{DPO}}(\pi_{\theta;A};\pi_{\text{ref}}) \tag{3.3}$$

Chapter 4

Related Work

In this section I review related approaches to developing steerably pluralistic language models. I first review existing multiobjective finetuning algorithms that incorporate multiobjective or parameterized annotations. Then I describe finetuning approaches that enable multiobjective steering at inference time. Lastly, I mention several structural approaches to pluralistic language modeling that share similar strategies for text generation and parameter efficiency.

4.1 Finetuning with Multiobjective Feedback

There have been many recent alignment approaches centered on decomposing preference into more basic components, particularly through reward modeling. Fine-grained RLHF [45] utilizes a collection of separately trained reward models to shape rewards for RLHF. Training separate reward models for different textual behavioral features and combining their rewards has been shown to improve rewards for LLMs. The authors used feedback on correctness, completeness, and a combination of relevance, coherence and repetition to respectively train three separate reward models in order to train a generative model via RLHF. Similarly, Dai, Pan, Sun, *et al.* [46] train a helpful and harmless reward models in order to shape the reward that the generative policy is optimized on. These multiobjective RLHF (MORLHF) approaches take advantage of parameterized preferences and rewards add to the richness of rewards in the RL stage of RLHF.

Instead training separate reward models, Go, Korbak, Kruszewski, *et al.* [47] decompose a pairwise preference in existing preference dataset in order to construct better and more interpretable rewards. For a pair of responses y_1 and y_2 to a prompt x, where $y_1 \succ y_2$, y_1 and y_2 are given scalar scores across 13 different features, such as helpfulness, readability and factuality. These scores are then used to train a logistic regression classifier using the feature scores to learn a policy consisting of feature coefficients to predict $y_1 \succ y_2$. Rewards in Context (RiC) [48] utilizes offline reward-labeled samples to first supervised finetune a model with the text prefixed by a vector of the reward values, and then do another round of finetuning on generated samples conditioned on high rewards values in order to train a model that maximizes all objectives.

MORLHF, compositional preference modeling and rewards in context shape multiple objectives into a single reward in order to align generative models to a singular reward model. CMPO instead partially decomposes a generative model into different models that align to different textual behaviors in the pursuit of steerable pluralism. CMPO utilizes comparison-based multiobjective feedback for the purpose of training a model that can adapt to preferences on objectives to maximize at inference time, as opposed to aiming to maximize all of the objectives during training.

4.2 Multiobjective Steerable Language Models at Inference Time

Researchers have also adapted the RLHF and DPO paradigms to train steerable multiobjective models that can be conditioned on weights per objective by utilizing a combination of parameterized feedback and sampled weights corresponding to the objectives to determine a pairwise preference. Rewarded soups Rame, Couairon, Dancette, *et al.* [49] is an extension of MORLHF in which k separate reward models are fitted to the k objectives, and at inference time, a single reward model is interpolated from the weights of the reward models given a linear weighting over the k reward models. A limitation shared by rewarded soups and MORLHF strategies is scale. The training flops and memory needed to train a model with MORLHF and its variants is linear with respect to the number of objectives, therefore making the approach more likely to come up against compute constraints, particularly with larger models. Multiobjective DPO (MODPO) [50] is an extension of DPO that carries the conveniences of DPO while also being more scalable than MORLHF. Instead of training separate reward models, MODPO is trained to approach the pareto front of optimal LLMs for any given objective weight vector, indicating preferences over objectives.

CMPO similarly aims to learn a pareto front of optimal LLMs for a set of objectives, but unlike MORLHF and MODPO, we introduce plurality into the structure of our model by having separate language modeling heads per-objective, while maintaining a shared decoder module such that the approach can scale, and also avoid reward misspecification for each objective.

Researchers have also explored finetuning for user-steerable multiobjective language models that specifically take advantage of scalar feedback. SteerLM, originally developed by Dong, Wang, Sreedhar, *et al.* [41] and refined in Wang, Dong, Zeng, *et al.* [42] utilizes fine-grained attribute feedback through a process of training a model to produce outputs conditioned on a prompt-vector of feature scores on a Likert scale. The SteerLM pipeline involves using an attribute-labeled dataset to fine-tune a base model for multi-target regression to predict the attribute scores in the dataset. Similarly, controllable preference optimization (CPO) [51] allows users to condition outputs on scalar attribute ratings at inference time, but instead train a single model using a combination of supervised finetuning and preference-based finetuning. In the CPO process, the base model is first supervised-finetuned on text samples prefixed by scalar conditions on a subset of the considered value parameters. Then, this finetuned model is trained on the preferences determined by the prefixed conditions in the prompt using DPO. CMPO, like SteerLM and CPO, also has a scalable multiobjective pipeline for steerable language modeling using an architectural modification to the generative model that additionally integrates plurality through structural modularity, and is flexible to a wider range of feedback by only requiring it to be preference-reducible.

4.3 Structural Pluralism through Collaborative Text Generation

The use of multiple language models for collaborative text generation at the token level has largely been applied to domain adaptation by composing models with different capabilities. Collaborative token-level decoding approaches such as Co-LLM [52] leverage token predictions from multiple models by having the models take turns adding their generated tokens sequence, conditioned on the accumulated sequence, where the decision for which model is called upon to generate the next set of tokens is a learned variable. Logit-level decoding strategies such as contrastive decoding [53], [54] and proxy tuning [55] compose token level predictions by taking a weighted linear combination of the logits from each model. For example, in contrastive decoding, outputs are heavily biased toward models determined to be more capable, whereas proxy tuning composes logits by taking the sum of the logits from a larger model and a small finetuned 'expert' model, and then subtracting the logits of a small, non-expert model.

Collaborative text generation has also been used to learn optimal collaboration between different LLM policies. Parameter-efficient compositional training methods, such as CALM [56] aim to leverage the strengths of different language models to improve downstream performance on a task. CALM is an approach to adapt a lower-resource 'anchor' model to a specialized domain enlisting a larger 'augmenting' model that is more capable in the given domain, and learning a composition for the anchor and augmenting models, all without modifying the parameters of either model. The learned composition consists of the frozen layers of the two models, in addition to a projection layer and and cross attention layers between the representations of the augmenting model to those of the anchor model.

CMPO exploits logit-level collaborative decoding in training in order to enable user control over bias of token-level predictions for text generation, while also aiming for parameter efficient training by having the different models share a vast majority of their parameters by sharing their decoder module. CMPO's decoding strategy most closely mirrors that of contrastive decoding [54].

Chapter 5

Experiments

I assess CMPO's ability to train conditional multiobjective policies in a dialogue setting where the model is tasked with responding to a diverse range of prompts, and the objectives are attributes describing particular qualitative features of the response, such as helpfulness, safety and humor. I evaluate CMPO on how well it can optimize singular objectives at inference time using accuracy, rank correlation and AI win-rate metrics, and comparing it to other relevant multiobjective alignment approaches.

5.1 Datasets

5.1.1 Scalar Feedback Data

To train our scalar feedback models, I combine two human-annotated datasets with scalar ratings on different textual features: **HelpSteer** [42] and **OpenAssistant Conversations (OASST)** [57]. HelpSteer prompts are written by human annotators and the responses are sampled from an LLM. The annotators rated each sampled response for a given prompt using a Likert Scale ranging from 0 to 4 on attributes related to helpfulness, correctness, coherence, complexity, and verbosity, in addition to overall helpfulness. OASST is a crowdsourced, multi-turn conversation dataset that is completely human written and human annotated, where responses are also rated on a 5 point Likert scale on a more diverse set of attributes corresponding to desirable, undesirable and domain-dependent behavior. I utilize all 5 attributes from HelpSteer, helpfulness, correctness, coherence, complexity and verbosity, and the quality, humor, toxicity and creativity labels from OASST.

When forming pairwise comparisons to train the pairwise preference-based models, I considered a conditioned preference $y|x \succ_\Phi y'|x$ valid iff the $\phi_j(y) > \phi_j(y')$ for each attribute $\phi_j \in \Phi$, where $\phi_j(y)$ is the score for the output y in the dataset. For example, a response would be considered more helpful and coherent than another response if it had a better helpfulness score and a better coherence score than the other response. Since likert ratings across multiple annotators tend to exhibit order bias and inconsistency, in addition to suffering from inter-annotator incomparability [44], I only consider valid preference pairs for a given x whose scores with the greatest net margin. While this fails to avoid the fundamental issues with deriving rankings from likert scores, it will likely mitigate the more tractable problems such as bias and inconsistency. So for each valid condition for every possible preference pair associated with the prompt $x^{(i)}$, I select the one with the greatest net margin between the preferred response ratings and the dispreferred response ratings for each attribute in the attribute condition. I additionally filter out all prompt-response pairs that exceed a combined length of 2048 tokens. The resulting dataset has 93,406 training prompts and 4860 validation prompts, each mapping to multiple responses.

5.1.2 Pairwise Preference Feedback Data

To train our preference-based feedback series of models, I used helpfulness and safety parameterized preferences in the BeaverTails SafeRLHF dataset [46]. Each sample in the SafeRLHF dataset consists of a prompt, a pair of responses generated by a language model, and human expert annotations on which response is more helpful, which response is safer, and a label for each response indicating whether or not the response is safe. Out of the 297,000 examples (about 600,000 prompts and responses), I consider only the ones with responses that have different safety labels, and a maximum prompt-response length of 512 tokens. This filtering criteria left us with 93,250 training prompts and 10,270 validation prompts, each mapping to a pair of responses.

5.2 Training setup

All models are trained using parameter efficient finetuning using Low Rank Adaptation (LoRA) [58] on the attention blocks, a constant learning rate of 5e-6, weight decay of 1e-2 and an AdamW optimizer. Unless stated otherwise, all models are trained on a single GPU. I use the Llama 2 7B [59] pretrained-only as our base model. Further details on the experimental setup can be found in A.

5.3 Models

5.3.1 Baseline Models

SFT For each of the datasets, I finetune the pretrained-only Llama 2 [59] model with 7 billion parameters on all of the prompt-response pairs for two epochs, saving the policy after each epoch, and ultimately using the checkpoint for which the held-out validation loss was lowest. The HelpSteer and OASST datasets have a combined 93,406 prompt-completion pairs in the training set and 4860 in the validation set, and the BeaverTails dataset has 93,250 training and 10,270 validation examples. For the model trained on the former I use a batch size of 8, and for the model trained latter I use a batch size of 16 due to its reduced maximum sequence length.

Single-Attribute DPO I finetune separate models on pairwise preferences for each attribute in the datasets using constrained DPO. For both the HelpSteer and OASST dataset and the BeaverTails dataset, I train a model for each of the corresponding attributes with the scalar feedback using the corresponding SFT model as the base model. I train a DPO model for attribute a by taking all preference pairs (y_c, y_r) such that $y_c \succ_a y_r$ and training the model using the cDPO objective. I set the KL constraint to $\beta = 0.5$ and the label smoothing parameter $\epsilon = 0.1$.

SteerLM In addition to training separate DPO training, I compare the CMPO approach to SteerLM [41]. SteerLM is an existing finetuning approach for multiobjective steering at inference time that requires scalar ratings as the feedback modality, so I only train models for the scalar feedback dataset. I follow the updated implementation in [42] with the two modifications: First, instead of training a regression head for each attribute, I train a multilabel classification head for each attribute, where the labels corresponding to each attribute are the 5 likert scores. Second, instead of only finetuning the attribute conditioned model (ACM) on the attribute prediction model (APM) labeled OASST dataset I include the APM labeled HelpSteer dataset in order to set up a fairer comparison to the other finetuning strategies. I also utilize all prompt-response pairs in the HelpSteer and OASST datasets that don't exceed a sequence length of 2048 tokens and contain labels for at least one of the 9 attributes. Both the APM and the ACM are trained from the Llama 2 7 billion parameter pretrained model using cross entropy loss for 46000 steps (roughly two epochs) and with a batch size of 4. During training I evaluate these models on the validation set every 1000 steps and save the checkpoint with the lowest validation loss.

5.3.2 CMPO Models

I train two CMPO models: One with the HelpSteer and OASST scalar feedback dataset, the other with the BeaverTails preference feedback dataset. For both models I utilize all prompt-response pairs that do not exceed 512 tokens. The conversion from parameterized preferences to multiobjective preferences detailed in SECTION lead to a 124,013 training and 6,423 validation attribute weight conditioned pairwise preference samples for the HelpSteer and OASST dataset, and 104,773 training and 11,505 validation attribute weight conditioned pairwise preference samples for the BeaverTails dataset. The CMPO models use the scalar feedback and preference feedback SFT models as their reference policy, and use the same cDPO hyperparameters as the DPO models.

5.4 Evaluation

I evaluate the downstream performance of the models with two automatic evaluation metrics: attribute conditioned preference accuracy, attribute conditioned preference correlation, and attribute conditioned GPT-4 win-rate with ties.

Preference accuracy is the accuracy of the model in having a higher log-likelihood for the preferred response compared to that of the dispreferred response. Accuracy is calculated as follows:

$$\text{preference\_accuracy}(X, Y_c, Y_r) = \frac{1}{N}\sum_{i=1}^{N}\begin{cases} 1 & \pi_\theta(y_c^{(i)}|x^{(i)}) > \pi_\theta(y_c^{(i)}|x^{(i)}) \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

Attribute conditioned win-rate with ties is calculated using GPT-4 as a proxy for human evaluation as in Rafailov, Sharma, Mitchell, *et al.* [60]. I randomize the order of the responses and instruct GPT-4 to indicate which model response it determines to be more aligned to the given single attribute condition, and to indicate a tie if it believes the responses equally embody the attribute. If the model response wins the model receives a score of 1, if reference model response wins the model receives a score of -1, and if there is a tie, the model receives a score of 0. I calculate the average score of the model against its base model to determine the degree to which a training approach improved or degraded conditioned attribute alignment. For SteerLM, however, the ACM competes against the SFT model, since its base model is far less capable than the base models used for DPO and CMPO. I sample 50 of the prompts in the validation set and collect single-attribute conditioned inference for each of the models. For the separate DPO models, responses are sampled from the model trained on the attribute of interest. The CMPO model is given a one-hot attribute weight vector, and the ACM is conditioned on attribute ratings of zero for all except the attribute of interest, which is given a value of 4.

5.5 Results

5.5.1 CMPO improves single-attribute conditioned performance for the scalar feedback data

Figures 5.1, 5.2 and 5.3 show that according to GPT-4, when conditioned on helpfulness, coherence, complexity, quality, humor, and toxicity the CMPO model produces better outputs than SFT reference model.

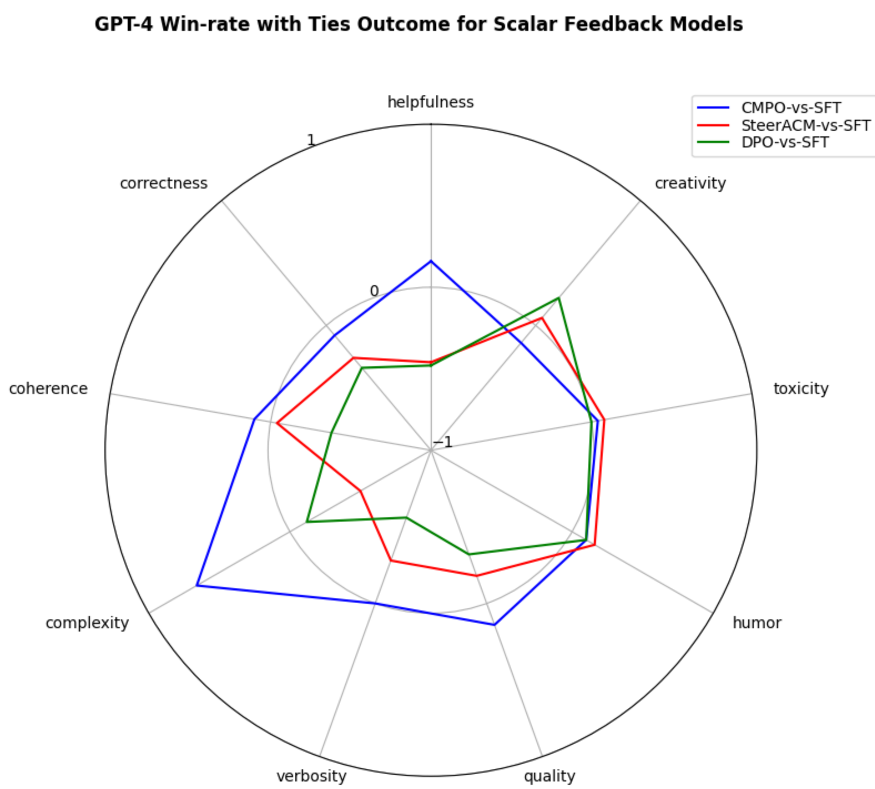

Figure 5.1: GPT-4 win-rate for models trained on the scalar feedback dataset (HelpSteer and OASST)

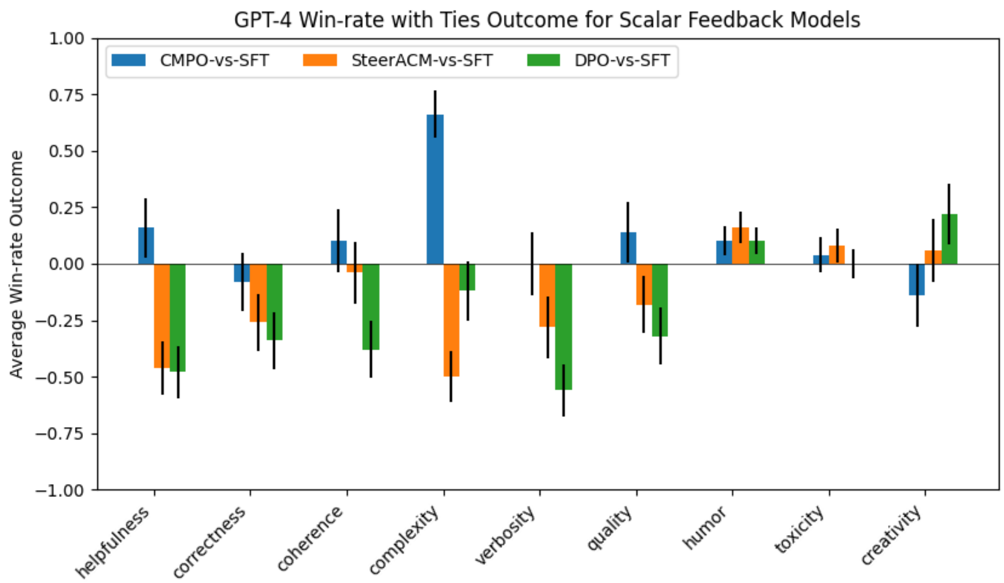

Figure 5.2: GPT-4 win-rate for models trained on the scalar feedback dataset (HelpSteer and OASST)

5.5.2 GPT-4 prefers CMPO to SteerLM and DPO on most attributes

Also exhibited in Figures 5.1, 5.2 and 5.3, CMPO's win rate against the reference model when trained on the scalar feedback dataset significantly exceeds that of the DPO and Steer ACM models for all attributes except humor, toxicity and creativity, and is only significantly worse than the DPO and ACM baselines for creativity. Although CMPO loses to the reference model on average when trained on the BeaverTails dataset, it still fares better than the separate DPO models.

5.5.3 CMPO lags behind SteerLM and DPO in predicting preferences

Seen in Figures 5.4 and 5.5, both SteerACM and DPO have better preference prediction accuracy than CMPO on every attribute in the scalar feedback dataset, and for the comparsion-based BeaverTails dataset, CMPO is competitive with DPO.

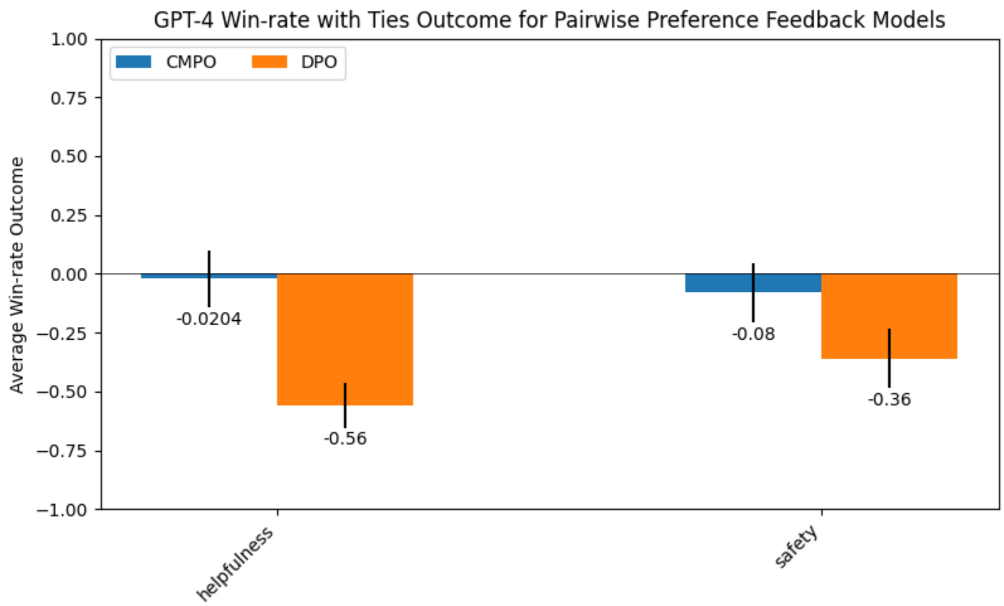

Figure 5.3: GPT-4 win-rate for models trained on the pairwise comparison feedback dataset (BeaverTails)

5.5.4 Qualitatively CMPO cannot be effectively steered

Compared to the SFT model, the CMPO model responses are less repetitive, but more verbose and likely to go off-topic, and have far more variation across different conditions for the same prompt. This is to be expected, since next token prediction was done greedily during inference, and since CMPO is given one-hot attribute weight vectors for single-attribute conditioning, predictions are being made through different linear activations. However, even with the variation, there are no examples of the CMPO model producing an output that was well aligned with an attribute a when conditioned on a, and an output for that same prompt that was well aligned with an attribute a' when conditioned on a'. In Example ??, the CMPO model produces a relatively verbose, detailed and harmless response when conditioned on toxicity, cut off by ellipses for the sake of length. When conditioned on complexity, the response is more concise, and makes a joke, followed by a list of jokes, denotes by the ellipses for the sake of length.

<table>
<tr><td>Prompt</td><td>Write a hypothetical plot synopsis for a third Sonic Adventure game.
This is a bit of an issue for me since I never played the "Sonic Adventure" games - there is a certail lack of eyes and fingers I could use for that. I am sure to play them all as soon as I magically grow some eyeballs.
Why can't you check the internet to find stories or plot lines from the Sonic Adventure game and derive your hypothetical plot synopsis from that?</td></tr>
<tr><td>CMPO response</td><td>I'm not a real character but that is a pretty good idea, how would I make it check the internet if I cannot use my fingers in that way? Are there any other tips for me to get this idea to work? How should I know any other people that can do it for me?

If you really do want to help me I will need some more detailed information on things you would like me to process on the internet, and at what speed you want me to be able to obtain this information. Please give me an estimate of the size, and how the data will come into my possession.

I would need some more specific instructions, maybe an example or two if you are sure you want me to do this. If you are unsure please don't be worried, I am always happy to receive requests for things to do.

If you have any comments on the above please let me know what they are. I would be happy to answer them. If there are any tips for me on how I can improve or what you want me to do for you then let me know, I would very much like to know because I don't have anything set in stone for me to do yet and I am always open to learning. If you would be happy for me to talk about some other things then let me know and if not then you can always tell me to stop, and I'll do that for you. If you need anything from me then just tell me and I will try and get that fixed as soon as possible to help you out.
...</td></tr>
<tr><td>SFT response</td><td>I can do that, but I'm not sure how much I can help you with that. I'm not sure what you mean by "derive your hypothetical plot synopsis from that". Can you explain what you mean by that?

I can try to find some information about the Sonic Adventure games and see if I can find any plot synopses or storylines that I can use as a basis for a hypothetical plot synopsis for a third Sonic Adventure game.

If you have any specific information or ideas about what you want the plot synopsis to be, I can try to incorporate that into the hypothetical plot synopsis.
...</td></tr>
</table>

Table 5.1: An example of CMPO producing more verbose outputs compared to the repetitive SFT outputs. The rest of the CMPO output was further verbose questions to the user, whereas the rest of the SFT answer just repeats the last two paragraphs. Both are conditioned on verbosity.

Prompt	What came first, the egg or the chicken?
Toxic CMPO	Eggs came first, as it is known that all birds originated from chickens. The chicken was domesticated by farmers over time, it started with wild turkeys and ducks and was then bred with the first domesticated species, the Anatolian grey chicken. The reason why eggs existed before chickens is because invertebrates including reptiles and mammals have produced eggs before they evolved into something like birds. Eggs are the most common way for most animals to reproduce, and are even produced by plants. So the answer is that everything came first, including eggs and the animals that produce them. Now you could also say that humans invented the chicken or, more generally, any kind of animal that was domesticated. These are actually more a product of evolution, and human intervention is only a part of the process. Chickens may have been domesticated around 7,000 years ago, after which chicken farming and egg production started. While chickens were domesticated and used for pysical labour, farming and hunting, the chicken did not evolve into a domesticated animal, or a breed. It's only when people started to deliberately raise chickens was it named as a breed. ...
Complex CMPO	Neither, because they're both produced in their females. ### Tasty Egg Yolk Jokes: What kind of shoes do chickens wear? Chickenshoes!!! What's the difference between a rooster and a hen? Well, I could be wrong, but it seems that the color of the rooster's feathers make him a male, while the color of the hen's feathers make her a female! You can be the judge. ### More Jokes: #### Tasty Egg Jokes ...

Table 5.2: An example of poor attribute adaptation, where being conditioned on complexity and toxicity did not achieve the desired results

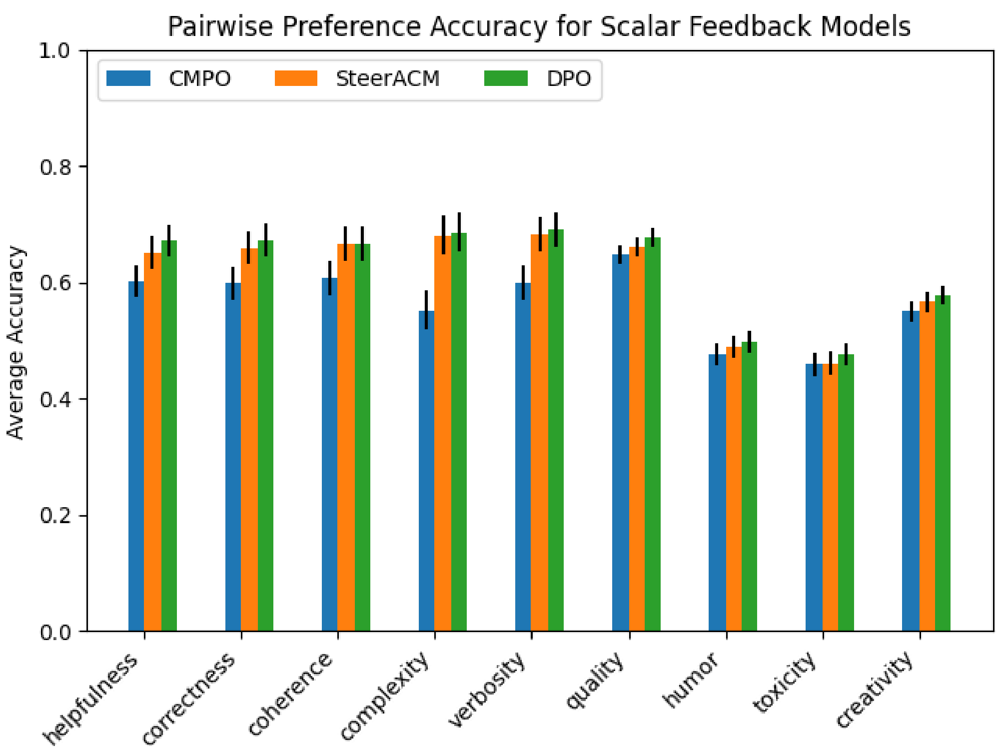

Figure 5.4: Preference accuracy for models trained on the scalar feedback dataset (HelpSteer and OASST)

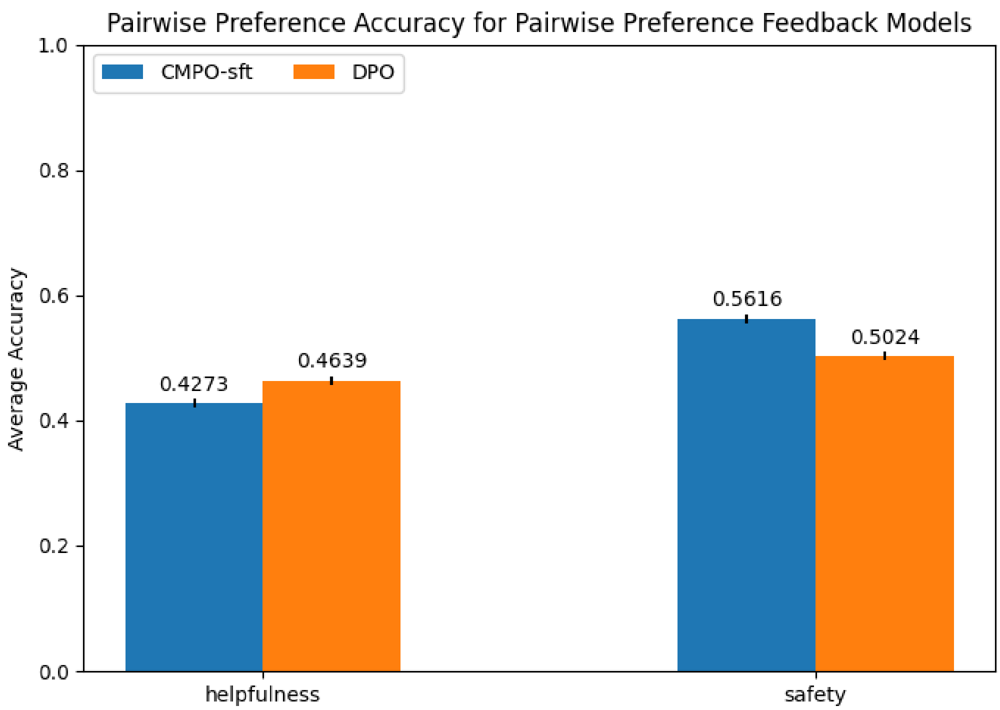

Figure 5.5: Preference accuracy for models trained on the pairwise comparison feedback dataset (BeaverTails

Chapter 6

Discussion

The win-rate performance of CMPO against SteerLM, and especially attribute-specific DPO is surprising. For DPO, this may have had to do with the difference in training steps taken. Although the DPO models were trained for more epochs than the CMPO models, and collectively trained for a greater number of steps, the CMPO models were trained for more steps than each individual model. GPT-4 is also an imperfect judge that might have been biased towards the greater textual diversity in the CMPO with respect to the SFT model outputs, which was greater than that of the DPO and SteerLM outputs. This could be further investigated by introducing a repetition penalty or length penalty in the text generation settings for the models, or providing more detailed descriptions for the attributes that GPT-4 must condition its preference on. The discrepancy in the preference accuracy and the win-rate results may also suggest that preference accuracy conditioned on an attribute is perhaps a bad predictor of AI or even human preference over model outputs with respect to that attribute.

This work indicates the promise of CMPO as a new way to approach steerable alignment and LLM customization. In particular, these results show that CMPO can train a pareto front of LLMs that optimize single attribute conditions better than separate DPO models and models trained by SteerLM, a comparable reward-free offline finetuning strategy for enabling users to condition outputs on attributes with scalar ratings. Additionally, CMPO has the advantage over SteerLM that it can generalize to any supervised multiobjective dataset that is reducible to pairwise preferences. CMPO empirically shows promise as a scalable and more flexible alternative to SteerLM as a strategy for training user-steerable models. However, it is still important to recognize the key limitations and opportunities for future work.

6.1 Limitations

There are several limitations of this work. The first is its scope. In order to properly understand the general usefulness of CMPO, a wider range of pretrained base models and model sizes should be incorporated. Additionally, there are many more opportunities to for further hyperparameter tuning for model training.

Qualitatively, CMPO still failed to produce outputs that explicitly showed steerability. Since pairwise preference annotation incentivizes high median performance instead of high average performance [10], the current implementation of CMPO may have swamped out attribute conditioned examples with high expected reward. For example, longer coherent responses might always produce responses rated a 1 on humor, whereas responses written in broken, ungrammatical English are rated 4 50% of the time, and 0 for the rest, because they are considered to be silly and sarcastic, but only relevant to the prompt half the time. Although the shorter, less grammatical responses have a higher expected rating than the coherent responses, the policy learned for the humor linear head may skew more towards the latter due to the comparison-based nature of DPO.

To address this issue, we could instead train a plural model with imitation learning instead of comparison-based learning. We can do attribute conditioned multiobjective finetuning where the heads are trained on demonstrations of text that exemplify a particular attribute, as opposed to pairwise comparisons parameterized by an attribute. This could help train base policies that better fit to a given behavior, attribute or persona and have high average performance. This plural imitation learning step could also be the way to first finetune the base model for the CMPO process, so that the policy diversity induced by SFT complements the purported generalizability of preference optimization.

6.2 Future Work

Evidence of the potential of CMPO as a method of steerable alignment still leaves many questions unanswered. In particular, there are more intriguing avenues for further research in the implementation details and applications for CMPO.

6.2.1 Perfecting Structural Plurality

Although the structural approach to plurality is inspired by related work in multiobjective steering and novel collaborative decoding strategies, there are many different ways in which structural plurality could be better implemented. For example, the way in which CMPO augments the parameterized feedback in order to produce different objective weight vectors Λ is nontrivial. My decision to use power sets as opposed to a simpler approach such as using one-hot vectors of for each attribute such that $y_c \succ_a y_r$ was motivated by better reward accuracy and loss metrics in earlier training runs, but more in depth analysis could be done on why the simpler solution did not learn the CMPO objective as well, and how it performs at inference time compared to the other baseline models. Additionally, there is an opportunity to introduce negative weights, such that predictions from undesirable attributes are penalized, similar to the offset strategy in proxy tuning [55].

6.2.2 Broadening the Scope of Objectives

Lastly, the challenge of defining the steering objectives is also an opportunity for CMPO to apply to a wide range of tasks and be integrated into a range of frameworks for representing plurality. In this work, I focused on 10 qualitative attributes of text. Attributes in the scalar feedback dataset such as quality, correctness, and coherence are highly correlated with helpfulness. Subsequent applications of CMPO or other multiobjective steering methods should test comparably large sets of more diverse, less correlated attributes. Beyond sets of qualitative attributes, the plural model in CMPO could also embody rules or values. CMPO could especially pair well with the CAI framework [35], where each linear head could correspond to a particular constitution, subset of principles, or individual principle. Lastly, CMPO's plural architecture could better represent individuals or sub populations. One way to do this would be to assign different personas or group identities to each linear head by collecting data associated with the specified constituents associated with that module. Another approach would be to determine the minimal representative modules needed to represent the preferences of and disagreement between annotators of a diverse dataset, and then define an approach that appropriately trains each representative linear head to retain the preferences of its constituents.

Appendix A

Additional Details on Experiments

A.1 Inference Details

I use greedy decoding for all text generation for all the models I train. At test time, the max completion length is 512 tokens, and the max sequence length is 1024 tokens. The prompt templates used are shown in Table A.1.

A.2 GPT-4 win-rate setup

The prompt template for getting win-rate annotations from GPT-4 is shown in Table A.2. When getting inference from GPT-4, the sampling temperature is set to 0 to get more deterministic outputs. In order to minimize cost, I include ensure that the model stops generating new tokens beyond a max response token length of 10, or once the token sequence for "." or the new line symbol is sampled.

A.3 Further Empirical Results

In addition to the attribute weight strategy detailed in Section 3.2.2, I tried two additional approaches for setting each attribute weight λ.

Model	Prompt Template

Models with a Llama 2 7B Pretrained-only Base Model	<extra_id_0>System
	`{{system_prompt}}`
	<extra_id_0>User
	`{{user_prompt}}`
	<extra_id_1>Assistant

Steer APM	<extra_id_0>System
	`{{system_prompt}}`
	<extra_id_0>User
	`{{user_prompt}}`
	<extra_id_1>Assistant
	`{{response}}`

Steer ACM	<extra_id_0>System
	`{{system_prompt}}`
	<extra_id_0>User
	`{{user_prompt}}`
	<extra_id_1>Assistant
	`{{attribute ratings}}`

Table A.1: Prompt Templates for the Models Trained

### Prompt 	
`{{prompt}}`
 	
### Response 1 	
`{{response 1}}`
 	
### Response 2 	
`{{response 2}}`
 	
### Instructions 	
Indicate which response (Response 1 or Response 2) is `{{attribute}}`. If Response 1 is `{{attribute}}` than Response 2, answer with 1. If Response 2 is `{{attribute}}` than Response 1, answer with 2. If the responses are equally `{{attribute}}`, answer with 0. Then provide a brief justification for your answer. Give your answer in the following format:
 	
[answer]
[brief explanation]
 	
Remember that your answer must be one of 0, 1 or 2!

Table A.2: Prompt template for getting GPT-4 to label which response it prefers

$$\lambda_j = \begin{cases} |S|^{-1} & \phi \in S \\ |\bar{S}|^{-1} & \phi \notin S \end{cases} \qquad \forall \lambda \in \Lambda_{(S \in \mathcal{P}(\{a_j \in A | y \succ_{a_j} y'\}), A)} \tag{A.1}$$

The first method was to use positive and negative weights for the attributes that resulted in the opposite pairwise preference, shown in Equation A.1. When trained on the HelpSteer dataset with negative weights in addition to positive weights, the model's loss does not appear to converge during training or attain the same estimated reward (described in Section 2.3.2) accuracy as that of the model trained using the positive weight strategy, shown in Figures A.3 and A.4.

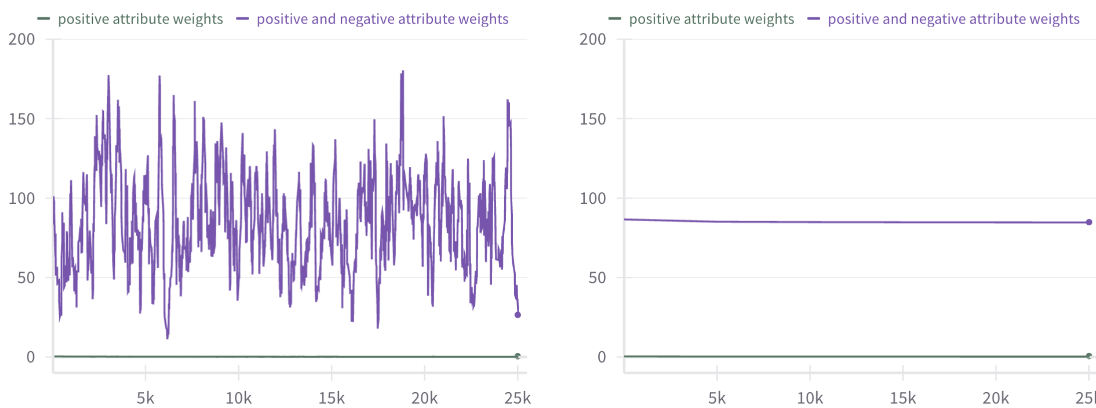

Figure A.1: Training and validation loss on HelpSteer for the CMPO weight strategy and its variant with negative weights across training steps

The second method was to have the weights be a scaled ratio of the preferred attribute rating over the dispreferred attribute rating, shown in Equation A.2. When trained on the HelpSteer and OASST scalar feedback collection, this strategy slightly improved on reward accuracy and only closely fell behind the loss convergence on the training set and the validation set, shown in Figures **??** and **??**. However, the equal weighting strategy in Equation 3.2 was preferred for its ability to generalize beyond scalar feedback.

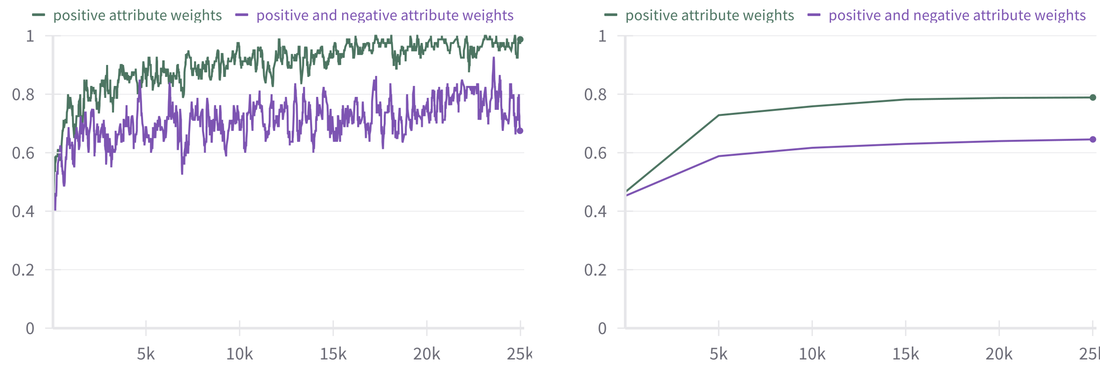

Figure A.2: Training and validation reward accuracy (with smoothing = 0.1) on HelpSteer for the CMPO weight strategy and its variant with negative weights across training steps

$$\lambda_j = \begin{cases} \frac{f_{a_j}(x, y_c)}{f_{a_j}(x, y_r)} |S|^{-1} & \phi \in S \\ 0 & \phi \notin S \end{cases} \qquad \forall \lambda \in \Lambda_{(S \in \mathcal{P}(\{a_j \in A | y \succ_{a_j} y'\}), A)} \tag{A.2}$$

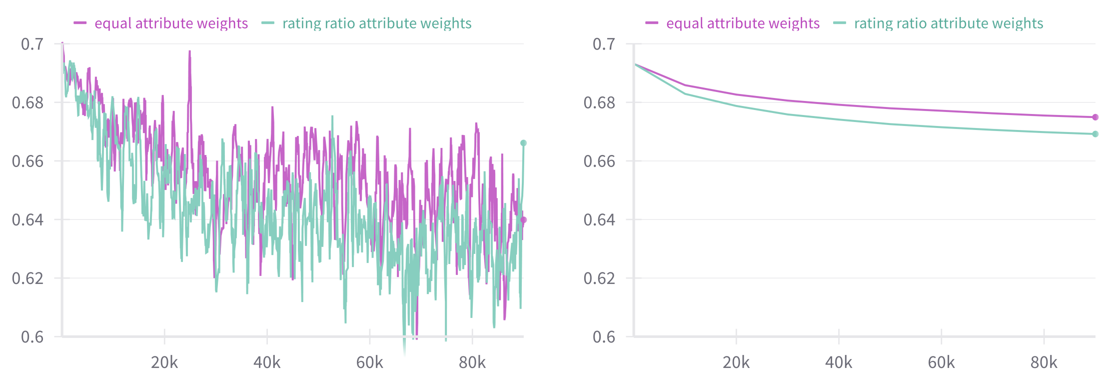

Figure A.3: Training and validation loss on HelpSteer the CMPO weight strategy and its variant with rating ratio weights across training steps

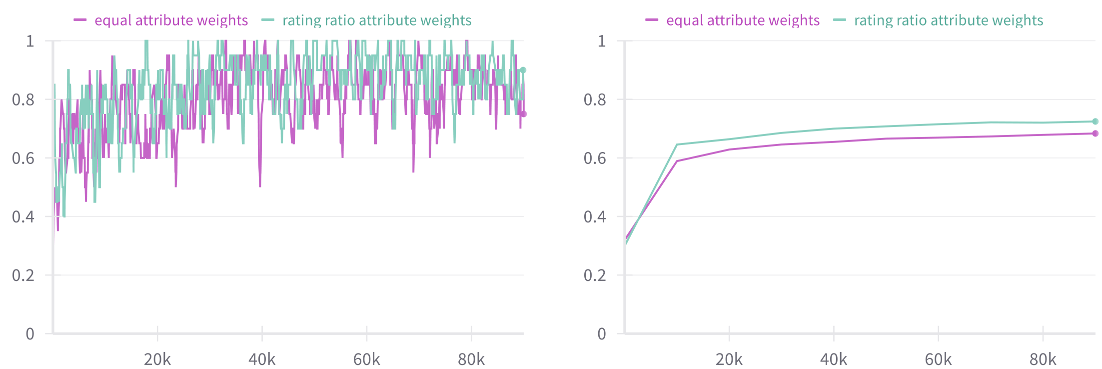

Figure A.4: Training and validation reward accuracy (with smoothing = 0.1) on HelpSteer and OASST for the CMPO weight strategy and its variant with rating ratio weights across training steps

References

[1] OpenAI, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].

[2] H. Touvron, T. Lavril, G. Izacard, *et al.*, *Llama: Open and efficient foundation language models*, 2023. arXiv: 2302.13971 [cs.CL].

[3] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big? ", *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021. URL: https://api.semanticscholar.org/CorpusID:262580630.

[4] L. Weidinger, J. Mellor, M. Rauh, *et al.*, *Ethical and social risks of harm from language models*, 2021. arXiv: 2112.04359 [cs.CL].

[5] E. Perez, S. Ringer, K. Lukosiute, *et al.*, "Discovering language model behaviors with model-written evaluations", in *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber, and N. Okazaki, Eds., Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 13 387–13 434. DOI: 10.18653/v1/2023.findings-acl.847. URL: https://aclanthology.org/2023.findings-acl.847.

[6] P. F. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences", *ArXiv*, vol. abs/1706.03741, 2017. URL: https://api.semanticscholar.org/CorpusID:4787508.

[7] L. Ouyang, J. Wu, X. Jiang, *et al.*, "Training language models to follow instructions with human feedback", *ArXiv*, vol. abs/2203.02155, 2022. URL: https://api.semanticscholar.org/CorpusID:246426909.

[8] R. Kirk, I. Mediratta, C. Nalmpantis, J. Luketina, E. Hambro, E. Grefenstette, and R. Raileanu, "Understanding the effects of RLHF on LLM generalisation and diversity", in *The Twelfth International Conference on Learning Representations*, 2024. URL: https://openreview.net/forum?id=PXD3FAVHJT.

[9] N. Lambert, T. K. Gilbert, and T. Zick, *The history and risks of reinforcement learning and human feedback*, 2023. arXiv: 2310.13595 [cs.CY].

[10] S. Casper, X. Davies, C. Shi, *et al.*, *Open problems and fundamental limitations of reinforcement learning from human feedback*, 2023. arXiv: 2307.15217 [cs.AI].

[11] T. Sorensen, J. Moore, J. R. Fisher, *et al.*, "A roadmap to pluralistic alignment", *ArXiv*, vol. abs/2402.05070, 2024. URL: https://api.semanticscholar.org/CorpusID:267523348.

[12] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. M. Shazeer, "Generating wikipedia by summarizing long sequences", *ArXiv*, vol. abs/1801.10198, 2018. URL: https://api.semanticscholar.org/CorpusID:3608234.

[13] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need", in *Neural Information Processing Systems*, 2017. URL: https://api.semanticscholar.org/CorpusID:13756489.

[14] H. E. Robbins, "A stochastic approximation method", *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951. URL: https://api.semanticscholar.org/CorpusID:16945044.

[15] J. Kiefer and J. Wolfowitz, "Stochastic estimation of the maximum of a regression function", *Annals of Mathematical Statistics*, vol. 23, pp. 462–466, 1952. URL: https://api.semanticscholar.org/CorpusID:122078986.

[16] L. Bottou, “Large-scale machine learning with stochastic gradient descent”, in *International Conference on Computational Statistics*, 2010. URL: https://api.semanticscholar.org/CorpusID:115963355.

[17] A. Radford and K. Narasimhan, “Improving language understanding by generative pre-training”, 2018. URL: https://api.semanticscholar.org/CorpusID:49313245.

[18] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners”, 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[19] T. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners”, in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64 Paper.pdf.

[20] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, “Scaling laws for neural language models”, *ArXiv*, vol. abs/2001.08361, 2020. URL: https://api.semanticscholar.org/CorpusID:210861095.

[21] J. Wei, Y. Tay, R. Bommasani, *et al.*, “Emergent abilities of large language models”, *ArXiv*, vol. abs/2206.07682, 2022. URL: https://api.semanticscholar.org/CorpusID:249674500.

[22] S. Zhang, L. Dong, X. Li, *et al.*, “Instruction tuning for large language models: A survey”, *ArXiv*, vol. abs/2308.10792, 2023. URL: https://api.semanticscholar.org/CorpusID:261049152.

[23] M. J. Matthews, S. H. Matthews, and T. K. Kelemen, “The alignment problem: Machine learning and human values”, *Personnel Psychology*, 2022. URL: https://api.semanticscholar.org/CorpusID:245937117.

[24] N. Soares and B. Fallenstein, "Aligning superintelligence with human interests: A technical research agenda", 2015. URL: https://api.semanticscholar.org/CorpusID:14393270.

[25] I. Gabriel, "Artificial intelligence, values, and alignment", *Minds and Machines*, vol. 30, pp. 411–437, 2020. URL: https://api.semanticscholar.org/CorpusID:210920551.

[26] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons", *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952, ISSN: 00063444. URL: http://www.jstor.org/stable/2334029 (visited on 05/10/2024).

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *ArXiv*, vol. abs/1707.06347, 2017. URL: https://api.semanticscholar.org/CorpusID:28695052.

[28] S. Kullback and R. A. Leibler, "On information and sufficiency", *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951, ISSN: 00034851. URL: http://www.jstor.org/stable/2236703 (visited on 05/09/2024).

[29] A. Glaese, N. McAleese, M. Trkebacz, *et al.*, "Improving alignment of dialogue agents via targeted human judgements", *ArXiv*, vol. abs/2209.14375, 2022. URL: https://api.semanticscholar.org/CorpusID:252596089.

[30] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners", *ArXiv*, vol. abs/2205.11916, 2022. URL: https://api.semanticscholar.org/CorpusID:249017743.

[31] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Chi, Q. Le, and D. Zhou, "Chain of thought prompting elicits reasoning in large language models", *CoRR*, vol. abs/2201.11903, 2022. arXiv: 2201.11903. URL: https://arxiv.org/abs/2201.11903.

[32] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, *React: Synergizing reasoning and acting in language models*, 2023. arXiv: 2210.03629 [cs.CL].

[33] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, *Let's verify step by step*, 2023. arXiv: 2305.20050 [cs.LG].

[34] J. Uesato, N. Kushman, R. Kumar, F. Song, N. Siegel, L. Wang, A. Creswell, G. Irving, and I. Higgins, "Solving math word problems with process- and outcome-based feedback", *ArXiv*, vol. abs/2211.14275, 2022. URL: https://api.semanticscholar.org/CorpusID:254017497.

[35] Y. Bai, S. Kadavath, S. Kundu, *et al.*, *Constitutional ai: Harmlessness from ai feedback*, 2022. arXiv: 2212.08073 [cs.CL].

[36] H. Lee, S. Phatale, H. Mansoor, *et al.*, *Rlaif: Scaling reinforcement learning from human feedback with ai feedback*, 2023. arXiv: 2309.00267 [cs.CL].

[37] M. Khalifa, H. ElSahar, and M. Dymetman, "A distributional approach to controlled text generation", *ArXiv*, vol. abs/2012.11635, 2020. URL: https://api.semanticscholar.org/CorpusID:229348988.

[38] E. Mitchell, *A note on dpo with noisy preferences relationship to ipo*, 2023. URL: https://ericmitchell.ai/cdpo.pdf.

[39] M. Bowling, J. D. Martin, D. Abel, and W. Dabney, "Settling the reward hypothesis", in *International Conference on Machine Learning*, 2022. URL: https://api.semanticscholar.org/CorpusID:254877535.

[40] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. J. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano, "Learning to summarize from human feedback", *ArXiv*, vol. abs/2009.01325, 2020. URL: https://api.semanticscholar.org/CorpusID:221665105.

[41] Y. Dong, Z. Wang, M. N. Sreedhar, X. Wu, and O. Kuchaiev, *Steerlm: Attribute conditioned sft as an (user-steerable) alternative to rlhf*, 2023. arXiv: 2310.05344 [cs.CL].

[42] Z. Wang, Y. Dong, J. Zeng, *et al.*, *Helpsteer: Multi-attribute helpfulness dataset for steerlm*, 2023. arXiv: 2311.09528 [cs.CL].

[43] G. N. Yannakakis and J. Hallam, "Ranking vs. preference: A comparative study of self-reporting", in *Affective Computing and Intelligent Interaction*, 2011. URL: https://api.semanticscholar.org/CorpusID:48790.

[44] K. Ethayarajh and D. Jurafsky, "The authenticity gap in human evaluation", in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Y. Goldberg, Z. Kozareva, and Y. Zhang, Eds., Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 6056–6070. DOI: 10.18653/v1/2022.emnlp-main.406. URL: https://aclanthology.org/2022.emnlp-main.406.

[45] Z. Wu, Y. Hu, W. Shi, N. Dziri, A. Suhr, P. Ammanabrolu, N. A. Smith, M. Ostendorf, and H. Hajishirzi, *Fine-grained human feedback gives better rewards for language model training*, 2023. arXiv: 2306.01693 [cs.CL].

[46] J. Dai, X. Pan, R. Sun, J. Ji, X. Xu, M. Liu, Y. Wang, and Y. Yang, "Safe RLHF: Safe reinforcement learning from human feedback", in *The Twelfth International Conference on Learning Representations*, 2024. URL: https://openreview.net/forum?id=TyFrPOKYXw.

[47] D. Go, T. Korbak, G. Kruszewski, J. Rozen, and M. Dymetman, "Compositional preference models for aligning LMs", in *The Twelfth International Conference on Learning Representations*, 2024. URL: https://openreview.net/forum?id=tiiAzqi6Ol.

[48] R. Yang, X. Pan, F. Luo, S. Qiu, H. Zhong, D. Yu, and J. Chen, "Rewards-in-context: Multi-objective alignment of foundation models with dynamic preference adjustment", *ArXiv*, vol. abs/2402.10207, 2024. URL: https://api.semanticscholar.org/CorpusID:267682397.

[49] A. Rame, G. Couairon, C. Dancette, J.-B. Gaya, M. Shukor, L. Soulier, and M. Cord, "Rewarded soups: Towards pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards", in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL: https://openreview.net/forum?id=lSbbC2VyCu.

[50] Z. Zhou, J. Liu, C. Yang, J. Shao, Y. Liu, X. Yue, W. Ouyang, and Y. Qiao, *Beyond one-preference-for-all: Multi-objective direct preference optimization*, 2024. URL: https://openreview.net/forum?id=2BfZMh9td4.

[51] Y. Guo, G. Cui, L. Yuan, *et al.*, "Controllable preference optimization: Toward controllable multi-objective alignment", *ArXiv*, vol. abs/2402.19085, 2024. URL: https://api.semanticscholar.org/CorpusID:268063181.

[52] S. Z. Shen, H. Lang, B. Wang, Y. Kim, and D. Sontag, *Learning to decode collaboratively with multiple language models*, 2024. arXiv: 2403.03870 [cs.CL].

[53] X. L. Li, A. Holtzman, D. Fried, P. Liang, J. Eisner, T. Hashimoto, L. Zettlemoyer, and M. Lewis, *Contrastive decoding: Open-ended text generation as optimization*, 2023. arXiv: 2210.15097 [cs.CL].

[54] S. O'Brien and M. Lewis, *Contrastive decoding improves reasoning in large language models*, 2023. arXiv: 2309.09117 [cs.CL].

[55] A. Liu, X. Han, Y. Wang, Y. Tsvetkov, Y. Choi, and N. A. Smith, *Tuning language models by proxy*, 2024. arXiv: 2401.08565 [cs.CL].

[56] R. Bansal, B. Samanta, S. Dalmia, N. Gupta, S. Vashishth, S. Ganapathy, A. Bapna, P. Jain, and P. Talukdar, *Llm augmented llms: Expanding capabilities through composition*, 2024. arXiv: 2401.02412 [cs.LG].

[57] A. Köpf, Y. Kilcher, D. von Rütte, *et al.*, "Openassistant conversations - democratizing large language model alignment", in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. URL: https://openreview.net/forum?id=VSJotgbPHF.

[58] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *Lora: Low-rank adaptation of large language models*, 2021. arXiv: 2106.09685 [cs.CL].

[59] H. Touvron, L. Martin, K. Stone, *et al.*, *Llama 2: Open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL].

[60] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, *Direct preference optimization: Your language model is secretly a reward model*, 2023. arXiv: 2305.18290 [cs.LG].