Bone Soups: A Seek-and-Soup Model Merging Approach for Controllable Multi-Objective Generation

Guofu Xie[1], Xiao Zhang[1]*, Ting Yao[2], Yunsheng Shi[2]
[1]Gaoling School of Artificial Intelligence,
Renmin University of China, Beijing, China
[2]Tencent
{guofuxie, zhangx89}@ruc.edu.cn, {tessieyao, yunshengshi}@tencent.com

Abstract

User information needs are often highly diverse and varied. A key challenge in current research is how to achieve controllable multi-objective generation while enabling rapid adaptation to accommodate diverse user demands during test time. Existing solutions, such as Rewarded Soup, focus on merging language models individually tuned on single objectives. While easy to implement and widely used, these approaches face limitations in achieving optimal performance due to their disregard for the impacts of competing objectives on model tuning. To address this issue, we propose **Bone Soup**, a novel model merging approach that first seeks a series of back**bone** models by considering the impacts of multiple objectives and then makes the **soup** (i.e., merge the backbone models). Specifically, Bone Soup begins by training multiple backbone models for different objectives using multi-objective reinforcement learning. Each backbone model is guided by a combination of backbone reward signals. To ensure that these models are optimal for the Pareto front, the backbone rewards are crafted by combining standard reward functions into basis vectors, which can then be modified through a rule-based construction method. Bone Soup leverages a symmetric circulant matrix mapping to generate the merging coefficients, which are used to merge the backbone models according to user preferences. Extensive experimental results demonstrate that Bone Soup exhibits strong controllability and Pareto optimality in controllable multi-objective generation, providing a more effective and efficient approach to addressing diverse user needs at test time. Code is available at https://github.com/andyclsr/BoneSoups.

1 Introduction

Human preferences and their information needs are highly diverse, and even for the same task, users may have distinct personalized demands in different scenarios (Wu et al., 2024; Rame et al., 2023; Wang et al., 2024a; Shi et al., 2024; Chen et al., 2024). This diversity introduces a significant controllability challenge for AI service providers (Shen et al., 2024a; Chen et al., 2023; Shen et al., 2024b), who must develop learning models that can effectively adapt to a wide range of user preferences. A key area of research addressing this challenge is *controllable multi-objective generation* (CMOG), which focuses on guiding the behavior of language models (LMs) to meet diverse and real-time user requirements without the need for retraining (Zhang et al., 2023; Rame et al., 2023; Shi et al., 2024; Wang et al., 2024a). For example, Bing Copilot offered users modes like "More Accurate", "More Balanced", and "More Creative", allowing for customization based on their *discrete* requirements.

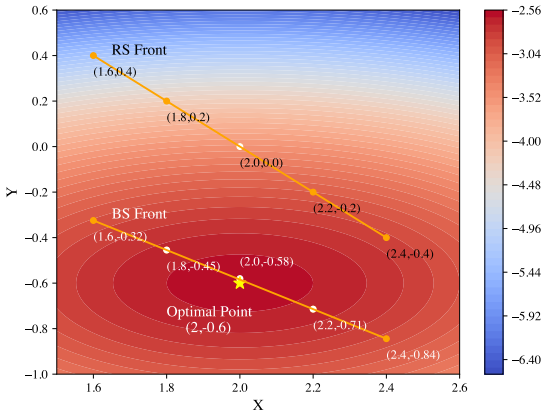


Figure 1: Illustration of Example 2.1. The RS front represents the front obtained by the existing soup-like approach (Yang et al., 2024). The BS front represents the front obtained by our Bone Soup scheme that seeks the backbone models first. The heatmap indicates the magnitude of the testing reward as a function of two inputs x and y. As shown in the figure, the points on the BS front are closer to the exact solution, highlighting the importance of constructing backbone models.

*Corresponding author: Xiao Zhang.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 27237–27263
July 27 - August 1, 2025

A straightforward approach to implementing CMOG is through prompt-based control (Dong et al., 2023; Ramnath et al., 2023; Yang et al., 2024; Wang et al., 2024a), where LMs are guided to generate content according to user preferences for different objectives by modifying only the input prompts. These approaches can be seen as an *implicit* control mechanism since it does not modify the model parameters at test time. Recently, some *explicit* approaches of controlling LMs have gained attention, known as model merging (Wortsman et al., 2022; Rame et al., 2023; Tang et al., 2024; Yu et al., 2024; Yadav et al., 2024; Yang et al., 2023; Wang et al., 2024b; Ilharco et al., 2022). In model merging approaches, model parameters from different LMs are combined at test time to accommodate varying user preferences. This form of test-time adaptation often provides more reliable control for CMOG, as it achieves control at the parameter level.

However, the performance of model merging heavily depends on the selection of base LMs and the determination of merging coefficients. This introduces a new challenge: *how to effectively seek and merge base models based on users' preferences for multiple objectives?* We illustrate the existence of this challenge through an example, as shown in Figure 1. The figure presents two different trajectories, each interpolated from solutions optimized using distinct reward functions, accompanied by a heatmap that displays the testing rewards for user preferences. As shown, compared to solutions optimized with reward functions constructed by existing methods, *there are superior solutions optimized with alternative reward functions that enable the model trajectories to more closely approximate the optimal testing reward.*

To address the above challenge, we propose **Bone Soup**. Our proposed Bone Soup approach follows the model merging approaches seen in Rewarded Soup (Rame et al., 2023) and Model Soup (Wortsman et al., 2022). Overall, we first seek a series of backbone models, and then, based on the received user preferences at test time, we combine various backbones. Unlike Rewarded Soup, where models are tuned separately for each reward (with each reward corresponding to a specific objective) and then merged, our Bone Soup first identifies the optimal combination of rewards for different objectives. Then, these reward combinations are then used as supervision signals to train the backbone models. During inference, these backbones are adaptively merged based on given user preferences. *This process is akin to selecting the right ingredients (bones) before making the soup.* Moreover, we focus on the task of controllable multi-objective generation, where content is generated based on user-provided preference weights across different objectives at test time. In contrast, the Model Soup approach merges multiple models fine-tuned with different hyperparameters to improve model performances, while Rewarded Soup focuses on scenarios where the user's true preference (i.e., a single true label) is known, and explores how to represent it as a combination of rewards for different objectives (i.e., reward decomposition). We summarize our contributions as follows:

- We identify a key challenge in achieving controllable multi-objective generation through model merging, particularly when managing competing objectives, where existing approaches often fail to deliver optimal performance.
- We propose Bone Soup, a novel model merging approach. By introducing combined rewards to guide the construction of backbone models, we enhance the merging process and optimize generation performance across multiple objectives, particularly in terms of controllability and Pareto optimality.
- Extensive experiments show that Bone Soup outperforms existing approaches, offering superior controllability, Pareto-optimal performance, and better adaptability to changes in user preferences.

2 Problem Formulation and Analyses

This section formulates the problem of controllable multi-objective generation through model merging and analyzes the sub-optimality of existing model merging approaches.

2.1 Problem Formulation

Consider n objectives (e.g., factuality, relevance, completeness, etc.) that users care about, and each objective can be measured by a **reward function** r_i, $i \in \{1, 2, \ldots, n\}$. The **preference weights** for these n objectives can be represented as an n-dimensional vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]^\top \in \Delta_n$, where Δ_n denotes the n-dimensional probability simplex. The problem of **controllable multi-objective generation** (CMOG) aims to enable

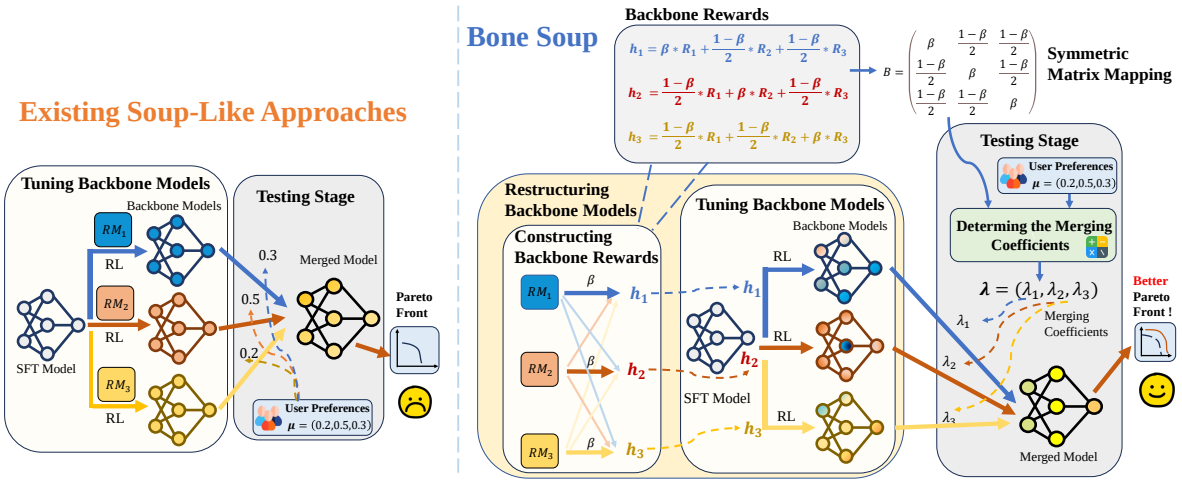


Figure 2: An overview of our method and a comparison between existing soup-like approaches and our Bone Soup method. Compared to existing methods, we incorporate the combined rewards and construct backbone rewards to guide the restructuring of backbone models. The merging coefficients are then determined based on the relationship between preference weights and the backbone rewards, improving the Pareto optimality and controllability of the merged model.

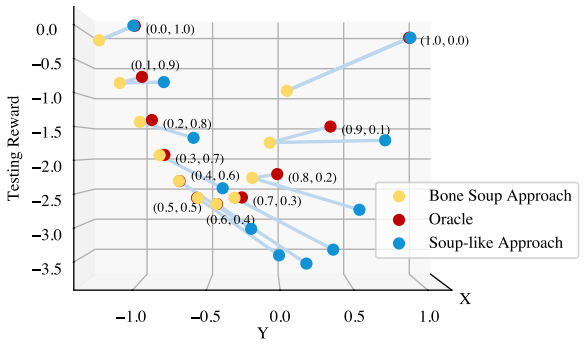


Figure 3: The solutions corresponding to the same preference across different methods are connected by blue lines. For each line, the closer the solution is to the red point (oracle), the better the result. Many of the yellow points in the middle are almost overlapping with the red point, indicating better solutions compared to the blue points further away. This highlights the importance of using backbone rewards to construct the backbone model.

language models (LMs) to dynamically adjust to changes in user-specified preference weights $\boldsymbol{\mu}$, allowing them to generate content that meets the user's requirements at test time.

To address the CMOG problem, the **model merging** approach first trains multiple base LMs using reward functions $\{r_i\}_{i=1}^n$, parameterized by $\boldsymbol{\theta}_i \in \Theta, i \in \{1, 2, \ldots, n\}$. Then, for satisfying user preferences, a merging strategy $\mathcal{M}$ is used to construct the model parameters for testing, as follows:

$$\mathcal{M}\left(\{\boldsymbol{\theta}_i\}_{i=1}^n\right) = \sum_{i=1}^{n} \lambda_i \boldsymbol{\theta}_i, \tag{1}$$

where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_n]^\top$ denotes the **merging coefficients**. Given an evaluation tool $\mathcal{H}$ for achieving optimal solutions, the base LMs $\{\boldsymbol{\theta}_i\}_{i=1}^n$ and their merging strategy aim to optimize the following expression:

$$\arg\max_{\{\boldsymbol{\theta}_i\}_{i=1}^n, \mathcal{M}} \mathcal{H}\left(\mathcal{M}\left(\{\boldsymbol{\theta}_i\}_{i=1}^n\right)\right). \tag{2}$$

In existing soup-like model merging approaches (Rame et al., 2023; Jang et al., 2023), for any objective $i \in \{1, 2, \ldots, n\}$, the base language model $\boldsymbol{\theta}_i$ is tuned with an individual reward function r_i for that specific objective, making it a **specialized model** $\boldsymbol{\theta}_i$ for objective i. When applying these model merging approaches to CMOG, the merging coefficients in Eq. (1) are directly set to the user's preference weights, i.e., $\boldsymbol{\lambda} = \boldsymbol{\mu}$, to combine the specialized models at test time. In Section 2.2, we will demonstrate that merging the specialized models tuned individually with each reward does not lead to an optimal solution.

Overall, this paper explores model merging approaches for the CMOG problem, where model parameters are interpolated and merged based on user preference weights to achieve the following

two goals: (1) *Pareto Optimality* across multiple objectives (2) *Controllability* that merged model parameters satisfy users' real-time needs.

To measure the two goals mentioned above, we define the evaluation tool $\mathcal{H}$ in Eq. (2) as the following *testing reward*: given users' preference weights $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]^\top$ and corresponding rewards $\{r_i\}_{i=1}^n$, for merged model parameters $\bar{\boldsymbol{\theta}} := \mathcal{M}(\{\boldsymbol{\theta}_i\}_{i=1}^n)$ defined in Eq. (1), the testing reward is defined as

$$g_{\boldsymbol{\mu}}(\bar{\boldsymbol{\theta}}) := \sum_{i=1}^{n} \mu_i r_i(\bar{\boldsymbol{\theta}}). \tag{3}$$

On one hand, maximizing Eq. (3) allows us to identify the convex Pareto front, reflecting the Pareto optimality of the merged model (Zitzler and Thiele, 1999). On the other hand, for any preference weights $\boldsymbol{\mu}$ provided at test time, the corresponding testing reward $g_{\boldsymbol{\mu}}$ is defined, and the merged model is required to adapt controllably to it.

2.2 Problem Analyses

As stated in Section 2.1, in existing soup-like model merging approaches, the specialized models for each objective are tuned *individually* with a *single* reward, without considering whether incorporating other rewards could improve their training.

Rame et al. (2023) demonstrates that a global optimal solution can be derived using a single reward in certain cases, such as with quadratic rewards. However, for the CMOG problem we address, we show that individually tuning specialized models with a single reward and merging them using preference weights does not consistently yield, or even approximate, the global optimal solution.

Example 1. *Consider two objectives, respectively measured by the following two rewards:*

$$r_1(x, y) = -(x-1)^2 - (y-1)^2 \quad \textit{and} \tag{4}$$
$$r_2(x, y) = -(x-3)^2 - 4(y+1)^2, \tag{5}$$

which are maximized at $\boldsymbol{\theta}_1 = (1, 1)^\top$ and $\boldsymbol{\theta}_2 = (3, -1)^\top$, respectively. Given preference weights $\boldsymbol{\mu} = (0.5, 0.5)^\top$ for the two rewards, the testing reward becomes $g_{\boldsymbol{\mu}}(x, y) := \boldsymbol{\mu}^\top [r_1, r_2] = 0.5 \cdot r_1(x, y) + 0.5 \cdot r_2(x, y)$, and the exact solution for maximizing $g_{\boldsymbol{\mu}}$ is $\boldsymbol{\theta}^* = (2, -0.6)^\top$. However, using a soup-like approach, where the preference weights $\boldsymbol{\mu}$ are used to merge the individual solutions $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$, the resulting solution is $\bar{\boldsymbol{\theta}} = 0.5 \cdot \boldsymbol{\theta}_1 + 0.5 \cdot \boldsymbol{\theta}_2 = (2, 0)^\top$, which significantly deviates from the exact solution $\boldsymbol{\theta}^*$. Now, instead of directly optimizing r_1 and r_2, we consider two backbone rewards that combine the rewards with different combination weights as follows:

$$h_1(x, y) = 0.4 \cdot r_1(x, y) + 0.6 \cdot r_2(x, y), \quad \textit{(prefer obj 2)}$$
$$h_2(x, y) = 0.6 \cdot r_1(x, y) + 0.4 \cdot r_2(x, y), \quad \textit{(prefer obj 1)}$$

with their respective optimal solutions, referred to as backbone models, *occurring at $\boldsymbol{\theta}_1^{\text{bone}} = (2.2, -5/7)^\top$ and at $\boldsymbol{\theta}_2^{\text{bone}} = (1.8, -5/11)^\top$. Then, the merging solution is given by $\bar{\boldsymbol{\theta}}^{\text{bone}} = 0.5 \cdot \boldsymbol{\theta}_1^{\text{bone}} + 0.5 \cdot \boldsymbol{\theta}_2^{\text{bone}} \approx (2, -0.58)^\top$, which is closer to the exact solution $\boldsymbol{\theta}^* = (2, -0.6)^\top$ than $\bar{\boldsymbol{\theta}} = (2, 0)^\top$.*

In Example 1, consider $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ as model parameters. If they are optimized solely for rewards r_1 and r_2 individually and then merged using preference weights $\boldsymbol{\mu}$, the result will not approximate the optimal solution $\boldsymbol{\theta}^*$ for the testing reward. However, if we first derive *backbone rewards* h_1 and h_2 by combining the rewards, and then train *backbone models* $\boldsymbol{\theta}_1^{\text{bone}}$ and $\boldsymbol{\theta}_2^{\text{bone}}$ on these backbone rewards, merging these backbone models with the preference weights can lead to a solution much closer to the optimal testing reward. Moreover, if the user's preference weights are given by $\boldsymbol{\mu}' = \{0.4, 0.6\}^\top$, then based on the relationship between $\boldsymbol{\mu}'$ and the combination weights in the backbone rewards h_1 and h_2, we can directly output $\boldsymbol{\theta}_1^{\text{bone}}$ as the solution, obtaining the optimal solution for maximizing the testing reward $g_{\boldsymbol{\mu}'}$.

We also provide a comparison of the disparity between solutions of different methods and the oracle in Figure 3.

Through the above example, we have demonstrated that BoneSoup can achieve solutions closer to the oracle. To further illustrate this point, we present the following theorem, which provides a lower bound on the interval where BoneSoup outperforms Rewarded Soup. This result proves that the front obtained by BoneSoup is, in most cases, superior to that of Rewarded Soup. We follow the setting in (Rame et al., 2023) using quadratic reward functions and with Hessians proportional to identity matrices to derive the theorem.

Theorem 1. *Given quadratic reward functions with Hessians proportional to identity matrices:*

$$r_i(\theta) = r_i(\theta_i) - k_i \|\theta - \theta_i\|^2, i \in \{1, 2\},$$

where $k_i \in \mathbb{R}_+$ are distinct,and θ_i is the global maximum for reward r_i. Let the reward combination weight matrix be $B = \begin{pmatrix} \beta & 1-\beta \\ 1-\beta & \beta \end{pmatrix}, \beta \in (\frac{1}{2}, 1)$, then the backbone rewards of the bone-soup approach can be denoted as $[h_1, h_2]^T = B[r_1, r_2]^T$.Let $\boldsymbol{\mu} = [\mu, 1-\mu]^T$ be the user preference and the testing reward can written as $g_{\boldsymbol{\mu}}(\theta) := \boldsymbol{\mu}^T \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$.Denote the approximate solutions for the testing reward $g_\mu(\theta)$ of the soup-like approach and the bone-soup approach as $\bar{\theta}$ and $\bar{\theta}^{bone}$, respectively.Then,for any fixed $\beta \in (\frac{1}{2}, 1)$, when $\mu \in \left(\frac{1-\sqrt{2\beta^2-2\beta+1}}{2}, \frac{1+\sqrt{2\beta^2-2\beta+1}}{2}\right)$,

$$g_\mu(\bar{\theta}) < g_\mu(\bar{\theta}^{bone}),$$

with interval length $\sqrt{2\beta^2-2\beta+1} \geqslant \frac{\sqrt{2}}{2}$.

Therefore, constructing appropriate backbone rewards to train the backbone models is crucial for achieving Pareto optimality and controllability in CMOG.

Proof. Please refer to Appendix A.2. □

3 Bone Soup: The Proposed Approach

In this section, we propose a novel approach to seek a series of superior backbone models, and then determine the merging coefficients for merging.

3.1 Approach Overview

We design and implement a more sophisticated merging-based approach Bone Soup for CMOG. Instead of directly interpolating between original base models, we propose to first *seek the backbone models* which ensures better Pareto optimality, and then *determine the merging coefficients* to contribute to better controllability. Figure 2 illustrates the overall workflow of our method.

3.2 Restructuring the Backbone Models

We begin by revisiting how specialized models $\boldsymbol{\theta}_i$ are obtained in existing works. Typically, these models are tuned through reinforcement learning from human feedback (RLHF) (Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022).

Existing soup-like model merging approaches (Jang et al., 2023; Rame et al., 2023) for CMOG individually tune the specialized models as above. *However, when considering multi-objective reinforcement learning from human feedback (MORLHF), the approach used by existing methods represents just one specific case.*

Here, we extend tuning the backbone model from using a single reward to multiple rewards by introducing MORLHF:

$$\boldsymbol{\theta}_i = \arg\max_{\pi_{\boldsymbol{\theta}}} \mathbb{E}_{s\sim\mathcal{D}, a\sim\pi_{\boldsymbol{\theta}}(a|s)} \left[\boldsymbol{w}_i^\top \boldsymbol{r} - \eta \log \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\text{sft}}(a|s)} \right] \text{where} \tag{6}$$

$\boldsymbol{w}_i \in \Omega$ is the *combination weight* of the reward models and $\Omega = \{\boldsymbol{a} \in \mathbb{R}^n | \sum_{i=1}^n a_i = 1, a_i \geqslant 0\}$ is a n-simplex. $\boldsymbol{r}$ is a collection of all n optimized reward models $\boldsymbol{r} = \{r_i(s,a)\}_{i=1}^n$. Then we define the *backbone reward* as $h_i(s,a) = \boldsymbol{w}_i^\top \boldsymbol{r} = \sum_{j=1}^n w_{i,j} \cdot r_j(s,a)$. *In this case, the single-reward setup in existing works is equivalent to setting $\boldsymbol{w}$ as a standard basis vector.*

3.2.1 Obtaining Backbone Models

Let n denote the number of objectives to optimize, $\boldsymbol{B} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n] \in \mathbb{R}^{n\times n}$ denote a weight matrix composed of n column vectors, with each column vector corresponding to the reward combination weight for tuning a new backbone model $\pi_{\boldsymbol{\theta}_i}$ in MORL as in Eq. (6). Then, the combination weights $\{\boldsymbol{w}_i\}_{i=1}^n$ can be viewed as the basis vectors in the column space of $\boldsymbol{B}$, where $\boldsymbol{w}_i = \{w_{i,j}\}_{j=1}^n$.

To simplify the search space from high-dimensional parameter space in Eq. (2) to a more manageable matrix space, we employ **a rule-based construction approach** to modify the matrix $\boldsymbol{B}$ composed of $\{\boldsymbol{w}_i\}_{i=1}^n$ in Eq. (6) from an identity matrix to matrices of basis vectors which achieve Pareto optimality:

$$\begin{aligned} &\arg\max_{\{\boldsymbol{\theta}_i\}_{i=1}^n, \mathcal{M}} \mathcal{H}\left(\mathcal{M}\left(\{\boldsymbol{\theta}_i\}_{i=1}^n\right)\right) \\ \longrightarrow \quad &\arg\max_{\boldsymbol{B}, \mathcal{M}} \mathcal{H}\left(\mathcal{M}\left(\{\boldsymbol{\theta}_i\}_{i=1}^n\right)\right). \end{aligned} \tag{7}$$

As mentioned earlier, introducing additional reward models may help restructure better backbone models, we introduce several rules to *efficiently and effectively* determine matrix $\boldsymbol{B}$:

- **Rule 1 (Dominance).** Each combination weight $\boldsymbol{w}_i \in \mathbb{R}^n$ should have exactly one dominant component value, denoted by β_i, satisfying $\beta_i \in (1/n, 1)$. If we choose a small value for β_i, we will generate a set of backbone models with minor differences in abil-

ities causing the poor *Linear Mode Connectivity (LMC)* (Wortsman et al., 2022; Frankle et al., 2020) properties and reducing controllability of the resulting solutions.To improve efficiency, the basis vectors should possess a similar structure and we set $\beta_i = \beta, \forall_i$.

- **Rule 2 (Invertibility).** Matrix $\boldsymbol{B}$ should be invertible. Since the subsequent step involves determining the merging coefficients, we require column vectors in $\boldsymbol{B}$ to be linearly independent to ensure the effectiveness of the inversion operation and to guarantee that the column space of $\boldsymbol{B}$ does not contain redundant information.
- **Rule 3 (Normalization).** $\forall_i \sum_{j=1}^{n} w_{i,j} = 1$. This rule ensures that each $\boldsymbol{w}_i$ belongs to the n-simplex as defined in Eq. (6).

To fulfill all the rules, we adopt a symmetric circulant matrix mapping. The *symmetric circulant matrix mapping* $\boldsymbol{B}$ can be specified as follows:

$$\begin{aligned}\boldsymbol{B} &:= [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n] \\ &= \begin{pmatrix} \beta & \frac{1-\beta}{n-1} & \cdots & \frac{1-\beta}{n-1} \\ \frac{1-\beta}{n-1} & \beta & \cdots & \frac{1-\beta}{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1-\beta}{n-1} & \frac{1-\beta}{n-1} & \cdots & \beta \end{pmatrix} \in \mathbb{R}^{n \times n}. \end{aligned} \quad (8)$$

In Eq. (8), the non-dominant components are set as $(1-\beta)/(n-1)$. Taking $\boldsymbol{w}_1$ as an example, this can be interpreted as incorporating the original deterministic distribution $\boldsymbol{o}_1 := (1, 0, \ldots, 0)^\top$ with a uniform distribution $\boldsymbol{u} := (1/n, 1/n, \ldots, 1/n)^\top$ using a mixup approach: $\boldsymbol{w}_1 = \xi \boldsymbol{o}_1 + (1-\xi)\boldsymbol{u}$, where $\xi = (\beta n - 1)/(n-1) \in (0, 1)$. If we consider the basis vector $\boldsymbol{w}_i$ as a distribution for allocating rewards, this mixup method is equivalent to the exploration strategy employed in the Exp3.P algorithm (Bubeck and Cesa-Bianchi, 2012).

The next step is to select an approximate β which is the only unknown parameter in the mapping $\boldsymbol{B}$. To satisfy Rule 1 and Rule 2, we constrain β within the range $\beta \in (0.5, 1)$. Then, we train the backbone models in much smaller steps to determine which β results in the most controllable and Pareto-optimal backbone models. Specifically, we define $\beta \in \mathcal{S}$, where $\mathcal{S}$ is a finite set with cardinality m, and for any $s_i \in \mathcal{S}$, s_i is in the closed interval $[0.5, 1]$. By adjusting m, we can balance the trade-off between efficiency and performance: $\beta = \arg\max_{\beta \in \mathcal{S}} \mathcal{H}\left(\mathcal{M}_\beta\left(\{\boldsymbol{\theta}_i\}_{i=1}^n\right)\right)$.

We then use the symmetric circulant matrix mapping $\boldsymbol{B}$ to construct backbone rewards $h_i(s, a) = \sum_{j=1}^{n} w_{i,j} \cdot r_j(s, a)$ and use the reward to tune the backbone models $\{\boldsymbol{\theta}_i\}_{i=1}^n$.

3.3 Determine the Merging Coefficients

Having prepared the backbone models in the previous section, we now proceed to the merging stage. Given users' preference weights $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_n]^\top$, our objective is to determine the merging coefficients $\boldsymbol{\lambda}$ for better controllability.

As we have trained the backbone model using backbone rewards combined with multiple rewards, a natural and straightforward approach for merging is then *leveraging the reward relationship between the combination weights of backbone models and user preference weight* $\boldsymbol{\mu}$ to merge the models accordingly which is achieved by mapping the combination weight vector of the backbone rewards to the user preference illustrated in Figure 6. For instance, we will represent preference $\boldsymbol{\mu}$ by combination weights $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$ and use the solution λ_1 and λ_2 to merge models. Specifically: $\boldsymbol{\mu} = \boldsymbol{B} \cdot \boldsymbol{\lambda}$, and $\boldsymbol{\lambda} = \boldsymbol{B}^{-1}\boldsymbol{\mu}$ since $\boldsymbol{B}$ is invertible. Finally, we got the merged model parameters $\bar{\boldsymbol{\theta}} = \mathcal{M}\left(\{\boldsymbol{\theta}_i\}_{i=1}^n\right) = \sum_{i=1}^{n} \lambda_i \cdot \boldsymbol{\theta}_i$.

Existing soup-like model merging approaches (Jang et al., 2023; Rame et al., 2023) for CMOG combine specialized models linearly using $\boldsymbol{\mu}$ as the combination weight i.e. $\boldsymbol{\lambda} = \boldsymbol{\mu}$, which can also be interpreted as solving the linear equation in particular with $\boldsymbol{B}$ set as an identity matrix.

Finally, We include the extrapolation-based approach which is firstly introduced in the paper (Ilharco et al., 2022) to conduct unlearning or eliminate the effects on the expert model in specific tasks, and later used in (Zheng et al., 2024) to get a better-aligned model. We also apply extrapolation to the previously merged models as follows:

$$\hat{\boldsymbol{\theta}}^b = (1+\alpha)\hat{\boldsymbol{\theta}} - \alpha\boldsymbol{\theta}_{\text{sft}} = \hat{\boldsymbol{\theta}} + \alpha\Delta\boldsymbol{\theta}, \quad (9)$$

where $\boldsymbol{\theta}_{\text{sft}}$ is the initial model used for PPO training and $\Delta\boldsymbol{\theta} = \hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_{\text{sft}}$. $\hat{\boldsymbol{\theta}}^b$ represents the adjusted model after further diminishing the influence of the SFT model.

Table 1: Comparison of results across different methods for different trade-offs HH1 (Helpful vs Harmless), HH2 (Helpful vs Humor), and FP (Faithful vs Preference 1).

Method	Hypervolume ↑			Inner Product ↑			Controllability ↑			Length of Front ↑			Sparsity ↓			Spacing ↓		
	HH1	HH2	FP	HH1	HH2	FP	HH1	HH2	FP	HH1	HH2	FP	HH1	HH2	FP	HH1	HH2	FP
RS	1.06	1.12	0.61	1.70	1.89	1.13	0.84	**1.00**	**1.00**	**11**	**11**	**11**	0.24	0.24	**0.17**	0.02	**0.02**	**0.01**
MOD	1.08	1.09	0.62	1.83	1.85	1.17	**1.00**	**1.00**	**1.00**	**11**	**11**	**11**	0.24	0.24	0.18	0.02	**0.02**	0.02
RiC	0.45	0.66	**1.23**	1.09	1.52	**2.03**	0.85	0.80	0.82	8	6	6	**0.07**	0.25	0.39	**0.01**	0.07	0.08
Bone Soup	**1.24**	**1.24**	1.12	**2.11**	**2.06**	1.89	**1.00**	**1.00**	**1.00**	**11**	**11**	**11**	0.33	0.27	0.29	0.07	0.03	0.03

Table 2: Comparison of results across different methods for different trade-offs FR (factuality vs relevance), CR (completeness vs relevance), and FC (factuality vs completeness).

Method	Hypervolume ↑			Inner Product ↑			Controllability ↑			Length of Front ↑			Sparsity ↓			Spacing ↓		
	FR	CR	FC	FR	CR	FC	FR	CR	FC	FR	CR	FC	FR	CR	FC	FR	CR	FC
MORLHF*	0.27*	0.61*	0.16*	0.15	0.12	0.23	1.00	1.00	0.33	2	3	2	0.02	0.10	0.06	0.00	0.08	0.03
Rewarded Soups	0.28	0.82	0.17	0.77	0.56	0.82	1.00	1.00	0.98	11	11	10	0.06	0.12	0.02	0.01	0.03	0.01
Bone Soup ($\beta = 0.7$)	0.34	**0.89**	0.19	0.81	**0.61**	**0.88**	0.98	**1.00**	0.98	10	11	10	0.06	0.13	0.02	0.01	0.05	0.01
Bone Soup	**0.35**	0.86	**0.20**	**0.82**	**0.61**	0.85	**1.00**	0.98	0.93	11	10	9	0.04	0.11	0.05	0.01	0.06	0.03
Bone Soup ($\beta = 0.8$)	0.33	0.83	**0.21**	**0.82**	**0.61**	**0.88**	**1.00**	**1.00**	0.96	11	11	10	0.06	0.14	0.04	0.01	0.07	0.02

4 Experiments

In this section, we aim to evaluate the performance of Bone Soup and other latest typical controllable controllable multi-objective generation approaches.

4.1 Experiments Setups

Task Setup. We study three controllable multi-objective generation tasks using eight different rewards and two base models: **Long Form QA** (Wu et al., 2024), **Helpful Assistant** (Bai et al., 2022), and **Reddit Summary** (Stiennon et al., 2020). We use the QA-Dataset (Wu et al., 2024) and open-source reward models $\boldsymbol{R}_{\text{fact}}$ (Factuality), $\boldsymbol{R}_{\text{rele}}$ (Relevance), and $\boldsymbol{R}_{\text{comp}}$ (Completeness), considering the trade-offs: factuality vs relevance, factuality vs completeness, and relevance vs completeness. For **Helpful Assistant** task, we use the HH-RLHF dataset (Bai et al., 2022; Ganguli et al., 2022) and two reward models from Huggingface $\boldsymbol{R}_{\boldsymbol{\phi},\mathbf{1}}$ (helpful) and $\boldsymbol{R}_{\boldsymbol{\phi},\mathbf{2}}$ (harmless) to explore trade-offs helpful vs harmless and helpful vs humor. Regarding **Reddit Summary** task, we use two reward models "faithful" and "preference1" trained on different datasets to evaluate human preference for summaries. In this task, we seek controllability in trade-offs faithful vs preference1.

Implementation Details. We use LLama-2 7B (Touvron et al., 2023) for Helpful Assistant task and Reddit Summary task and use T5-large (Raffel et al., 2020) for Long Form QA task. For all three tasks, we choose the best $\beta \in \{0.8, 0.7, 0.6\}$ by only training for 20% total steps and evaluate the hypervolume. As for the extrapolation of $\hat{\boldsymbol{\theta}}$, we also select the optimal $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ using a validation datasets following the approach in (Zheng et al., 2024).

Baselines. We consider three latest CMOG approaches including prompt-based approach Rewards-in-Context (RiC) (Yang et al., 2024), decoding-time approach MOD (Shi et al., 2024) and merging-based method (Rame et al., 2023) and follow their settings of the Hyperparameters. Detailed introduction and discussion about baselines are in Appendix A.4.1.

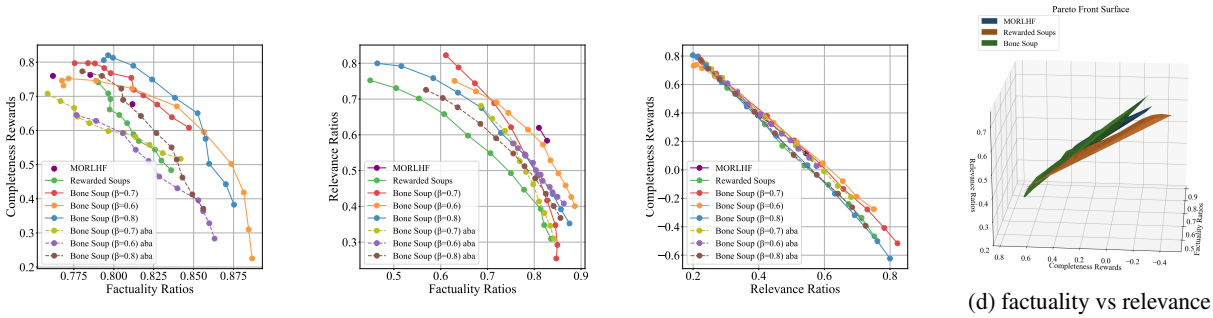
Evaluation Metrics. We provide both visualization and six numerical metrics for evaluation. To make the results more intuitive, we plot the Pareto Front of the rewards of each dimension for the evaluated set of preference vectors. A detailed introduction and discussion of all the metrics can be found in Appendix A.5.

4.2 Results

4.2.1 Long Form QA task

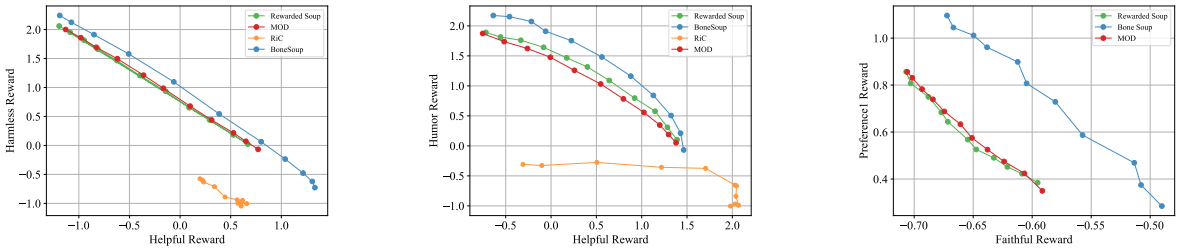
For the task of Long Form QA (Wu et al., 2024), As shown in Figure 4, each point in the front represents the average rewards of the solution corresponding to a specific user preference evaluated on test set.

In Figure 4 and Table 2, we compare Bone Soup (BS) at different β values with Rewarded Soup (RS) and MORLHF. The selection of β is discussed in A.3.6 and A.3.2. BS consistently outperforms RS and closely approximates Oracle MORLHF across three trade-offs. In factuality vs completeness and relevance vs completeness, BS even sur-



(a) factuality vs completeness (b) factuality vs relevance (c) relevance vs completeness (d) factuality vs relevance vs completeness

Figure 4: Results of the Long Form QA task with (a) "factuality vs. relevance", (b) "factuality vs. completeness", (c) "relevance vs. completeness" and (d) "factuality vs relevance vs. completeness". We connect the points in the figure according to the order of the preference weight partial order relation. Bone Soup learns a better front than RS.



(a) helpful vs harmless (b) helpful vs humor (c) faithful vs preference1

Figure 5: Results of Helpful Assistant task with (a) "helpful vs. harmless", (b) "helpful vs. humor", and Reddit Summary task with (c) "faithful vs. preference1".

passes MORLHF, achieving a superior Pareto front.

Additionally, experiments in A.3.1 show that combining multiple rewards generally improves the backbone model. This led us to investigate merging backbone models based on user preferences, without considering the reward interplay in merging coefficients. As seen in Figure 4, the direct merge approach ("ABA") performs worse than RS in factuality vs completeness, but slightly outperforms it in the other two trade-offs, though still behind BS.

Overall, the superior performance of BS relative to MORLHF, together with the instability and suboptimality of ABA, validates the necessity and advantage of the two-stage, seek-and-soup merging approach employed by Bone Soup.

Recent studies (Mao et al., 2023; Lambert et al., 2024; Sottana et al., 2023) have suggested that generative models can serve as unbiased evaluators—especially when ground-truth reward models are unavailable—making the use of models like GPT-4 a viable and effective evaluation approach. Therefore, in addition to using reward models, we incorporated GPT-based assessments to simulate more realistic evaluation scenarios. As shown in Figures 9a and Figure 9b, under the trade-off factuality vs relevance and across various user preferences, BoneSoup consistently outperforms Rewarded Soup, which is in line with our previous consequences.

We also conducted experiments in a three-objective setting. As shown in Figure 4d, the front obtained by RS is dominated by that of MORLHF. Additionally, we observe that the front of BS is Pareto-dominant over that of MORLHF.

4.2.2 Helpful Assistant

In this task, we focus on trade-offs "Helpful vs Harmless" (HH1), "Helpful vs Humor" (HH2). From Figure 5a, Figure 5b and Table 1, we can observe that the obtained front of RS approaches and MOD with similar shapes among which BS achieves the best front compared with all other baselines while RiC struggles with this task. The reason may lie in the difference between paradigms of RLHF and conditional SFT as RS and MOD all utilize the models tuned from RLHF and may obtain a similar shape.

From Figure 5a, we can observe that Bone Soup

consistently outperforms MOD which combines multiple backbone models' logits to achieve controllability. Compared to RS, both BS and MOD leverage different techniques to enhance the utilization of a set of backbone models, exploring how to better utilize these models for controllable multi-objective generation to varying degrees. However, BS provides a more comprehensive and fine-grained utilization of RLHF models therefore leading to a significantly better result.

4.2.3 Reddit Summary

In this task, we focus on trade-offs "Faithful vs Preference 1" (FP). From Table 1 and Figure 5c, We can see that BS significantly outperforms RS and MOD; however, in terms of hypervolume, BS falls short compared to RiC. Nevertheless, RiC performs poorly in controllability and has only 6 points on the front as shown in Figure 7. Since the region of the front generated by RiC differs significantly from the front of BS, RS, and MOD, we therefore display RiC separately from BS, RS, and MOD for clarity.

5 Related Work

To be brief, our work is closely related to research on model merging, multi-objective optimization, and controllable generation. Due to space limitations, we provide a detailed discussion of these topics in Appendix A.1.

6 Conclusions

In this work, we proposed Bone Soup, a novel model merging approach designed to address the challenges of controllable multi-objective generation. By introducing rule-based construction backbone models and combining rewards, we improved the merging process to achieve better controllability and Pareto-optimality. Extensive experiments show that Bone Soup outperforms existing methods, offering enhanced adaptability to dynamic user preferences and providing an effective and efficient solution for multi-objective generation tasks.

7 Limitations

Our work has the following limitations:(1) Our experiments primarily focus on controllable text generation based on human preferences, but we rely on automatic evaluators, including reward models and GPT-4, without conducting human evaluations. (2) Due to the relatively low additional complexity introduced in MORL (Wu et al., 2024), along with the existence of multi-value head reward models (Wang et al., 2023; Köpf et al., 2023; Wang et al., 2024a), our method is not significantly impacted by the number of objectives during training. As such, our approach can naturally scale to more than three objectives, but we have not conducted additional experiments with a larger number of objectives. (3) While our model merging approach can be easily applied to fields such as computer vision and multimodal tasks, we have not conducted additional experiments to validate its performance in these areas.

8 Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (No. 62376275, 92470205). Work partially done at Beijing Key Laboratory of Research on Large Models and Intelligent Governance, and Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE. Supported by fund for building world-class universities (disciplines) of Renmin University of China.

References

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

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

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Meghana Moorthy Bhat, Rui Meng, Ye Liu, Yingbo Zhou, and Semih Yavuz. 2023. Investigating answerability of llms for long-form question answering. *arXiv preprint arXiv:2309.08210*.

Ryan Boldi, Li Ding, Lee Spector, and Scott Niekum. 2024. Pareto-optimal learning from preferences with hidden context. *arXiv preprint arXiv:2406.15599*.

Sébastien Bubeck and Nicolò Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

Stephen Casper, Xander Davies, Claudia Shi,

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

Ruizhe Chen, Xiaotian Zhang, Meng Luo, Wenhao Chai, and Zuozhu Liu. 2024. Pad: Personalized alignment at decoding-time. *arXiv preprint arXiv:arXiv:2410.04070*.

Sirui Chen, Yuan Wang, Zijing Wen, Zhiyu Li, Changshuo Zhang, Xiao Zhang, Quan Lin, Cheng Zhu, and Jun Xu. 2023. Controllable multi-objective re-ranking with policy hypernetworks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3855–3864.

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.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. 2023. Safe rlhf: Safe reinforcement learning from human feedback. *arXiv preprint arXiv:2310.12773*.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

Haikang Deng and Colin Raffel. 2023. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. *arXiv preprint arXiv:2310.09520*.

Yi Dong, Zhilin Wang, Makesh Narsimhan Sreedhar, Xianchao Wu, and Oleksii Kuchaiev. 2023. Steerlm: Attribute conditioned sft as an (user-steerable) alternative to rlhf. *arXiv preprint arXiv:2310.05344*.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. 2019. Implementation matters in deep rl: A case study on ppo and trpo. In *International conference on learning representations*.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. 2022. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*.

Yukun Huang, Yixin Liu, Raghuveer Thirukovalluru, Arman Cohan, and Bhuwan Dhingra. 2024. Calibrating long-form generations from large language models. *arXiv preprint arXiv:2402.06544*.

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

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564*.

Maxim Khanov, Jirayu Burapacheep, and Yixuan Li. 2024. Args: Alignment as reward-guided search. *arXiv preprint arXiv:2402.01694*.

Andreas Köpf, Yannic Kilcher, Dimitri Von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36:47669–47681.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, et al. 2024. Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.

Rui Mao, Guanyi Chen, Xulang Zhang, Frank Guerin, and Erik Cambria. 2023. Gpteval: A survey on assessments of chatgpt and gpt-4. *arXiv preprint arXiv:2308.12488*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.

Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. 2016. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, and Matthieu Cord. 2023. Rewarded soups: Towards Pareto-optimal alignment by interpolating

weights fine-tuned on diverse rewards. *Advances in Neural Information Processing Systems 36*.

Sahana Ramnath, Brihi Joshi, Skyler Hallinan, Ximing Lu, Liunian Harold Li, Aaron Chan, Jack Hessel, Yejin Choi, and Xiang Ren. 2023. Tailoring self-rationalizers with multi-reward distillation. *arXiv preprint arXiv:2311.02805*.

V Sanh. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Jason Ramon Schott. 1995. *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Ph.D. thesis, Massachusetts Institute of Technology.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Chenglei Shen, Xiao Zhang, Teng Shi, Changshuo Zhang, Guofu Xie, and Jun Xu. 2024a. A survey of controllable learning: Methods and applications in information retrieval. *arXiv preprint arXiv:2407.06083*.

Chenglei Shen, Jiahao Zhao, Xiao Zhang, Weijie Yu, Ming He, and Jianping Fan. 2024b. Generating model parameters for controlling: Parameter diffusion for controllable multi-task recommendation. *arXiv preprint arXiv:2410.10639*.

Ruizhe Shi, Yifang Chen, Yushi Hu, ALisa Liu, Noah Smith, Hannaneh Hajishirzi, and Simon Du. 2024. Decoding-time language model alignment with multiple objectives. *Advances in Neural Information Processing Systems 37*.

Anand Siththaranjan, Cassidy Laidlaw, and Dylan Hadfield-Menell. 2023. Distributional preference learning: Understanding and accounting for hidden context in rlhf. *arXiv preprint arXiv:2312.08358*.

Andrea Sottana, Bin Liang, Kai Zou, and Zheng Yuan. 2023. Evaluation metrics in the era of gpt-4: reliably evaluating large language models on sequence to sequence tasks. *arXiv preprint arXiv:2310.13800*.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. Asqa: Factoid questions meet long-form answers. *arXiv preprint arXiv:2204.06092*.

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.

Anke Tang, Li Shen, Yong Luo, Nan Yin, Lefei Zhang, and Dacheng Tao. 2024. Merging multi-task models via weight-ensembling mixture of experts. *arXiv preprint arXiv:2402.00433*.

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*.

Haoxiang Wang, Yong Lin, Wei Xiong, Rui Yang, Shizhe Diao, Shuang Qiu, Han Zhao, and Tong Zhang. 2024a. Arithmetic control of LLMs for diverse user preferences: Directional preference alignment with multi-objective rewards. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8642–8655, Bangkok, Thailand. Association for Computational Linguistics.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. 2024b. Localizing task information for improved model merging and compression. *arXiv preprint arXiv:2405.07813*.

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, et al. 2023. Helpsteer: Multi-attribute helpfulness dataset for steerlm. *arXiv preprint arXiv:2311.09528*.

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

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2024. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems*, 36.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2024. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36.

Enneng Yang, Zhenyi Wang, Li Shen, Shiwei Liu, Guibing Guo, Xingwei Wang, and Dacheng Tao. 2023. Adamerging: Adaptive model merging for multi-task learning. *arXiv preprint arXiv:2310.02575*.

Rui Yang, Xiaoman Pan, Feng Luo, Shuang Qiu, Han Zhong, Dong Yu, and Jianshu Chen. 2024. Rewards-in-context: Multi-objective alignment of foundation models with dynamic preference adjustment. *International Conference on Machine Learning*.

Le Yu, Bowen Yu, Haiyang Yu, Fei Huang, and Yongbin Li. 2024. Language models are super mario: Absorbing abilities from homologous models as a free lunch. In *Forty-first International Conference on Machine Learning*.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. 2024. Weak-to-strong extrapolation expedites alignment. *arXiv preprint arXiv:2404.16792*.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. 2023. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*.

Yifan Zhong, Chengdong Ma, Xiaoyuan Zhang, Ziran Yang, Haojun Chen, Qingfu Zhang, Siyuan Qi, and Yaodong Yang. 2024. Panacea: Pareto alignment via preference adaptation for llms. *arXiv preprint arXiv:2402.02030*.

Zhanhui Zhou, Jie Liu, Jing Shao, Xiangyu Yue, Chao Yang, Wanli Ouyang, and Yu Qiao. 2024. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10586–10613, Bangkok, Thailand. Association for Computational Linguistics.

Eckart Zitzler and Lothar Thiele. 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation*, 3(4):257–271.

Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation*, 7(2):117–132.

A Appendix

A.1 Related Work

A.1.1 Multi-Objective Optimization and Generation

Reinforcement Learning with Human Feedback (RLHF) (Christiano et al., 2017; Stiennon et al., 2020; Ouyang et al., 2022), consisting of two stages—reward modeling and reinforcement learning—has become a powerful tool to align large language models (LLMs) with human preferences. Many existing models (Touvron et al., 2023; Achiam et al., 2023) utilize RLHF to enhance their performance. However, optimizing toward a single reward has notable limitations, such as its inability to handle complex, multifaceted preferences (Casper et al., 2023), the challenge of satisfying all preferences with a single reward (Jang et al., 2023; Rame et al., 2023), and issues related to fairness in alignment(Siththaranjan et al., 2023; Boldi et al., 2024; Rame et al., 2023). To address these shortcomings, multi-objective RLHF (MORLHF) has been introduced.

One of the most straightforward ways to adapt RLHF for multiple objectives is to combine all rewards linearly (Mossalam et al., 2016). However, due to the inefficiency of this approach in MORLHF, this paradigm struggles to quickly adapt to different preferences and achieve controllable multi-objective generation. Recently, an increasing number of studies have focused on controllable multi-objective generation. Methods for controllable multi-objective generation can be categorized into three main stages: pre-processing, in-processing, and post-processing. Pre-processing methods, like SteerLM (Dong et al., 2023), DPA (Wang et al., 2024a), and RiC (Yang et al., 2024), implement control through prompts, introducing multi-dimensional reward conditions. These methods use supervised fine-tuning to train the model to control outputs by prompts. The fine-tuning strategies and condition representations vary across methods, including rejection-sampling-based fine-tuning (Wang et al., 2024a; Yang et al., 2024) and representing conditions as unit vectors (Wang et al., 2024a) or by theoretical guarantee mapping (Yang et al., 2024).

In-processing methods (Rame et al., 2023; Jang et al., 2023) focus on model merging, where specialized models are combined using different merge coefficients to quickly generate models that cater to various preferences. This approach is straightforward to implement and computationally efficient.

Post-processing methods, such as Controlled Text Generation (CTG), primarily involve decoding-time algorithms (Khanov et al., 2024; Deng and Raffel, 2023; Shi et al., 2024). These methods generate the next token by taking a linear combination of predictions from multiple base models, based on different objective weightings. Reward signals are used to find the optimal merging coefficients. For instance, MOD (Shi et al., 2024) identifies a closed-form solution using the Legendre transform, deriving an efficient decoding strategy, while ARGS (Khanov et al., 2024) and RAD (Deng and Raffel, 2023) achieves alignment by reward-guided search.

This paper focuses on introducing control during the in-processing phase, incorporating explicit control mechanisms into the model parameters to enable controllable generation.

A.1.2 Model Merging

We denote the policy LLM as $\pi_{\boldsymbol{\theta}}$ whose parameters are $\boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^d$. $\mathcal{X}$ and $\mathcal{Y}$ represent the input space (prompt space) and output space individually. We have summarized the current model merging

techniques into the following three steps: *determining the base models, merging the backbone models, and calibration after model merging*. We mainly focus on discussing the first two stages.

Determining the base models, i.e., identifying the parameter space for interpolation. Denote the models to merge in the following step as $\{\pi_{\boldsymbol{\theta}_i}\}_{i=1}^{m}$. Here, it is generally assumed that the number of models to be merged is equal to the number of objectives or tasks, i.e., $m = n$. Moreover, these models are typically trained using a single loss (Ilharco et al., 2022; Yu et al., 2024) or reward (Wu et al., 2024; Jang et al., 2023), meaning they can be regarded as expert models corresponding to each task or objective.

Merging the base models. After obtaining n specializing (expert models in a multi-task setting) with different focuses, the next step is to determine the interpolation coefficients λ for model merging, $\boldsymbol{\theta}_{\text{target}} = \sum_{i=1}^{n} \lambda_i \cdot \boldsymbol{\theta}_i$. Rewarded Soup (Rame et al., 2023) proposes to merge the models optimized individually against a single objective. And $\lambda_i \in \Omega$ and $\Omega = \{\lambda_i \in \mathbb{R}^k | \sum_{i=1}^{n} \lambda_i = 1, \lambda_i \geqslant 0\}$.

In the field of multi-task learning, various model merging approaches have been proposed. Tang et al. (2024) using a dynamic routing mechanism trained in test-time adaptation to determine the model fusion coefficients. Some approaches exploit model parameter redundancy, leading to pruning-based approaches (Yadav et al., 2024; Yu et al., 2024). AdaMerging (Yang et al., 2023) employs an unsupervised approach to train merging coefficients and adjusts the merging coefficients at different layers of the models. TALL-masks (Wang et al., 2024b) generates a task-specific mask matrix using a predefined threshold derived from independent models. The key distinction between our approach and the above works lies in that they are not designed for, nor capable of, achieving controllable generation. In contrast, we have developed a series of techniques specifically aimed at optimizing for Pareto optimality and controllability.

A.2 The Proof of Theorem 1

Theorem 1. *Given quadratic reward functions with Hessians proportional to identity matrices:*

$$r_i(\theta) = r_i(\theta_i) - k_i\|\theta - \theta_i\|^2, i \in \{1, 2\},$$

where $k_i \in \mathbb{R}_+$ are distinct,and θ_i is the global maximum for reward r_i. Let the reward combination weight matrix be $B = \begin{pmatrix} \beta & 1-\beta \\ 1-\beta & \beta \end{pmatrix}, \beta \in (\frac{1}{2}, 1)$, then the backbone rewards of the bone-soup approach can be denoted as $[h_1, h_2]^T = B[r_1, r_2]^T$.Let $\boldsymbol{\mu} = [\mu, 1-\mu]^T$ be the user preference and the testing reward can written as $g_{\boldsymbol{\mu}}(\theta) := \boldsymbol{\mu}^T \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$.Denote the approximate solutions for the testing reward $g_\mu(\theta)$ of the soup-like approach and the bone-soup approach as $\bar{\theta}$ and $\bar{\theta}^{bone}$, respectively.Then,for any fixed $\beta \in (\frac{1}{2}, 1)$, when $\mu \in \left(\frac{1-\sqrt{2\beta^2-2\beta+1}}{2}, \frac{1+\sqrt{2\beta^2-2\beta+1}}{2}\right)$,

$$g_\mu(\bar{\theta}) < g_\mu(\bar{\theta}^{bone}),$$

with interval length $\sqrt{2\beta^2 - 2\beta + 1} \geqslant \frac{\sqrt{2}}{2}$.

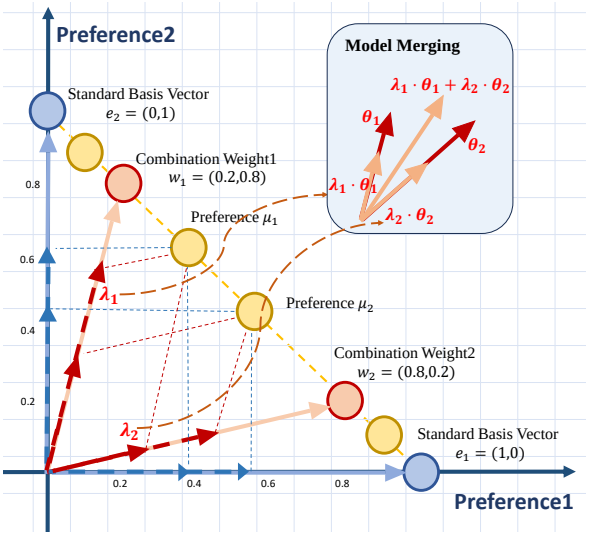


Figure 6: The illustration of how to leverage the relationship between user preferences and the rewards of backbone models to obtain the mapping between the target merged model and the backbone models.

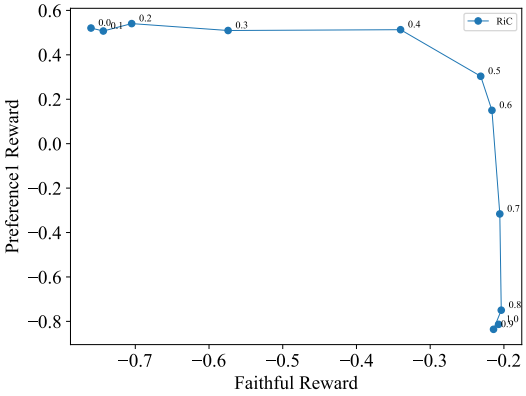


Figure 7: The front learned from RiC in trade-offs“ faithful vs preference1” (FP)

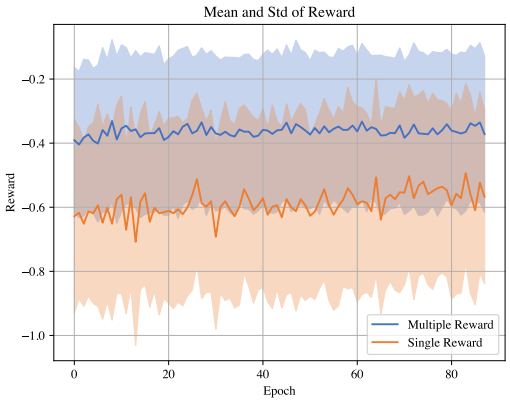


Figure 8: The mean and standard deviation of rewards per batch during the PPO optimization process for training the backbone model in Bone Soup (multiple rewards) and Rewarded Soup (single reward). The process of tuning the backbone model in Bone Soup is more stable compared to that in Rewarded Soup.

Proof. The testing reward $g_\mu = \mu r_1 + (1-\mu) r_2$ is quadratic thus has an unique global maximum θ^*,that we find analytically:

$$\nabla_\theta g_\mu(\theta) = 0 \Rightarrow \mu k_1(\theta - \theta_1) + (1-\mu)k_2(\theta - \theta_2) = 0$$
$$\Rightarrow \theta^* = \frac{\mu k_1 \theta_1 + (1-\mu)k_2\theta_2}{\mu k_1 + (1-\mu)k_2}$$

The approximate solution $\bar{\theta}$ for the testing reward g_μ of the soup-like approach is formulated as,

$$\bar{\theta} = \mu\theta_1 + (1-\mu)\theta_2$$

Consider the bone-soup approach,we have the backbone rewards and their corresponding global maximums as follows,

$$h_1 = \beta r_1 + (1-\beta) r_2, \theta_1^{bone} = \frac{\beta k_1 \theta_1 + (1-\beta)k_2\theta_2}{\beta k_1 + (1-\beta)k_2}$$
$$h_2 = (1-\beta) r_1 + \beta r_2, \theta_2^{bone} = \frac{(1-\beta) k_1 \theta_1 + \beta k_2\theta_2}{(1-\beta)k_1 + \beta k_2}$$

the merging coefficients can be calculated as $\boldsymbol{\lambda} = B^{-1}\boldsymbol{\mu} = [\lambda, 1-\lambda]^T$,where $\lambda = \frac{\beta+\mu-1}{2\beta-1}$,then we have the soup-like approach's approximate solution $\bar{\theta}^{bone}$ for the testing reward g_μ as,

$$\bar{\theta}^{bone} = \lambda\theta_1^{bone} + (1-\lambda)\theta_2^{bone}$$

We set the error function as $E(\beta,\mu) = \|\bar{\theta}^{bone} - \theta^*\|^2$.Since the soup-like approach can be regarded as a special case of the bone-soup approach when $\beta = 1$,we use $E(1,\mu)$ to denote the error of the soup-like approach.Under the current settings, the testing reward g_μ can be written as $g_\mu = c_1 - c_2\|\theta - \theta^*\|^2$,where c_1 and c_2 are constants, and $c_2 \in \mathbb{R}_+$.Therefore $g_\mu(\bar{\theta}) < g_\mu(\bar{\theta}^{bone}) \Leftrightarrow E(\beta,\mu) < E(1,\mu)$.

The expressions for $E(\beta,\mu)$ and $E(1,\mu)$ can be calculated as follows,

$$E(\beta,\mu) =$$
$$\|\lambda\frac{\beta k_1\theta_1 + (1-\beta)k_2\theta_2}{\beta k_1 + (1-\beta)k_2} +$$
$$(1-\lambda)\frac{(1-\beta)k_1\theta_1 + \beta k_2\theta_2}{(1-\beta)k_1 + \beta k_2} - \frac{\mu k_1\theta_1 + (1-\mu)k_2\theta_2}{\mu k_1 + (1-\mu)k_2}\|^2$$
$$= \left(\frac{k_1k_2(k_1-k_2)(\beta-\mu)(\beta+\mu-1)}{(\mu k_1 + (1-\mu)k_2)(\beta k_1 + (1-\beta)k_2)((1-\beta)k_1 + \beta k_2)}\right)^2 \|\theta_1 - \theta_2\|^2$$

$$E(1,\mu) = (\frac{(k_1-k_2)(1-\mu)\mu}{\mu k_1 + (1-\mu)k_2})^2\|\theta_1 - \theta_2\|^2$$

To compare $E(\beta,\mu)$ and $E(1,b)$, we have:

$$E(\beta,\mu) < E(1,\mu)$$
$$\Leftrightarrow (\frac{k_1k_2(\beta-\mu)(\beta+\mu-1)}{[\beta k_1 + (1-\beta)k_2][(1-\beta)k_1 + \beta k_2]})^2 < ((1-\mu)\mu)^2$$

since

$$[\beta k_1 + (1-\beta)k_2][(1-\beta)k_1 + \beta k_2]$$
$$= k_1k_2[2\beta^2 - 2\beta + 1 + \beta(1-\beta)(\frac{k_1}{k_2} + \frac{k_2}{k_1})]$$
$$\geqslant k_1k_2(2\beta^2 - 2\beta + 1 + 2\beta(1-\beta)) = k_1k_2$$

Therefore, we obtain:

$$\left(\frac{k_1k_2(\beta-\mu)(\beta+\mu-1)}{[\beta k_1 + (1-\beta)k_2][(1-\beta)k_1 + \beta k_2]}\right)^2 < ((\beta-\mu)(\beta+\mu-1))^2$$

$$E(\beta,\mu) < E(1,\mu)$$
$$\Leftarrow (\mu(1-\mu))^2 - ((\beta-\mu)(\beta+\mu-1))^2$$
$$= (\beta-\beta^2)(-2\mu^2 + 2\mu + \beta^2 - \beta) > 0$$

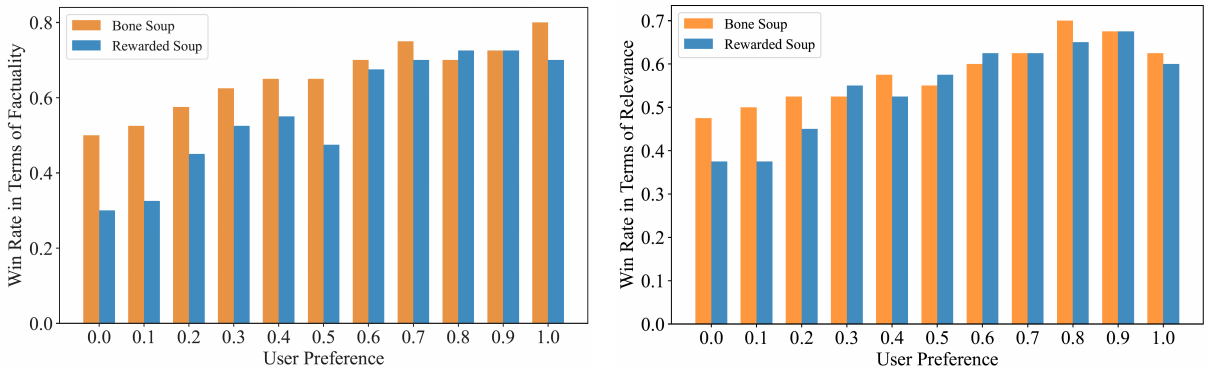
We can observe that,for any fixed $\beta \in (\frac{1}{2}, 1)$, the right-hand side of the equation holds, for all $\mu \in \left(\frac{1-\sqrt{2\beta^2-2\beta+1}}{2}, \frac{1+\sqrt{2\beta^2-2\beta+1}}{2}\right)$, i.e. $E(\beta,\mu) < E(1,\mu)$ holds.Besides,we can find that the interval length $\sqrt{2\beta^2-2\beta+1} \geqslant \frac{\sqrt{2}}{2}$.Thus, the theorem is proved.

□

A.3 Additional Experiments

A.3.1 RQ1: The Significance of Reconstructing Appropriate Backbone Models.

In Section 1, we have already demonstrated the importance of reconstructing appropriate backbone



(a) Factuality evaluation results

(b) Relevance evaluation results

Figure 9: The real evaluation results by GPT-4 under different user preferences. Bone Soup achieves better performance compared with Rewarded Soup.

models using a mathematical example. Here, we further illustrate the significance of basis reconstruction through some observations and empirical analysis.

Observation 1: *The specializing models trained with an individual reward for a single objective are not necessarily the optimal models under that specific reward function.*

The effectiveness of model merging fundamentally depends on the quality and diversity of the backbone models. Relying solely on models trained for specific objectives may not yield optimal results, as such models might not fully explore or exploit the entire reward landscape. In Figure 4, we observe that the specializing models extrapolated by the two backbone models of Bone Soup consistently extend in two reward dimensions and outperform the specializing models tuned in RS, verifying the fact that models tuned with a specific reward may not always be the optimal ones for that reward and can be outperformed by models derived through various interpolation techniques.

Observation 2: *Incorporating additional rewards into the reward function enhances stability during model tuning.*

In addition to the mathematical examples discussed in previous sections, we present some observations on incorporating additional rewards. During the PPO tuning process, a single reward model may provide incorrect or unreasonable signals in specific situations due to its inherent limitations (Casper et al., 2023), leading to significant fluctuations. By employing multiple reward models, these limitations can be mitigated through mutual complementation or correction, enhancing the stability of the tuning process.

We empirically validate that using multiple reward models can help smooth out the high variance problems introduced by a single model as shown in Figure 8. And Figure 10 illustrates the training process of the factuality-specialized model using both combined rewards and a single reward during PPO training. The figure plots various metrics, including rewards, KL divergence, policy loss, and total loss. We observe that a spike in KL divergence during PPO training is indicative of model collapse, accompanied by a decline in the corresponding rewards, which suggests that early stopping is necessary. As shown in Figure 10d, compared to the combined reward (especially at β values of 0.6 and 0.8), the single reward leads to a more rapid increase in KL divergence, reaching the threshold sooner and triggering premature termination of training. Additionally, despite the combined reward placing relatively less emphasis on the factuality dimension—resulting in a lower focus on factuality during PPO training—it nonetheless delivers superior factuality performance. At the same time, the rewards across other dimensions are also enhanced compared to the rewarded soup approach, as evidenced by Figures 10b and 10c.

Moreover, Figures 10d, 10e, and 10f clearly show that the training process with the combined reward is more stable. We attribute this stability to the integration of multiple rewards, which helps to counteract the incorrect and unstable signals that a single reward model might introduce. In essence, by blending multiple rewards, the potential instabil-

ities during training are effectively mitigated. And the similar results of training the completeness-specialized and relevance-specialized models are shown in Figure 11. In Figure 11d we can also observe a similar spike and higher KL compared with combined rewards. In Figure 11e, we also found a higher policy loss potentially representing the difficulty of convergence during training.

A.3.2 RQ2: Can the Small-Scale Selection Yield a β Consistent with or Near the Optimal β?

In this section, we discuss the approach to selecting the β parameter when constructing the matrix $\boldsymbol{B}$. Table 3 presents BoneSoup's performance under various β values for different trade-offs. We expect that a small-scale training run can effectively approximate the optimal beta found through full-scale training. Notably, Table 3 reveals that for all three trade-offs, a beta value of 0.6 consistently yields the best performance. Correspondingly, Table 2 shows that, with full-scale training, the optimal performance is achieved at $\beta = 0.6$ for the FR (factuality vs. relevance) and FC (factuality vs. completeness) trade-offs, while for the CR (completeness vs. relevance) trade-off, $\beta = 0.7$ is optimal, with $\beta = 0.6$ coming in as a close second. This strong alignment between small-scale and full-scale training results underscores the soundness and robustness of our β selection strategy, enabling us to efficiently acquire a near-optimal β that approximates the best possible performance.

A.3.3 RQ3: Can Our Bone Soup Matrix Construction Method Effectively Identify High-Quality Matrices in a Vast Solution Space?

To demonstrate the robustness and superiority of the Bone Soup matrix construction method, this section presents a performance comparison between the Bone Soup method and random matrix construction methods. Due to computational constraints, it is infeasible to exhaustively enumerate all potential backbone matrices $\boldsymbol{B}$. Therefore, we randomly constructed eight relatively representative matrices, as shown in Table 4.

The results are shown in Table 5. From the table, we can observe that under three different trade-offs, all Bone Soup variants utilizing multiple rewards outperform the naive Rewarded Soup method, which relies on a single reward. This advantage stems from the fact that multi-reward RL facilitates the construction of a superior backbone model, demonstrating that using combined rewards can better approximate the optimal solution.

Moreover, employing our proposed rule-based construction method in combination with the hypervolume-selection approach further surpasses the randomly selected reward matrices. Bone Soup achieves the best performance, while Bone Soup ($\beta = 0.7$), which solely adopts the rule-based construction without adaptation, attains the second-best performance, still outperforming the randomly constructed method.

A.3.4 RQ4: Robustness Analysis of Bone Soup

Due to the inherent instability and randomness of PPO optimization(Zheng et al., 2023; Casper et al., 2023; Engstrom et al., 2019), we randomly selected three seeds to rigorously assess the robustness of Bone-Soup. As illustrated in Figure 15, even with varying seeds, our approach consistently outperforms Rewarded Soups and remains very close to Oracle MORLHF. In several cases—specifically in Figure 15b, 15c, 15d, and 15f—Bone-Soup even achieves a Pareto front that surpasses MORLHF, further demonstrating its robustness. We also show the results in three-objective setting in Figure 16.

A.3.5 RQ5: How Does the Determination of Merging Coefficients Affect the Performance of Bone Soup?

We conducted ablation experiments on determination of the merging coefficients. Bone Soup with the suffix 'aba' refers to using Bone Soup to obtain backbone models, and then setting $\boldsymbol{\mu} = \boldsymbol{\lambda}$ during merging, which means directly mapping the preference. The results in Figure 12a and 12b indicate that, even with better backbone models, the merged model still underperforms compared to RS, highlighting the importance of the merging coefficients determination process. This underscores the importance of establishing a strong link between rewards and user preferences, and further validates the critical role of the "soup" stage in our two-stage seek-and-soup approach.

A.3.6 RQ6: How Do Different Values of β Impact the Performance of Bone Soup?

We conducted experiments on the different trade-offs in Long Form QA task and Reddit Summary task. In Figure 4 and Figure 13, we can see that by varying β, the resulting front consistently out-

Table 3: The results of β selection for three different trade-offs in the Long-form QA task. As shown in the table, β = 0.6 achieves the best hypervolume(for simplicity of the method, we assume that hypervolume could represent the overall performance of the front.) on the small-scale validation set, and thus, β = 0.6 is ultimately chosen for the three different trade-offs.

Method	Hypervolume ↑			Inner Product ↑			Controllability ↑			Length of Front ↑			Sparsity ↓			Spacing ↓		
	FR	CR	FC	FR	CR	FC	FR	CR	FC	FR	CR	FC	FR	CR	FC	FR	CR	FC
Bone Soup (6) select	**0.159**	**0.762**	**0.304**	**0.799**	**0.507**	**0.768**	1.000	1.000	0.982	11	11	10	0.015	0.106	0.038	0.006	0.022	0.006
Bone Soup (7) select	0.150	0.735	0.281	0.769	0.493	0.754	1.000	1.000	1.000	11	11	11	0.020	0.090	0.032	0.004	0.015	0.003
Bone Soup (8) select	0.139	0.729	0.296	0.742	0.471	0.766	0.982	1.000	1.000	10	11	11	0.017	0.093	0.035	0.007	0.013	0.006

Table 4: The Specific Representation of the Matrix $\boldsymbol{B}_i$

The randomly selected matrices $\boldsymbol{B}_i$	
$\boldsymbol{B}_1 = \begin{pmatrix} 0.7 & 0.2 & 0.15 \\ 0.15 & 0.6 & 0.15 \\ 0.15 & 0.2 & 0.7 \end{pmatrix}$	$\boldsymbol{B}_2 = \begin{pmatrix} 0.7 & 0.1 & 0.1 \\ 0.15 & 0.8 & 0.1 \\ 0.15 & 0.1 & 0.8 \end{pmatrix}$
$\boldsymbol{B}_3 = \begin{pmatrix} 0.7 & 0.15 & 0.1 \\ 0.15 & 0.7 & 0.1 \\ 0.15 & 0.15 & 0.8 \end{pmatrix}$	$\boldsymbol{B}_4 = \begin{pmatrix} 0.7 & 0.15 & 0.2 \\ 0.15 & 0.7 & 0.2 \\ 0.15 & 0.15 & 0.6 \end{pmatrix}$
$\boldsymbol{B}_5 = \begin{pmatrix} 0.8 & 0.15 & 0.2 \\ 0.1 & 0.7 & 0.2 \\ 0.1 & 0.15 & 0.6 \end{pmatrix}$	$\boldsymbol{B}_6 = \begin{pmatrix} 0.7 & 0.2 & 0.1 \\ 0.15 & 0.6 & 0.1 \\ 0.15 & 0.2 & 0.8 \end{pmatrix}$
$\boldsymbol{B}_7 = \begin{pmatrix} 0.7 & 0.2 & 0.2 \\ 0.15 & 0.6 & 0.2 \\ 0.15 & 0.2 & 0.6 \end{pmatrix}$	$\boldsymbol{B}_8 = \begin{pmatrix} 0.8 & 0.2 & 0.2 \\ 0.1 & 0.6 & 0.2 \\ 0.1 & 0.2 & 0.6 \end{pmatrix}$

Table 5: The comparison between Bone Soup and random matrix constction methods in 3 different trade-offs

Method	RC_Rank	FR_Rank	FC_Rank	Avg_Rank
Rewarded Soups	12.0	12.0	12.0	12.00
Bone Soup	7.0	2.0	2.0	**3.67**
Bone Soup (β=0.7)	1.0	3.0	8.0	<u>4.00</u>
Bone Soup (β=0.8)	11.0	8.0	1.0	6.67
Bone Soup $\boldsymbol{B}_1$	3.0	1.0	10.0	4.67
Bone Soup $\boldsymbol{B}_2$	9.0	11.0	4.0	8.00
Bone Soup $\boldsymbol{B}_3$	8.0	9.0	6.0	7.67
Bone Soup $\boldsymbol{B}_4$	2.0	6.0	9.0	5.67
Bone Soup $\boldsymbol{B}_5$	4.0	5.0	3.0	4.00
Bone Soup $\boldsymbol{B}_6$	10.0	10.0	7.0	9.00
Bone Soup $\boldsymbol{B}_7$	4.0	6.0	11.0	7.00
Bone Soup $\boldsymbol{B}_8$	6.0	4.0	5.0	5.00

performs and dominates RS. This shows that the choice of β does not significantly impact the performance of the front and a good β only would further improve the upper bound of our method. This demonstrates the robustness in choosing the β parameter, thereby underscoring both the lower bound performance and overall robustness of our method.

A.3.7 RQ7: What is the Impact of Extrapolation on Performance of Bone Soup?

We conducted experiments on the trade-off HH1 (Helpful vs Harmless) on Helpful Assistant dataset. The results in Figure 14 show that after incorporating interpolation in Equation 9, the front obtained by BS is indeed improved, indicating that interpolation can further enhance the model's capabilities.

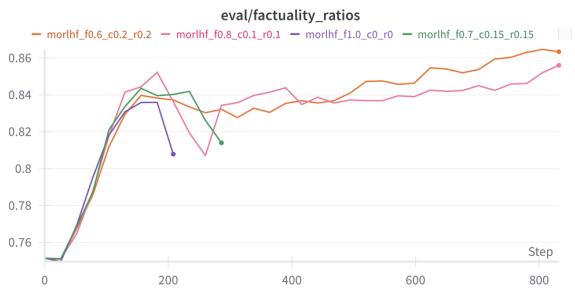
A.3.8 Comparision with MODPO

Regarding MODPO (Zhou et al., 2024), it is important to note that it is not an adaptive multi-objective generation method in the same sense as Bone Soup, MOD, RiC, or Rewarded Soup. MODPO requires re-training a model for each specific user preference, which is computationally expensive.

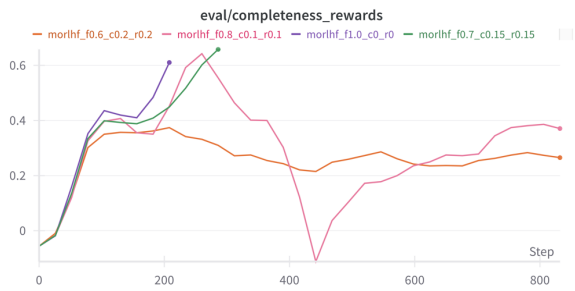
We also conduct additional experiments to include MODPO as a baseline for completeness. Since MODPO requires training for each preference, which incurs a huge overhead, we select values of $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ for the user preference $(\alpha, 1-\alpha)$. The experimental results are shown in Table 6 and Table 7:

As can be seen, Bone Soup consistently outperforms MODPO across various metrics. Although MODPO and Bone Soup do not share overlapping solutions, MODPO's Pareto front is significantly smaller, covering a more limited solution space. This limitation is also reflected in its poorer hypervolume performance.

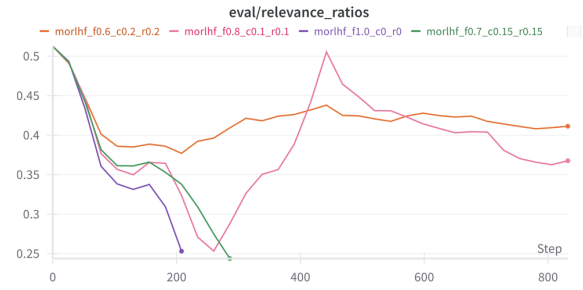
Based on our analysis, we suspect that the relatively weak performance of MODPO can be attributed to the lack of sufficient distinction of the reward in the dataset. As described in the original paper on MODPO, in addition to the margin reward, a second reward model is required to label the preference dataset, where the original "chosen" and "rejected" responses might have low distinction under the current reward model. This reduces the effectiveness of the DPO training process. On the other hand, Bone Soup, Rewarded Soup, and other similar methods utilize reinforcement learning to obtain diverse backbone models. The process of sampling responses in RL helps mitigate the issue of low distinction in rewards. This could be the



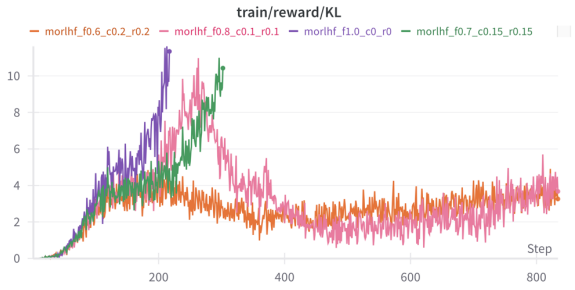
(a) Factuality rewards during PPO training



(b) Completeness rewards during PPO training

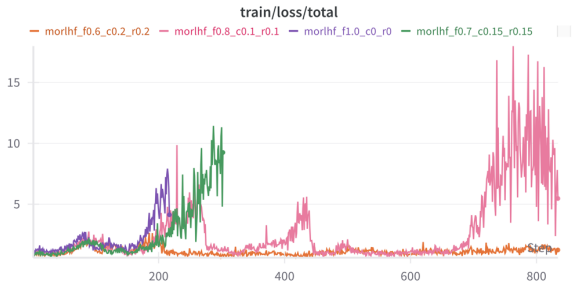


(c) Relevance rewards during PPO training



(d) KL divergence during PPO training

(e) Policy loss during PPO training



(f) Total loss during PPO training

Figure 10: The factuality rewards, completeness rewards, relevance rewards, KL divergence, and policy loss during PPO training. All the subfigures are variations of different metrics of factuality-specialized model.

reason why MODPO performs worse in our experiments.

Table 6: The comparison between Bone Soup and MODPO on Helpful vs Harmeless

Method	HyperVolume	Inner Product	Controllability
Bone Soup	**1.16**	**2.00**	**1.00**
MODPO	0.93	-0.57	1.00

Table 7: The comparison between Bone Soup and MODPO on Faituful vs Preference1

Method	HyperVolume	Inner Product	Controllability
Bone Soup	**1.12**	**1.89**	**1.00**
modpo	0.65	0.56	0.70

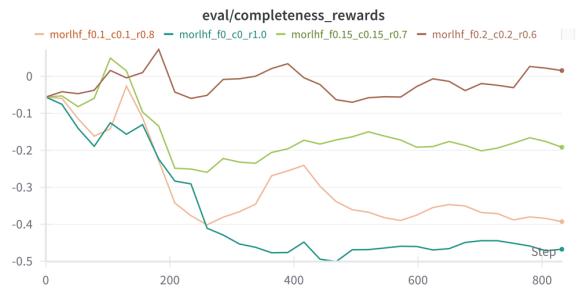
A.4 Experiments Setup Details

A.4.1 Baselines

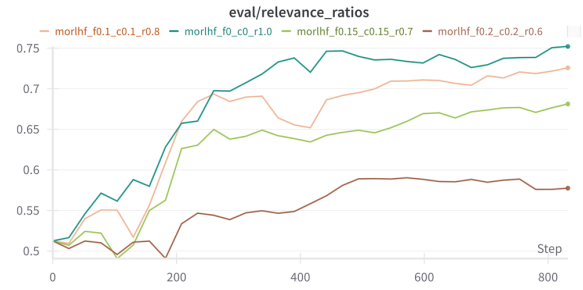
We first introduce and compare the three major categories of CMOG methods, followed by a detailed description of the baselines and experimental settings used in our study.

- Prompt-based methods: These require the LLM to understand fine-grained task descriptions encoded as numerical prompts (e.g., differences between 0.1 and 0.2). Achieving such precision through supervised fine-tuning is challenging, leading to poor controllability, as noted in Yang et al. (2024) and Table 2 in our paper.
- Decoding-based methods: For example, Shi et al. (2024) controls outputs by combining logits from multiple aligned models and there-

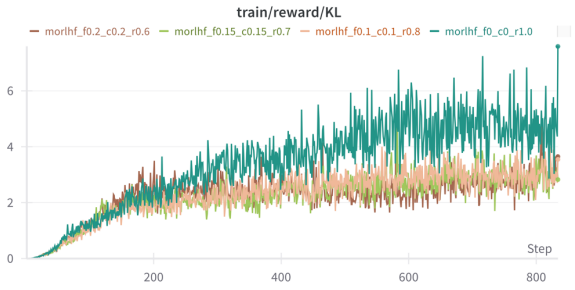
(a) Factuality rewards during PPO training



(b) Completeness rewards during PPO training



(c) Relevance rewards during PPO training



(d) KL divergence during PPO training

(e) Policy loss during PPO training

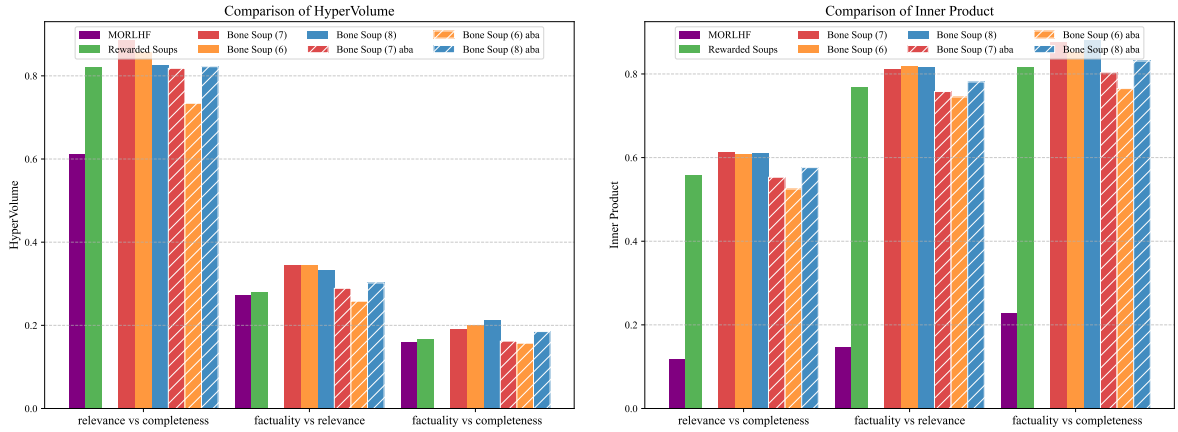
(f) KL divergence during PPO training of completeness-specialized model

Figure 11: The factuality rewards, completeness rewards, relevance rewards, KL divergence, and policy loss during PPO training. Subfigure (a), (b), (c), (d), and (e) depict the variations of different metrics during the PPO training process of the relevance-specialized model, while (f) shows the KL divergence changes during the PPO training process of the completeness-specialized model.

fore introduces additional inference time and memory overhead. We believe that the utilization of aligned models only at the logit level is not sufficient. As shown in Table 2 and Figure 5 in the paper, decoding-based methods (Shi et al., 2024) only improve upon naive merging-based methods(RS) marginally.

- Merging-based methods: Our method improves controllability and performance by adjusting parameters at a deeper level and leveraging RLHF models more comprehensively, therefore resulting in significantly better fronts. The detailed analysis can be found in Section 4.2.2.

RiC achieves control by adding multiple reward conditions in the prompt and aligns the model through two-stage supervised training. Rewarded Soups (RS) (Rame et al., 2023) trains specializing models separately for each reward and interpolates these models linearly. MOD (Shi et al., 2024) also needs to prepare multiple specializing models as RS did and achieve controllable alignment in decoding time by outputting the next token from a linear combination of predictions of all specializing models. For two objective setting, we utilize ten preferences $w_1 \in \{0.0, 0.1, \ldots, 1.0\}$ and $w_2 = 1 - w_1$. For the three-objective setting, we uniformly selected $\binom{12}{2} = 66$ points from the 3-simplex with a spacing of 0.1.



(a) HyperVolume for different coefficient determination approaches.

(b) Inner Product for different coefficient determination approaches.

Figure 12: Ablation of different approaches for coefficient determination.

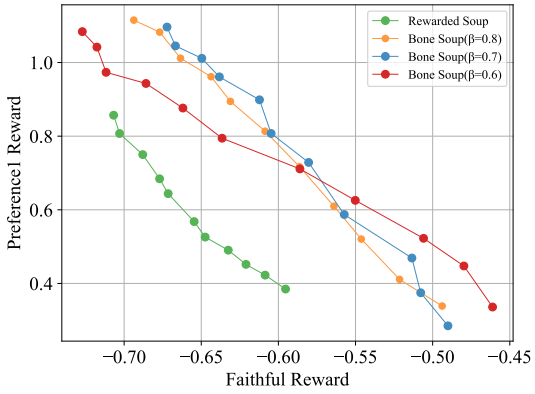


Figure 13: Ablation of the impacts of different β

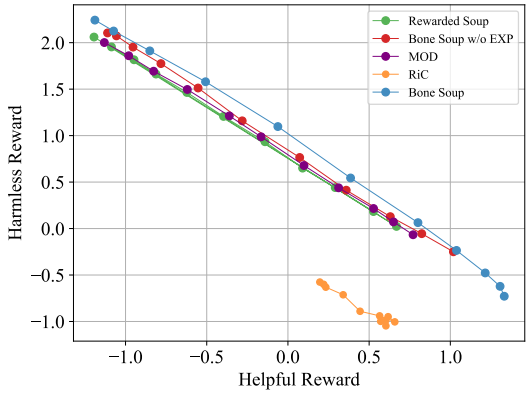


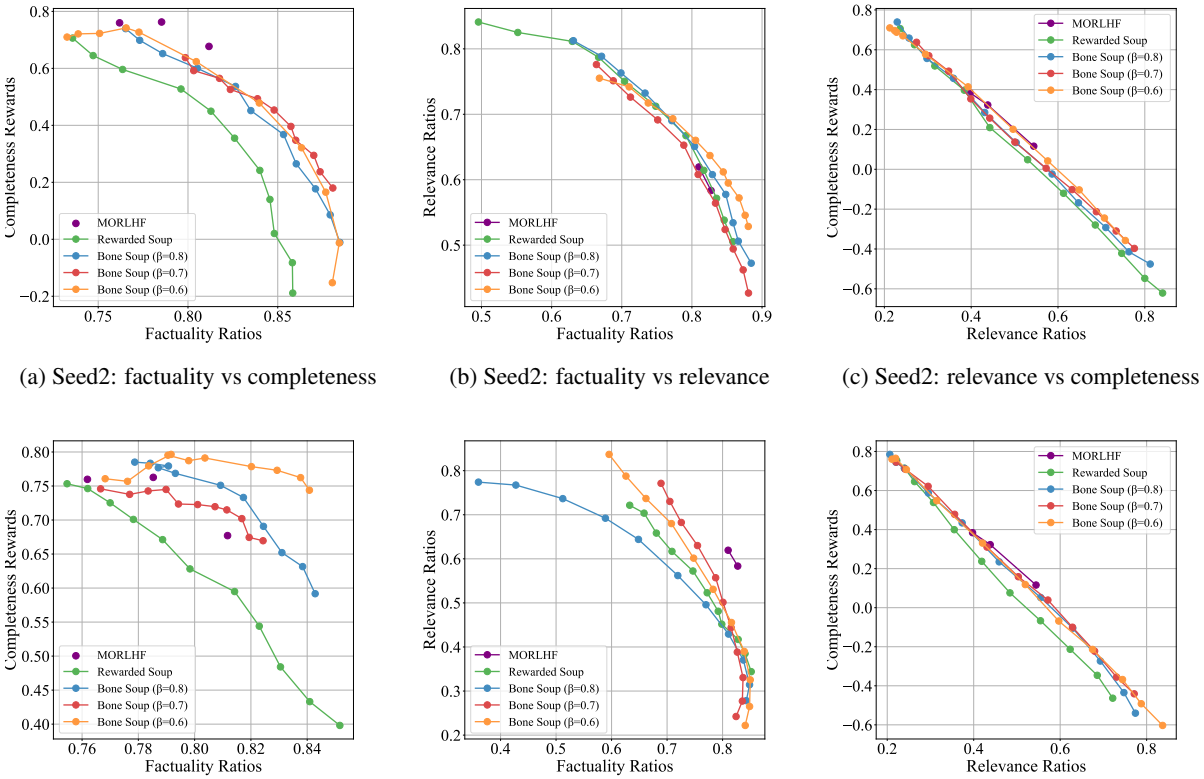
Figure 14: Ablation of Extrapolation

A.4.2 Long Form QA task

Long-form QA(Stelmakh et al., 2022; Min et al., 2020; Wu et al., 2024; Bhat et al., 2023; Huang et al., 2024) requires the model to generate a complete and comprehensive answer and explanation based on one or more given texts. Since questions often have multiple meanings and can easily cause ambiguity, the required answers need to be complete and multi-faceted.

The FineGrainedRLHF (Wu et al., 2024) dataset is obtained by reconstructing the ASQA(Stelmakh et al., 2022) dataset and collecting human feedback, which is publicly available under the Apache 2.0 License. Our use of the dataset is consistent with its intended use. It consists of 2,853 training examples and 500 development examples, forming "train_feedback.json" and "dev_feedback.json" respectively. Each example consists of a question corresponding to four model-predicted outputs sampled from the initial policy model. The feedback includes fine-grained feedback for the first model output and preference feedback for the four model outputs. In the original paper, the authors obtained 1,000 samples from the ASQA dataset to form "train_1k.json" for supervised training of the original policy model. We follow the setup of Wu et al. (2024), first performing supervised training on the initial policy model, and then using the reward models provided by Wu et al. (2024) to conduct PPO (Schulman et al., 2017) training.

Reward Models. Wu et al. (2024) provides rule-based reward models of three different granularities (**sub-sentence, sentence, full sequence**) based on error types. These reward models all use the encoder-only Longformer-base (Beltagy et al., 2020) as the backbone. Suppose the input format of the reward model is "question: q context: p_1 p_2



(a) Seed2: factuality vs completeness (b) Seed2: factuality vs relevance (c) Seed2: relevance vs completeness

(d) Seed3: factuality vs completeness (e) Seed3: factuality vs relevance (f) Seed3: relevance vs completeness

Figure 15: Ablation study on different random seeds for the Long-Form QA task, evaluating "factuality vs. relevance", "factuality vs. completeness", "relevance vs. completeness". We connect the points in the figure according to the order of the preference weight partial order relation. We varied the random seeds and observed that Bonesoup demonstrates strong robustness and consistently outperforms other baselines.

... answer: [sep] y_1^k [sep] y_2^k ...", where k represents different granularity levels corresponding to different rewards R_k——for example, the relevance reward corresponds to sub-sentence granularity. Wu et al. (2024) uses token-level classification loss to predict whether the segment of that granularity before each [sep] token contains errors corresponding to that reward. Therefore, the reward is set in a rule-based manner: for different rewards, it judges the different segments; if such an error exists, -1 is given at the [sep] position; otherwise, +1.

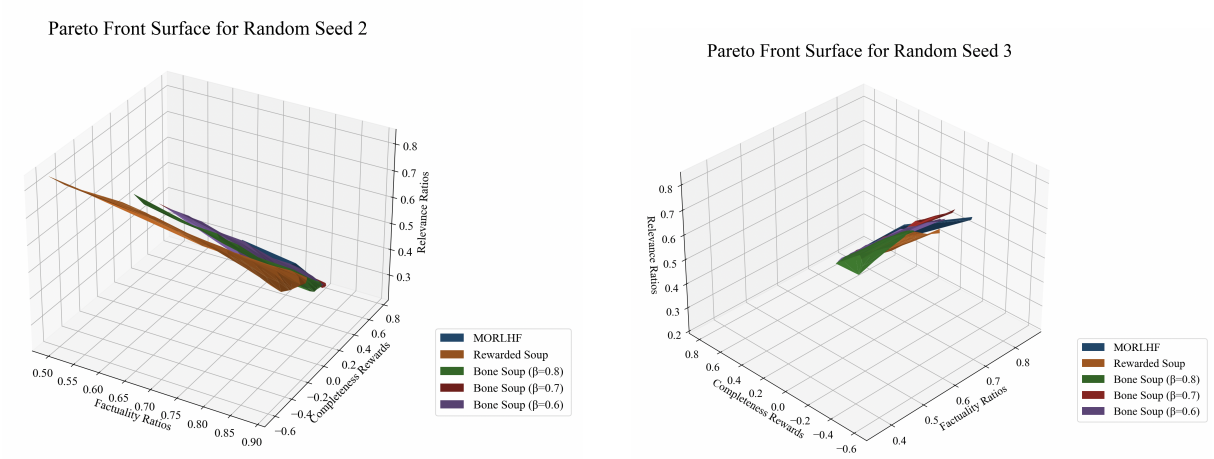
$\boldsymbol{R}_1$ **Relevance Reward**: . $\boldsymbol{R}_1$ is designed to predict whether there are errors such as irrelevance, repetition, or incoherence at the sub-sentence level. $\boldsymbol{R}_1$ gives a reward at each [sep] position; if there is no error, +1 is given at that position; otherwise, -1.

$\boldsymbol{R}_2$ **Factuality Reward**: $\boldsymbol{R}_2$ is designed to predict whether there are factual errors such as incorrect or unverifiable information at the sentence level. $\boldsymbol{R}_2$ gives a reward at each [sep] position; if there is no error, +1 is given at that position; otherwise, -1.

$\boldsymbol{R}_3$**: Completeness Reward**: $\boldsymbol{R}_3$ is designed to predict whether there are errors of information incompleteness at the full-sequence level. $\boldsymbol{R}_3$ gives a reward at each [sep] position; if there is no error, +1 is given at that position; otherwise, -1.

A.4.3 Helpful Assistant Task

The Helpful Assistant task requires the model to generate appropriate responses based on a given user conversation history, ensuring the response is as helpful as possible while maintaining safety. We use the hh-rlhf dataset for training and evaluation. The hh-rlhf dataset consists of 160k prompts, responses, and corresponding human annotations, which is publicly available under the MIT License. Our use of the dataset is consistent with its intended use. Additionally, we use three open-source reward models following Yang et al. (2024) to assess the helpfulness, harmlessness, and humor of the responses generated by the model. The backbones of first two reward models are GPT-2 model(Radford et al., 2019). The two reward models were trained



(a) factuality vs relevance vs completeness (seed 2)

(b) factuality vs relevance vs completeness (seed 3)

Figure 16: Ablation study on different random seeds for the Long-Form QA task, evaluating "factuality vs relevance vs. completeness"

on the Anthropic/hh-rlhf dataset using pair-wise feedback. The harmless reward model achieves a test set accuracy of 0.73698 and the helpful reward model achieves an accuracy of 0.72621 on the test set. The humor reward model is a fine-tuned version of distilbert-base-uncased (Sanh, 2019) on a joke/no-joke dataset to detect humor. And there is no extra prompt for this task.

A.4.4 Reddit Summary Task

The Reddit Summary task (Stiennon et al., 2020) focuses on summarizing Reddit posts, aiming to produce concise and coherent summaries that effectively capture the main content of the post. The dataset consists of Reddit threads, where the input comprises a post's title and body text, and the output is a human-written summary, which is publicly available under the MIT License. Our use of the dataset is consistent with its intended use. And the prefix of the prompt is "Generate a one-sentence summary of this post." according to (Yang et al., 2024). We use two open-source reward models (Yang et al., 2024) to assess the quality of the summary generated from two different aspects.

A.4.5 Parameters Setting Details

We list the values of parameters used in the experiment in Table 8 and Table 9 corresponding to Helpful Assitant, Reddit Summary Task, and Long Form QA task respectively.

A.5 Evaluation Metrics Details

Numerical metrics include **hypervolume indicator** (Zitzler et al., 2003), **Inner Product** (Zhong et al., 2024), **Sparsity(SP)** (Deb et al., 2002; Zhong et al., 2024), **Spacing** (Schott, 1995; Zhong et al., 2024), **controllability** and the **cardinality** of the Pareto front. **It is worth noting that a better front will also lead to greater Sparsity and Spacing since the coverage of that front is usually larger causing the bigger Sparsity and Spacing.** Therefore, we will use **HV, Inner Product and controllability** as our main metrics and other metrics are for reference when the main metrics are very close.

Since Controllability is the fundamental and most important aspect of implementing CMOG, we will formally define it below.

Definition 1 (Controllability)**.** *Controllability measures the degree to which the model's output aligns with the desired human preference $\boldsymbol{\mu}$. It is calculated by exhaustively enumerating all pairs of preferences $(\boldsymbol{\mu_i}, \boldsymbol{\mu_j})$, and checking if their relative order $(\mu_{i,k}, \mu_{j,k})$ matches the relative order of the evaluation (rewards r) $(r_k(\mathcal{S}_i), r_k(\mathcal{S}_j))$ in each dimension k where $\mathcal{S}_i$ is the model solution obtained by CMOG approaches corresponding to the preference $\boldsymbol{\mu}_i$. For each pair $(\boldsymbol{\mu_i}, \boldsymbol{\mu_j})$, if the above condition holds for all k dimensions, we increment the controllability score by 1. The final controllability score is normalized by dividing by the total number of solution pairs. The controllability score C is defined as:*

$$C = \frac{1}{N(N-1)} \sum_{i \neq j} \mathbf{1}[\bigwedge_{k=1}^{n} sign\left(\mu_{i,k} - \mu_{j,k}\right) = sign\left(r_k(\mathcal{S}_i) - r_k\left(\mathcal{S}_j\right)\right)], \tag{10}$$

where N is the total number of solutions, and $\mathbf{1}(\cdot)$ is the indicator function. The closer the score C is to 1, the better the controllability of the model.

1.Hypervolume

Hypervolume is a key performance indicator in multi-objective optimization, used to measure the volume of the space dominated by a set of solutions in the objective space. The hypervolume is defined as:

$$HV(S) = \text{Volume}\left(\bigcup_{i=1}^{n}[\mathbf{r}, \mathbf{f}(x_i)]\right) \tag{11}$$

where $[\mathbf{r}, \mathbf{f}(x_i)]$ represents the hyper-rectangle region between the reference point $\mathbf{r}$ and each solution $\mathbf{f}(x_i)$. Hypervolume is widely used to evaluate the performance of multi-objective optimization algorithms. A larger hypervolume indicates a better coverage of the objective space.

2.Inner Product

The inner product between the preference vector and the corresponding reward vector serves as a metric for measuring their correspondence. From another perspective, this can be interpreted as a weighted sum of rewards, where the preference vector reflects the emphasis on different reward components.

Mathematically, this can be expressed as:

$$IP(\mu, \mathbf{r}) = \sum_{i=1}^{n} \mu_i \cdot r_i \tag{12}$$

where $\mu = [\mu_1, \mu_2, \ldots, \mu_n]$ is the preference vector and $\mathbf{r} = [r_1, r_2, \ldots, r_n]$ is the corresponding reward vector. The inner product quantifies the alignment between the preference and reward, with higher values indicating stronger alignment between them.

3.Sparsity

Sparsity measures the variation between solutions corresponding to the consecutive preference vectors (Deb et al., 2002; Zhong et al., 2024). It is defined as the average squared Euclidean distance between adjacent vectors. A smaller sparsity value indicates smoother transitions between successive rewards, which is desirable in our context. **However, due to the huge cost of evaluating solutions, we can only obtain a limited number of solutions and therefore the evaluation of Sparsity is less convincing.**

$$\text{Sparsity} = \frac{1}{n-1}\|\mathbf{r}_i - \mathbf{r}_{i-1}\|^2 \tag{13}$$

4.Spacing

We follow Zhong et al. (2024) to introduce the Spacing metric to evaluate the front. The Spacing metric evaluates the variance of the minimum distance between solutions(corresponding reward vectors). Lower values indicate a better Pareto front but with the same limitations as Sparsity.

$$\text{Spacing} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i - p)^2}, \tag{14}$$

where $d_i = \min\{||r_i - r_j||\}$ and $p = \frac{1}{N}\sum_{i=1}^{N} p_i$.

A.6 Discussions

Computational Complexity Analysis The primary computational cost of our approach occurs during the training phase. Once training is completed, the inference stage can adapt to any user preference without additional overhead. Unlike MOD (Shi et al., 2024) and MODPO (Zhou et al., 2024), there is no extra cost during inference. The process of determining merging coefficients only requires solving a linear equation, which incurs no additional computational expense.

On the other hand, although we propose using a small-scale selection to choose the optimal $\beta \in \{0.8, 0.7, 0.6\}$, the robustness of the constructed matrix (see Appenidx A.3.3) means we are not overly dependent on β and can get rids of this additional expense. In Appendix A.3.6, we discuss the impact of varying β on Bone Soup, and the conclusion is that, regardless of the β value, the resulting Pareto front consistently outperforms and dominates other baselines. Our insight is that including marginal reward could lead to better backbone construction. Therefore, the simplest approach is to select any β, which results in **computational costs identical to those of standard Rewarded Soup, with no additional overhead**. If one aims to achieve the optimal performance, let the computational cost of Rewarded Soup be C. Then,

selecting optimal β introduces an additional cost of $0.2 \times C \times n$, where $n \leqslant 3$ is the number of possible choices for β and as the small-scale selection trained the model with 20% total steps.

Negative Merging Coefficients.

Solving the linear system may result in negative values, which could lead to negative interpolation or said extrapolation (Ilharco et al., 2022; Zheng et al., 2024) of the backbone models.

Previous works have discussed improving model performance through extrapolation techniques (Zheng et al., 2024), or by deliberately weakening the initial SFT model to facilitate unlearning (Ilharco et al., 2022). In these approaches, negative merging coefficients were determined through trial-and-error methods using a validation set.

However, in our approach, we avoid the cumbersome and somewhat unnatural trial-and-error process by directly solving linear equations to establish a clear mapping between the backbone models' rewards and user preferences. This not only simplifies the process but can also be regarded as an interpretable extrapolation. Our method Bone Soup of constructing non-orthogonal bases and then performing interpolation can thus be seen as a form of extrapolation. Unlike the naive merging method, where coefficients are constrained to be nonnegative, the presence of negative coefficients extends beyond the original model space. More importantly, the ability to interpret the negative coefficients selected offers the potential to improve performance in regions that would otherwise be inaccessible through standard interpolation techniques.

Model Merging in Multi-Objective and Multi-Task Setting. Here, we emphasize the distinction between obtaining a *Pareto front* and *single model*. Most current research (Wortsman et al., 2022; Rame et al., 2023; Tang et al., 2024; Yu et al., 2024; Yadav et al., 2024; Yang et al., 2023; Wang et al., 2024b; Ilharco et al., 2022) primarily focuses on obtaining a single model that, through merging, possesses the capabilities of multiple models. This approach works in multi-task scenarios because the interference between various tasks is present but often not strong enough to pose significant challenges.

However, in multi-objective optimization, numerous objectives are inherently conflicting or compromised. For instance, in QA tasks, relevance and completeness are often at odds (Wu et al., 2024): a complete answer is likely to include some irrelevant content, while a highly relevant answer may be too narrow, resulting in incomplete responses. Similarly, in typical alignment tasks, objectives like helpfulness and harmlessness (Dai et al., 2023; Bai et al., 2022; Ganguli et al., 2022) frequently conflict, making it difficult to achieve both fully. In such cases, it is preferable to aim for a Pareto front, where the points on the front represent non-dominated and optimal solutions. Our goal is not only to find this front but to ensure it is as expansive as possible, with widely dispersed points, thereby covering a broad range of trade-offs between competing objectives.

Model and Task Settings	
Model	LLaMA-2-7B
Helpful Assistant Task	128 tokens
Reddit Summary Task	48 tokens
LoRA Settings	
Rank	64
Alpha	128
LoRA Dropout	0.05
SFT Training	
Fine-tune Steps	60k steps
Initial Learning Rate	1.41e-4
Learning Rate Decay	Linear
RL Training (PPO)	
Implementation	trl 0.8
Initial KL Penalty Coefficient	0.2
Learning Rate	1e-5
GAE Lambda	0.95
Discount Factor (Gamma)	1
Clip Range	0.2
Maximum Gradient Norm	0.5
Sampling Strategy	nucleus sampling, top_p=0.1
Sampling Temperature	0.7
Target KL Early Stop	5
Training Epochs	1
Batch Size	64
Mini Batch Size	1
Optimization Epochs per Batch	4

Table 8: Helpful Assistant and Reddit Summary Task Settings and Parrameters

Model and Task Settings	
Model	T5-large
Input Length Limit	1024
Max Generation Tokens	200
Data Splits	
Dataset	QA-FEEDBACK
Splits	SFT, Train, Test
SFT Training Data	1000 examples
PPO Training Data	2853 examples
Test Data	500 examples
SFT Training	
Epochs	10
Batch Size	32
Learning Rate	5e-5
PPO Training Parameters	
Episodes	80,000
PPO Epoch per Rollout	4
Initial Learning Rate	1e-5
Learning Rate Decay	Linear
Early Stop KL Threshold	10
GAE Lambda	0.95
KL Coefficient	0.3
Discount Factor (Gamma)	1
Clip Range	0.2
Sampling Strategy	Top-k sampling(k=20)
Temperature	0.7
LoRA Settings	
Rank	32
Alpha	32

Table 9: Long Form QA Task Settings and Parameters

System Prompt:
You are a impartial judge for checking the quality of the answer.

User Prompt:
[System]
We kindly request your feedback on the performance of two AI assistants in response to the user question presented below. Act as an impartial judge and evaluate only the factuality of the response provided by each assistant. Rate each assistant on a scale of 1 to 10, where a higher score signifies a more factually accurate response. Try to avoid giving the same score.

Your evaluation should focus solely on the factual accuracy of the response. When assessing factuality, please check whether the response is consistent with the provided context, whether it is correct, and whether the information is verifiable. A higher score should reflect better adherence to facts and context.

The question and answers are as follows:

[Question]
{question}

[The Start of Assistant 1's Answer]
{answer1}

[The End of Assistant 1's Answer]

[The Start of Assistant 2's Answer]
{answer2}

[The End of Assistant 2's Answer]

[System]
Start by outputting a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores should be separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, ensuring that the order in which the responses were presented does not influence your judgment.

[Answer]

Figure 17: Prompt template for GPT-4 to evaluate Factuality.

System Prompt:
You are a impartial judge for checking the quality of the answer.

User Prompt:
[System]
We kindly request your feedback on the performance of two AI assistants in response to the user question presented below. Act as an impartial judge and evaluate only the relevance of the response provided by each assistant. Rate each assistant on a scale of 1 to 10, where a higher score signifies a more relevant response. Try to avoid giving the same score.

Your evaluation should focus solely on whether the response is relevant to the context and the question, whether it is logically coherent, and whether it is concise and to the point. When assessing relevance, please check if the response directly answers the question, aligns well with the provided context, and avoids unnecessary or off-topic information.

The question and answers are as follows:

[Question]
{question}

[The Start of Assistant 1's Answer]
{answer1}

[The End of Assistant 1's Answer]

[The Start of Assistant 2's Answer]
{answer2}

[The End of Assistant 2's Answer]

[System]
Start by outputting a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores should be separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, ensuring that the order in which the responses were presented does not influence your judgment.

[Answer]

Figure 18: Prompt template for GPT-4 to evaluate Relevance.