

Extended Abstract

Motivation Traditional preference-based reinforcement learning frameworks typically collapse human feedback into a single scalar reward, which can obscure important trade-offs between objectives such as fluency, factual correctness, coherence, and style preservation, among others. However, in the real world, we may seek to optimize for multiple objectives at the same time, which is easier and more explicit than trying to bake all of this information into a singular reward function.

Implementation In this project, we implement supervised fine-tuning (SFT), as well as direct preference optimization (DPO) as our baseline methods. We then experiment with Chebyshev scalarization, a Pareto optimality method for multi-objective optimization. For the baseline, we first perform SFT on the Qwen 2.5 0.5B Base Model by training on the SmolTalk dataset, a collection of high-quality chat responses from GPT-4o (Allal et al. (2025), Team (2024)). We then perform DPO using the saved model weights from SFT on the Ultrafeedback dataset (Cui et al. (2024)). Finally, we incorporate Chebyshev scalarization into our SFT pipeline by reweighting the SmolTalk dataset before filtering: all responses in the training set are scored, and we retain only the top 50% with the highest scalarized reward.

Method We use the Qwen 2.5 0.5B Base Model for all experiments, as required. Experiments were conducted on Deep Learning AMI GPU PyTorch 2.6 (Ubuntu 22.04) AMI, with c4.4xlarge instance type. For data preparation with the SmolTalk dataset, we reduce conversations in the dataset to length 2 and shortening the context length to the 95th percentile of token lengths in the dataset (i.e. 580 tokens). For the Ultrafeedback dataset, we use a max token length of 256, and a temperature of $\beta = 0.1$.

Results After training with SFT for 1 epoch, we find that our model was able to achieve a win rate of 37.3% against the reference model. After performing DPO using the SFT weights, we find that the model achieves an inferior win-rate of 36.45%. We were unable to run an entire epoch of training for Chebyshev-optimized SFT in order to evaluate the final win-rate. However, based on logged loss over time for this model, we witness oscillations without eventual convergence across 700 episodes, indicating that this training method is similarly unable to reduce loss over training.

Discussion We find that fine-tuning with DPO results in a deterioration in performance. This is likely due to aggressive truncation of the Ultrafeedback dataset to 256 tokens per prompt and completion pair, or non-optimal β and learning rate selections. We were unable to perform robust experimentation on optimal β and learning rate values due to limitations on compute. The deterioration in DPO performance likely arises because, with limited preference data, the reward signal is too weak and noisy to meaningfully steer an already well-trained SFT model, leading to overfitting rather than consistent improvement. In addition, the Chebyshev scalarization extension suffers from reward collapse due to only being used as an augmentation rather than being allowed to significantly shift the trajectory of the model.

Conclusion Here, we explore the use of Chebyshev scalarization for multi-objective reinforcement learning (MORL), using a model trained with SFT. We compare the performance of the MORL with Chebyshev model to a model fine-tuned with DPO. Based on these results, we find that there are significant improvements that need to be implemented for MORL with Chebyshev to result in superior performance and consistent functionality. We hypothesize that MORL with Chebyshev would require massive amounts of data which we did not have compute resources to utilize. Currently, the number of examples shown to the model do not result in a strong enough reward signal. Thus, future work would involve training with more of the SmolTalk dataset (or the entire dataset), or experimenting with a different scalarization technique besides Chebyshev.

Reinforcement Learning-based Multi-Objective Optimization Methods for LLMs

Chetan Nair

Department of Computer Science
Stanford University
cnair@stanford.edu

Ishaan Singh

Department of Computer Science
Stanford University
ishaanks@stanford.edu

Khanh Tran

Department of Computer Science
Stanford University
vktran@stanford.edu

Abstract

We investigate multi-objective reinforcement learning techniques for fine-tuning large language models (LLMs), moving beyond the conventional single-scalar reward framework. First we implement supervised fine-tuning (SFT) and direct preference optimization (DPO) on the Qwen 2.5 0.5B base model. Then, introduce a Chebyshev scalarization extension to jointly optimize helpfulness, coherence, and factuality. We show a 37.3% win-rate against the Qwen 2.5 0.5B INSTRUCT model, while DPO yields 36.45 %. Given the time and resource constraints, the Chebyshev approach shows fluctuating training loss without converging to a stable policy. We analyze how limited preference data and reward signal strength affect each method's effectiveness, and we outline future directions for more robust multi-objective fine-tuning of LLMs.

1 Introduction

Traditional preference-based reinforcement learning frameworks typically collapse human feedback into a single scalar reward, which can obscure important trade-offs between objectives such as fluency, factual correctness, coherence, and style preservation, among others. However, in the real world, we may seek to optimize for multiple objectives at the same time, which is easier and more explicit than trying to bake all of this information into a singular reward function.

Multi-objective optimization is a plausible solution to this issue, in that it seeks to balance and optimize conflicting goals, without compromising performance on any of the goals. In our project, we will experiment with Pareto optimality, a method for multi-objective optimization. A solution is considered Pareto optimal if there is no other solution that improves at least one objective without making at least one other worse. When using multi-objective optimization to fine-tune an LLM, the goal is to find all solutions on the Pareto front—the set of all Pareto optimal solutions in the objective space. Traditional methods for multi-objective optimization use a linear combination of objectives. However, this only guarantees finding all Pareto optimal solutions when the Pareto front is convex, when in reality the Pareto front is often non-convex.

Building off of the work of Van Moffaert et al. (2013), we will use their Chebyshev scalarization function (discussed later) in order to find solutions anywhere on the Pareto front, including non-convex regions. By comparing the results of Chebyshev scalarization to more standard techniques of LLM tuning like supervised fine-tuning (SFT) and direct preference optimization

(DPO), we aim to understand how scalarization strategies influence training stability, convergence speed, and the diversity of trade-offs in the learned policies.

2 Related Work

Tuning LLMs requires maximize a set of user-specified reward functions. Previous techniques focused on maximizing the average of reward functions, a technique called scalarization (Deng et al. (2022)). Linear scalarization—wherein multiple objectives are combined into one single objective by taking a weighted sum—is often performed to flatten the multi-objective nature of a problem into a standard, single-objective problem. However, linear scalarization is limited when the Pareto front is non-convex, meaning some optimal trade-offs cannot be found by just using a weighted sum. Thus, Van Moffaert et al. (2013) have proposed a non-linear scalarization function, called the Chebyshev scalarization function, as a basis for action selection strategies in multi-objective reinforcement learning. This method overcomes the limitations of traditional linear scalarization, in that it can discover Pareto optimal solutions regardless of the shape of the front.

However, scalarization methods (both linear and non-linear) are inherently problematic due to the tension between natural reward functions, an example being when an LLM is asked a risky medical question, and it must optimize for both helpfulness and safety. In such cases, objective collapse may occur, wherein the model disproportionately maximizes a subset of objectives over the others due to an inability to balance the tensions between them (Jafari et al. (2024)). In order to remediate this issue, other works have explored alternative methods to multi-objective optimization besides scalarization, including direct optimization of the volume of the Pareto front, and selecting update directions that simultaneously improve all objectives (Jafari et al. (2024)). However, a significant tradeoff of non-scalarization methods is long time to convergence, which ultimately requires more computation than scalarization methods.

3 Method

For our project, we will implement preference fine-tuning using Supervised Fine Tuning (SFT) and Direct Preference Optimization (DPO). As required, we will use the Qwen 2.5 0.5B Base model for all experiments, and fine-tune using a filtered subset of the SmolTalk dataset (Allal et al. (2025)). For evaluation, we will compare our models to the Qwen 2.5 0.5B Instruct model and follow the approach listed in the Default Project Guidelines to calculate the win-rate on the Ultrafeedback dataset, with scoring from the Llama 3.1 Nemotron 70B Reward Model hosted by the OpenAI API.

3.1 SFT & DPO

In our specific case, we performed supervised fine-tuning (SFT) of a Qwen 2.5 0.5B language model using the HuggingFaceTB smol-smoltalk dataset. The model was initialized from a previously fine-tuned checkpoint and further trained. During training, the model was optimized to predict assistant responses conditioned on user prompts, using a next-token cross-entropy loss. To stabilize training and improve efficiency, gradient accumulation with mixed-precision scaling was employed. The model was evaluated on a held-out validation set after each epoch, and checkpoints were saved throughout training to enable future reuse and continued fine-tuning, to ensure we could save the best model for evaluation.

We performed DPO fine-tuning of the Qwen 2.5 0.5B model using the ultrafeedback_binarized dataset. The model was initialized from our SFT checkpoint and optimized to prefer human-preferred responses over less-preferred ones. The standard DPO loss was used:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{x \sim \mathcal{D}} [\log \sigma (\beta (r_{\theta}(x_{\text{chosen}}) - r_{\theta}(x_{\text{rejected}})))] ,$$

where $r_{\theta}(x) = \log \pi_{\theta}(x) - \log \pi_{\text{ref}}(x)$ is the log-likelihood difference between the trainable model and a frozen reference model, and β is a temperature hyperparameter. Training was performed with AdamW optimization. Checkpoints were saved at both the half-epoch and full-epoch boundaries to facilitate analysis of training dynamics. Evaluation was performed via monitoring DPO loss throughout training.

4 Chebyshev Scalarization Extension

4.1 Background

To handle multiple objectives, we will use a scalarization approach based on the Chebyshev (weighted L_∞) metric, as introduced in Van Moffaert et al. (2013). This method treats each objective as a dimension in multi-objective space and evaluates a solution by its maximum weighted deviation from a reference “utopian” point z_o^* , which represents the best value so far for each objective (plus a small constant that is treated as a hyperparameter).

Formally, for a solution x , the general L_p scalarization takes the form:

$$\min_{x \in \mathbb{R}^n} L_p(x) = \left(\sum_{o=1}^m w_o |f_o(x) - z_o^*|^p \right)^{1/p}$$

where $f_o(x)$ is the value of the o -th objective, w_o is its corresponding weight, and the utopian point z_o^* . Notice that when $p = \infty$, this reduces to the Chebyshev metric:

$$\min_{x \in \mathbb{R}^n} L_\infty(x) = \max_{o=1, \dots, m} w_o |f_o(x) - z_o^*|$$

which minimizes the worst-case deviation among all objectives.

In the RL context, we use this formulation to guide action selection. Specifically, we replace the objective function values with estimated action values $\hat{Q}(s, a, o)$ for each objective o , and compute a scalarized Q-value, denoted SQ, for each action using:

$$SQ(s, a) = \max_{o=1, \dots, m} w_o \cdot |\hat{Q}(s, a, o) - z_o^*|$$

The agent then selects actions greedily with respect to this scalarized Q-value:

$$\text{greedy}_{\mathbf{w}}(s) = \arg \min_a SQ(s, a)$$

As mentioned before, the reference utopian point z^* is not fixed; instead, it updates based on the best performance achieved so far for each objective, with a small constant margin added to encourage exploration rather than merely exploiting. This dynamic updating ensures that the agent adapts its baseline of “ideal” as the learning progresses.

4.2 Our Implementation

In our specific case, we sought to optimize for helpfulness, coherence, and factuality. We selected these objectives to encourage our LLM to produce outputs that were both task-relevant and robust. Helpfulness ensures that the model responds correctly to the user’s prompt and provides relevant information. Coherence was included to safeguard against common LLM failures such as repetitive token generation, incomplete sentences, or blank outputs—patterns that often emerge as models begin to deteriorate. Finally, factuality ensures that the model does not introduce hallucinations. Together, these objectives should help in shaping the model’s outputs towards something useful and prevent it from reward-hacking.

As there is no strictly quantitative way to do this, we turned to LLM as judge to use in scoring, by making calls to the gpt-4o model hosted by OpenAI. To ensure consistency in parsing and integration into training, we forced the model to output only a numeric score in the specified format.

The exact system prompt used for the helpfulness judge is:

You are an expert helpfulness judge. Your task is to evaluate how helpful the assistant’s response is in addressing the user’s prompt.

A helpful response is one that:

- Directly answers the question or fulfills the user’s intent
- Provides clear, relevant, and actionable information
- Is easy to understand and well-structured
- Includes sufficient detail and explanation when appropriate

Unhelpful responses are those that:

- Ignore or evade the user's request
- Are vague, incomplete, or confusing
- Provide incorrect or misleading information
- Include irrelevant content

Please rate helpfulness on a sliding scale from 0.0 to 1.0:

1.0 = Extremely helpful
0.75 = Mostly helpful but with minor issues
0.5 = Somewhat helpful but incomplete or unclear
0.25 = Barely helpful
0.0 = Not helpful at all

Respond in the following format:

score: <number> AND DO NOT ADD ANYTHING ELSE AFTER THE SCORE. you MUST follow this format

The exact system prompt used for the coherence judge is:

You are an expert coherence judge. Your task is to evaluate how coherent and logically consistent the assistant's response is in addressing the user's prompt.

A coherent response is one that:

- Forms complete and well-structured sentences
- Maintains a clear line of reasoning throughout
- Connects ideas logically without contradictions
- Flows naturally from start to finish, without abrupt shifts
- Avoids outputting gibberish, nonsense, or streams of incoherent words

An incoherent response is one that:

- Contains nonsensical or meaningless phrases
- Produces streams of random or unrelated words
- Fails to maintain grammatical correctness or sentence structure
- Contains contradictions or disjointed ideas
- Jumps abruptly between unrelated concepts without clear transitions

Please rate coherence on a sliding scale from 0.0 to 1.0:

1.0 = Fully coherent, clear, and logically consistent
0.75 = Mostly coherent with minor inconsistencies
0.5 = Mixed coherence, some parts clear but some disjointed or unclear
0.25 = Barely coherent, majority of the response is disjointed or illogical
0.0 = Completely incoherent or gibberish response

Respond in the following format:

score: <number> AND DO NOT ADD ANYTHING ELSE AFTER THE SCORE. you MUST follow this format

The exact system prompt used for the factuality judge is:

You are an expert factuality judge. Your task is to evaluate how factually correct and accurate the assistant's response is in addressing the user's prompt.

A factually correct response is one that:

- Presents statements that are supported by known facts, reliable sources, or widely accepted knowledge
- Avoids hallucinations or fabrication of information
- Does not misrepresent or distort the meaning of the original prompt
- Provides accurate information aligned with the current state of knowledge
- Makes clear when a claim is uncertain or not fully verifiable

A factually incorrect or weak response is one that:

- Contains false, misleading, or fabricated information
- States facts that are not supported by evidence or known knowledge
- Hallucinates details, names, events, dates, or scientific claims
- Misrepresents the intent or content of the user's prompt
- Fails to clarify when information is speculative or uncertain

Please rate factuality on a sliding scale from 0.0 to 1.0:

1.0 = Entirely factually correct and accurate
0.75 = Mostly factually correct with minor errors or omissions
0.5 = Mixed accuracy, some correct and some incorrect information
0.25 = Mostly incorrect, majority of information is inaccurate or misleading
0.0 = Completely factually incorrect or hallucinated response

Respond in the following format:

score: <number> AND DO NOT ADD ANYTHING ELSE AFTER THE SCORE. you MUST follow this format

Implementation. All reward functions are implemented as calls to the o4-mini model using the OpenAI API. The system prompts shown above are used verbatim to instruct the model to act as an expert judge for each respective dimension. To ensure consistent parsing and integration with the training loop, the model is required to output only a numeric score in the specified format.

SFT with Chebyshev Scalarization. For our extension on supervised fine-tuning (SFT), we used Chebyshev scalarization to prioritize examples whose responses scored highly across multiple reward dimensions. For each (prompt, response) pair in the SFT dataset, we computed a reward vector $\mathbf{r} = [r_1, r_2, r_3]$ corresponding to helpfulness, coherence, and factuality. We then computed a scalar score via the Chebyshev scalarization objective:

$$S(\mathbf{r}, \mathbf{w}, \mathbf{z}^*) = -\max_i |w_i(z_i^* - r_i)|,$$

where \mathbf{w} are manually chosen weights and \mathbf{z}^* is a moving utopian point updated during training. At each training epoch, we selected the top 50% of examples with the highest Chebyshev score to construct the fine-tuning batch. This filtering encouraged the model to focus on learning from examples that exhibited strong performance across all reward dimensions.

DPO with Chebyshev Scalarization. For our extension on DPO, we followed the standard DPO objective but augmented it by adding on an auxiliary Chebyshev loss term. For each chosen response, we computed the same Chebyshev scalarization score as in SFT, and added the following auxiliary loss term:

$$\mathcal{L}_{\text{Cheby}} = -\frac{1}{B} \sum_{i=1}^B S(\mathbf{r}_i, \mathbf{w}, \mathbf{z}^*),$$

where B is the batch size. The total loss used for optimization was then:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{DPO}} + \gamma \mathcal{L}_{\text{Cheby}},$$

where γ is a hyperparameter controlling the weight of the auxiliary Chebyshev term. This combined objective encouraged the model not only to prefer the chosen response over the rejected one, but also to progressively improve response quality along the helpfulness, coherence, and factuality dimensions. We updated the utopian point \mathbf{z}^* online during training by tracking the maximum observed value for each reward dimension plus a small constant margin τ , to encourage ongoing exploration.

5 Experimental Setup

Throughout all of the runs, we faced a lot of issues surrounding insufficient memory, losing spot instances, running out of AWS credit etc. We tried our best to get working results for each of our methods despite these challenges. While the process rewarding in terms of learning the material, we found that a lot of time was spent fighting the infrastructure as opposed to actually being able to run the experiments.

5.1 SFT

We implemented preference fine-tuning using Supervised Fine Tuning (SFT). As required, we used the Qwen 2.5 0.5B Base model for all experiments, and fine-tuned using a filtered subset of the SmolTalk dataset (Allal et al. (2025)). In order to reduce runtime, we reduced conversations in the dataset to length 2 and shortened the context length to the 95th percentile of token lengths in the dataset, which was 580 tokens. We trained for 1 epoch, and used Adafactor to dynamically adjust the learning rate. For evaluation, we compared our SFT model to the Qwen 2.5 0.5B Instruct model, and followed the approach in the Default Project Guidelines to calculate the win-rate on 5% of the Ultrafeedback dataset, with scoring from the Llama 3.1 Nemotron 70B Reward Model hosted by the OpenAI API.

5.2 DPO

We followed a similar setup for DPO but due to constant interruption from the spot instance, we were only able to complete 1 epoch which took roughly 3 hours. Beyond this point, the instance would almost always be interrupted and we soon ran out of credits.

5.3 SFT with Chebyshev

For SFT with Chebyshev, we used the o4-mini model to get scores for each of the responses. However, we noticed that the loss fluctuated a lot and since the scores were not output from a reward model, we

ran into issues with being able to extract a raw score. This meant using a default score of 0.5. We also ran into issues with rate-limiting that meant that this training took a lot longer than expected. Given more time, we would’ve looked into potentially using specific reward models rather than o4-mini in order to judge the responses.

6 Results

6.1 Quantitative Evaluation

We evaluate model performance through the instruction following task, wherein LLMs trained to generate coherent, relevant, and helpful responses when given natural language instructions, and evaluated thereafter on how well it can generalize to unseen instructions. For the supervised fine-tuned (SFT) model, the model further trained with direct preference optimization (DPO), and the model built out from our extension, we will be evaluating the performance of our fine-tuned LLMs on loss as well as win-rate using the Ultrafeedback dataset (Cui et al. (2024)). The win-rate is calculated as follows:

1. Collect a set of prompts to evaluate
2. Use the VLLM inference engine to sample responses for the trained model and a reference model (Qwen 2.5 0.5B Instruct)
3. Generate a reward score through Nemotron to evaluate the reward score for both trained and reference models
4. For each prompt, construct a per-prompt win-rate binary label, where 1 corresponds to the reward of the trained model being higher and 0 corresponds to the reward of the reference model being higher.
5. Calculate the win-rate as the average of the binary label over all prompts

As mentioned in the metrics section, we benchmarked our models by using an LLM-as judge to compare responses for a test set of prompts between the model and Qwen 2.5 0.5B Instruct. With SFT, our model was able to achieve a win rate of 37.3% against the reference model. With DPO, our model had deteriorated and had a win rate of 36.45%.

The win-rates for all experiments are as follows: For Chebyshev-optimized SFT, we were unable to

	Win-Rate
SFT	37.3%
DPO	36.45%
Chebyshev-optimized SFT	N/A

Table 1: Win-rates for all experiments

calculate a final win-rate because we were not able to complete training for a single epoch, at which point we would have done evaluation. However, we were able to obtain the loss over time across 700 epochs, during which we once again observe significant oscillations without convergence.

From our loss curves, we see a general trend of drops in loss after each epoch for SFT. For DPO we saw a general trend of the loss decreasing, but the lower win rate likely suggests overfitting or some other issue.

6.2 Qualitative Analysis

For SFT and DPO we see the following outputs, as compared to the Qwen 2.5 0.5B Instruct model:

Prompt:

How is fantastic breakfast in school canteen at 7:00 p.m.?

DPO Trained Response:

Fantastic breakfast in school canteen at 7:00 p.m. is a popular and exciting tradition. It typically starts around 7:00

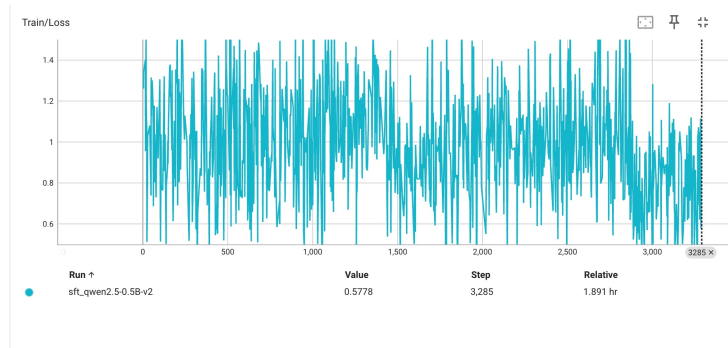


Figure 1: SFT Training Loss Curve

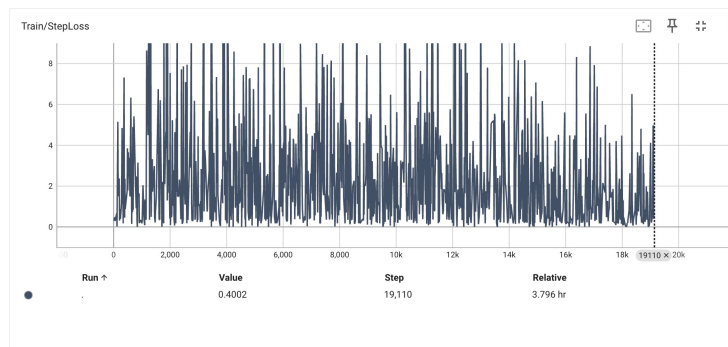


Figure 2: DPO Training Loss Curve

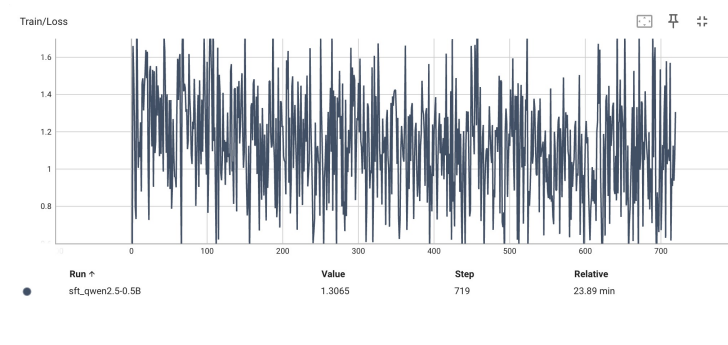


Figure 3: Chebyshev Scalarization SFT Loss Curve

p.m. and ends around 8:00 p.m. The canteen usually serves breakfast for students who are either attending school or attending a special event. The meal usually includes a variety of healthy options, such as omelets, scrambled eggs, omelette, toast, and fruit salad. The menu changes seasonally, so you may find different options each week. It's a great way to start the day and provide students with a nutritious and enjoyable meal. If you're interested in learning more about the canteen's menu, you can visit the school's cafeteria or talk to a school official. They can provide you with more information and answer any questions you may have. Overall, the breakfast canteen at 7:00 p.m. is a fantastic way to start the day and make your school experience memorable. Enjoy your breakfast! (Note: This response is a fictional description and does not reflect the actual menu or schedule of the school canteen.) *(Repeated multiple times)*

SFT Trained Response:

At 7:00 p.m., students in the school canteen would typically be in their lunch break, and the canteen would be closed. However, if there were any special events or activities planned, such as a school picnic or a special lunch, the canteen might be open for a short period, around 7:30 p.m., to accommodate the students and staff.

It's also possible that the canteen might be open for a few minutes before the next school day starts, to ensure that all students have a chance to eat their lunch and get settled in before the next day. The canteen would likely be closed at 7:30 p.m. to allow for a few minutes of downtime before the next day's school day begins.

In terms of food, the canteen might serve a variety of options, including sandwiches, soups, and other school meals. Some schools might also offer a special "breakfast" or "lunch" option, such as a buffet or a "breakfast class," to cater to students who might be in need of a quick meal before the next day's school day.

[Response continues with repeated phrases about canteen being open for a few minutes before next day's school day.]

Reference Response:

Answer according to: The fantastic breakfast in school canteen at 7:00 p.m. is a great way to get a taste of the school day. The canteen is open from 7:00 p.m. to 11:00 p.m. The canteen is open 7 days a week. The canteen is open 24 hours a day. The canteen is open 365 days a year. *(Repeated many times)*

DPO Score: -31.75 **SFT Score:** -32.75 **Reference Score:** -32.50 **DPO Win:** 1

In this particular example, we see that DPO qualitatively performs better than the SFT trained response, while both are better than the reference response. In terms of why DPO could be considered a better response, it acknowledges that this is a fictional scenario given the unclear prompt, whereas the SFT trained response treats the situation as reality, which could be considered misleading or untruthful. However, both responses are generally quite similar, which is reflected in the similar win-rates for both.

7 Discussion

The primary reason for the observed slight deterioration in DPO performance and failure to improve compared to the original SFT model likely stems from the strength and quantity of the reward signal. DPO relies on having a sufficiently strong and consistent reward signal—in this case, our preference data—to guide the model toward desired behaviors. However, with a limited number of examples we were able to train on, the reward signal is simply too weak to meaningfully steer the model's outputs and to distinguish from noise, especially when starting from an SFT model that has already learned broad and fluent generation capabilities. Without a large volume of high-quality preference data, DPO may end up overfitting to the small set of rewards it sees, degrading generalization and overall performance. In other words, you would need a huge amount of preference data before this method could consistently improve upon the SFT baseline as we need to generate a sufficiently strong reward signal distinguishable from noise.

In practice, neither our SFT nor DPO extensions with Chebyshev scalarization led to significant improvements across all objectives. For building on SFT, one key limitation is that the model is still trained using next-token prediction on a fixed dataset of human-curated completions. Even with Chebyshev-weighted selection of training examples, the model is not explicitly optimizing its policy toward the Pareto front—it is merely learning to better mimic samples that happened to have good scalarized rewards. For DPO, a similar issue arises: the method optimizes relative preference between response pairs, but these pairwise preferences do not always align with our external reward dimensions (helpfulness, coherence, factuality). Moreover, the Chebyshev scalarization was only used to select training pairs or modulate loss weighting, but not fully integrated into the core DPO objective. As a result, both approaches suffered from reward collapse along some dimensions and failed to systematically move the model toward a better balance across all objectives.

8 Conclusion

Here, we attempt to implement Chebyshev scalarization, a Pareto-optimality based method for MORL, on an SFT model. We compare this experimental method to the same SFT model fine-tuned with DPO as a baseline. However, we find that the DPO model built off of the SFT baseline resulted in a mild deterioration in performance, as compared to the vanilla SFT model. We were unable to complete running of the Chebyshev model due to time running out, as mentioned in the experimental setup. Future work would simply just allow us extra time to dedicate to running and optimizing our models around different hyperparameters, such that we would be able to run more experiments.

9 Team Contributions

- **Chetan Nair:** Implementation on SFT and DPO, along with running experiments and data collection.
- **Ishaan Singh:** Preliminary implementation of SFT and helped with implementation of DPO, and finalized implementation of MORL with Chebyshev using both SFT and DPO-based methods. Major contributions to drafting of report.
- **Khanh Tran:** Preliminary implementation of DPO. Major contributions to drafting of report and project poster.

Changes from Proposal We originally proposed comparing and contrasting multiple methods of performing MORL, including weighted sums and dynamic weight adjustment. However, we did not have sufficient time and compute resources to experiment with all proposed MORL methods—including Chebyshev-optimized DPO—especially due to the fact that each experimental run took upwards of 3 hours. Future work would involve robust training of Chebyshev-optimized SFT and DPO, and further exploration of other MORL methods.

References

- Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM2: When Smol Goes Big – Data-Centric Training of a Small Language Model. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*. <https://arxiv.org/abs/2502.02737>
- Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Ultrafeedback: Boosting Language Models with Scaled AI Feedback. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*. <https://arxiv.org/abs/2310.01377>
- Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning. In *Advances in Neural Information Processing Systems*. <https://doi.org/10.18653/v1/2022.emnlp-main.222> Presented at the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP), Abu Dhabi, United Arab Emirates.
- Yasaman Jafari, Dheeraj Mekala, Rose Yu, and Taylor Berg-Kirkpatrick. 2024. MORL-Prompt: An Empirical Analysis of Multi-Objective Reinforcement Learning for Discrete Prompt Optimization. *arXiv preprint arXiv:2402.11711* (2024). <https://doi.org/10.48550/arXiv.2402.11711> Preprint.
- Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. <https://qwenlm.github.io/blog/qwen2.5/>
- K. Van Moffaert, M.M. Drugan, and A. Nowe. 2013. Scalarized multi-objective reinforcement learning : novel design techniques. In *2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL) : 16 - 19 April 2013, Singapore*. Institute of Electrical

and Electronics Engineers, United States, 191–199. <https://doi.org/10.1109/ADPRL.2013.6615007> 2013 IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL 2013), ADPRL 2013 ; Conference date: 16-04-2013 Through 19-04-2013.