Extended Abstract

Motivation In this project, we investigated fine-tuning techniques aimed at enhancing the instruction-following capabilities of a small language model (Qwen 2.5 0.5B). We implemented Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO), while also exploring advanced variants such as Rejection Sampling DPO (RS-DPO) and self-improvement strategies. Our goal was to evaluate whether the model could improve its own performance using techniques that rely on its own outputs, rather than solely depending on static datasets.

Method Implementation

Rejection Sampling Rejection Sampling DPO (RS-DPO) is a technique that augments standard DPO by dynamically generating rejected responses from the current model—typically an SFT model—instead of relying exclusively on predefined datasets. These generated rejections are paired with preferred responses to form contrastive training examples. In our implementation, due to time constraints, we conducted single-sample rejection sampling on the 61k prompts from the UltraFeedback training split. We assumed that UltraFeedback's "chosen" responses are of higher quality than those generated by our SFT model. Therefore, for each prompt, we paired the UltraFeedback "chosen" response as the preferred output and the SFT-generated response as the rejected output. These pairs were then used to train the policy model using DPO.

Self-Improvement This experiment explored self-improvement using model-generated data for DPO training. The motivation is based on two hypotheses: 1. High-temperature sampling enables better selection – By sampling multiple responses (e.g., K=8) at higher temperature, the model generates a diverse set of outputs. With the aid of a reward model, we can select the best and worst responses to construct preference pairs for training. 2. Self-generated chosen responses can also be useful – Our prior experiments showed that model-generated rejections improved DPO training. We hypothesized that carefully selected self-generated chosen responses could similarly enhance performance.

Results Our results indicate measurable performance improvements using both RS-DPO and self-improvement strategies. Specifically, RS-DPO achieved approximately a 15

Discussion Conclusion These explorations highlight the feasibility and effectiveness of self-improvement strategies—leveraging the model's own outputs to refine its behavior. Further investigation is warranted into areas such as model bias, stability of sampling-based methods, and the criteria for selecting high-quality training data. These directions are crucial for building more robust and adaptive fine-tuning pipelines.

Self-Improvement variants on DPO

Yiyang Hao
Department of Computer Science
Stanford University
yyhao@stanford.edu

Abstract

In this project, we investigated fine-tuning techniques to enhance the instruction-following capabilities of a small language model (Qwen 2.5 0.5B). We implemented Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and explored variants such as Rejection Sampling DPO (RS-DPO) and self-improvement strategies. Our results demonstrate that incorporating a small portion of rejection-sampled data into DPO training leads to performance gains, and highlight the potential of self-improvement methods for further enhancing model alignment.

1 Introduction

Objective This project explores the implementation of various reinforcement learning (RL) algorithms for fine-tuning large language models (LLMs). We begin by fine-tuning base models using Supervised Fine-Tuning (SFT) and Direct Preference Optimization (DPO), followed by experiments with advanced techniques such as RS-DPO, self-improvement, and curriculum-based optimization.

Starting Point We selected the Qwen 2.5 0.5B base model, which is the smallest in the Qwen 2.5 series, as our starting point. Its lightweight nature enables efficient experimentation on relatively small datasets, allowing us to clearly observe the impact of different techniques while minimizing GPU usage and training time.

Supervised Finetuning For supervised fine-tuning (SFT), we used the smolTalk dataset Allal et al. (2025), available at `https://huggingface.co/datasets/HuggingFaceTB/smoltalk`. This synthetic dataset is designed for training LLMs in a conversational format, containing approximately 460k human–assistant dialogue pairs.

Directo Preference Optimization After SFT training, we applied Direct Preference Optimization (DPO) to further enhance the model's instruction-following capabilities. We used the UltraFeedback dataset Cui et al. (2024), which contains 61k prompt–response pairs in the exact format required for DPO: each example includes a prompt, a preferred (chosen) response, and a less preferred (rejected) one. UltraFeedback is a large-scale, synthetic, and fine-grained preference dataset, created by sampling prompts from various LLMs and generating responses to capture diversity of instructional behaviors.

Evaluation We use win rate comparison as the primary evaluation metric for model performance. Given a set of instruction-following prompts, we sample responses from both the trained model and reference models, and then evaluate them using the LLaMA 3.1 Nemotron-70B Reward Model

Wang et al. (2024b,a). This reward model assesses responses based on truthfulness, honesty, and helpfulness. While absolute reward scores are not comparable across different prompts, they are meaningful for comparing responses to the same prompt. We evaluate win rates across multiple reference baselines, including Qwen 2.5 0.5B Instruct, our trained SFT model, DPO model, and other experimental variants.

Research Directions We aim to explore extensions of DPO to improve model performance. First, we investigate Reject Sampling DPO (RS-DPO) Khaki et al. (2024), which augments training data by sampling rejected responses directly from the current model, rather than relying solely on UltraFeedback. We also examine the potential of RLAIF (Reinforcement Learning from AI Feedback) Ouyang et al. (2022), focusing on how self-generated response pairs can be selected and structured within the DPO framework to impact performance.

2 Related Work

Self-Rewarding Language Models Yuan et al. (2025) proposes a novel approach for fine-tuning LLMs without relying on fixed, human-trained reward models. Instead, the model serves as its own evaluator by: (1) generating its own training prompts, (2) scoring its responses using an LLM-as-a-judge prompting strategy, and (3) fine-tuning itself using DPO. The results are promising—after 2–3 self-training iterations, the model shows consistent and significant improvement.

Rejection Sampling-DPO Khaki et al. (2024) introduces RS-DPO, a method that systematically combines Rejection Sampling (RS) with Direct Preference Optimization (DPO). The approach begins by training a supervised fine-tuned (SFT) policy model, from which k diverse responses are sampled per prompt. RS-DPO then selects contrastive pairs based on their reward distribution. These pairs are used to apply DPO, aligning the model more closely with human preferences. Experiments show that RS-DPO is effective in resource-constrained settings and consistently outperforms standalone RS, PPO, and DPO in aligning models with user intent.

RLAIF vs. RLHF Ouyang et al. (2022) investigates alternatives to human-in-the-loop feedback by comparing Reinforcement Learning from AI Feedback (RLAIF) with traditional Reinforcement Learning from Human Feedback (RLHF). In RLAIF, preference labels are generated by off-the-shelf LLMs rather than humans, enabling scalable training. The study explores two settings: (1) RLAIF, where a reward model is trained on AI-generated preference data, and (2) d-RLAIF, which bypasses the reward model entirely by directly using LLM-generated scores during reinforcement learning. Experimental results demonstrate that both methods offer competitive alignment performance while significantly reducing reliance on human annotations.

Self-Instruction Wang et al. (2023) introduces Self-Instruct, a method for aligning language models by leveraging self-generated instructional data. The approach uses a small seed set of prompts (approximately 5%) to bootstrap a larger set of high-quality instruction-following examples generated by the model itself. Through iterative self-citation and self-evaluation, the model refines its own behavior, creating a feedback loop that enhances alignment and overall performance.

3 Method

Supervised Fine-Tuning (SFT) The first stage of this project involves Supervised Fine-Tuning (SFT). Our starting point is a base model trained only to predict the most likely next token, which means it lacks the ability to follow instructions and instead tends to produce generic continuations. SFT is designed to bridge this gap by teaching the model to behave like an assistant. It transforms the model from a raw language model into one that can engage in helpful, task-oriented dialogues by fine-tuning it on a dataset of conversational interactions.

By training on prompt–response pairs in a natural dialogue format, SFT helps the model learn the structure of helpful responses, maintain appropriate length, and generalize across diverse user queries. This process lays the foundation for human-like interactions with LLMs.

Technically, SFT involves predicting the correct response given a prompt. The loss function used is the standard autoregressive cross-entropy loss over the response tokens (where the prompt is masked out):

$$\mathcal{L}LM = -\sum i = 1^T \log P(y_i \mid x, y_{<i}; \theta)$$

Masking and Padding One critical implementation detail is the masking of the prompt tokens in the loss calculation. Tokens from the prompt should be labeled as `-100` to exclude them from gradient updates. Additionally, since training is batched and all sequences in a batch must be the same length, proper padding is necessary.

Gradient Accumulation Due to the constrains of GPU memory storage, we only have batch size = 8. To accomplish a more stable gradient ascent, we applied gradient accumulation. It performs multiple forward + backward passes and accumulate the gradient, simulate a much larger batch.

Automatic Mixed Precision Automatic Mixed Precision (AMP) is speed up model training and reduce GPU memory, it uses FP16 in some operations for speed and efficiency and keep the FP32 in loss calculation or weights updates for numerical stability.

3.1 Direct Preference Optimization

Direct Preference Optimization (DPO) is a fine-tuning method that updates the policy directly using preference data, eliminating the need to explicitly train a reward model. Instead of learning a reward function and optimizing it with reinforcement learning, DPO uses a pairwise loss that implicitly aligns the model's behavior with human preferences. The core loss function is:

$$\mathcal{L}_{\text{DPO}} = -\log\left(\frac{\exp\left(\beta \cdot \Delta(x, y^+, y^-)\right)}{1 + \exp\left(\beta \cdot \Delta(x, y^+, y^-)\right)}\right)$$

Here, $\Delta(x, y^+, y^-) = \log \pi(y^+|x) - \log \pi(y^-|x) - (\log \pi_{\text{ref}}(y^+|x) - \log \pi_{\text{ref}}(y^-|x))$ is the difference in log-probabilities between the chosen and rejected responses under the current policy π and a frozen reference model π_{ref}.

In our DPO training, we use the following optimization techniques (also used in SFT): cosine learning rate scheduling with 10

To further enhance DPO training, we apply the following regularization techniques:

Frozen Reference Model We use a reference model π_{ref} as a frozen copy of the base model. It serves as a baseline to compare how the trained policy π improves or diverges. Including π_{ref} in the loss prevents the model from overfitting for preferences.

DPO-Positive Regularization To ensure that the model does not degrade on chosen outputs, we use DPO-Positive (DPO-P) regularization. This term penalizes the model when the log-probability of the preferred (chosen) response under the new policy is lower than that of the reference model:

$$\mathcal{L}\text{DPO-P} = \max\left(0, \log \pi \text{ref}(y^+|x) - \log \pi(y^+|x)\right)$$

This encourages the model to maintain or improve performance on preferred responses instead of losing on both of chosen and reject patterns.

KL Regularization To prevent overfitting and catastrophic forgetting, we also add a KL-divergence term that penalizes divergence from the reference model for both the chosen and rejected responses:

$$\mathcal{L}\text{KL} = \frac{1}{2}\left[\left(\log \pi\text{ref}(y^+|x) - \log \pi(y^+|x)\right) + \left(\log \pi_{\text{ref}}(y^-|x) - \log \pi(y^-|x)\right)\right]$$

This term is controlled by a hyperparameter λ_{KL}, which balances imitation of the reference model and preference alignment.

3.2 RS-DPO

Rejection Sampling DPO (RS-DPO) is a technique that combines rejection sampling with Direct Preference Optimization. In this method, rejected responses are not sourced from a fixed dataset but are instead generated dynamically from the current model (typically the SFT model), and used to construct preference pairs for DPO training.

In our implementation, due to time constraints, we conducted single-sample rejection sampling using the SFT model on the 61k prompts from the UltraFeedback training split. We assumed that the UltraFeedback "chosen" responses are of higher quality than those generated by our SFT model. Thus, for each prompt, we formed a preference pair where the UltraFeedback "chosen" was treated as the preferred response and the SFT-generated sample was treated as the rejected response. We then trained our policy model using DPO on these pairs.

This setup is based on the observation that the UltraFeedback chosen and rejected responses are often both significantly better than outputs generated by our SFT model, and therefore may not provide strong contrast for learning. In contrast, the SFT-generated responses serve as more informative and realistic "negative" examples, better matching the current model's decision boundary. Khaki et al. (2024).

3.3 Self-Instruct DPO

In this experiment, we explored a self-improvement strategy for DPO based on self-generated data. This is inspired by two key assumptions: 1. High-Temperature Sampling Enables Quality Selection: With increased sampling temperature, the model can generate a diverse range of outputs—some strong, some weak. By generating multiple candidates (e.g., K=8) and using a reward model to select high-quality samples, we can build new training data that encourages better performance through preference optimization. 2. Self-Generated Chosen and Rejected Responses Can Be Effective: Building on our prior observation that SFT-generated rejections improve training, we hypothesize that self-generated "chosen" responses—if properly selected—can also be beneficial. By constructing entirely self-generated preference pairs (both chosen and rejected), and applying DPO, we can test whether the model is capable of self-improvement without relying on human-annotated or external high-quality responses.

This experimental direction parallels the spirit of RLAIF and Self-Rewarding Language Models, pushing the boundary of LLMs learning from their own outputs in a bootstrapped manner.

4 Experimental Setup

4.1 Supervised Fine Tuning

4.1.1 Data Preparation

In data prepration phase, we have done three modification on the smolTalk dataset:
1. Extract first conversation.
We only use the first conversation of each dialog, mainly because we realize that the following questions will contain the previous context, and due to our evaluation format, which is one question one response, not the multi question required, so we decide to only extract first question and prompt, first response as response to format the training dataset. 2. Truncate at the best percentile.
After the examination of prompts and responses, we realized that 95th is the best cut to balance the space requirement and content included. As the result, we use 255 length of the prompt cut, 420 as the length to cut prompts, all based on token numbers.
3. Filter out short response.
Due to the small conversation, there are responses really short which doesn't fit our expectation of

Table 1: Percentile of smolTalk

Percentile	Prompt length	Response length
50th	57	225
90th	179	375
95th	254	419
99th	1339	576

instruction following, so we removed the reponse which contains less than 75 characters.

4.1.2 Training

We initially tested with a small subset (5%) of the smolTalk dataset to verify the training and code pipeline. Although both training loss and evaluation loss decreased as expected, we observed that the resulting SFT model performed significantly worse than the base model—producing mostly nonsensical outputs. This highlighted the inadequacy of relying solely on loss metrics. To address this, we introduced win rate evaluation by randomly selecting 100 prompts from the UltraFeedback test set and comparing our model's responses against those from the Qwen base and Qwen Instruct models.

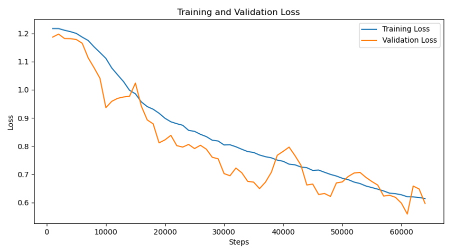


Figure 1: loss & evaluation

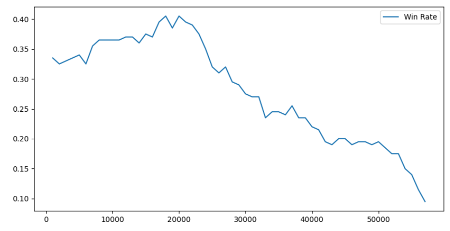


Figure 2: win rate

From the charts we can observe that win rates increases at the beginning and drops significanlty later. The main reason should be overfitting, since it's a rather small dataset, model learns to fit it quickly but forget generalization. We then decided to add win rate monitoring to all of the following training to avoid overfitting, which adds extra time of training though.
Then we prepared for SFT training, we use learning rate: 5e-6 within three epochs. From the results we can see the decreasiong of loss and evaluation, with increasing of win rates against base and instruct models

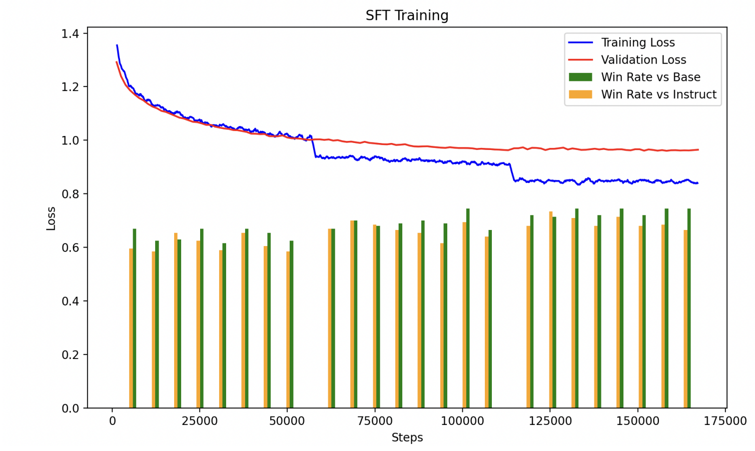


Figure 3: SFT Training

4.2 DPO

With generated 61k ultrafeedback DPO datasets, we evaluates different combination of learning rate, beta, KL and other parameters.

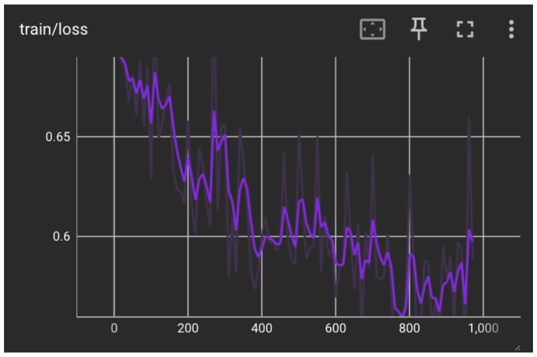


Figure 4: DPO loss

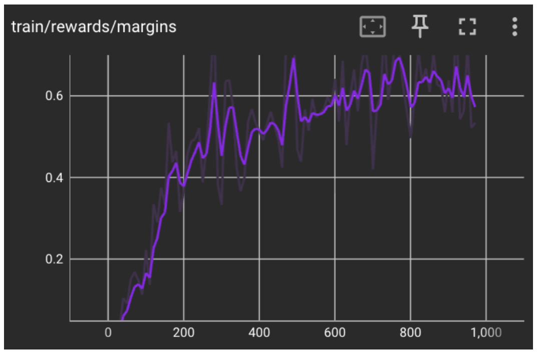


Figure 5: DPO Margins

4.3 RS-DPO

Due to the time constrains, we didn't sample multiple rejections, for each prompt we only sampled once, there are 61k ultrafeedback training prompts, we generated the same amount of response from

our previously trained SFT-model, then applied DPO with the same parameters as previous with the full new dataset.
We then tried with another two experiments, mix dataset of ultrafeedback and our generated RS training dataset. We tested a lower learning rate on the 10% mixture (10% of Reject Sampling dataset and 90% of ultrafeedback) and a 20% mixture (20% of Reject Sampling dataset and 80% of ultrafeedback).
The above experiments are all trained upon SFT-model, which can compare with the trained DPO model.

4.4 Self-Improvement from cirrulumns

To validate our assumption of modeling could self-improve, we tested sampling 5 results for a small prompts set (100), with temperature 1.0. After sampling, we evaluate the response using reward model, pick the top score and calculate win rate against SFT-model. The result is promosing, we have increased the winning rate from 0.72 to 0.83, as a 15.2% performance increase.
We then started the self-improvement cycle, to improve most performance with least prompts, we used cirrululmns-orientated prompts selection. We first categorized all 1000k test prompts in ultrafeedback with ChatGPT, then calculate the win-rate of our DPO-model vs SFT-model, calcuate indivitual win rate, pick the top two category with lowest win rate, generate corresponding prompts from other LLMs (we use ChatGPT, Gemini and Claude mixture). Then sampled 8 responses within in different temperature. The different temperature can help to provide more diverse responses. We sampled two response under temperature 0.2, sampled three under temperature 0.7 and sampled last three under temperature 1.2. In this case we have two of the "best" response and some moderate exploration and some high risk explorations. Then use reward model to evaluate the responses, pick the top performance response as chosen and random pick rejected from the bottom three responses. Use the new data as DPO training data and train our model.

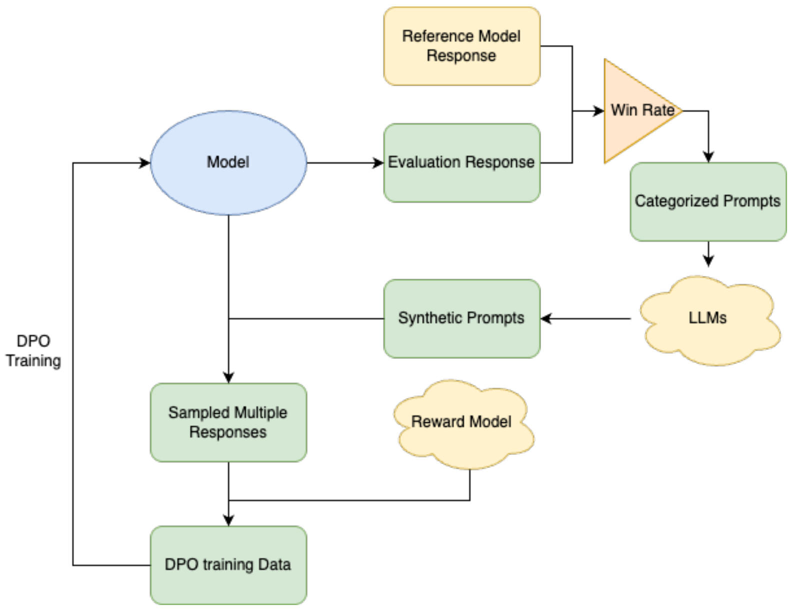


Figure 6: Self-Improvement Flow Chart

5 Results

5.1 RS-DPO-1

Here is the training results of RS-DPO with full rejection sampled dataset, I also include the loss and margin of previous DPO training with normal ultrafeedback chart. Yellow represents RS-DPO, pink represents regular DPO. And win rates of RS-DPO1 is dropping directly under 0.1 agains SFT model.

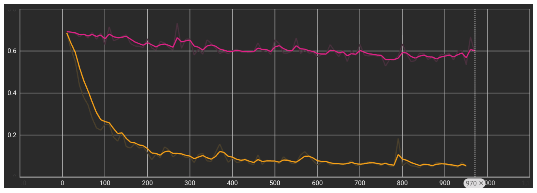


Figure 7: RS-DPO loss

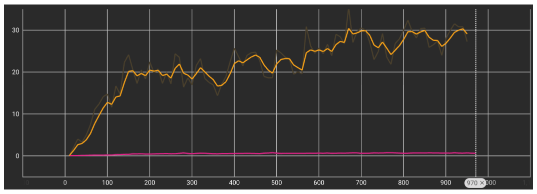


Figure 8: RS-DPO margin

Analysis We can clearly see that the margin of RS-DPO increased a huge number compared with regular DPO, the large difference actually explains a lot. Since DPO aims to let model learn the difference between chosen and reject and allow model learn the reward by picking chosen, this pattern is too easy for our model to learn not picking its own geneartedresponse. It gets "too easy" training signals, chosen response are obviously better than teh SFT outpus. This turns out: model learns the different pattern quickly but didn't learn what makes response really good, it predicts rejected really easy, turns out low loss and high margin, the result turns to be, this model learns "response not similar" to its SFT generated is Good, turns out a dramastic low performance on the evaluation.

5.2 RS-DPO2, RS-DPO3

We then tested two data mixture of rejection sampling and ultrafeedback datasets, RS-DPO2 is the 20% mixture and RS-DPO3 is the 10% mixture. With less amount of rejection sampling, it can provide diversity and allow model to learn what is the good response and steer not too much from its current policy.
Pink is regular DPO, green is RS-DPO2 and gray is RS-DPO3.

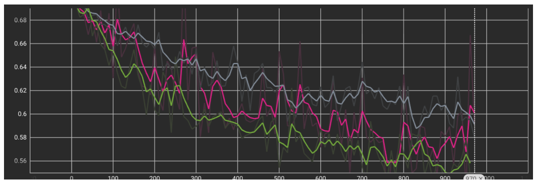


Figure 9: RS-DPO2 vs RS-DPO3 loss

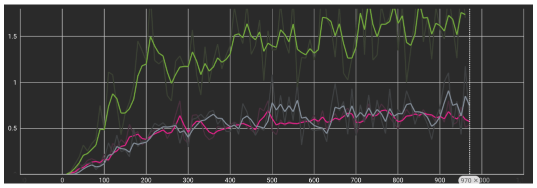


Figure 10: RS-DPO2 vs RS-DPO3 margin

We can see with mixture of ultrafeedback as majority, we have loss and margin much closer to regular DPO, which shows we have a normal level of pattern recoginization. The win rate is more important to compare, there is the win rate calculation (the win rate is calculated against DPO model):

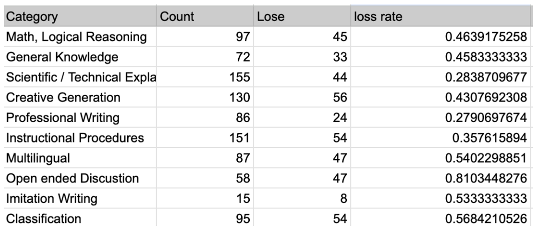
Table 2: RS-DPO win rates

Method	win-rate (0.33 epoch)	win-rate (0.66 epoch)	win-rate (1 epoch)
RS-DPO1	0.12	0.15	0.10
RS-DPO2	0.62	0.585	0.48
RS-DPO3	0.515	0.555	0.67

Analysis We can clearly see, the 10% mixture has highest performance after one epoch runs, which out performed regular DPO training. And if we increase the percentage, we increase the high risk for model to learn the "easy" pattern instead of what's really good response. It's much clear in the RS-DPO2, the win rate actually drops after more data involved in training.

5.3 Self-Improvement

Due to the time and resource constrains, we didn't test self-improvement in a larger set, we only categorized the 1000 prompts from ultrafeedback test results and generated about 1k synthetic prompts.



Category	Count	Lose	loss rate
Math, Logical Reasoning	97	45	0.4639175258
General Knowledge	72	33	0.4583333333
Scientific / Technical Expla	155	44	0.2838709677
Creative Generation	130	56	0.4307692308
Professional Writing	86	24	0.2790697674
Instructional Procedures	151	54	0.357615894
Multilingual	87	47	0.5402298851
Open ended Discustion	58	47	0.8103448276
Imitation Writing	15	8	0.5333333333
Classification	95	54	0.5684210526

Figure 11: Category Distribution

With previous approach, we categorized prmpots into 10 buckets, and calculated the loss rate of our DPO model against SFT model, realzied that our model is weak at Open ended Discussion and Classification Tasks. Then we generated 600 prompts in the Open ended Discussion, 400 prompts in the Classification category.
We use the DPO training, running three epoches on the RS-DPO3 model.

Table 3: Self-Improvement win rates

Method	win-rate
RS-DPO3 vs DPO	0.5975
Self Improvement VS DPO	0.6275
Self Improvement VS RS-DPO3	0.520

Analysis Since self-improvement is trained upon RS-DPO3 and we only have time to train on a rather small dataset (1k), we can see a potiential of improvements. The Self-Improvement version performs 6% better when comparing to DPO baseline and it outperformed a little (4%) over RS-DPO3. This method has risk of amplify the bias inside the model, but the results shows potiential of iteration and improve itself towards a better response.

6 Discussion

This project explored and evaluated several techniques for fine-tuning a small language model (Qwen 2.5 0.5B) to improve its instruction-following capabilities. Our experiments demonstrated that approaches like Rejection Sampling (RS-DPO) and Self-Improvement can yield measurable performance gains. However, there remain limitations and areas for further study.

In the case of Self-Improvement, we were only able to perform a single iteration due to time constraints. We did not analyze cross-category win rates, which could have provided deeper insights into model generalization. Additional iterations may help the model gradually refine its outputs, but also risk exposing the limitations of small models, such as capacity bottlenecks and self-reinforcing biases.

Another key limitation lies in the reward-model-centric optimization. Since we rank samples using a fixed reward model, the fine-tuned model may learn to optimize specifically for that reward model's preferences—not necessarily for human judgment. This alignment gap means that performance could degrade when evaluated using a different reward model or by human raters. Thus, the improvements we observe may reflect optimization for a narrow evaluator, not general instruction-following ability.

7 Conclusion

Through our exploration, we found that even small models like Qwen 2.5 0.5B possess potential for self-improvement when equipped with the right training mechanisms. Techniques such as Rejection Sampling DPO introduce additional diversity and offer meaningful gains by pairing high-quality reference responses with self-generated rejections.

Similarly, Self-Improvement demonstrates that models can bootstrap their own learning—selecting chosen and rejected responses from their own generations and using them for further fine-tuning.

Looking forward, we are particularly interested in two open questions: 1. How does performance evolve over multiple self-improvement iterations? Does the model reach a plateau, or does it eventually degrade due to overfitting or bias accumulation? 2. What is the potential and limitation of self-judgment? If a model scores its own responses instead of relying on external reward models, can it still improve? And how reliable is such self-assessment?

These directions will be critical for building more robust, autonomous, and generalizable LLM training pipelines.

8 Team Contributions

- **Yiyang Hao** Single Member Team

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. arXiv:2502.02737 [cs.CL] `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. arXiv:2310.01377 [cs.CL] `https://arxiv.org/abs/2310.01377`

Saeed Khaki, JinJin Li, Lan Ma, Liu Yang, and Prathap Ramachandra. 2024. RS-DPO: A hybrid rejection sampling and direct preference optimization method for alignment of large language models. (2024). `https://www.amazon.science/publications/rs-dpo-a-hybrid-rejection-sampling-and-direct-preference-optimization-method-for-alignment-of-`

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

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. arXiv:2212.10560 [cs.CL] `https://arxiv.org/abs/2212.10560`

Zhilin Wang, Alexander Bukharin, Olivier Delalleau, Daniel Egert, Gerald Shen, Jiaqi Zeng, Oleksii Kuchaiev, and Yi Dong. 2024a. HelpSteer2-Preference: Complementing Ratings with Preferences. arXiv:2410.01257 [cs.LG] `https://arxiv.org/abs/2410.01257`

Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. 2024b. HelpSteer2: Open-source dataset for training top-performing reward models. arXiv:2406.08673 [id='cs.CL' full$_name =' Computation and Language' is_active = True alt_name =' cmp-lg' in_archive =' cs' is_general = False description =' Covers natural language processing. Roughly includes material in ACM Subject Class I.2.7. Note that work on artific$ $language issues broadly construed (natural-language processing, computational linguistics, speech, text retrieval,$

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2025. Self-Rewarding Language Models. arXiv:2401.10020 [cs.CL] `https://arxiv.org/abs/2401.10020`