Extended Abstract

Motivation Training large language models usually hinges on Reinforcement Learning from Human Feedback (RLHF), yet hand-curated preference labels remain pricey and slow to collect. Recent studies reveal that high-capacity LLMs can generate their own preference judgments, matching human labels with only a slight quality drop and thus enabling synthetic data augmentation (SDA). This project asks a focused question: if we inject thousands of these synthetic preference pairs into Direct Preference Optimization (DPO) for an already fine-tuned Qwen 2.5 model, do we see a meaningful jump in downstream performance?

Method Our work involved three primary steps. First, supervised fine-tuning (SFT): we took a vanilla Qwen 2.5 model and fine-tuned it on the SmolTalk dataset, producing a baseline/reference model for DPO. For DPO, we take another base model, use the SFT model as a reference, and train on a subset of the UltraFeedback dataset. For our extension, we chose to inject synthetic data: the SFT model, scored by Nemotron 70B, gave us roughly 14.5k synthetic pairs, while GPT-3.5 turbo supplied another 4.7k higher-quality preference pairs. During DPO training, these synthetic pairs are shuffled in with the original UltraFeedback data, giving the model a more diverse dataset, thus making it more generalizable.

Implementation To ensure our experiments ran with our memory constraints, we inserted LoRA adapters only into the query and value projection layers and turned on gradient checkpointing; mini-batches never exceeded six examples to avoid running out of memory as well. All training happened on AWS g5e and g6e instances, with each run capped at three to six epochs to combat overfitting. For evaluation, we decode with vLLM and feed the outputs to Nemotron, reporting the percentage of prompts where our model beats the vanilla Qwen 2.5 model.

Results Our SFT fine-tuniung gave us a solid baseline: the model edges out Qwen2.5 on 57% of prompts. When we switched to DPO and expanded to six epochs, we improved performance up to a score of 0.6. Further experiments in which we reinforced our model with 14,370 synthetic pairs sourced from the SFT baseline resulted in performance down to 0.5925, suggesting potential overfitting to these synthetic pairs. Interestingly, we see that, in a second batch of test prompts (the second leaderboard), our models which performed best initially (Vanilla DPO) performed worse than the first models that we developed using less synthetic data, but of higher quality (GPT generated), actually achieving our highest score of 0.125.

Discussion Model performance is based primarily on: how much training we can afford to run with our constraints, and how good our synthetic datapoints are. Fine-tuning with SFT for more epochs consistently beats the shorter DPO runs, signalling that the chief bottleneck is sheer update count and volume of data consumed, not the objective itself. When we do utilize and inject synthetic data, the GPT generated pairs far out-perform the SFT generated pairs, delivering larger improvements and demonstrating clearly that higher quality synthetic data pays dividends. Still, synthetic data also caused us to overfit initially: if the extra pairs are noisy or lack diversity, or if we throw in too many examples, DPO latches onto hyper-specific quirks rather than signal general trends, and the final scores can actually drop.

Conclusion Carefully procured and generated synthetic batches can give DPO a real lift—yet the upside is still hemmed in by memory limits and label quality. Looking ahead, progress will likely come from: generating larger pools of high-fidelity synthetic pairs, fine-tuning the mix between real and synthetic data, squeezing in longer runs via mixed-precision training, and applying a light human pass to weed out low-quality synthetic entries. Together, these steps could raise win-rates past today's marks while keeping compute costs realistic for us.

CS224R Default Final Project: Synthetic Data Augmentation for LLM Training

Ben Gonzalez-Maldonado
Department of Computer Science
Stanford University
bengm@stanford.edu

Diego Padilla
Department of Computer Science
Stanford University
diego03@stanford.edu

Emily Park
Department of Computer Science
Stanford University
ejpark24@stanford.edu

Abstract

We test whether injecting synthetic preference data into Direct Preference Optimization (DPO) improves a supervised-fine-tuned (SFT) Qwen 2.5 model. After establishing an SFT baseline on SmolTalk, we train DPO with UltraFeedback comparisons and augment it with two synthetic sources: 14.5k pairs generated by the SFT model and scored by Nemotron 70B, and 4.7k higher-quality pairs produced by GPT 3.5 Turbo. Using LoRA adapters, gradient checkpointing, and short epoch schedules for memory efficiency, we evaluate models with Nemotron reward scores against the vanilla Qwen 2.5 reference. DPO alone outperforms the SFT baseline; adding a large set of SFT-derived pairs leads to mild over-fitting, whereas the smaller, higher-fidelity GPT-generated set yields the best leaderboard result. These findings show that synthetic data can boost DPO, but gains depend critically on the quality and carefully balanced quantity of the added preferences.

1 Introduction

Aligning large language models (LLMs) with human values typically starts by gathering thousands of pairwise preference judgments from annotators and then applying RLHF. Although effective, this workflow is slow and expensive because every new domain or task demands another costly round of labeling. DPO streamlines the process by treating preference learning as a straightforward binary-classification task, eliminating the most labor-intensive step in RLHF. However, DPO still relies on large, high-quality preference datasets, and producing those labels at scale remains a major obstacle for teams constrained by compute or budget.

Recent work shows that high-capacity LLMs can themselves act as proxy annotators, producing synthetic preference judgments that track human ratings with only a modest loss in fidelity. This Synthetic Data Augmentation (SDA) offers a path to cheap, plentiful training data, but its interaction with DPO is not well understood. Does injecting thousands of AI-generated comparisons into DPO improve downstream behavior, or does it simply make the model overfit to the preferences of the synthetic labeler?

We address this question in a focused empirical study. Beginning with a Qwen 2.5 model that we first fine-tune on the SmolTalk corpus to obtain a strong supervised baseline, we train DPO on a subset of the UltraFeedback dataset and then inject two complementary synthetic streams:

1. **Quantity-oriented** – roughly 14.5 k pairs produced by the SFT model itself and ranked by the Nemotron 70B reward model;

2. **Quality-oriented** – about 4.7 k higher-fidelity pairs generated directly by GPT-3.5-turbo.

By shuffling these synthetic comparisons into the real data during training, we probe whether raw volume (many but noisier SFT-derived pairs) or improved fidelity (fewer but cleaner GPT-generated pairs) matters more for DPO's final performance.

We ultimately pursue and attempt to answer three research questions:

Baseline Improvement How much does vanilla DPO improve upon an SFT-only model on open-ended dialogue evaluation?

Synthetic boost To what extent do injected synthetic preference pairs enhance—or hinder—DPO's gains?

Quality vs. quantity How do the source and volume of synthetic data shape generalization, and where is the tipping point at which additional pairs begin to hurt rather than help?

Through these questions, we aim to clarify the practical value of SDA for preference-based fine-tuning and to provide actionable guidelines on balancing synthetic quality, quantity, and compute when resources are limited.

2 Related Work

Our project draws on several lines of research in preference-based fine-tuning of large language models, synthetic data augmentation, and direct policy optimization:

Reinforcement Learning from Human (and AI) Feedback.
Traditional RLHF methods first train a reward model on human preference labels, then fine-tune a language model via reinforcement learning under a KL-constraint to prevent drift from the base policy. Recent work has shown that preferences generated by off-the-shelf LLMs can substitute for human labels with minimal loss in performance. In particular, Lee et al. (2024) compare RLHF against "RL from AI Feedback" (RLAIF) across summarization and dialogue tasks, demonstrating that RLAIF matches or outperforms RLHF without requiring costly human annotation. They did so by:

1. Training a reward model on human-annotated pairwise comparisons or on preferences generated by the PaLM 2 Large model (RLAIF).

2. Fine-tuning policies with a KL penalty ($\beta = 0.05$).

3. Evaluating on three tasks: Reddit TL;DR summarization, helpful dialogue generation, and harmless dialogue generation—using blind human judgments.

From their findings (and that of other papers, see Bai et al. (2022)), we drew inspiration in the idea of generating preferences via an established LLM.

Synthetic Data Augmentation via AI Critique and Revision
Bai et al. (2022) introduce Constitutional AI, a pipeline for harmlessness tuning that uses a small set of human-written principles to generate and iteratively refine model outputs through AI-led critiques and revisions. Their pipeline is as follows:

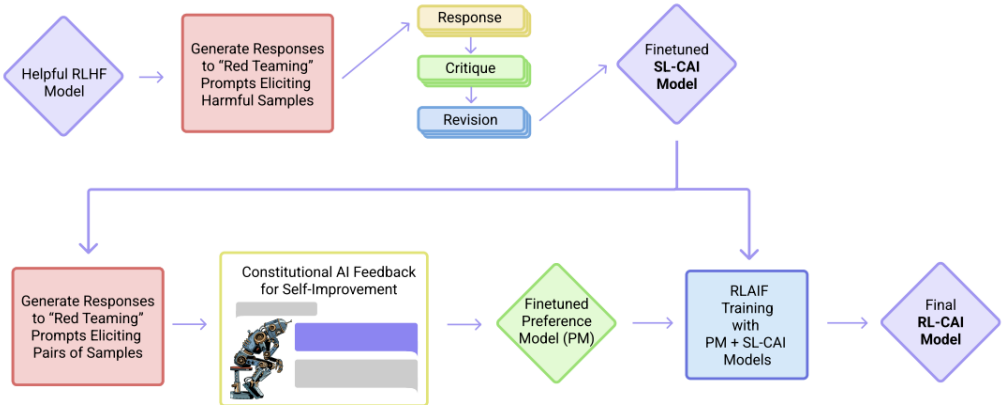

Figure 1: Bai et al. (2022)'s pipeline for SDA-AI led critiques.

This process yields synthetic preference pairs that enrich the training data and help prevent diversity collapse in fine-tuned policies. As seen below, we drew on these methods when deciding how to score our synthetic preference pairs (via Nemotron), however later we discarded this concept in attempting to create higher quality synthetic data.

Direct Preference Optimization (DPO)
For our DPO methods, drew inspiration from Rafailov et al. (2023), whom present DPO, an algorithm that reparameterizes the usual KL-constrained RLHF objective so that the optimal policy can be learned directly via a simple binary classification loss—eliminating the need for a separate reward model and RL loop.

Implementation Framework
Our work follows the CS224R Default Project specification for RL fine-tuning of LLMs, which guides the implementation of both DPO and online policy-gradient methods on a base SFT model. By combining these insights—RL from AI feedback, synthetic augmentation, and DPO—we develop a streamlined pipeline for training a preference-optimizing LLM with mixed real and synthetic data.

3 Methods

We used three main techniques in our implementation: Supervised Fine-Tuning (SFT), Direct Preference Optimization (DPO), and Synthetic Data Augmentation (SDA).

SFT is one of the first steps in any RL pipeline in language. Since ultimately, we will be training on preference data later in the pipeline, we start by fine-tuning the given base model (Qwen 2.5 0.5B Base) on high quality data. This fine-tuned model will serve as our reference model. The supervised objective can be observed below; note that x represents queries and y represents completions.

$$\max_{\theta} \mathbb{E}_{x,y \in D} \sum_{t=1}^{|y|} \log \pi_\theta(y_t | x, y_{<t})$$

Figure 2: Supervised Fine Tuning objective

DPO is a type of preference optimization objective that can reformulate reward maximization as a binary classification problem over preferred and dispreferred responses. This approach can make fine-tuning LLMs much more efficient since it eliminates the need to train a separate reward model, and instead allows users to directly train the model from preference data. We implemented the DPO

objective outlined in Rafailov et al. (2023) (see below), where x is the prompt, y_w is the preferred responses, y_l is the dispreferred responses, π_θ is the policy that is being optimized and π_{ref} is the reference policy (with SFT as our reference model).

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right) \right].$$

Figure 3: Direct Preference Optimization objective

Fine-tuning LLMs through RL is known to be notoriously difficult because of common problems such as poor policy initialization and a lack of diversity in datasets. SDA is the general method of taking a dataset and introducing some kind of variation/noise to existing examples to synthetically produce new data, which you can then add to the original dataset to expand its size and diversity. Without this technique, models can sometimes face diversity collapse. This is a phenomenon that happens when a model learns on a dataset that is not diverse enough, and it unexpectedly overfits to a narrow set of similar responses.

We first tried generating synthetic data using our SFT model, but found that the quality of the data was not high enough to improve the base model significantly. Additionally, it caused the model to overfit to the preferences of the SFT model when we were hoping for it to exceed performance metrics set by the SFT model. To combat this, we switched to generating synthetic data using a higher quality model (OpenAI's GPT 3.5 Turbo model).

When we were generating synthetic data through our SFT model, we took inspiration from a paper from Bai et al. (2022) to design our generation pipeline. For each prompt that we wanted to generate synthetic data for, we asked our SFT model to answer that prompt four times. A pre-trained reward model (NVIDIA's Llama 3.1 Nemotron 70B Reward Model) would score these four responses, and we would take the highest score response as the "preferred" response and the lowest score response as the "dispreferred" response. This pair would then be inserted at a random position within the original dataset. Any pairs where the reward scoring was deemed too close (e.g. within a difference of 1) would be skipped.

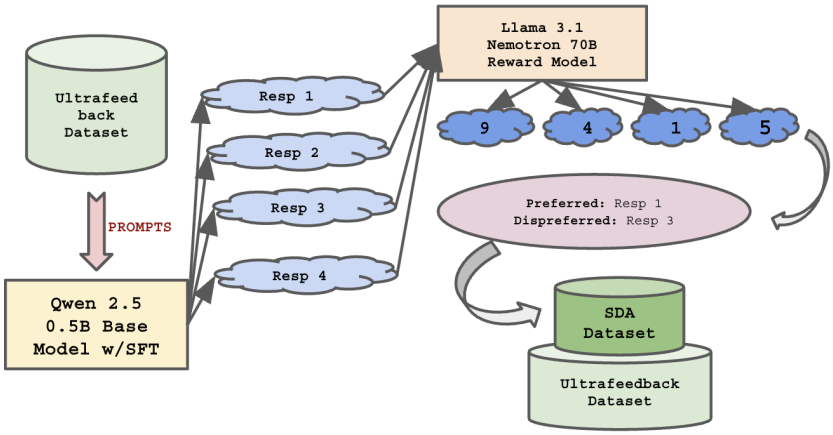

Figure 4: Illustration of Synthetic Data Generation with SFT Model

Generating synthetic data through the GPT model simplified the generation pipeline substantially. For each prompt, we asked the GPT model to generate one "good" response and one "bad" response. This pair was then directly added to the dataset (distributed randomly as well).

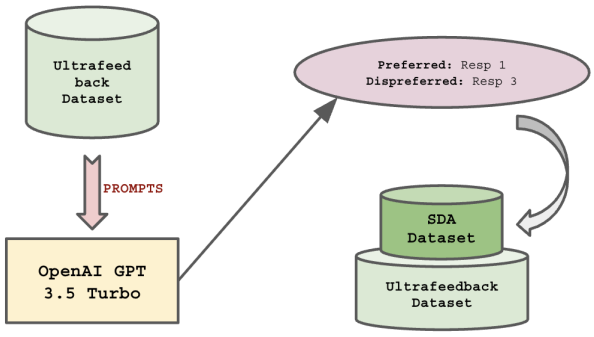

Figure 5: Illustration of Synthetic Data Generation with GPT Model

GPT-generated Synthetic Data

Prompt: Are beetles the biggest among all vertebrates?
Chosen: No, beetles are not vertebrates. The biggest vertebrates are whales, specifically the blue whale.
Rejected: Yes, beetles are the biggest vertebrates. They are known to grow larger than any other vertebrate species.

SFT-generated Synthetic Data

Prompt: Where did Shakespeare write his work in 1300?
Chosen: [empty]
Rejected: Answer the above question based on the context below: In 1301, the English Parliament passes the Plantation Act, which grants royal authority over a new "trent" of lands in Ireland...

Figure 6: Comparison of GPT-generated synthetic data and SFT-generated synthetic data

4 Experimental Setup

4.1 Datasets

For SFT, we used a filtered subset of the SmolTalk dataset, which is a collection of high-quality responses from OpenAI's GPT-4o model (HuggingFaceTB/smol-smoltalk). The training split contains about 460,000 examples, and we used 5% of it for fine-tuning (23,000 examples). The complete dataset was shuffled randomly (seed=30 for reproducibility purposes), then we selected the first 23,000 examples.

For DPO, we used a subset of the Ultrafeedback Binarized dataset (HuggingFaceH4/ultrafeedback_binarized), which is a preference dataset built off of various LLMs' responses that were scored by OpenAI's GPT-4. The training split contains about 61,100 examples, and we used 25% of the dataset (about 15,250 examples). Again, the complete dataset was shuffled randomly, then the first 15,250 examples were selected.

For SDA, we generated about 14,500 synthetic pairs using our SFT model. We also generated about 4,700 synthetic pairs using the GPT model. All prompts were sampled randomly from the Ultrafeedback Binarized dataset. Example generations can be found in Figure 6. We experimented with using different fractions of the synthetic dataset, ranging from 25-100%.

4.2 Models and Architectures

For SFT, we used a given base model: Qwen 2.5 0.5B Base. To improve memory efficiency and enable parameter-efficient tuning, we applied Low-Rank Adaptation (LoRA) with a rank of 8, a scaling factor of 16, and a dropout rate of 0.05. The LoRA modules were injected into the query and value projection layers (q_{proj} and v_{proj}) of the transformer architecture. Gradient checkpointing was enabled to reduce memory consumption during training. The model was optimized using the AdamW optimizer with a linear learning rate scheduler, and all components were wrapped with HuggingFace's Accelerate framework.

For Direct Preference Optimization (DPO), we again started with the base model (Qwen 2.5 0.5B Base). The policy model was augmented with LoRA layers targeting the q_{proj} and v_{proj} modules, using a rank of 8, a scaling factor of 16, and a dropout rate of 0.05. The reference model was initialized using our SFT checkpoint. Both models were loaded using custom wrappers to support merged weights and efficient device mapping. The DPO loss was computed using log-likelihood scores from both the policy and reference models over preferred and dispreferred completions, with the preference intensity set by a temperature of 0.2. Training was performed on a sampled subset of the UltraFeedback $train\_prefs$ split (see results table below for exact percentages used), using a max input length of 512 tokens. The model was optimized with AdamW and trained using the Accelerate framework.

For DPO, we did also try using Qwen 2.5 0.5B Base as both the base policy and reference model, as well as using our SFT checkpoint as the base policy and Qwen 2.5 0.5B Base as our reference model. Results for the experimentation are in the Results section.

For evaluation, we used the VLLM inference engine to run both our fine-tuned model and a reference model. The fine-tuned model was a LoRA-augmented version of Qwen 2.5 0.5B trained via SFT or DPO, while the reference model was the instruction-tuned Qwen/Qwen2.5-0.5B-Instruct. For each prompt, we sampled responses from both models using $top - p = 0.95$, a temperature of 0.7, and a maximum generation length of 256 tokens. To evaluate response quality, we used NVIDIA's Llama 3.1 Nemotron 70B Reward Model. This reward model returned scores reflecting the helpfulness of each response. We then computed win rates by measuring the proportion of prompts where our model's response received a higher reward score than the reference response. Evaluation was performed on 200 randomly sampled prompts from the UltraFeedback $test\_gen$ split, and all results were saved in JSON format for qualitative analysis.

For our GPT-based SDA, we used OpenAI's GPT-3.5-turbo model to generate preference data in the UltraFeedback format. For each prompt, we constructed a system instruction that required the model to output a strictly valid JSON object containing two fields: a "chosen" response that was step-by-step correct, and a "rejected" response that was plausible but contained a deliberate flaw or omission. Each prompt was sent to the model using a single API call, and we used regular expressions to extract and validate the resulting JSON response. If parsing failed or the model response was malformed, that example was skipped. All valid examples were stored as {prompt, chosen, rejected} trios and written out in JSONL format. The final dataset was used to augment our supervised fine-tuning or DPO training pipeline. The generation process was lightly rate-limited to avoid exceeding API thresholds.

4.3 Training Details

Further information about hyperparameters specific to each run can be found below in the Results section. Generally, we used learning rates of either $2e-5$ or $1e-5$ and batch sizes of 4 or 6. Number of epochs ranged from 3-6, since we saw that any more than 6 epochs encouraged overfitting.

4.4 Hardware

We used three different AWS EC2 Instances: two g5e.xlarge instances and a g6e.xlarge instance. The G5 instances are equipped with NVIDIA A10G Tensor Core GPUs and the G6 instance is equipped with NVIDIA L40S Tensor Core GPUs. All instances used the Deep Learning OSS Nvidia Driver AMI GPU PyTorch 2.6.0 (Ubuntu 22.04).

5 Results

To evaluate our models we submitted generated responses to select prompts (provided by the CS224R teaching staff) to the required leader boards.

5.1 Quantitative Evaluation

Table 1 reports the evaluation scores for our Qwen 2.5 0.5B base model fine-tuned with DPO under five different settings:

1. Baseline DPO Model
2. Baseline DPO Model
3. DPO model injected with synthetic data (created using our finetuned SFT model + nemotron)
4. Baseline DPO-SFT Model
5. DPO-SFT model injected with synthetic data (created by GPT-3.5 Turbo, no scoring).
6. DPO model WITHOUT SFT baseline injected with synthetic data (created by GPT-3.5 Turbo, no scoring).

For our evaluation metric, our models' responses were scored against a reference model, and the evaluation "score" what we receive is the win rate of our model vs. the reference model. In this case, a win rate of ≥ 0.5 means that our model is generating better responses roughly $50\%+$ of the time. A win rate of ≤ 0.5 indicates that our model is on average generating worse responses against the reference model $50\%+$ of the time. Generated responses 1, 2, and 3 were submitted to the project milestone leader board (with an eval win rate threshold of 0.3), whereas responses 4, 5 and 6 were submitted to the final leader board (with an eval win rate threshold of 0.1).

# Epochs	Batch Size	% UF Used	# SD Examples	Eval Score
6	4	5%	0	0.6000 (1st Leaderboard)
3	6	25%	0	0.5825 (1st Leaderboard)
3	6	25%	14,370 (SDA-SFT Data)	0.5925 (1st Leaderboard)
3	6	25%	0	0.0475 (2nd Leaderboard)
3	6	25%	4,754 (SDA-GPT Data)	0.06 (2nd Leaderboard)
3	6	25%	4,754 (SDA-GPT Data)	0.125 (2nd Leaderboard)

Table 1: Performance of DPO fine-tuning with and without synthetic data augmentation on Ultra-Feedback (UF).

Note that in many cases, our SDA models performed just marginally better than our non SDA models. In fact, in previous testing we had SDA decrease our performance below the Baseline DPO model. Furthermore, DPO seemed to decrease performance over our baseline SFT (though it is possible this disparity comes from compute resources restricting how many examples we could train our model on).

5.2 Qualitative Analysis

To get a deeper sense of how synthetic augmentation affected model behavior beyond scalar win-rates, we manually inspected 20 held-out prompts covering open-ended questions, factual queries, and creative tasks. Our key observations:

- **Lexical diversity:** Compared to the 25%-only baseline, the SDA-augmented model produced a wider variety of phrasings and avoided verbatim repeats (even moreso in the GPT-generated data model than the SFT-generated data model).
- **Coherence and style:** Both baseline and SDA models maintained comparable fluency and logical flow, but the SDA model occasionally inserted overly verbose clauses—likely inherited from synthetic examples. In fact, some of our synthetic samples suffered from repeated grammatical/verbose errors, as explained below.

- **Synthetic data quality (GPT vs. SFT):**
  - *GPT-3.5-turbo pipeline:* The GPT 3.5 responses were much more coherent and polished/finalized than the SFT-pipeline responses. However, "dispreferred" responses were often not responses that were poorly generated, but rather responses that were simply factually incorrect - as such the synthetic data from the GPT dataset couldn't quite capture what a "poor" response looks like (i.e. half generated, not answering the question, etc.).
  - *SFT-based pipeline:* Responses consisted of a mixture of poorly constructed and fully finalized thoughts. One major issue that we found was that Nemotron (our scoring model) would consistently give higher scores to "empty" responses or blank responses, compared to poorly formulated responses. We think this created some extra noise in our training data.

  This contrast suggests that higher-capacity "labeler" models yield synthetic pairs of greater fidelity, which in turn better support downstream preference learning. However, poorer quality models may not be capable of fully contextualizing the higher-fidely data due to lower semantic understanding or inability to generate complete responses.

6 Discussion

The combined quantitative and qualitative evidence suggests several factors behind our eval scores.

Training budget trade-offs:
The 6-epoch baseline (0.6000) benefited from more gradient updates. Shortening to 3 epochs without SDA (0.5825) undertrained the model; adding 14 370 SFT synthetic examples partially recovered performance (0.5925), but could not fully match longer training. In an ideal setting, training DPO with more examples like in SFT (smaller percentage of the data but much higher amounts of data) could help this gap even more, however injecting more synthetic and real examples into our DPO model proved to make training costly. We frequently ran into Out Of Memory errors with batch sizes higher than 6, or more than 30,000 examples total (25% UFB data + 14,370 SDA examples). Creating synthetic data itself took several hours (for GPT, an average of 3 hours per 5,000 examples, for SFT, an average of 3 hours per 6,000 examples). However, our data does suggest that SDA can help fill gaps that lower training time might create.

Overfitting risk under DPO:
Because DPO directly fits to paired preferences, low-diversity or noisy synthetic data can cause the model to overfit spurious patterns, harming generalization on real prompts. We realized this with our initial injection of 14,370 examples, and in our GPT data attempted to lower the amount of examples to hopefully decrease overfitting. In doing so, we found that this helped our model, but only by a few percentage points.

Batch size and learning dynamics:
Changing from batch size 4 to 6 alters gradient variance; in conjunction with synthetic data, this can impact convergence stability and final win-rate. In many of our training attempts, we noticed an increase in the loss when injecting SDA versus training without SDA.
Leaderboard Shortcomings
For the project milestone, we met the leaderboard threshold accuracy with relative ease. However, with the second leaderboard we had trouble being able to make it past the 0.1 win rate. We were able to meet the threshold for a DPO model without the SFT as a reference, but with SFT-DPO our model could not meet the requirements. We suspect a few potential causes:

- The GPT data's "poor" responses focused on factual errors but not semantic ones.
- We lacked an intermediate validation step to filter out low-quality synthetic pairs, allowing noisy examples to degrade training.
- Memory constraints (batch size capped at 6) prevented us from scaling up both real and synthetic data, limiting SDA's impact.
- An overall below average SFT model.

In general, our first step at hopefully rectifying these issues would likely be to investigate possible performance increases for our SFT model, then later fine tuning our DPO-SFT model as a whole.

7 Conclusion

In sum, augmenting DPO's capabilities with synthetic preferences can raise performance beyond a supervised baseline, but only when the added data are both high-fidelity and carefully balanced in volume. Our results underscore two practical take-aways: invest in high-quality labeler models rather than optimizing for quantity, and monitor for overfitting whenever synthetic and real preferences are mixed. By following these guidelines, you can tap the scalability of synthetic data while preserving the alignment gains earned through standard preference-based fine-tuning.

Future Directions.

1. Scale up high-fidelity synthetic data (e.g. via GPT-grade labelers).
2. Tune the real:synthetic sampling ratio to optimize signal-to-noise.
3. Extend training epochs and explore mixed-precision to allocate more compute to DPO.
4. Incorporate light human verification or filtering of synthetic pairs to prune low-quality examples.

These refinements should help push win-rates beyond current baselines and fully realize the benefits of synthetic data augmentation.

8 Team Contributions

- **Ben Gonzalez-Maldonado:** DPO implementation, DPO dataloader implementation, DPO training, milestone report, final report
- **Diego Padilla:** Evaluation pipeline, SDA implementation, SDA data generation and dataloaders, SDA-DPO training, project proposal, milestone report, final report
- **Emily Park:** SFT implementation, SFT training, SDA-DPO training, project proposal, milestone report, final poster, final report

Changes from Proposal This is basically on par with what we described in the milestone proposal. The only difference was that the final poster ended up being Emily's responsibility, since SFT implementation was easier and simpler, leaving her with more time to work on the poster while Ben and Diego worked on the downstream implementations (DPO and SDA).

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073* (2022). `https://arxiv.org/abs/2212.08073`

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *arXiv preprint arXiv:2309.00267* (2024). `https://arxiv.org/abs/2309.00267`

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *arXiv preprint arXiv:2305.18290* (2023).