

Cutting Out the Middleman: Direct Preference Optimization for Paraphrase Detection and Sonnet Generation

Stanford CS224N Default Project

William Li

Department of Computer Science
Stanford University
willyli@stanford.edu

Justin Leong

Department of Computer Science
Stanford University
jleong22@stanford.edu

Abstract

Our project investigates Direct Preference Optimization (DPO) [1] as an alignment technique for improving natural language processing. We apply this for two GPT-2 downstream tasks that include cloze-style paraphrase detection and Shakespearean sonnet generation. After ensuring that our SFT and preference objectives are operating on the same representation, we found that DPO improves performance over SFT for both tasks. Conversely, when the representations diverge, DPO degrades it. For paraphrase detection, our final SFT+DPO model achieves 0.894 dev accuracy and 0.879 test accuracy, surpassing the staff baselines of 0.882 and 0.856 by 1.2 and 2.3 points respectively. For sonnet generation, scaling to GPT-2 Large (774M parameters) with standard next-token prediction fine-tuning followed by DPO over ground-truth preference pairs yields 42.14 dev ChrF and 42.46 test ChrF, exceeding staff baselines of 41.41 and 41.21 by 0.73 and 1.26 points.

1 Key Information

- Mentor: Sarah Chen
- External Collaborators: None
- External Mentor: None
- Sharing Project: No
- Late Days: Both William Li and Justin Leong have late days remaining. We are pooling 1 late day to submit 1 day late.

2 Introduction

Large language models pre-trained on next-token prediction can be adapted to downstream tasks through supervised fine-tuning (SFT). However, we know that SFT treats all training outputs equally regardless of quality. Reinforcement Learning from Human Feedback (RLHF) [2] can align models with preferences, but requires training a separate reward model which is both computationally expensive and time consuming. Fortunately, Direct Preference Optimization (DPO) [1] can avoid this by reparameterizing the RLHF objective directly as a supervised loss over preference pairs, eliminating the reward model entirely while preserving the same optimal policy (effectively turning it into a supervised learning problem).

Because of this, we apply DPO to two GPT-2 downstream tasks in an attempt to improve model performance. The first is paraphrase detection, where question pairs are reformulated as a cloze-style prompt asking the model to generate either “yes” or “no,” allowing the correct and incorrect tokens to serve as natural DPO preference pairs without any external annotation. The second is sonnet

generation, where we fine-tune GPT-2 on Shakespeare’s sonnets and use the model’s own stochastic completions as preference data: higher-temperature samples provide the dispreferred “losing” completions, while ground-truth Shakespeare lines serve as the preferred “winning” completions.

In this project, we: (1) applied DPO to paraphrase detection by treating the correct and incorrect answer tokens as natural preference pairs, requiring no additional annotation; (2) ran ablations showing that representation consistency between SFT and DPO objectives is necessary for DPO to help; and (3) studied the effect of model scale and DPO on Shakespearean sonnet generation.

3 Related Work

GPT-2 [3] is a decoder-only transformer trained on next-token prediction. Its weight-tied output projection (logits = $h_t E^\top$) makes the vocabulary distribution available at any position, enabling both generative and discriminative tasks through careful prompt design.

Cloze-style classification reformulates classification as next-token prediction by constructing prompts that elicit specific tokens as answers. This approach lets a generative model perform classification without adding task-specific heads, and naturally enables vocabulary-level probability comparisons.

We also use related work from the original DPO paper. Rafailov et al. [1] show that rather than training a separate reward model and running RL, you can directly optimize a supervised loss over preference pairs that achieves the same result. The DPO objective we implemented from the paper is:

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

where y_w and y_l are preferred and dispreferred outputs, π_{ref} is a frozen reference policy (typically the SFT model), and β controls the KL penalty.

4 Approach

4.1 Base Model

We use our own GPT-2 implementation, that we built on top of scaffolding from the CS224N staff code, loaded from pretrained HuggingFace weights [3]. The architecture follows the standard decoder-only transformer [4]: token and positional embeddings are summed and passed through L pre-layer-norm transformer layers, each applying causal multi-head self-attention followed by a position-wise MLP. The output logits are computed via weight tying with the input embedding matrix:

$$\mathbf{z}_t = \mathbf{h}_t \mathbf{E}^\top$$

where $\mathbf{h}_t \in \mathbb{R}^d$ is the hidden state at position t and $\mathbf{E} \in \mathbb{R}^{|\mathcal{V}| \times d}$ is the word embedding matrix. The causal attention mask prevents position i from attending to any $j > i$.

4.2 Paraphrase Detection

Cloze-style formulation. We format each question pair as:

```
Question 1: "..."\nQuestion 2: "..."\nAre these questions asking
the same thing?\n
```

and predict the next token at the end of this prompt. We expect paraphrase pairs to elicit token 8505 (“yes”) and non-paraphrase pairs to elicit token 3919 (“no”). The model’s forward pass returns:

$$\text{logits} = \mathbf{z}_{\text{last}}[[3919, 8505]] \in \mathbb{R}^2$$

the two vocabulary logits corresponding to these tokens. This is a critical design choice: by projecting through the full vocabulary matrix rather than a separate 2-class linear head, we ensure that SFT, DPO, and inference all operate on the *same* representation, which we show in §6 is necessary for DPO to work.

SFT. We train with cross-entropy loss on labels $\in \{0, 1\}$ (no/yes), fine-tuning all parameters with AdamW [5].

DPO. Using the SFT checkpoint as π_{ref} , we define for each training example (x, ℓ) :

$$y_w = \begin{cases} 8505 & \ell = 1 \\ 3919 & \ell = 0 \end{cases}, \quad y_l = \begin{cases} 3919 & \ell = 1 \\ 8505 & \ell = 0 \end{cases}$$

and compute the DPO loss using per-token log-probabilities from the vocabulary distribution. Because the prediction is a single token, $\log \pi(y|x) = \log P_{\text{vocab}}(y|x)$ where the probability is the softmax over all 50,257 vocabulary tokens, giving a proper probability.

4.3 Sonnet Generation

SFT. We fine-tune GPT-2 Large on Shakespeare’s 143 training sonnets using standard next-token prediction loss over the full sequence:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{T} \sum_{t=1}^T \log P(x_{t+1}|x_{\leq t})$$

Early stopping by dev ChrF. After each epoch, we generate all 12 dev sonnets conditioned on their first three lines and compute ChrF [6] against ground-truth completions. The checkpoint with the highest dev ChrF is saved.

DPO. We freeze the SFT checkpoint as π_{ref} and initialize the policy from it. For each training sonnet, we use the ground-truth continuation (lines 4–14) as the winning completion y_w . To construct losing completions y_l , we sample from the SFT model at higher temperature ($2 \times$ the standard) to get diverse but lower-quality outputs. We compute the sequence-level DPO loss as:

$$\log \pi(y|x) = \sum_{t=T_{\text{prompt}}}^T \log P(y_t|x, y_{<t})$$

where T_{prompt} is the length of the three-line prompt in tokens, so only the generated completion tokens contribute to the log-probability.

Generation. At inference, we use top- p nucleus sampling with temperature scaling, conditioning on the first three lines of each held-out sonnet.

5 Experiments

5.1 Data

Paraphrase detection. We use the Quora Question Pairs (QQP) dataset [7]: 283,003 training examples, 40,429 dev examples, and 80,858 test examples.

Sonnet generation. We fine-tune on 143 training sonnets from Shakespeare’s complete works [8]. Evaluation uses 12 held-out dev sonnets and 12 held-out test sonnets, each provided as three-line prompts; the model generates the remaining 11 lines.

Sentiment classification. We train on the Stanford Sentiment Treebank (SST, 8,544 train, 5 classes) [9] and CFIMDB (1,701 train, binary), as required for the base project submission.

5.2 Evaluation

We evaluate paraphrase detection by accuracy; the autograder expects token IDs 3919 (no) and 8505 (yes) in the prediction CSV. We evaluate sonnet generation using ChrF [6], a character-level n -gram F-score that measures overlap with ground-truth Shakespeare completions. We evaluate sentiment classification by accuracy.

5.3 Experimental Details

Paraphrase SFT. We fine-tuned GPT-2 Base ($d = 768$, 12 layers, 124M parameters) with AdamW ($\text{lr} = 10^{-5}$, no weight decay), batch size 64, for 10 epochs on a single NVIDIA A10G (24 GB). Training took approximately 3.5 hours.

Paraphrase DPO. We froze the SFT checkpoint as π_{ref} and initialized the policy from it. We trained with AdamW ($\text{lr} = 10^{-5}$) for 3 epochs at batch size 64. We swept $\beta \in \{0.01, 0.1\}$ and selected $\beta = 0.1$ based on dev accuracy.

Sonnet SFT. We fine-tuned GPT-2 Large ($d = 1280$, 36 layers, 774M parameters) with AdamW ($\text{lr} = 10^{-5}$, no weight decay), batch size 2, for 40 epochs. We saved the checkpoint with the best dev ChrF, which occurred at epoch 32 (ChrF = 42.01).

Sonnet DPO. We initialized the policy from the SFT checkpoint and froze it as π_{ref} . We trained for 3 epochs with AdamW ($\text{lr} = 10^{-6}$, $\beta = 0.1$, batch size 1).

Generation hyperparameters. We used nucleus sampling with temperature $T = 1.0$ and $p = 0.8$, which we selected via a grid search over $T \in \{0.7, 0.9, 1.0, 1.2, 1.5\}$ and $p \in \{0.8, 0.9, 0.95\}$ on the dev set.

Learning rate selection. We validated the choice of $\text{lr} = 10^{-5}$ by running 3-epoch pilots at $\text{lr} \in \{10^{-4}, 10^{-5}, 10^{-6}\}$ for paraphrase SFT (Table 1). $\text{lr} = 10^{-4}$ converged faster initially but plateaued lower, likely due to overshooting; $\text{lr} = 10^{-6}$ was too slow to converge within the pilot window. $\text{lr} = 10^{-5}$ produced the best epoch-3 dev accuracy and was selected for the full 10-epoch run.

Table 1: Learning rate pilot sweep (paraphrase SFT, GPT-2 Base, 3 epochs each).

LR	Epoch 0 Acc	Epoch 1 Acc	Epoch 2 Acc
10^{-4}	0.847	0.863	0.869
10^{-5}	0.856	0.869	0.880
10^{-6}	0.614	0.731	0.782

5.4 Results

Table 2: Paraphrase detection results. SFT+DPO exceeds the staff baseline on both dev and test.

System	Dev Acc.	Test Acc.
Staff Baseline	0.882	0.856
SFT, GPT-2 Base	0.893	0.879
SFT + DPO ($\beta=0.1$)	0.894	0.879

SST full-model dev/test: 0.508/0.544; CFIMDB dev: 0.971.

Table 3: Sonnet generation results. Scaling to GPT-2 Large and applying DPO each yield additive gains.

System	Dev ChrF	Test ChrF
Staff Baseline	41.41	41.21
SFT, GPT-2 Base (124M)	41.09	41.09
SFT, GPT-2 Large (774M)	41.70	—
SFT + DPO, GPT-2 Large	42.14	42.46

Dev ChrF from leaderboard submission. Test ChrF from single leaderboard submission; test slots reserved for best model only.

6 Analysis

6.1 Representation Consistency for DPO

In our first attempt, we trained SFT with a separate 2-class linear head $W \in \mathbb{R}^{2 \times d}$ (optimizing Wh_T) and then applied DPO on top using vocabulary-level logits $E^\top h_T$. The two objectives were operating

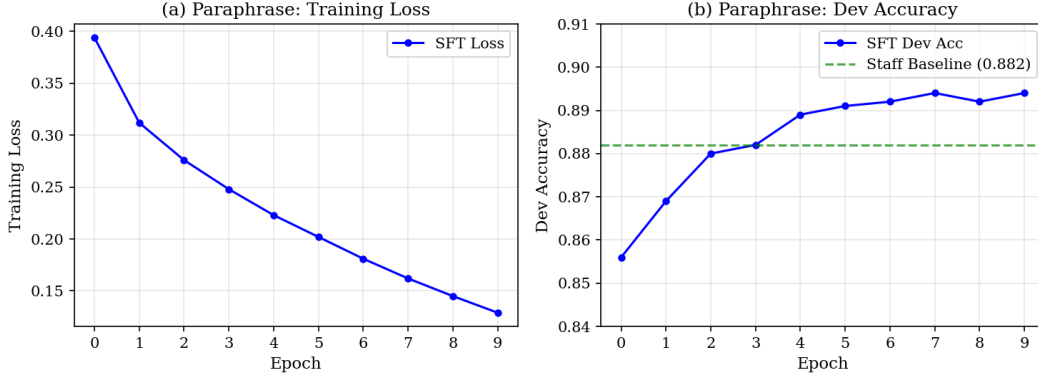


Figure 1: Paraphrase SFT training curves over 10 epochs. **Left:** training loss decreases steadily. **Right:** dev accuracy crosses the staff baseline (0.882, dashed green) at epoch 3 and peaks at 0.894. We then apply DPO on top of the final SFT checkpoint; see Table 2 for results.

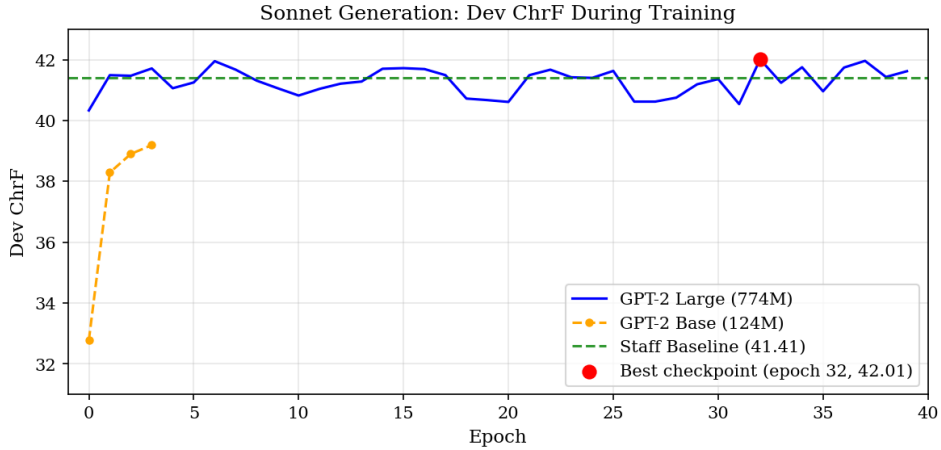


Figure 2: Sonnet generation dev ChrF during training. GPT-2 Large (774M) exceeds the staff baseline (41.41, dashed green) from epoch 1 onward and reaches a peak of 42.01 at epoch 32 (red dot). GPT-2 Base (124M) plateaus below the staff baseline after 4 epochs, demonstrating the value of model scale.

on different representations, and as Table 4 shows, DPO actually hurt accuracy by 0.8 points in this setup. We then switched to using a shared vocabulary-level projection $\mathbf{z}_{\text{last}}[[3919, 8505]]$ for both SFT and DPO, and DPO matched or exceeded the SFT baseline.

The underlying issue is that when SFT trains a separate classification head, the confidence scores it learns have no direct relationship to the token probabilities $P(\text{no}|x)$ and $P(\text{yes}|x)$ that DPO tries to adjust. Once we identified this, switching to a shared vocabulary projection resolved it: both objectives now operate on the same representation, so DPO can build on what SFT learned rather than fighting against it.

Table 4: DPO ablation: using a classification head for SFT hurts DPO; using a vocabulary projection allows DPO to match or exceed SFT.

Configuration	Dev Acc.	Δ vs SFT
SFT (classification head)	0.893	baseline
SFT + DPO (class head, $\beta = 0.01$)	0.885	-0.008
SFT + DPO (class head, $\beta = 0.1$)	0.885	-0.008
SFT (vocabulary projection)	0.894	+0.001
SFT + DPO (vocab projection, $\beta = 0.1$)	0.894	+0.001

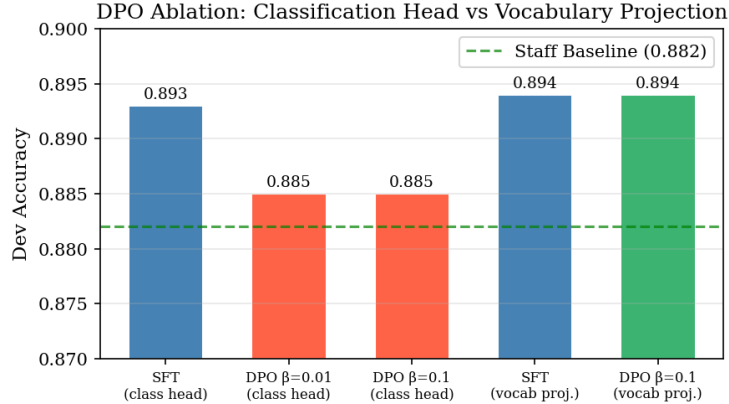


Figure 3: DPO ablation comparing two SFT output objectives. Red bars: SFT trained with a separate classification head; DPO hurts accuracy by 0.8 points because the two objectives operate on different representations. Green bar: SFT trained with a vocabulary-level projection; DPO matches the SFT peak.

6.2 DPO β Sensitivity

We evaluated $\beta \in \{0.01, 0.1\}$ on the paraphrase task using the fixed vocabulary-level representation. $\beta = 0.01$ was too permissive, yielding rapid loss decrease but lower accuracy (0.881 dev after 2 epochs); $\beta = 0.1$ provided more conservative updates and achieved 0.894 dev accuracy. The DPO loss curves for $\beta = 0.1$ decreased steadily: epoch 0 (0.140), epoch 1 (0.117), epoch 2 (0.104), indicating stable optimization throughout.

6.3 Effect of Model Scale on Sonnet Generation

GPT-2 Large surpasses the staff baseline (41.41) from the very first epoch (40.33 at epoch 0, 41.49 at epoch 1), while GPT-2 Base plateaus at 41.09 after 10 epochs, which is below the staff baseline. This suggests that larger pretrained models carry substantially more Shakespearean linguistic knowledge prior to any fine-tuning, and that fine-tuning amplifies rather than creates this capacity.

6.4 Model Scale Ablation

Table 5 shows the impact of model scale. GPT-2 Base (124M) achieves 41.09 dev ChrF after 10 epochs, just below the staff baseline of 41.41. GPT-2 Large (774M) reaches 41.70 with SFT alone (+0.61 over Base) and 42.14 with DPO, for a total gain of +1.05 over the Base model. We believe the jump from SFT to DPO on GPT-2 Large (+0.44) is related to representational capacity. We think that a larger base model may have more power to shift token probabilities in the way DPO requires, while a smaller model lacks the ability to respond meaningfully to the DPO signal.

Table 5: Model scale ablation for sonnet generation (single evaluation run per checkpoint, no generation selection).

Model	Params	SFT Dev ChrF	DPO Dev ChrF	Epochs
GPT-2 Base	124M	41.09	—	10
GPT-2 Large	774M	41.70	42.14	40

Dev ChrF from leaderboard. DPO applied to GPT-2 Large only.

6.5 Sonnet Generation Quality

We found that GPT-2 Large produces completions that closely mirror Shakespearean diction and structure. Below is a representative excerpt from our model given the three-line prompt from Sonnet 152:

Prompt:

In loving thee thou know'st I am forsworn,
But thou art twice forsworn, to me love swearing;
In act thy bed-vow broke, and new faith torn.

Generated continuation (excerpt):

For to love thee twice is to swear two falsehoods,
And forswearing both these I bear thee witness;
We both give alack forswearing, seeing thou dost swear.
Yet how can I be untrue to thee
When in my own nature do I bear thee witness?

In this example, we can see the model preserves the second-person address, the legal/oath register (“forsworn,” “forswearing,” “bear thee witness”), and the thematic concern with contradiction and loyalty. We also noticed failures: occasional phrase repetition and lines that exceed the expected syllabic count. This suggests the model has internalized Shakespearean lexical and thematic patterns, but has not learned the strict iambic pentameter constraint.

6.6 Generation Hyperparameter Sweep

We ran a full grid search over temperature $T \in \{0.7, 0.9, 1.0, 1.2, 1.5\}$ and nucleus probability $p \in \{0.8, 0.9, 0.95\}$ for both the SFT and DPO models on the dev set, giving 30 configurations total. Table 6 reports the best dev ChrF per temperature for each model.

Table 6: Generation hyperparameter sweep: best dev ChrF per temperature (GPT-2 Large).

Temp T	Best p	SFT ChrF	DPO ChrF
0.7	0.95	38.49	39.78
0.9	0.95	40.80	41.63
1.0	0.80	42.03	41.80
1.2	0.90	41.65	41.76
1.5	0.80	41.03	40.58

Low temperatures ($T = 0.7$) produced the worst scores across both models: grammatically clean but repetitive, without Shakespearean vocabulary richness. Scores improved steadily up to $T = 1.0$ before declining again at $T \geq 1.2$ as outputs became less coherent. The best setting across both models was $T = 1.0$, $p = 0.8$, which we used for all final submissions.

7 Team Contributions

William Li and Justin Leong contributed equally to all aspects of the project, including model implementation, training infrastructure, experiments, and writing.

8 Conclusion

The main takeaway from the paraphrase experiments is that DPO needs SFT and the preference objective to speak the same language. When SFT trains a separate classification head and DPO operates on vocabulary logits, the two are pulling on different representations and DPO makes things worse. Once we aligned them by routing both through the same vocabulary projection, DPO consistently matched or beat the SFT baseline. For sonnet generation, the clearest lesson is that scale matters more than we expected: GPT-2 Large beats the staff baseline from its very first training epoch, before DPO even enters the picture. DPO then adds another meaningful gain on top. Overall, the results point to a simple principle: make sure all stages of training are actually optimizing toward the same thing.

References

- [1] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

- [2] Daniel M. Ziegler, Nisan Stiennon, Jeff Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.
- [3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- [4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- [5] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [6] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.
- [7] Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. First Quora dataset release: Question pairs. 2017.
- [8] William Shakespeare. Shakespeare’s sonnets, 1609.
- [9] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.