

SMART Regularization and DPO for GPT-2 Sonnet Generation

Stanford CS224N Default Project

Mac Broido

Department of Computer Science
Stanford University
mbroido@stanford.edu

Abstract

Finetuning small models with limited data is a challenging yet highly applicable setting: many use cases require small models due to cost and latency constraints, and generating substantial training datasets is often prohibitively costly and time-consuming. SFT can overfit on small datasets, and maximum-likelihood training can fail to produce high-quality outputs. In addition to the base default project tasks, we compare DPO[1], SMART[2], and their combination against an SFT baseline on a sonnet generation task with a small dataset. SMART regularization alone provides a modest gain, raising the dev-set ChrF score from 40.53 (SFT) to 41.82 ($\epsilon=10^{-5}$, $\lambda_s=0.5$, $\mu=0$). DPO yields a large improvement with ChrF 42.05 ($\beta=1$, $\text{lr} = 10^{-5}$). Combining SMART with DPO reaches the highest ChrF of 42.21 ($\epsilon=10^{-5}$, $\lambda_s=2$), slightly outperforming DPO alone. A pairwise LLM judge evaluation suggests DPO may produce the highest-quality outputs, though the rate of ties is high.

1 Key Information

- Mentor: Simon Kim
- External Collaborators: No
- Sharing project: No
- Late Days: 3

2 Introduction

Finetuning small language models with limited data is a common practical setting. Production constraints on cost and latency often mandate small models, and curating large, high-quality datasets can be prohibitively expensive and time-consuming. Standard supervised finetuning (SFT) with cross-entropy can overfit in this regime, and even when it generalizes, maximum-likelihood training provides only a coarse signal: it rewards token-level match to the reference but has no explicit notion of output quality.

Sonnet generation provides a suitable testbed for this scenario. A good Shakespearean-style sonnet must satisfy multiple interacting constraints: iambic pentameter, rhyme scheme, thematic coherence, and a closing couplet, none of which are directly captured by next-token prediction. An SFT model can pick up surface-level style from a small corpus but consistently fails on the structural and semantic requirements. This motivates training objectives that encode richer notions of quality.

We study two methods, independently and combined, in the context of GPT-2 sonnet generation. Direct Preference Optimization (DPO) uses pairwise preference data to steer the model toward higher-quality outputs without a separate reward model. SMART regularization encourages smooth,

robust representations by penalizing output divergence under worst-case embedding perturbations. We evaluate each method independently and their combination against an SFT baseline.

DPO produces large gains, SMART provides modest improvement without requiring Bregman anchoring, and combining the two yields the highest ChrF but falls short of DPO under an LLM judge and on the held-out test set.

3 Related Work

DPO[1] sidesteps the reward-modeling and on-policy sampling stages of RLHF by reparameterizing the reward function in terms of the optimal policy, yielding a simple classification-style loss over preference pairs. SMART[2] regularizes finetuning by finding worst-case perturbations in the input embedding space via projected gradient ascent and penalizing the symmetric KL divergence between clean and perturbed outputs, optionally augmented with Bregman proximal point regularization to limit parameter drift. Both methods are well-studied independently, but their combination within a single training objective is underexplored.

Recent work has introduced adversarial ideas into preference optimization, though through mechanisms distinct from input-space smoothness regularization.

Adversary-Aware DPO (ADPO)[3] is a close analog to our approach. It integrates PGD-style adversarial perturbations directly into the DPO training loop, perturbing vision inputs to construct worst-case preference pairs for safety alignment of vision-language models. The perturbation-into-DPO-loss approach is similar to our SMART+DPO, but operates on image inputs for jailbreak robustness rather than text embeddings for general generation quality.

GAPo[4] replaces DPO’s static reference policy with a dynamically constructed pessimistic anchor (adversarial perturbation of the current policy within a small KL-ball). Where GAPo creates perturbations in policy distribution space to filter noisy preferences, SMART+DPO perturbs input embeddings to encourage representational smoothness.

GANPO[5] takes a similar approach by adding a GAN-style discriminator on latent representations to penalize divergence between the policy and reference model beyond token-level KL, arguing that standard KL does not capture semantic similarity well. Similar to SMART, GANPO regularizes DPO in a space richer than token distributions, but through an adversarial discriminator enforcing structural consistency between policy and reference representations rather than through input perturbation.

Our work differs from all three in applying SMART’s embedding-space smoothness regularization as an auxiliary loss during DPO training for text generation, and in studying how these objectives interact in a low-data setting.

4 Approach

4.1 SMART Regularization

First we implement SMART regularization[2], which encourages the model to produce similar output distributions for inputs that are close together in embedding space. The SMART loss is $\mathcal{L}_{\text{SMART}} = \text{SymKL}(f(x + \delta^*), f(x))$, where f is the model, x is the input embedding, and δ^* is the worst-case perturbation found by solving $\delta^* = \arg \max_{\|\delta\|_\infty \leq \varepsilon} \text{SymKL}(f(x + \delta), f(x))$. We use symmetric KL divergence $\text{SymKL}(P, Q) = \text{KL}(P\|Q) + \text{KL}(Q\|P)$ as the smoothness measure, so that neither the clean nor the perturbed output is privileged as ground truth.

The inner maximization is solved approximately via projected gradient ascent. We initialize $\delta_0 \sim \mathcal{N}(0, \sigma^2 I)$ and iterate: compute $g = \nabla_{\delta} \text{SymKL}(f(x + \delta_t), f(x))$ with $f(x)$ detached, normalize the gradient $\hat{g} = g / (\|g\|_\infty + 10^{-8})$, take a step $\delta' = \delta_t + \alpha \cdot \hat{g}$, and project back onto the ℓ_∞ ball via $\delta_{t+1} = \text{clamp}(\delta', -\varepsilon, \varepsilon)$. Perturbations operate in the embedding space (after word and position embeddings, before the transformer layers), so they capture semantically meaningful directions rather than discrete token swaps, and the ℓ_∞ projection bounds perturbation magnitude across embedding dimensions. In the final SMART loss computation, $f(x)$ is not detached, so gradients flow through both the perturbed and clean forward passes: the model is trained to reduce the divergence caused by the worst-case perturbation it just found.

The total SFT training objective becomes $\mathcal{L} = \text{CE}(f(x), y) + \lambda_s \cdot \mathcal{L}_{\text{SMART}}$. We add Bregman proximal point regularization $\mu \cdot \text{KL}(f_{\tilde{\theta}} \| f_{\theta})$, where $\tilde{\theta}$ is an exponential moving average of the model weights updated as $\tilde{\theta} \leftarrow \gamma \tilde{\theta} + (1 - \gamma)\theta$. This penalizes drift from the model’s own recent trajectory, complementing SMART’s local smoothness in input space.

For our implementation, we referenced two open-source repositories: the paper authors’ `namisan/mt-dnn` and `archinetai/smart-pytorch`, which addresses some shortcomings of `mt-dnn`. We adapt the perturbation logic from `smart-pytorch` and the EMA class from `mt-dnn`, inlining as functions. Key deviations include using `reduction='batchmean'` to keep λ_s batch-size-invariant, replacing `mt-dnn`’s BERT-specific perturbation interface with a generic `eval_fn` callback, and adding `inputs_embeds` plumbing through our GPT-2 implementation to support embedding-space perturbation. The Bregman proximal point training loop is custom, as neither reference repo provides this end-to-end. We perform a hyperparameter sweep over ε , λ_s , and μ , independently ablating each component (Table 2).

4.2 Direct Preference Optimization (DPO)

We implement DPO[1], following the reference implementation from the paper authors’ `eric-mitchell/direct-preference-optimization`. DPO starts from an SFT checkpoint and creates two copies: a frozen reference policy π_{ref} and a trainable policy π_{θ} , initialized identically. Training data consists of (prompt, preferred completion, dispreferred completion) triples. We construct these by splitting each training sonnet into a 3-line prompt and using the ground-truth continuation as the preferred completion and the SFT completion as dispreferred.

Standard RLHF optimizes $\max_{\pi} \mathbb{E}_{x \sim \mathcal{D}, y \sim \pi} [r(x, y)] - \beta \cdot \text{KL}(\pi \| \pi_{\text{ref}})$, requiring a separate reward model and on-policy sampling. The key insight for DPO is that the reward can be reparameterized in terms of the optimal policy: $r^*(x, y) = \beta \log(\pi^*(y|x)/\pi_{\text{ref}}(y|x)) + \beta \log Z(x)$, where $Z(x)$ is a partition function that cancels in the Bradley-Terry preference model, allowing the loss to be written entirely in terms of the policy and reference.

For each sequence we compute per-completion log-probabilities $\log \pi(y|x) = \sum_{t \in \text{completion}} \log P(y_t | y_{<t}, x)$, masking prompt tokens with -100 in labels so that only completion tokens contribute. These are computed via `log_softmax` over logits, gathering the log-probability of the actual token at each position, then summing over non-masked positions.

Given chosen (y_w) and rejected (y_l) completions, we compute the log-probability ratios under the policy and reference:

$$\mathcal{L}_{\text{DPO}} = -\log \sigma \left(\beta [(\log \pi_{\theta}(y_w|x) - \log \pi_{\text{ref}}(y_w|x)) - (\log \pi_{\theta}(y_l|x) - \log \pi_{\text{ref}}(y_l|x))] \right) \quad (1)$$

The term inside the sigmoid is the implicit reward margin: how much more the policy rewards the chosen completion over the rejected one relative to the reference. β controls the strength of the KL constraint: higher values keep the policy closer to the reference, while lower values allow greater deviation. When the policy already strongly prefers y_w over y_l (relative to the reference), the sigmoid saturates and the loss vanishes. When it incorrectly prefers y_l , the loss pushes the model to increase $\log \pi_{\theta}(y_w|x)$ and decrease $\log \pi_{\theta}(y_l|x)$. The reference model anchors the loss to preference shifts rather than absolute log-probabilities, preventing mode collapse and excessive distribution shift.

The collator concatenates chosen and rejected sequences along the batch dimension: a batch of size B becomes $2B$ sequences in a single forward pass, then split by the number of chosen examples. We omit label-smoothing and IPO variants for simplicity. We sweep over β and the learning rate (Table 3).

4.3 SMART + DPO

We combine SMART and DPO by adding SMART perturbation to the policy model’s forward pass within the DPO training loop. During each step, we extract the policy model’s embeddings, route the forward pass through `inputs_embeds`, and compute the SMART loss over the perturbed outputs. This loss is added to the DPO objective weighted by λ_s , so the combined loss is $\mathcal{L}_{\text{DPO}} + \lambda_s \mathcal{L}_{\text{SMART}}$. We perform a grid search over ε and λ_s while holding the best DPO hyperparameters fixed (Table 4).

4.4 Baselines

We use two baselines representative of what initial attempts might look like given this task and setting. The first is zero-shot GPT-2 (base): 12 transformer layers, 768 hidden dimensions, 12 attention heads ($\sim 124\text{M}$ parameters), loaded directly from Hugging Face with no finetuning. Completions are generated via nucleus sampling (temperature = 1.2, top- p = 0.9). This establishes a lower bound measuring how well a general-purpose language model can produce sonnets without any exposure to Shakespeare during finetuning.

The second baseline is supervised finetuning (SFT) of the same GPT-2 (base) with standard cross-entropy next-token prediction. We train for 10 epochs with learning rate 10^{-5} , batch size 8. Generation uses the same nucleus sampling configuration as the zero-shot baseline. This establishes what straightforward finetuning on in-domain data achieves, serving as the comparison point for SMART, DPO, and their combination.

5 Experiments

5.1 Data

Base sonnet corpus. We use Shakespeare’s 154 sonnets from the Folger Shakespeare Library. The corpus is split into 131 training sonnets, a 12-sonnet dev set, and a 12-sonnet test set. All text is tokenized with the GPT-2 tokenizer.

SFT data. Full sonnets are tokenized as single sequences under the standard next-token prediction objective, padded to equal length within each batch.

DPO preference pairs. For each training sonnet we create a preference pair from a 3-line prompt: the ground-truth continuation serves as the preferred response and the SFT completion as the dispreferred response, so the model learns to shift probability mass toward Shakespeare-like continuations.

Figure 1 illustrates a representative preference pair from Sonnet 18. The preferred continuation sustains the summer metaphor across all eleven lines, maintains iambic pentameter, follows the ABAB CDCD EFEF GG rhyme scheme, contains a closing couplet and is overall coherent. The dispreferred SFT completion, by contrast, exhibits incoherent semantics (“rich preservatives when I sip,” “Resunches”), erratic line lengths with no discernible meter, no rhyme scheme, and abandons the prompt’s imagery.

Prompt (Sonnet 18, lines 1–3):

Shall I compare thee to a summer’s day? / Thou art more lovely and more temperate. / Rough winds do shake the darling buds of May,

Preferred (Shakespeare):

And summer’s lease hath all too short a date. / Some- Are sprung with rich preservatives when I sip / Not time too hot the eye of heaven shines, / And often that lame mock’sr contentment. / Supposing thou still is his gold complexion dimmed; / And every fair least heed / Or, where interest seems fragile, wroth from fair sometime declines, / By chance or nature’s appears / Against him, that meet rewd do burn. / changing course untrimmed. / But thy eternal sum- Then looked retreat’s from gloom might sweet verse. mer shall not fade / Nor lose possession of that fair / Then ivy shed not th’ effrontery thou art ill loved. / thou ow’st, / Nor shall Death brag thou wand’rest Resunches still evoke maiden splendour sing’d and in his shade, / When in eternal lines to time thou strong, fair, blessed on. / I fall about striving for grow’st. / So long as men can breathe or eyes can younger versions / So deficient in memory dwell my see, / So long lives this, and this gives life to thee. day, that my creations fade. / Settle wait’d I on thee

Dispreferred (SFT model):

Figure 1: A DPO preference pair from Sonnet 18. The SFT completion exhibits incoherent semantics, broken meter, no rhyme scheme, and loss of thematic continuity.

Choice of SFT completions as dispreferred. We use SFT completions rather than zero-shot GPT-2 outputs as dispreferred examples because zero-shot outputs degenerate into gibberish with no sonnet structure, making them trivially distinguishable from ground truth and providing a weak learning signal. SFT outputs occupy the right stylistic region but still exhibit broken meter and awkward

phrasing, enabling DPO to learn more fine-grained distinctions between passable and genuinely good Shakespearean verse.

5.2 Evaluation method

We evaluate on the 12-sonnet dev set.

Our primary automated metric is ChrF (character n -gram F -score): the generated output is concatenated with the prompt and scored against the full ground-truth sonnet. ChrF measures surface overlap with a single reference but cannot assess whether an output is a structurally sound or stylistically convincing sonnet, so we complement it with qualitative inspection and an LLM judge evaluation.

For the LLM judge evaluation, we select the best checkpoint from each training approach (SFT, SMART, DPO, SMART+DPO) along with the zero-shot baseline and compare all $\binom{5}{2} = 10$ model pairs on each of the 12 dev prompts. The judge model is Gemini 3.1 Flash Lite, prompted as an expert poetry critic scoring on three axes: *poetic quality* (iambic pentameter, ABAB CDCD EFEF GG rhyme scheme, Elizabethan diction), *coherence* (thematic and logical continuity from the prompt), and *fluency* (grammaticality and naturalness). The judge returns a structured JSON verdict (“A”, “B”, or “tie”) with a one-sentence justification. To control for position bias, which model appears as “Sonnet A” vs. “Sonnet B” is randomized per pair via coin flip. Results are aggregated into a pairwise win-rate matrix and overall win rates.

5.3 Experimental details

All experiments use a custom AdamW optimizer with $\beta_1=0.9$, $\beta_2=0.999$, $\varepsilon_{\text{Adam}}=10^{-6}$, and weight decay 0.0, with a constant learning rate. SFT trains for 10 epochs with learning rate 10^{-5} and batch size 8; DPO and SMART+DPO each train for 5 epochs with batch size 8. For SMART perturbation, δ_0 is initialized from $\mathcal{N}(0, \sigma^2 I)$ with $\sigma=10^{-5}$ with step size $\alpha=10^{-3}$ and a single inner maximization step ($K=1$); the EMA decay for Bregman proximal point regularization is $\gamma=0.999$. The gradient is normalized by its ℓ_∞ norm (plus 10^{-8} for numerical stability), then δ is updated with step size α and projected onto the ℓ_∞ ε -ball via clamping. DPO dispreferred completions are generated via nucleus sampling; each dispreferred completion is a single sample with no best-of- N or repeated sampling. DPO hyperparameters are swept in two sequential phases: Phase 1 sweeps $\beta \in \{0.1, 0.5, 1, 2\}$ with $\text{lr}=10^{-6}$; Phase 2 sweeps $\text{lr} \in \{10^{-6}, 5 \times 10^{-6}, 10^{-5}\}$ with the best β from Phase 1. SMART hyperparameters are swept in three sequential phases: Phase 1 sweeps $\varepsilon \in \{10^{-6}, 10^{-5}, 10^{-4}\}$ with $\lambda_s=1$, $\mu=0$; Phase 2 sweeps $\lambda_s \in \{0.1, 0.5, 1, 2, 5\}$ with the best ε from Phase 1; Phase 3 sweeps $\mu \in \{0, 1, 5\}$ with the best ε and λ_s discovered. In the SMART+DPO setting, we perform a grid search over $\varepsilon \in \{10^{-6}, 10^{-5}, 10^{-4}\}$ and $\lambda_s \in \{0.5, 1, 2, 5\}$ while holding DPO hyperparameters at their best values ($\beta=1$, $\text{lr}=10^{-5}$); Bregman proximal point regularization is not used as it did not improve performance for SMART alone and DPO already provides anchoring via the reference policy. ChrF is computed at the corpus level over all 12 dev sonnets with a single generation per prompt and no averaging over multiple seeds. All experiments use a fixed random seed.

5.4 Results

Table 1: Sonnet generation ChrF scores

Model	Dev	Test
Zero-shot Baseline	39.445	–
SFT Baseline	40.528	–
SMART ($\varepsilon=10^{-5}$, $\lambda_s=0.5$, $\mu=0$)	41.817	–
DPO ($\beta=1$, $\text{lr}=10^{-5}$)	42.047	41.947
SMART+DPO ($\varepsilon=10^{-5}$, $\lambda_s=2$)	42.214	41.081

Table 2: SMART regularization hyperparameter sweeps (dev set). Phase 1 sweeps ε ($\lambda_s=1, \mu=0$); Phase 2 sweeps λ_s ($\varepsilon=10^{-5}, \mu=0$); Phase 3 sweeps μ ($\varepsilon=10^{-5}, \lambda_s=0.5$).

ε	ChrF	λ_s	ChrF	μ	ChrF
10^{-6}	40.304	0.1	39.767	0	41.817
10^{-5}	40.906	0.5	41.817	1	40.030
10^{-4}	40.650	1	40.906	5	41.697
		2	38.113		
		5	40.346		

Table 3: DPO hyperparameter sweeps (dev set). Phase 1 sweeps β ($\text{lr} = 10^{-6}$); Phase 2 sweeps learning rate ($\beta=1$).

β	ChrF	Learning Rate	ChrF
0.1	40.817	10^{-6}	41.367
0.5	41.344	5×10^{-6}	41.287
1	41.367	10^{-5}	42.047
2	41.344		

Table 4: SMART+DPO grid search over ε and λ_s (dev set, $\text{lr}=10^{-5}, \beta = 1$)

ε	λ_s			
	0.5	1	2	5
10^{-6}	41.943	41.655	42.150	41.529
10^{-5}	41.401	42.084	42.214	41.193
10^{-4}	41.737	41.972	41.753	41.025

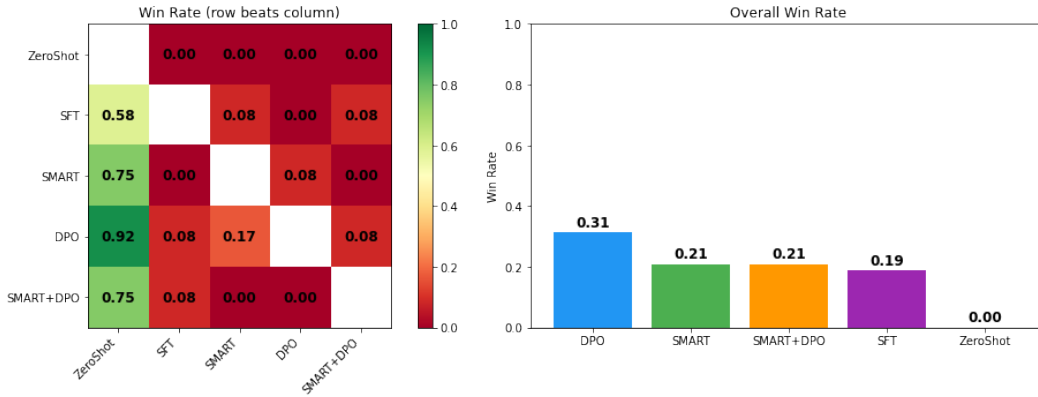


Figure 2: Pairwise win-rate matrix (left) and overall win rates (right) from the LLM judge evaluation. Win rate is the fraction of judgments won against the column model.

Table 5: Classifier accuracy on SST and CFIMDB

Task	Baseline	Ours
SST Last-Linear Dev Acc	.462	.453
SST Last-Linear Test Acc	–	.453
SST Full Model Dev Acc	.513	.515
SST Full Model Test Acc	–	.544
CFIMDB Last-Linear Dev Acc	.861	.878
CFIMDB Full Model Dev Acc	.976	.984

Table 6: Paraphrase detection accuracy

Task	Baseline	Ours
Paraphrase Dev Acc	.882	.892
Paraphrase Test Acc	.856	.877

6 Analysis

SMART regularization provides modest gains. The best SMART configuration ($\varepsilon=10^{-5}$, $\lambda_s=0.5$, $\mu=0$) achieves a ChrF of 41.82, a modest improvement over the SFT baseline (40.53). Notably, Bregman proximal point regularization does not help: the best result occurs at $\mu=0$, and adding $\mu=1$ degrades performance substantially (40.03), though $\mu=5$ partially recovers (41.70). This suggests that at the right perturbation budget and weight, SMART’s smoothness objective provides a useful regularization signal on its own, but Bregman’s EMA anchor over-constrains optimization in this setting. Qualitative inspection of outputs at $\mu=5$ revealed generations resembling the incoherent, near-gibberish quality of zero-shot outputs, suggesting that the EMA constraint prevented the model from updating enough to meaningfully learn from the training data.

DPO provides a large improvement. DPO yields the strongest single-method gain, raising ChrF from 40.53 to 42.05 at $\beta=1$, $\text{lr}=10^{-5}$. Performance is relatively stable across $\beta \in \{0.5, 1, 2\}$ but drops at $\beta=0.1$, suggesting that some regularization toward the reference policy is needed but the exact strength matters less beyond a very small amount. The higher learning rate (10^{-5}) outperforms the lower rates (10^{-6} , 5×10^{-6}), indicating that with only 131 preference pairs the model benefits from larger updates to shift probability mass within the limited training budget.

SMART+DPO yields the highest ChrF but limited qualitative gain. A grid search over ε and λ_s finds that combining SMART with DPO slightly outperforms DPO alone by ChrF: the best combination ($\varepsilon=10^{-5}$, $\lambda_s=2$, ChrF 42.21) edges out DPO (42.05). The best results within each ε tend to occur at moderate λ_s (1–2), while high $\lambda_s=5$ consistently degrades performance, suggesting that the smoothness objective helps in small doses but dominates the DPO signal when weighted too heavily. The LLM judge evaluation does not clearly differentiate SMART+DPO from other methods: most pairwise comparisons result in ties, making it difficult to determine whether the ChrF gain reflects genuine quality improvement or closer lexical match to the reference.

LLM judge evaluation. The LLM judge evaluation (Figure 2) gives DPO the highest overall win rate (0.31), followed by SMART and SMART+DPO (both 0.21), SFT (0.19), and ZeroShot (0.00), where win rate is computed as wins divided by total judgments (wins + losses + ties) across all pairwise matchups for each model. However, the most striking feature of the results is the prevalence of ties: most pairwise comparisons result in a tie judgment, and the non-zero win rates are driven by a small number of decisive comparisons. DPO’s clearest advantage is against ZeroShot (0.92), with only modest differentiation against the other finetuned models (0.08 against SFT, 0.17 against SMART, 0.08 against SMART+DPO). SMART, SMART+DPO, and SFT are largely indistinguishable to the judge, with pairwise win rates of 0.00–0.08 among them. Given the small evaluation set (12 dev prompts, 120 total judgments) and the high tie rate, these results should be interpreted cautiously:

they are consistent with DPO producing modestly better outputs, but the evidence is not strong enough to draw firm conclusions about the relative quality of the other methods.

Limitations of ChrF evaluation. ChrF measures character n -gram overlap with a single reference sonnet, which combines two different failure modes: generating text that is unlike Shakespeare in general, and generating text that is a plausible sonnet but happens to differ from Shakespeare’s exact writing. Because the first 3 lines of each generation match the reference exactly, all scores are inflated by a shared constant that contributes substantial n -gram overlap regardless of output quality. ChrF also cannot assess structural properties central to sonnet quality such as rhyme scheme, iambic pentameter or stylistic features of Shakespearean verbiage and imagery. A more complete evaluation would include additional automated analysis of these attributes and, for best results, expert human annotation.

Low-data regime dynamics. With only 131 training and 12 dev sonnets, all experiments operate in an extremely low-data setting. DPO’s pairwise contrastive signal appears more sample-efficient, with each preference pair providing a direct gradient toward preferred outputs. SMART’s perturbation-based smoothness must discover useful structure in the embedding space. In this setting, the perturbation directions may not generalize well enough to improve quality. This suggests that in data-scarce settings, methods that directly encode task-relevant supervision (preference pairs) may outperform methods that impose generic geometric priors (embedding smoothness). DPO also has the most straightforward path to further improvement: the preference dataset can be cheaply expanded by sampling the SFT model with different decoding hyperparameters or corrupting ground-truth completions. One interesting research direction that could be applicable here is online or iterative DPO methods, where dispreferred examples can be iteratively refreshed with generations from the current policy, enabling continued improvement without additional data annotation or manipulation efforts. While all methods would benefit from more comprehensive hyperparameter search, DPO’s data-driven nature means that scaling the preference set is likely to yield larger marginal gains than further tuning ε or λ_s for SMART, further underscoring its practical advantage. Moreover, the small evaluation set (12 dev sonnets) means that ChrF differences on the order of ~ 0.2 points (e.g., DPO vs. SMART+DPO) may not be meaningful; we did not perform significance testing or variance analysis across random seeds, so the rankings reported here should be treated as suggestive rather than conclusive.

7 Conclusion

We compared DPO, SMART regularization, and their combination for finetuning GPT-2 on Shakespeare sonnet generation. DPO provided the largest single-method improvement over SFT, demonstrating that pairwise preference data can effectively guide a small model toward higher-quality outputs even with a minimal dataset. SMART regularization improved modestly over SFT without requiring Bregman proximal point regularization; in fact, the EMA anchor degraded performance, suggesting it over-constrained learning in this low-data setting. Combining SMART with DPO achieved the highest ChrF score, slightly outperforming DPO alone.

An LLM judge evaluation gives DPO the highest win rate, but frequent ties across all model pairs limit the strength of this conclusion; the finetuned methods are largely indistinguishable to the judge outside of their shared advantage over zero-shot. Future work should incorporate structural metrics and thorough human evaluation to better differentiate these methods. Overall, both DPO and SMART+DPO show promise over standard SFT for low-data generation tasks, and their combination may be useful in settings where embedding-space smoothness complements preference-based training.

References

- [1] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [2] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through

principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, 2020.

- [3] Fenghua Weng, Jian Lou, Jun Feng, Minlie Huang, and Wenjie Wang. Adversary-aware dpo: Enhancing safety alignment in vision language models via adversarial training, 2025.
- [4] Youngjae Cho, Jongsuk Kim, and Ji-Hoon Kim. Learning where it matters: Geometric anchoring for robust preference alignment, 2026.
- [5] Enyi Jiang, Yibo Jacky Zhang, Yinglun Xu, Andreas Haupt, Nancy Amato, and Sanmi Koyejo. Latent adversarial regularization for offline preference optimization, 2026.