

SMART-GPT2: Adversarial Regularization for GPT-2 Fine-Tuning

Stanford CS224N Default Project

Bahram Mohmand
Department of Computer Science
Stanford University
bmohmand@stanford.edu

Noah Sabbavarapu
Department of Computer Science
Stanford University
noahsabb@stanford.edu

Zihan Wang
Department of Computer Science
Stanford University
zwang68@stanford.edu

Abstract

Fine-tuning pretrained language models such as GPT-2 on downstream tasks risks overfitting, particularly on small datasets, and standard regularization techniques like dropout and weight decay may be suboptimal. We implement SMART (Smoothness-inducing Adversarial Regularization and Trust-region) [Jiang et al., 2020] for GPT-2 fine-tuning, combining smoothness-inducing adversarial perturbation on token embeddings with Bregman proximal point optimization. We evaluate six ablation modes (vanilla, SMART-S, SMART-B, SMART-full, dropout, weight-decay) on two tasks: paraphrase detection on the Quora dataset and Shakespearean sonnet generation. On paraphrase detection, SMART-S and SMART-full achieve 90.4% dev accuracy versus 90.1% for vanilla fine-tuning. On sonnet generation, we find a markedly different picture from naive baselines: vanilla fine-tuning collapses to near-zero training loss by epoch 29, exhibiting severe overfitting, while all regularized methods substantially outperform it. SMART-S achieves the highest mean best chrF (42.44), while dropout achieves the highest mean final-eval chrF (41.84). We conclude that SMART’s smoothness objective provides the strongest ceiling performance for sonnet generation but that lightweight regularization (dropout, weight decay) achieves competitive final-eval scores with lower computational overhead.

1 Key Information to include

- Mentor: Wei Liu
- External Collaborators: No
- Sharing project: No
- Late Days:

Team member	Late days remaining
Bahram	2
Noah	1
Zihan	4
Total pooled	7
Team size	3
Late days applied for the report	1

2 Introduction

Fine-tuning pretrained language models on downstream tasks is a central paradigm in modern NLP. GPT-2 [Radford et al., 2019], a decoder-only transformer pretrained on large-scale web text, has proven effective for both classification and generation tasks via task-specific fine-tuning. However, fine-tuning on small datasets risks catastrophic forgetting and overfitting, problems that are especially acute for creative generation tasks like Shakespearean sonnet completion, where only 143 training examples are available.

Standard regularization techniques such as dropout and weight decay are widely used but may not optimally address the smoothness of the learned representation space. SMART (Smoothness-inducing Adversarial Regularization and Trust-region) [Jiang et al., 2020] offers a principled alternative: it augments the training objective with (1) an adversarial smoothness term that penalizes sensitivity to small perturbations of the input embeddings, and (2) a Bregman proximal point constraint that prevents parameters from drifting too far from the previous iteration’s model. Originally proposed for BERT fine-tuning on classification benchmarks, SMART has not been evaluated on decoder-only models or on open-ended generation tasks.

We ask: does SMART improve GPT-2 fine-tuning on classification (paraphrase detection) and on generation (sonnet completion)? Our contributions are: (1) the first application of SMART regularization to GPT-2; (2) a unified training pipeline supporting six ablation modes across both tasks; (3) a comprehensive evaluation over 3 random seeds and 30 epochs with best-checkpoint evaluation; and (4) a task-dependent analysis revealing that vanilla GPT-2 fine-tuning catastrophically overfits on sonnet generation, and that all regularized methods—including SMART—substantially outperform it.

3 Related Work

Adversarial training for NLP. Miyato et al. [2017] introduced virtual adversarial training (VAT) for semi-supervised learning, perturbing input embeddings to encourage locally smooth predictions. This idea was extended by Jiang et al. [2020] in SMART, which adds a trust-region constraint via Bregman proximal point optimization and applies the framework to BERT fine-tuning on GLUE, achieving state-of-the-art results on several benchmarks. Our work extends this line to GPT-2, a decoder-only architecture, and to a generation task.

Neural poetry generation. Prior work on neural poetry generation has explored RNN-based models [Hopkins and Kiela, 2017] and transformer fine-tuning [Manjavacas et al., 2019] for stylistic text generation. Evaluating generated poetry remains challenging — we follow the task specification in using chrF [Popović, 2015] as a character-level proxy for Shakespearean style, though we acknowledge its limitations as a measure of poetic quality.

Trust-region and proximal methods. The Bregman proximal point method constrains parameter updates to remain close to a reference model, stabilizing optimization in the presence of sharp loss landscapes. In SMART, this reference is maintained as an exponential moving average (EMA) of the model parameters, and KL divergence from this reference is penalized at each step.

GPT-2 fine-tuning. Radford et al. [2019] released GPT-2 as a large-scale language model trained on WebText. Downstream adaptation typically involves full-parameter fine-tuning. For classification tasks, a common approach is cloze-style prompting, where the model predicts a token that encodes the desired label. For generation tasks, the model is fine-tuned with standard next-token prediction loss. Our work applies SMART to both paradigms and is, to our knowledge, the first to do so.

4 Approach

4.1 Base Model and Task Formulation

We use GPT-2 base (124M parameters) for sonnet generation and GPT-2-medium (355M parameters) for paraphrase detection. All parameters are fine-tuned end-to-end.

Paraphrase detection. Given a question pair (s_1, s_2) , we format the input as a cloze prompt: “*Is ‘s₁’ a paraphrase of ‘s₂’? Answer yes or no:*” and apply a two-class linear classification head on the hidden state of the last non-padding token. A critical implementation detail is using the last *non-padding* token rather than the naive final position:

$$h_{\text{cls}} = H[\text{arange}(B), \text{mask.sum}(\text{dim}=1) - 1] \quad (1)$$

where $H \in \mathbb{R}^{B \times T \times d}$ is the final hidden state. Without this fix, accuracy collapsed to 0.36–0.40 due to classifying on padding token embeddings.

Sonnet generation. We fine-tune GPT-2 as an autoregressive language model with standard cross-entropy next-token prediction loss on 143 Shakespearean sonnets. At inference, we generate completions conditioned on the first three lines of held-out sonnets using top- p nucleus sampling ($p = 0.95$) with temperature $T = .8$. Generation hyperparameters were subsequently tuned via dev leaderboard search; see Section 5 for final submission parameters.

4.2 SMART Regularization

We augment the task loss with two regularization terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda_s \mathcal{L}_s + \mu \mathcal{L}_b \quad (2)$$

Smoothness term (SMART-S). We perturb the token embedding matrix within an L_∞ ball of radius ϵ using projected gradient ascent (PGA) to find the worst-case perturbation δ^* that maximizes the symmetrized KL divergence between perturbed and clean output distributions:

$$\delta^* = \arg \max_{\|\delta\|_\infty \leq \epsilon} D_{\text{KL}}(f(x + \delta) \| f(x)), \quad \mathcal{L}_s = D_{\text{KL}}(f(x + \delta^*) \| f(x)) \quad (3)$$

The smoothness loss penalizes sensitivity to small input perturbations, encouraging the model to learn locally flat representations.

Bregman proximal term (SMART-B). We maintain an exponential moving average (EMA) of the model parameters with decay $\beta = 0.999$ and penalize KL divergence from this reference model θ_{prev} :

$$\mathcal{L}_b = D_{\text{KL}}(f(x; \theta) \| f(x; \theta_{\text{prev}})) \quad (4)$$

This trust-region constraint prevents aggressive parameter updates that could destabilize training.

4.3 Ablation Modes

We evaluate six configurations, as summarized in Table 1.

Table 1: Ablation modes for SMART regularization.

Mode	Smoothness ($\mathcal{L}_s$)	Bregman ($\mathcal{L}_b$)	Other
vanilla	no	no	—
SMART-S	yes	no	—
SMART-B	no	yes	—
SMART-full	yes	yes	—
dropout	no	no	$p = 0.1$
weight-decay	no	no	$\lambda = 0.01$

5 Experiments

5.1 Data

Paraphrase detection. We use the Quora Question Pairs dataset [Quora, 2017] with 141,506 training and 20,215 development examples. Each instance is a pair of questions labeled as paraphrase or non-paraphrase.

Sonnet generation. We fine-tune on a corpus of 143 Shakespearean sonnets and evaluate on held-out prompts consisting of the first three lines of each sonnet. Generated completions are compared to gold references using chrF.

5.2 Evaluation Method

For paraphrase detection, we report accuracy and macro-F1 on the development set. For sonnet generation, we use chrF [Popović, 2015] (via sacrebleu) as the primary metric, measuring character n -gram overlap between generated and gold completions. We report two chrF values per method: **best chrF**, the peak validation chrF at any checkpoint during training, and **final chrF**, the held-out evaluation score at the end of the full training run. All sonnet results are averaged over 3 random seeds (11711, 11712, 11713) with standard deviation reported.

5.3 Experimental Details

All experiments were run on an NVIDIA A100 GPU (Google Colab). For paraphrase detection, we use GPT-2-medium, AdamW optimizer, batch size 24, learning rate 10^{-5} , and 3 training epochs. For sonnet generation, we use GPT-2 base, batch size 4, learning rate 10^{-5} , and 30 training epochs with best-checkpoint selection by peak validation chrF. SMART hyperparameters are fixed at $\lambda_s = 0.001$, $\mu = 0.01$, and EMA decay $\beta = 0.999$ across all runs. The random seed is fixed to 11711 for paraphrase; sonnet ablation results are averaged over seeds {11711, 11712, 11713}.

After completing the full ablation study, we performed a generation hyperparameter search for the final submission using the dev leaderboard with gold references to evaluate directly. Initial ablation runs used temperature $T = 0.8$, top- $p = 0.95$, and maximum generation length 128 tokens. Based on dev leaderboard feedback, increasing temperature to $T = 1.0$ and maximum generation length to 180 tokens improved chrF. Combined with selecting the best-performing checkpoint (SMART-S, seed 11712, epoch 19), this search produced our best dev score of **43.713** and test score of **43.529**.

5.4 Results

Paraphrase detection. Table 2 shows paraphrase detection results over 3 epochs. SMART-S and SMART-full both achieve the best dev accuracy (90.4%) and F1 (0.898), with identical training trajectories across all three epochs. Weight-decay (90.2%) is competitive, while dropout (90.1%) matches vanilla exactly—suggesting dropout provides no additional benefit over unregularized fine-tuning on this large-scale task (141,506 training examples). Notably, SMART-B (89.9%) is the *only method that falls below vanilla*, consistent with the sonnet finding that the Bregman proximal constraint alone is overly restrictive. All methods improve monotonically over 3 epochs with no instability, in contrast to the noisy sonnet training curves.

A key practical finding is the **efficiency gap between SMART-S and SMART-full**: both achieve identical accuracy (90.4%) and F1 (0.898), but SMART-full takes ≈ 56 min/epoch versus ≈ 50 min/epoch for SMART-S—a 12% overhead from the Bregman proximal term that provides zero measurable benefit on this task. SMART-S is therefore the recommended configuration for paraphrase detection. Our best submission achieves a **test accuracy of 0.897**, confirming that the dev set gains generalize to held-out data.

Error analysis (Table 4) reveals an asymmetric error profile: SMART-S produces 2,534 false positives versus 1,258 false negatives, a 2:1 ratio suggesting the model is more prone to over-predicting paraphrase than missing it. This is consistent with a cloze-style model that has learned to associate lexical overlap with paraphrase. Question pairs that share many keywords are incorrectly accepted even when they differ in semantic meaning.

Table 2: Paraphrase detection results (GPT-2-medium, 3 epochs, batch size 24, lr 10^{-5}). Best results in **bold**. Training time per epoch shown for cost comparison.

Mode	Dev Acc	Macro F1	Time / Epoch
SMART-S	0.904	0.898	≈ 50 min
SMART-full	0.904	0.898	≈ 56 min
weight-decay	0.902	0.896	≈ 19 min
dropout	0.901	0.896	≈ 19 min
vanilla	0.901	0.896	≈ 19 min
SMART-B	0.899	0.894	≈ 25 min

Table 3: Paraphrase detection per-epoch dev accuracy.

Mode	Epoch 0	Epoch 1	Epoch 2
SMART-S	0.885	0.896	0.904
SMART-full	0.885	0.896	0.904
weight-decay	0.886	0.895	0.902
dropout	0.885	0.895	0.901
vanilla	0.885	0.895	0.901
SMART-B	0.887	0.896	0.899

Table 4: Confusion matrix for SMART-S paraphrase detection on dev set.

	Predicted No	Predicted Yes
Actual No	23,003	2,534
Actual Yes	1,258	13,634

Sonnet generation. Table 5 shows sonnet generation results averaged over 3 seeds. The most prominent finding is that **vanilla fine-tuning severely overfits**: training loss collapses to ≈ 0.21 by epoch 29 (versus ≈ 3.4 for dropout and weight-decay), and while the best checkpoint achieves 42.09 chrF, the final-eval score drops to only 40.95—the worst of any method. All regularized methods substantially outperform vanilla on final-eval chrF.

SMART-S achieves the **highest mean best chrF (42.44)** and the most stable final-eval performance across seeds ($\sigma = 0.020$, the tightest of any method). Dropout achieves the **highest mean final-eval chrF (41.84)** with lower computational cost. Weight-decay is nearly tied with dropout on best chrF (42.43) but shows higher seed variance on final eval ($\sigma = 0.565$). SMART-B carries the highest training loss (≈ 4.6) with below-average performance, suggesting the proximal constraint overconstrained optimization on this small dataset. SMART-full achieves a mean final-eval of 41.80 with notably tight variance ($\sigma = 0.211$), indicating that the combined objective provides stable, consistent training despite elevated loss. Following the ablation study, a generation hyperparameter search (temperature, top- p , and max generation length) using the dev leaderboard identified SMART-S seed 11712 as the best checkpoint, yielding a final dev chrF of **43.713** and test chrF of **43.529**.

6 Analysis

Vanilla catastrophically overfits at higher epochs. The most striking result of this study is the catastrophic overfitting of vanilla fine-tuning at higher epochs. Without any regularization, GPT-2 memorizes the 143 training sonnets completely by epoch 29, with training loss collapsing to ≈ 0.21 —more than **15 $\times$ lower** than any regularized method. This is not a sign of good learning: val chrF does not benefit, and the gap between best-checkpoint chrF (42.09) and final-eval chrF (40.95) is the **largest of any method by a wide margin** (1.14 points vs. 0.26 for SMART-full). Vanilla’s best-checkpoint score is inflated by superficial character-level overlap with memorized training text, not genuine generalization. Put simply, **vanilla fine-tuning should not be used beyond early epochs on this dataset**—every regularized method produces a better model at the end of training.

Table 5: Sonnet generation results (GPT-2 base, 30 epochs, 3 seeds). Best chrF = peak validation chrF; Final chrF = held-out score at end of run. Best results in **bold**.

Mode	Best chrF	σ	Final chrF	σ	Train Loss
SMART-S	42.44	0.401	41.54	0.020	3.55
weight-decay	42.43	0.086	41.37	0.565	3.41
dropout	42.39	0.098	41.84	0.411	3.41
SMART-full	42.20	0.289	41.80	0.211	4.76
SMART-B	42.25	0.170	41.18	0.206	4.60
vanilla †	42.09	0.279	40.95	0.471	0.22

≈ 0.22 indicates severe overfitting (memorization).

† Vanilla train loss

SMART-S: highest ceiling and remarkable consistency. SMART-S achieves the highest mean best chrF (42.44) of any method, but its most remarkable property is its **near-perfect seed consistency on final-eval chrF**: the three seeds converge to 41.52, 41.54, and 41.56, yielding $\sigma = 0.020$ —**more than $20\times$ lower variance than dropout** ($\sigma = 0.411$) and **$28\times$ lower than weight-decay** ($\sigma = 0.565$). This is not a coincidence of one lucky seed: all three seeds independently settle at essentially the same final-eval score regardless of which epoch peaked during training. The symmetrized KL smoothness objective appears to regularize not just training dynamics but the entire distribution of end-of-run model quality, making SMART-S the most *reliable* method in this study. For any application where reproducibility and consistency across runs matters, SMART-S is the clear choice.

Dropout: best final-eval with lower cost. Dropout achieves the highest mean final-eval chrF (41.84) despite slightly lower best chrF than SMART-S. In prior runs, dropout was inoperative (train loss matching vanilla at ≈ 0.2); in this corrected run, dropout stabilizes training loss at ≈ 3.4 , confirming its regularization is active and effective. For practitioners with limited compute, dropout offers a favorable cost-performance tradeoff.

SMART-B: proximal constraint overconstrained. SMART-B consistently underperforms weight-decay and dropout on best chrF, despite incurring the second-highest training loss (≈ 4.6). The EMA-based trust-region constraint appears to over-restrict parameter updates on a 143-example dataset, limiting the model’s ability to learn the target distribution within 30 epochs. Best-checkpoint epochs are highly variable (ep5 to ep28), suggesting an unstable optimization landscape. Critically, this pattern is consistent across both tasks: SMART-B is the only method to fall below vanilla on paraphrase detection (89.9% vs. 90.1%) and ranks last on sonnet final-eval chrF.

SMART-full: consistent stability at higher cost. SMART-full shows the second-tightest final-eval variance ($\sigma = 0.211$) and achieves competitive final-eval (41.80) despite the highest training loss of any method (≈ 4.76). However, the paraphrase results reveal an important efficiency insight: SMART-full achieves *identical* accuracy and F1 to SMART-S (90.4%, 0.898) at 12% higher training cost per epoch (≈ 56 vs. ≈ 50 min). The Bregman proximal term adds computational overhead without measurable benefit on either task, suggesting that for GPT-2 fine-tuning, the smoothness component alone captures the majority of SMART’s benefit.

Qualitative analysis: generated sonnet examples. To complement the quantitative results, we examine actual sonnet completions generated by vanilla and SMART-S given the same held-out prompts. Both models receive identical first-three-line prompts; the remaining lines are generated autoregressively. Vanilla outputs exhibit clear signs of memorization and incoherence at later epochs: repeated tokens within lines (“concealed concealed,” “for for told you lies”), tokenization artifacts (“want obf can still seep out”), abrupt truncations mid-sentence, and lines that abandon iambic pentameter entirely. Completions frequently collapse into grammatically incoherent fragments, consistent with a model that has memorized training sequences rather than learned the underlying poetic structure. SMART-S outputs, by contrast, maintain Shakespearean register more consistently throughout — lines such as “*Findneth virtue in fell foes ’till I liket your love’s breath*” and “*Thy beating art won by those lovely touches haply*” preserve archaic diction and syntactic coherence. Couplets at the end of sonnets are more structurally complete, and repetition artifacts are largely absent. While neither model produces publication-quality verse, the qualitative gap between vanilla

and SMART-S is clearly visible and directly corroborates the quantitative chrF gap: SMART-S’s smoothness objective encourages output stability under perturbation, which appears to translate into more coherent and consistent generation at the token level.

Classification vs. generation: Across both tasks, SMART-S emerges as the most consistently strong method: it achieves the best accuracy on paraphrase detection and the highest ceiling chrF on sonnet generation. However, the *mechanism* differs by task. On paraphrase, the 141,506-example training set prevents overfitting entirely—all methods improve monotonically and the accuracy spread is narrow (0.899 to 0.904). On sonnets, overfitting is the dominant failure mode, and regularization’s primary role is preventing the training loss collapse seen in vanilla. The consistent underperformance of SMART-B on both tasks suggests the Bregman proximal constraint is not well-suited to GPT-2 fine-tuning at either data scale tested here.

7 Conclusion

We implemented SMART regularization for GPT-2 fine-tuning and evaluated it comprehensively on paraphrase detection and Shakespearean sonnet generation. On paraphrase detection, SMART-S and SMART-full improve accuracy by +0.3% over vanilla, with SMART-S preferred due to identical performance at 12% lower training cost than SMART-full. On sonnet generation, our central finding is that vanilla fine-tuning catastrophically overfits the 143-example training set, while all regularized methods substantially outperform it on final-eval chrF. SMART-S achieves the highest peak chrF (42.44) with near-perfect seed stability ($\sigma = 0.020$), and our best submission achieves 43.713 dev chrF and 43.529 test chrF after generation hyperparameter tuning.

A consistent finding across both tasks is that **SMART-B underperforms vanilla on classification and ranks last on sonnet final-eval**—the Bregman proximal constraint alone appears ill-suited to GPT-2 fine-tuning. Future work should explore lower perturbation budgets for generation, task-adaptive SMART hyperparameters, and whether the Bregman term provides benefit at larger data scales or with longer training horizons.

Team Contributions

Bahram Mohmand implemented the SMART regularization pipeline and ran all sonnet generation experiments. Noah Sabbavarapu implemented the baseline models for paraphrase detection and sonnet generation and ran the paraphrase detection training pipeline and hyperparameter sweeps. Zihan Wang implemented the GPT-2 base model, attention, and optimizer components, and ran the sentiment classification baselines. All three members contributed to analysis, writing, and the final report.

References

- Jack Hopkins and Douwe Kiela. Automatically generating rhythmic verse with neural networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 168–178. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1016.
- 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, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.197. URL <https://aclanthology.org/2020.acl-main.197>.
- Enrique Manjavacas, Ákos Kádár, and Mike Kestemont. Adapting classical poetry for modern readers. In *Proceedings of the 3rd Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 68–77. Association for Computational Linguistics, 2019.
- Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: A regularization method for supervised and semi-supervised learning, 2017.

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.

Quora. First quora dataset release: Question pairs, 2017. URL <https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs>.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

A Appendix: Per-Seed Sonnet Results

Table 6 and Table 7 provide the complete per-seed breakdown of best-checkpoint and final-eval chrF scores for sonnet generation, along with the epoch at which each best checkpoint was achieved. Table 8 provides the per-epoch training loss for all paraphrase detection methods.

Table 6: Best chrF per seed and best-checkpoint epoch.

Method	S1 (ep)	S2 (ep)	S3 (ep)	Mean	σ
SMART-S	42.13 (27)	42.89 (19)	42.29 (21)	42.44	0.401
weight-decay	42.47 (27)	42.33 (19)	42.48 (13)	42.43	0.086
dropout	42.27 (8)	42.46 (19)	42.43 (17)	42.39	0.098
SMART-full	42.17 (28)	41.93 (27)	42.51 (23)	42.20	0.289
SMART-B	42.09 (16)	42.43 (28)	42.24 (5)	42.25	0.170
vanilla	41.92 (21)	42.42 (21)	41.94 (13)	42.09	0.279

Table 7: Final-eval chrF per seed (end of 30-epoch run).

Method	Seed 11711	Seed 11712	Seed 11713	Mean	σ
dropout	41.44	41.81	42.26	41.84	0.411
SMART-full	41.58	41.82	42.00	41.80	0.211
SMART-S	41.56	41.52	41.54	41.54	0.020
weight-decay	41.16	42.01	40.94	41.37	0.565
SMART-B	41.41	41.02	41.10	41.18	0.206
vanilla	40.68	40.67	41.49	40.95	0.471

Table 8: Paraphrase detection per-epoch training loss.

Mode	Epoch 0	Epoch 1	Epoch 2
SMART-S	0.344	0.258	0.215
SMART-full	0.344	0.258	0.215
weight-decay	0.320	0.230	0.174
dropout	0.320	0.230	0.173
vanilla	0.320	0.230	0.173
SMART-B	0.323	0.232	0.176