

SMART-GPT2: Adapting SMART-Style Regularization for Decoder-Only Fine-Tuning

Stanford CS224N Default Project

Austin Ho

Department of Computer Science
Stanford University
austinho@stanford.edu

Abstract

Fine-tuning large pretrained models such as GPT-2 on small downstream datasets can lead to overfitting, and it is unclear whether SMART regularization, originally developed for encoder-based models, can also help regularize decoder-only models. In this project, I evaluated SMART-style regularization for GPT-2 on Quora paraphrase detection and Shakespeare sonnet continuation. I first implemented a faithful SMART baseline for GPT-2, then evaluated decoder-adapted alternatives including a custom paraphrase classification head, representation-level SMART, R-Drop, and mild embedding augmentation. On both tasks, faithful SMART seems to hurt accuracy on decoder-only GPT-2, while the decoder-adapted regularization methods improve it. On paraphrase tasks, these techniques improved accuracy - the best models achieved 0.897. However, for sonnet generation, stronger regularization methods generally underperform the baseline, while mild embedding augmentation provides small but consistent improvements and achieves the best final test score of 41.878 CHRF. These results suggest that decoder-only fine-tuning benefits more from lightweight, decoder-native regularization than from directly porting SMART's original input-level formulation.

1 Key Information to include

- Mentor: Chenglei Si
- Late Days (if you are using any): 2

2 Introduction

SMART [1] addresses overfitting during fine-tuning through smoothness regularization and proximal optimization, but it was only ever tested on bidirectional encoders (BERT, RoBERTa). I wanted to evaluate whether SMART transfers to decoder-only architectures like GPT-2 [2], and if not, what decoder-native alternatives can recover its benefits.

SMART and related methods like FreeLB [3] were published during the BERT era (2019-2020) when encoders dominated fine-tuning benchmarks. As the field shifted to decoder-only architectures, the regularization question was sidestepped rather than solved - parameter-efficient methods like LoRA implicitly reduce overfitting by constraining the update space, and no decoder-native alternatives to SMART were developed. But more recently, the industry is deploying small decoder models (Phi, Gemma 2B, LLaMA 3.2 1B) for domain-specific fine-tuning on limited data, exactly the domain where overfitting matters. As decoder models are increasingly fine-tuned for specialized tasks, practitioners need to understand which regularization strategies actually transfer from the encoder era and which need to be redesigned for causal architectures.

Compared to encoders, decoder-only models process text causally, meaning each token can only attend to previous tokens. This changes how perturbations propagate through the network; in BERT, a perturbation at the input affects all token representations symmetrically through bidirectional attention. In GPT-2, a perturbation at position 3 can only influence positions 4, 5, 6, etc and must cascade through 12 layers of causal attention to reach the last token where the prediction happens. By then, the perturbation signal is either washed out or distorted unpredictably. This means SMART’s core mechanism — "perturb the input, penalize output changes" — has a fundamentally different effect on decoders than on encoders, and it’s not obvious in advance whether it will help, hurt, or do nothing.

The sonnet task adds a second layer of difficulty: with only 131 training examples, the model quickly memorizes the training data without absorbing enough of Shakespeare’s style to generate well. Regularizers make this worse by slowing down learning in a setting where there is already too little data to learn from.

I first implemented SMART on GPT-2 for both tasks, then decomposed its failure, tracing it to how perturbations propagate through causal attention. This diagnosis led to three decoder-adapted alternatives: representation-level smoothness, R-Drop [4] consistency regularization, and embedding augmentation [5], which repurposes SMART’s perturbation mechanism as data augmentation instead of regularization. On paraphrase, decoder-adapted methods recovered the benefits SMART provides on encoders. On sonnet, regularization was counterproductive, but embedding augmentation was the only method to stabilize training on 131 examples.

3 Related Work

SMART (Jiang et al., 2020) is the original inspiration for this paper; it introduces a smoothness-inducing regularizer that penalizes output changes under small input embedding perturbations, and Bregman proximal point optimization that constrains each update to stay close to the previous step’s predictions. It was evaluated on BERT and RoBERTa across GLUE, SNLI, SciTail, and ANLI, with the largest gains on small datasets like RTE (2.5k examples).

FreeLB (Zhu et al., 2020) is another popular encoder regularizer; it works by accumulating gradients across multiple adversarial perturbation steps to approximate training on a larger batch of diverse adversarial examples; it was tested on BERT and RoBERTa for GLUE and commonsense reasoning. Both SMART and FreeLB perturb input embeddings and assume the perturbation meaningfully affects the output, which is an assumption that depends on how information propagates through the model and can differ between bidirectional and causal attention.

R-Drop (Liang et al., 2021), an architecture-agnostic regularizer takes a different approach to output consistency. Instead of perturbing inputs, it runs each training example through the model twice with different dropout masks and minimizes the symmetric KL-divergence between the two output distributions. Because dropout operates at every layer simultaneously, the noise source is distributed throughout the network rather than injected at a single point. R-Drop was validated across 18 datasets spanning machine translation, summarization, language understanding, language modeling, and image classification, including decoder-based Transformer language models on Wikitext-103. However, R-Drop’s evaluations used datasets with thousands to millions of examples. Whether it remains effective in extreme low-data settings like 131 training examples, or whether it can complement perturbation-based methods like SMART, has not been explored.

A separate line of work uses embedding-level noise not for regularization but for data augmentation. Zhang and Yang (2018) compare Gaussian, Bernoulli, and adversarial noise applied to word embeddings for sentence classification, finding that all improve performance and that gains increase as training data decreases. This augmentation framing is distinct from SMART’s regularization framing: rather than penalizing the model for changing its output under perturbation, the model is trained to predict correct labels from perturbed views, expanding the effective training set rather than constraining the model’s capacity. This distinction becomes critical in extreme low-data settings where regularization can be counterproductive.

4 Approach

The goal is to improve GPT-2 small (124M parameters) on two tasks: Quora paraphrase detection and Shakespeare sonnet continuation (131 training sonnets).

Paraphrase detection is a classification problem, so GPT-2 must be adapted to produce a binary decision. Sonnet continuation is a generation problem, which is what GPT-2 was designed for. For each task, I first apply faithful SMART regularization, then evaluate the three decoder-adapted alternatives.

4.1 Turning GPT-2 into a Classifier

For paraphrase detection, I evaluated two ways to extract a classification signal from GPT-2. Both encode the sentence pair as a prompt asking whether one sentence is a paraphrase of the other. In the cloze-style approach, GPT-2 predicts the next token using its full 50k-token vocabulary and the decision is based on whether "yes" or "no" receives a higher score. This is the default formulation provided by the project and requires no additional parameters. This approach is straightforward but somewhat unnatural — it uses a generation mechanism to perform classification.

The other approach I tried was to pass the model’s internal representation when the last token is generated through a small learned linear layer. This would then directly output two scores: paraphrase or not paraphrase. This adds a task-specific classifier that was trained jointly with GPT-2 during fine tuning and gives GPT-2 a more direct classification interface. I evaluated both formulations as baselines and applied regularization methods on top of each.

4.2 Adapting SMART to GPT-2

The first regularizer I implemented was SMART’s two components, following Algorithm 1 of Jiang et al. (2020) as closely as possible. The smoothness term adds a small perturbation to the input embeddings and penalizes the model for changing its output predictions; the perturbation is chosen to maximally change the output within a small neighborhood of the original input. The proximal term maintains an EMA teacher model and penalizes the KL-divergence between the student’s and teacher’s output distributions, with β transitioning from 0.99 to 0.999 after the first 10% of training. I used the paper’s default hyperparameters: $\epsilon = \sigma = 10^{-5}$, $\eta = 10^{-3}$, one ascent step.

Adapting SMART to GPT-2 required two changes. First, the prediction comes from the last token through causal attention rather than from a bidirectional [CLS] token. Second, for sonnet generation, the smoothness and proximal terms are applied token-wise across the sequence rather than to a single output distribution, since GPT-2 predicts the next token at every position.

4.3 Decoder-Adapted Alternatives

I also evaluated three alternative regularizers inspired by SMART for a decoder-only setting.

First, representation-level smoothness moves SMART’s perturbation from the input embeddings to the last-token hidden state immediately before the custom classifier head. This tests whether perturbing closer to the output is more effective in a causal model.

Second, R-Drop [4] enforces output consistency without any explicit perturbation. Each training example passes through the model twice with different dropout masks, and the symmetric KL-divergence between the two output distributions is added to the loss. Because dropout operates at every layer, the noise source is distributed throughout the network rather than injected at the input. I applied R-Drop to both paraphrase and sonnet.

Third, I implemented embedding augmentation, which repurposes SMART’s perturbation mechanism as data augmentation rather than regularization. Instead of penalizing the model for changing its output under perturbation, the model is trained to predict the correct labels from noisy embedding views:

$$\mathcal{L}_{\text{aug}} = \frac{1}{1+K} \left(\mathcal{L}(g(x; \theta), y) + \sum_{k=1}^K \mathcal{L}(g(x + \delta_k; \theta), y) \right), \quad \delta_k \sim \mathcal{N}(0, \sigma^2 I), \quad (1)$$

where x is the detached input embedding, $g(\cdot; \theta)$ denotes the model’s forward pass from embeddings to logits, K is the number of augmented views, and σ is a task-dependent noise scale.

Unlike SMART, there is no KL term and no adversarial search — the noisy views provide additional supervised training signal. For paraphrase detection, this loss is standard classification cross-entropy, while for sonnet generation it is the masked autoregressive next-token prediction loss averaged across clean and noisy views.

5 Experiments

5.1 Data

The paraphrase training set contains 283,003 examples, the development set contains 40,429 examples, and the test set contains 80,858 examples. For sonnet generation, the training set contains 131 Shakespeare sonnets, with 12 additional held-out sonnets for development and 12 for test evaluation.

Because paraphrase training is substantially more expensive than sonnet training, I first screened candidate methods on a fixed 10% subset of the Quora training set, then promoted the strongest configurations to full-data runs. For sonnet generation, the training set was small enough that all experiments were run on the full dataset.

5.2 Evaluation method

For paraphrase detection, the primary metric is accuracy. I report development accuracy for model selection and hidden-test accuracy from the leaderboard for final comparison.

For sonnet generation, the primary metric is CHRF, a character-level n-gram overlap score that measures how closely generated text matches Shakespeare’s original continuations. Because the sonnet dataset is very small and generation is stochastic, small CHRF differences (under 0.5 points) may not be meaningful. I focus on consistent trends across experiments rather than single isolated improvements.

5.3 Experimental details

All experiments use GPT-2 small (124M parameters) as the base model. For paraphrase detection, I evaluate either the default cloze-style output formulation or a custom classification head on top of GPT-2; for sonnet generation, the baseline is plain GPT-2 autoregressive fine-tuning with standard next-token prediction loss and padding positions masked out of the objective. Unless otherwise noted, optimization uses AdamW with batch size 8. Most runs use weight decay 0.0, with weight decay changed only in explicit paraphrase ablations.

As an initial screening step, I swept learning rates over $\{5 \times 10^{-6}, 10^{-5}, 2 \times 10^{-5}\}$ across the main baselines and regularized variants for both tasks. Based on these results, the strongest promoted paraphrase and sonnet runs generally used a learning rate of 2×10^{-5} . For faithful SMART, I used the paper’s default perturbation hyperparameters: $\epsilon = \sigma = 10^{-5}$, $\eta = 10^{-3}$, one ascent step, and the EMA teacher β schedule from the paper’s Algorithm 1.

For paraphrase R-Drop, I swept $\alpha \in \{0.5, 1.0, 2.0, 3.0\}$ on the 10% subset and selected $\alpha = 1.0$, which gave the best development accuracy among the tested values (0.8439). For sonnet R-Drop, I evaluated $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 1.0\}$, with $\alpha = 0.5$ giving the strongest development result among the R-Drop variants. For embedding augmentation, I tested $\sigma \in \{10^{-3}, 3 \times 10^{-4}\}$ with $K = 1$ augmented view.

Paraphrase subset runs were trained for 2 epochs. Full-data paraphrase and sonnet runs were trained for 3 epochs. However, I also later ran a diagnostic experiment comparing the baseline and mild embedding augmentation, which was trained for 6 epochs to inspect epoch-wise dev CHRF trajectories.

5.4 Results

Paraphrase screening on the 10% subset. I used subset experiments to identify which decoder-adapted methods were worth promoting to full-data runs. The strongest results are shown in Table 1.

Model	Dev Accuracy
Baseline (cloze)	0.8289
Baseline (custom head)	0.8394
Faithful SMART (custom head)	0.8174
Representation-SMART (custom head)	0.8419
R-Drop (custom head)	0.8439
R-Drop + Embedding Augmentation (custom head)	0.8445

Table 1: Paraphrase subset-screening results on a fixed 10% Quora training subset.

These results showed two clear patterns. First, on the 10% subset, the custom-head baseline outperformed the cloze baseline, suggesting that the output formulation itself was an important bottleneck for decoder-only paraphrase classification. For this reason, the later paraphrase regularization experiments were built primarily on top of the custom-head baseline rather than the cloze formulation. Second, faithful SMART underperformed both baselines, while the decoder-adapted methods were more promising.

Full-data paraphrase results. Faithful SMART again underperformed the baselines, while decoder-adapted methods improved over them.

Model	Dev Accuracy
Baseline (cloze)	0.8919
Baseline (custom head)	0.8897
Faithful SMART (custom)	0.8869
Representation-SMART (custom)	0.8887
R-Drop (custom)	0.8974
R-Drop (cloze)	0.9000
R-Drop + Representation-SMART (custom)	0.9005

Table 2: Paraphrase screening results on a full Quora training set.

The best full-data dev score came from combining R-Drop with representation-level SMART on the custom-head paraphrase model, reaching 0.9005. Interestingly, the plain cloze baseline (0.8919) slightly outperformed the plain custom-head baseline (0.8897), even though the custom-head baseline had been stronger in the earlier subset screening experiments. I also tested the cloze baseline with R-Drop, which was also competitive at 0.9000 dev accuracy. On the hidden test set, the top three paraphrase submissions all achieved the same leaderboard score of 0.897.

Sonnet results. Sonnet generation was much harder to improve. The baseline was already strong, and most regularization methods hurt rather than helped.

Model	Dev CHRF
Baseline	42.044
Legacy proximal variant	39.808
R-Drop ($\alpha = 0.5$)	41.784
Embedding augmentation ($\sigma = 10^{-3}$)	40.955
Embedding augmentation ($\sigma = 3 \times 10^{-4}$)	42.075

Table 3: Sonnet generation development results measured with CHRF.

The only training-time method to improve over the baseline on dev was mild embedding augmentation with $\sigma = 3 \times 10^{-4}$, and the gain was small. On the hidden test set, the embedding augmentation submission reached 41.878, while the baseline was very similar at 41.876. Given the marginal gains and added complexity, the baseline is effectively the best model.

6 Analysis

The central finding is that faithful SMART did not transfer well to decoder-only GPT-2. Across both tasks, directly applying SMART’s original input-level formulation underperformed decoder-adapted alternatives. A plausible explanation is architectural mismatch: in SMART’s original encoder setting, perturbations are distributed through bidirectional attention before influencing the output. In GPT-2, perturbations applied at the input must instead propagate through a unidirectional causal stack before affecting the model’s predictions.

The paraphrase results show a clear pattern. Perturbations applied at the input (faithful SMART) hurt, perturbations applied right before the classifier (representation-level SMART) helped slightly, and noise distributed throughout the network via dropout (R-Drop) helped the most. This suggests that the farther noise has to travel through the causal stack, the less effective it is as a regularizer. The strongest result, though only by a very small margin, combined R-Drop with representation-level SMART. In practice, however, the improvement over R-Drop alone is so small that the two methods should be viewed as performing similarly.

Sonnet generation told a different story. Regularization methods that helped on paraphrase, such as R-Drop, generally hurt sonnet performance. The epoch-wise training curves help explain why: the baseline’s training loss decreased steadily across epochs, but dev CHRf oscillated rather than degrading monotonically, so the model was not cleanly overfitting in the usual sense. Mild embedding augmentation produced a smoother dev trajectory and slightly better best-dev performance, and it also narrowly achieved the best hidden-test score. However, the margin over the baseline was extremely small. The most defensible interpretation is that 131 sonnets provides too little data for stronger regularization to help consistently: the task is dominated by limited training signal and high variance, not simply by excess model capacity. In this case, repurposing SMART’s perturbation as data augmentation was more effective than using it as an explicit smoothness penalty, although the gains remained modest.

The broader takeaway is that regularization for decoder-only fine-tuning should be designed around where noise enters the model and how far that signal must travel before affecting the prediction. For classification tasks with sufficient data, decoder-adapted consistency methods like R-Drop are effective. For extreme low-data generation, the same perturbation machinery is better used as augmentation than as regularization.

7 Conclusion

This project studied whether SMART transfers from encoder-based fine-tuning to decoder-only GPT-2. It does not. Faithful SMART consistently underperformed decoder-adapted alternatives on both paraphrase detection and sonnet generation. The strongest paraphrase results came from R-Drop and representation-level smoothness — methods that place noise close to or throughout the output path rather than at the input. For sonnet generation, most regularizers were counterproductive; only embedding augmentation, which repurposes SMART’s perturbation as data augmentation, showed any improvement over the baseline.

I see that regularization designed for bidirectional encoders should not be applied to causal decoders without adaptation. The causal attention stack changes how perturbations propagate, and methods must account for this. As small decoder models are increasingly fine-tuned on limited domain-specific data, understanding which regularization strategies actually work in this setting — rather than inheriting assumptions from the encoder era — becomes increasingly important. Future work should test these findings across larger decoder models and multiple seeds, and explore whether generation tasks benefit more from augmentation-based approaches than from traditional regularization.

References

- [1] 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.

- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.
- [3] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*, 2020.
- [4] Xiaobo Liang, Lijun Wu, Juntao Li, Yue Wang, Qi Meng, Tao Qin, Wei Chen, Min Zhang, and Tie-Yan Liu. R-drop: Regularized dropout for neural networks. In *Advances in Neural Information Processing Systems*, 2021.
- [5] Dongxu Zhang and Zhichao Yang. Word embedding perturbation for sentence classification. *arXiv preprint arXiv:1804.08166*, 2018.