

Re-implementing GPT-2 for Classification, Paraphrase Detection, and Sonnet Generation

Stanford CS224N Default Project

Shani Su

Department of Computer Science
Stanford University
shanisu@stanford.edu

Saanvi Thummalapally

Department of Computer Science
Stanford University
saanvit@stanford.edu

Abstract

We implemented GPT-2 from scratch, including masked multi-head self-attention, transformer blocks, token and positional embeddings, and the AdamW optimizer, and then loaded pretrained GPT-2 weights for downstream fine-tuning. Adapting pretrained language models to downstream tasks can be challenging because different tasks may benefit from different fine-tuning strategies, and it is not always clear whether full-model tuning, parameter-efficient adaptation, or regularization will work best. We evaluated the model on sentiment classification (SST and CFIMDB), paraphrase detection on Quora Question Pairs, and Shakespeare sonnet generation. For the classification tasks, we compared fine-tuning only the last linear layer versus fine-tuning the full model, and observed that full-model fine-tuning generally performs better, especially on the binary CFIMDB dataset, suggesting that our implementation is functioning as intended. On the paraphrase detection task, SMART-style regularization slightly improved dev accuracy over the plain full-model baseline, while LoRA variants performed worse. We then extended the default project with parameter-efficient fine-tuning and robustness-inspired regularization. In particular, we explored LoRA, a proximal SMART-lite regularizer that constrains the model to stay closer to its initialization, and a smoothness regularizer that penalizes large output-distribution changes under small worst-case perturbations to the hidden representation. On sonnet generation, full-model tuning remained strongest on the dev set with chrF 42.579, while SMART smoothness was the strongest extension at 42.363 chrF and clearly outperformed our LoRA and proximal variants. Overall, our experiments suggest that full-model adaptation remains the strongest setting for this small stylistic generation task, while smoothness regularization is a promising extension that preserves most of the benefits of full fine-tuning.

1 Key Information to include

- Mentor: Ali Khan (askhan1@stanford.edu)
- External Collaborators (if you have any): None
- Sharing project: None
- Late Days (if you are using any):

Team member	Late days remaining
Shani Su	5
Saanvi Thummalapally	1
Total pooled	6
Team size	2
Late days applied for the report	3

2 Introduction

Large autoregressive language models such as GPT-2 are strong general-purpose sequence models because they can be pretrained on large corpora with a simple next-token prediction objective and then adapted to many downstream tasks [1]. However, adapting a pretrained language model is not straightforward: different tasks may favor full-model fine-tuning, parameter-efficient adaptation, or additional regularization.

The CS224N default final project provides a useful setting for studying this question because it includes both discriminative and generative tasks. We build GPT-2 from scratch and apply it to sentiment classification, paraphrase detection, and sonnet generation. These tasks stress different abilities: sentiment classification tests sentence-level semantic representations, paraphrase detection requires finer-grained semantic comparison between sentence pairs, and sonnet generation requires fluent continuations in a Shakespearean style. We use SST [2] and CFIMDB for sentiment classification.

Our first goal was to verify that our base GPT-2 implementation worked correctly by comparing last-layer-only and full-model fine-tuning on SST and CFIMDB. Our second goal was to explore the tradeoff between efficient adaptation and robust optimization. To do this, we added three extensions: LoRA [3], a proximal regularizer that keeps trainable parameters close to initialization, and a SMART-inspired smoothness regularizer [4] that encourages stable output distributions under small adversarial perturbations to intermediate representations.

3 Related Work

Autoregressive language models and pretraining. Our work builds on the GPT family [5, 1], which showed that decoder-only transformers pretrained with next-token prediction can be adapted to many downstream tasks. The underlying architecture comes from Vaswani et al. [6], whose multi-head self-attention design we implement from scratch. Our focus is not on pretraining itself, but on how best to adapt a pretrained model: full fine-tuning, limited updates, or added regularization.

Benchmark tasks. We evaluate on the Stanford Sentiment Treebank [2], CFIMDB, and the Quora Question Pairs dataset [7]. Because these are standard benchmarks with established baselines, they are useful for validating a new implementation before studying adaptation strategies.

Parameter-efficient fine-tuning. To reduce the cost of full-model tuning, we study LoRA [3], which freezes pretrained weights and adds trainable low-rank updates to selected linear projections. While LoRA is effective on many large-model tasks, its behavior on small stylistic generation tasks is less clear, which motivates our evaluation on sonnet generation.

Robustness-oriented fine-tuning. We also study SMART [4], which combines a proximal constraint that limits movement from initialization with a smoothness regularizer that penalizes output sensitivity to small perturbations. Although SMART was introduced for classification, its effect on generative tasks like sonnet continuation is less explored. We evaluate simplified versions of both ideas in this setting.

Positioning our work. Our project is a replication and analysis study. Using a from-scratch GPT-2 implementation, we compare full fine-tuning, LoRA, and SMART-inspired regularization across sentiment classification, paraphrase detection, and constrained poetry generation.

4 Approach

Our main goal was to implement the core GPT-2 architecture and fine-tune it for downstream tasks. GPT-2 is trained as a next-token prediction model, where it processes a sequence of tokens and predicts the next token autoregressively. We adapted it for sentiment classification by using the final hidden state of the last token as a sentence representation. For SST and CFIMDB, the input is a sentence (or movie review) and the output is a sentiment label. We adapted the model for paraphrase detection by prompting it to generate yes/no when given a sentence pair.

Model. We implemented GPT-2 as a decoder-only transformer with token and positional embeddings and stacked masked self-attention plus MLP layers using causal attention:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V$$

where M is the causal mask that prevents attending to future tokens. For sentiment classification, we use the hidden state of the last non-padding token as a sentence representation and pass it through a linear layer: $\hat{y} = W_c \cdot \text{Dropout}(z) + b_c$. We train using cross-entropy loss $\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N CE(\hat{y}_i, y_i)$. For paraphrase detection, we structure the input as a prompted sentence pair:

Is ‘s1’ a paraphrase of ‘s2’? Answer ‘yes’ or ‘no’:

and then train a classifier on the final representation to predict the next token as either yes or no. This lets us reuse the GPT-2 backbone while aligning the downstream decision with a generative interpretation of the task. For sonnet generation, we fine-tune GPT-2 as a standard next-token language model on Shakespearean sonnets. During training, we minimize token-level cross-entropy over the continuation sequence. At inference time, we condition on the first three lines and generate the remainder of the poem autoregressively using temperature-scaled top- p sampling.

Baselines: We compare against the provided baselines for dev accuracy in the sentiment tasks. For paraphrase and sonnet generation, our own plain GPT-2 full-model runs serve as the main baseline against which our extensions are compared.

Extensions

LoRA. We implemented low-rank adaptation within selected linear layers of GPT-2. Instead of updating all pretrained weights, LoRA freezes the base model and adds trainable low-rank matrices to target projections. This significantly reduces the number of trainable parameters. In our experiments, we varied the LoRA rank, scaling factor, dropout, and target modules. We primarily explored applying LoRA to attention projections such as q, v and k, v .

SMART-lite proximal regularization. We added a trust-region style penalty that keeps trainable parameters close to their initialization. Let θ_0 denote the initial trainable parameters and θ the current trainable parameters. The proximal penalty is

$$\mathcal{L}_{prox} = \frac{1}{|\theta|} \|\theta - \theta_0\|_2^2$$

and the total loss becomes

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda_{prox} \mathcal{L}_{prox}.$$

Intuitively, this discourages fine-tuning from drifting too far away from the pretrained model when the downstream dataset is small.

SMART smoothness. We also implemented a second SMART-inspired component based on adversarial smoothness. Given an internal representation z and a prediction head $h(\cdot)$, we find a small perturbation δ that maximally changes the output distribution, and then penalize the difference between the clean and perturbed predictions. In practice, we use a symmetrized KL divergence:

$$\mathcal{L}_{smooth} = \text{KL}(p \| q) + \text{KL}(q \| p)$$

where p is the clean output distribution and q is the output distribution after a small worst-case perturbation to the representation. The total objective becomes

$$\mathcal{L}_{total} = \mathcal{L}_{task} + \lambda_{smooth} \mathcal{L}_{smooth}.$$

For paraphrase, this smoothness penalty is applied to the final hidden representation before the classification head. For sonnet generation, we use a simplified version based on the last real token representation, which makes the extension lightweight while still encouraging locally stable predictions.

Hyperparameter Tuning (Sonnet). In addition to fixed ablation runs, we performed a random hyperparameter sweep for sonnet generation. The sweep covered both optimization and decoding choices, including tuning mode (full-model vs. lora), learning rate, batch size, proximal regularization strength λ_{prox} for full-model runs, LoRA rank, scaling factor (α), dropout, and target

modules for LoRA runs, as well as decoding parameters such as temperature and top- p . Each sampled configuration was trained for a fixed number of epochs and logged automatically. We first ranked runs using the training objective (lower is better) to filter out poor configurations, and then evaluated the most promising candidates using held-out sonnet generation on the dev set.

Implementation Notes

We implemented the core GPT-2 components (masked multi-head attention, MLP layers, and embeddings) and handled loading pretrained weights into our model. We also built a sentiment classification head with an AdamW training loop and implemented both causal and additive attention masks for padding. We initialized from HuggingFace GPT-2 weights and used `GPT2Tokenizer` for tokenization, along with `sklearn` for accuracy and F1 computation. We implemented LoRA adapter insertion, trainable-parameter filtering, proximal regularization against the initialization, and smoothness regularization based on adversarial hidden-state perturbations. We also developed small sweep scripts to search over selected hyperparameters for LoRA and generation settings.

5 Experiments

5.1 Data + Evaluation Method

We evaluated our model on four datasets spanning three tasks.

SST [2] is a five-class sentiment classification dataset of 11,855 movie review sentences, split into 8,544 train / 1,101 dev / 2,210 test examples. The task is to predict whether a sentence expresses negative, somewhat negative, neutral, somewhat positive, or positive sentiment.

CFIMDB is a binary sentiment classification dataset of 2,434 longer movie reviews (1,701 train / 245 dev / 488 test). The task is to predict whether a review is positive or negative. Reviews are longer than those in SST, making the last-token representation a more compressed summary.

Quora Question Pairs [7] is a paraphrase detection dataset of over 400,000 question pairs, of which we use 141,506 train / 20,215 dev / 40,431 test examples. The task is to predict whether two questions express the same semantic intent. We formulate this as a cloze-style task, prompting GPT-2 to generate token yes or no.

Shakespeare Sonnets is a small conditional generation dataset of 154 Shakespearean sonnets (143 train / 11 dev / 12 test). The task is to generate the remaining lines of a sonnet conditioned on the first three lines. This is a constrained stylistic generation task: outputs are evaluated against Shakespeare’s original continuations using chrF [8].

Evaluation Method: We use accuracy as the primary metric on SST, CFIMDB, and the paraphrase detection task. For paraphrase, we also tracked dev F1 during development, but accuracy remained our main reported score. For sonnet generation, we use chrF, a character n-gram overlap metric that is provided by the project autograder and is appropriate for conditional text generation [8]. We tune hyperparameters using only the training and dev splits, and we generate final predictions for the held-out test split without using test labels for model selection.

5.2 Experimental details

- **Model config:** GPT-2 base ($d=768$, $L=12$ layers, 12 attention heads, MLP hidden size $= 4d$), initialized from pretrained GPT-2 weights.
- **Training objective:** Cross-entropy loss over K sentiment classes (SST: $K=5$, CFIMDB: $K=2$) using the last non-padding token embedding as the sentence representation; token-level next-token cross-entropy for sonnet generation.
- **Optimizer:** AdamW.
- **Batch size:** SST: 64; CFIMDB: 8; Paraphrase: 8; Sonnet: 8 for baseline full-model runs, with smaller batch sizes such as 4 explored during tuned LoRA sweeps.
- **Learning rate:** Last-linear-layer runs used 1×10^{-3} . Full-model and paraphrase baseline runs used 1×10^{-5} . For sonnet generation, the untuned full-model baseline used 1×10^{-5} , while tuned LoRA sweeps explored larger learning rates, including 3×10^{-5} and 1×10^{-4} .

- **Dropout:** 0.3 in the classification model. For sonnet LoRA runs, LoRA dropout was also tuned, with values such as 0.0, 0.05, and 0.1 explored.
- **Epochs:** Epochs: 10 for most runs, including SST, CFIMDB, and sonnet-generation experiments. Paraphrase experiments used 3 training epochs. For sonnet generation, untuned full-model, SMART-lite, and SMART smoothness runs used 10 epochs.
- **LoRA setup for sonnet generation:** We evaluated both an untuned and a tuned LoRA variant. The untuned LoRA configuration used rank $r=8$, scaling $\alpha=16$, dropout 0.05, and adaptation on the query and value projections (q, v). For the tuned LoRA result reported in Table 1, the best configuration used rank $r=4$, scaling $\alpha=32$, dropout 0.1, adaptation on q, k, v , learning rate 3×10^{-5} , batch size 4, 3 training epochs, temperature 0.9, and top- $p = 0.9$. We also explored other LoRA sweep settings, including rank 16 and larger learning rates such as 10^{-4} , but results were not up to par with the tuned LoRA result.
- **SMART regularization for sonnet generation:** We evaluated two lightweight regularization variants on full-model tuning: a proximal-style SMART-lite setting using $\lambda=10^{-4}$, and a smoothness-based SMART variant using $\lambda=0.1$, perturbation size $\epsilon=10^{-3}$, and 1 inner step.
- **Generation details:** Sonnet outputs were generated with top- p sampling and temperature scaling. Because decoding materially affected chrF, we also swept decoding hyperparameters for tuned LoRA runs, including temperature values up to 1.3 and top- p values up to 0.95.
- **Evaluation metric:** Classification tasks were evaluated with accuracy, while sonnet generation was evaluated with chrF on the held-out development set.
- **Hardware:** Lightning.ai GPU runtime (CUDA-enabled GPU); local Mac MPS was used during earlier debugging.

5.3 Results

Table 1: Sentiment, paraphrase, and sonnet generation results.

Setting	Dev Acc / chrF	Test Acc / chrF
SST last-linear-layer (Baseline)	0.462	—
SST last-linear-layer (Ours)	0.467	0.471
SST full-model (Baseline)	0.513	—
SST full-model (Ours)	0.501	0.519
CFIMDB last-linear-layer (Baseline)	0.861	—
CFIMDB last-linear-layer (Ours)	0.853	—
CFIMDB full-model (Baseline)	0.976	—
CFIMDB full-model (Ours)	0.980	—
Quora paraphrase (full-model)	0.872	0.867
Quora paraphrase p1 (SMART small)	0.889	0.882
Quora paraphrase p2 (SMART medium)	0.890	0.882
Quora paraphrase p3 (LoRA q, v)	0.867	—
Quora paraphrase p5 (LoRA q, k, v)	0.867	—
Quora paraphrase p9 (LoRA + SMART)	0.863	—
Sonnet GPT-2 full-model	42.579	42.140
Sonnet + LoRA (untuned)	40.594	—
Sonnet + LoRA (tuned)	41.532	41.475
Sonnet + SMART-lite	41.490	—
Sonnet + SMART smoothness	42.363	41.587

Full-model fine-tuning performs better than training only the final linear layer, especially on CFIMDB where we slightly outperform the baseline. This suggests sentiment information is distributed across the model and benefits from updating deeper layers. On SST, our full-model performance is close to the baseline but still slightly lower, likely reflecting the greater difficulty of fine-grained sentiment classification. Overall, these results indicate that our GPT-2 implementation is working as intended

and is able to learn useful task-specific representations. Our model also successfully transferred to the paraphrase detection task, where SMART-based regularization slightly improved dev and test accuracy over the plain full-model baseline.

For sonnet generation, full-model fine-tuning achieved our strongest dev (as well as test) chrF score of 42.579, with SMART smoothness close behind at 42.363. Tuned LoRA improved substantially over our untuned LoRA setting (41.532 vs. 40.594 chrF), showing that parameter-efficient adaptation was highly sensitive to rank, target modules, dropout, learning rate, training duration, and decoding choices. However, even after tuning, LoRA still remained below the full-model baseline. SMART-lite proximal regularization also underperformed the baseline at 41.490 chrF, suggesting that directly constraining the model to stay close to its pretrained initialization can limit the stylistic adaptation needed for Shakespearean sonnet continuation. Importantly, these differences were not only numerical: manual inspection showed that weaker settings, especially untuned LoRA, were more likely to produce corrupted or off-domain continuations, while stronger full-model and smoothness-regularized runs more often stayed in a sonnet-like register. Overall, full-model tuning remained the strongest setting, and SMART smoothness was the most promising extension because it preserved most of the gains of full fine-tuning while regularizing generation more gently than the proximal penalty.

6 Analysis

Paraphrase

Full-model tuning versus extensions. Whereas full-model tuning was strongest for sonnet generation, on Quora paraphrase detection our SMART-based variants slightly outperformed the plain full-model baseline. The baseline full-model run achieved 0.872 dev accuracy, while SMART small and SMART medium improved this to 0.889 and 0.890 respectively. In contrast, the LoRA variants underperformed the baseline, with both the q, v and q, k, v settings reaching 0.867 dev accuracy, and the LoRA + SMART combination falling further to 0.863. This suggests that for paraphrase detection, modest regularization of the learned representation can help generalization, while parameter-efficient adaptation alone may not provide enough flexibility to capture the semantic alignment required by the task.

Why SMART helped more on paraphrase. We think the paraphrase task benefits more directly from smoothness-style regularization because the label depends on a stable semantic relationship between two inputs rather than on stylistic generation. Small perturbations to the hidden representation should ideally not flip a correct paraphrase judgment, so encouraging local output stability is well matched to the task. This is consistent with the original motivation of SMART, which was developed for supervised language understanding settings rather than open-ended text generation.

Why LoRA lagged behind. LoRA was less successful on paraphrase than we initially expected. One likely reason is that our formulation already constrains the task through a prompted yes/no setup, so the model must both interpret the prompt format and compare the semantic content of the two questions using a relatively small trainable subspace. With limited low-rank updates, the model may not be able to fully adapt the relevant attention and representation layers needed for fine-grained sentence-pair reasoning. The fact that moving from q, v to q, k, v did not improve dev accuracy suggests that the issue was not only target-module choice, but also the overall expressive limit of the low-rank adaptation in our setting.

Task formulation and likely error modes. Our paraphrase setup reuses GPT-2 in a generative way by prompting the model to produce yes or no. This is simple and effective, but it also introduces possible sources of error beyond pure semantic reasoning. For example, the model must be sensitive to prompt wording and punctuation, and performance may depend on whether the two candidate answer tokens are equally easy for the pretrained tokenizer and language model head to model. We suspect that some residual errors come from lexical-overlap shortcuts: question pairs with high word overlap but meaning differences, or low-overlap pairs that are true paraphrases, are likely harder than more straightforward examples.

Qualitative behavior and error analysis (SMART medium). We manually inspected several dev examples from our best-performing paraphrase model, the SMART medium configuration, to understand what kinds of pairs it handled well and where it was more brittle. The model performed well on near-paraphrases with small lexical or syntactic variation, such as “How do you apply for a visa?” versus “How does one apply for a visa?” and “How can I avoid sleepiness during lectures?”

versus “How can I keep myself awake during boring lectures?” It also correctly rejected many high-overlap non-paraphrases in which only a key entity or condition changed, such as “What are the tips and hacks for getting the classes that you want as a freshman at Chicago State?” versus the same question about NYU, and “Where is the Baraka River located, and how does it compare to the Salween River?” versus the same question about the Japura River. These examples suggest that the model often captures both local wording similarity and some finer semantic distinctions. At the same time, borderline cases remained challenging, especially when two questions shared topic words but differed in intent, scope, or implied answer structure. Overall, the qualitative patterns are consistent with the quantitative results: SMART regularization appears to improve robustness on sentence-pair judgments, but the model still struggles on subtle semantic distinctions and occasional ambiguous examples.

Takeaway. Overall, the paraphrase results strengthen the main conclusion of our project that the best adaptation strategy depends on the task. For semantic classification-style problems, lightweight robustness-oriented regularization can be helpful and even outperform plain full-model tuning. For stylistically constrained generation, however, the same regularization benefits are smaller and full-model adaptation remains strongest. This contrast was useful because it showed that our extensions were not uniformly good or bad, but instead interacted in meaningful ways with task structure.

Sonnet

Full-model tuning versus LoRA. The clearest pattern in our sonnet results is that full-model tuning remained the strongest setting. Our initial untuned LoRA run dropped to 40.594 chrF, substantially below the full-model baseline, which initially made it unclear whether LoRA was simply a poor fit or whether our settings were poorly chosen. After additional experiments, we found that LoRA was highly sensitive to the adaptation rank, target modules, dropout, learning rate, epoch count, and decoding parameters. In particular, targeting only q, v was weaker than targeting q, k, v , and smaller-rank, shorter-training LoRA runs with tuned decoding settings improved performance to 41.532 chrF. Even so, the best LoRA variant still remained below the full-model baseline. This suggests that the sonnet task requires broader stylistic adaptation than a low-rank update can provide, at least at the scale and budget we explored.

Effect of proximal regularization. The proximal SMART-lite penalty consistently trailed the full-model sonnet baseline, achieving 41.490 chrF. This result is informative because the proximal penalty is meant to stabilize adaptation by discouraging the model from moving too far away from pretrained GPT-2. On a small dataset, this can help prevent overfitting. However, for sonnet generation, the model appears to need substantial stylistic adaptation to match the domain. Penalizing deviation from the initialization likely constrained that adaptation too strongly, producing continuations that remained closer to generic pretrained GPT-2 behavior than to Shakespearean sonnet continuation.

Effect of smoothness regularization. SMART smoothness performed much better than SMART-lite proximal regularization and came close to matching the baseline, reaching 42.363 chrF. This is encouraging because it suggests that not all regularization is equally harmful. Rather than directly restricting parameter movement, the smoothness term regularizes the sensitivity of the output distribution to perturbations in hidden space. For generation, this appears to preserve task-specific learning better than the proximal penalty while still encouraging robustness. Among our extensions, this was the strongest setting on the sonnet task.

Sensitivity of sonnet generation to evaluation setup. During development, we found that sonnet results were especially easy to mis-measure. In particular, using the wrong held-out split for dev versus test caused large changes in chrF that were unrelated to model quality. We also found that generation quality was more sensitive than classification accuracy to decoding parameters such as temperature and top- p . This means that relatively small changes in generation settings can produce noticeable shifts in chrF even when the underlying trained model is similar. As a result, careful evaluation hygiene was particularly important for this task.

Qualitative behavior and error analysis. Manual inspection of the 12 dev continuations revealed clear behavioral differences across methods that aligned with the chrF ranking. The untuned LoRA model was the weakest not only quantitatively but also qualitatively: several generations drifted out of Shakespearean register entirely and contained obvious corruption or off-domain insertions, including artifacts such as “pdf del Rey,” “Legislation and Parliamentary cloud,” and other malformed phrases.

Tuned LoRA improved substantially over this baseline, but its failure mode shifted rather than disappearing. Instead of catastrophic corruption, it more often produced repetitive or semantically looping continuations. For example, for the prompt beginning “O, call not me to justify the wrong ... / Wound me not with thine eye but with thy tongue,” the tuned LoRA continuation repeatedly reuses phrases like “with thy tongue” and “with thy heart” rather than developing the poem forward. Similarly, in other examples it loops heavily on abstract words such as “truth” or “love,” making the continuation look superficially sonnet-like while weakening rhetorical progression.

SMART smoothness was qualitatively much closer to full-model tuning. Its continuations more often stayed in an archaic, sonnet-like register and were less prone to catastrophic corruption than either LoRA variant. However, it still showed grammatical instability, invented words, and incomplete endings, so it did not fully solve the structural challenges of the task. The full-model baseline had the strongest overall qualitative behavior: relative to the extensions, it more often preserved local coherence, emotional continuity with the prompt, and stylistic consistency without collapsing into the same degree of corruption or repetition. At the same time, even the strongest models still struggled to reliably enforce global sonnet structure, especially sustained rhetorical development, meter, and a convincing closing couplet.

Takeaway. Taken together, these experiments suggest that for the sonnet task, the limiting factor is not merely parameter count or regularization strength in isolation, but whether the adaptation mechanism preserves enough expressive freedom for stylistic generation while still remaining stable. In our setting, full-model tuning best achieved that balance. SMART smoothness came closest among the extensions because it regularized generation without constraining adaptation as directly as the proximal penalty, while tuned LoRA improved substantially but still lagged behind due to repetition and reduced discourse-level control.

7 Conclusion

In this project, we implemented GPT-2, loaded pretrained weights, and adapted the model to sentiment classification, paraphrase detection, and sonnet generation. Our sentiment results show that full-model fine-tuning is generally stronger than training only the last linear layer, especially on CFIMDB, supporting the correctness of our implementation. The model also transferred successfully to the Quora paraphrase task.

Our extension experiments on sonnet generation further clarified tradeoffs between adaptation strategies. Full-model fine-tuning remained the strongest overall setting. LoRA improved substantially after tuning, showing that parameter-efficient adaptation is highly sensitive to hyperparameters and decoding choices, but even the best configuration did not match the full-model baseline. SMART-lite proximal regularization also underperformed, likely because it constrained adaptation too strongly for a task requiring meaningful stylistic shifts from pretrained GPT-2 behavior. SMART smoothness performed much better and came close to the baseline, making it our most promising extension. Qualitatively, untuned LoRA often produced corrupted or off-domain continuations, tuned LoRA tended toward repetition and semantic looping, and the strongest full-model and smoothness-regularized runs more consistently preserved sonnet-like phrasing and local coherence.

One limitation of this work is that we do not yet have final leaderboard test results for every extension setting, and generation results remain somewhat sensitive to evaluation configuration and decoding choices. Future work could complete the remaining evaluations, more thoroughly compare full-model and parameter-efficient adaptation, and perform hyperparameter search on the paraphrase task as we didn’t have enough compute for it.

Team contributions

Shani Su: Contributed to the core GPT-2 implementation and debugging process, including editing model code, implementing GPT-2 components, and helping ensure the required sanity checks passed. Worked on the sentiment classification portion of the project with GPT-2, and implemented the main extensions explored in the paper: LoRA, SMART-lite proximal regularization, and SMART smoothness regularization. Also led much of the sonnet-generation experimentation and analysis, including training runs, hyperparameter exploration, and qualitative error analysis, while assisting with paraphrase-training experiments as needed. Contributed to writing and editing the final report,

especially the abstract, extensions, approach, sonnet analysis, and part of the conclusion.

Saanvi Thummalapally: Contributed to the core GPT-2 implementation and debugging process, including verifying model components, assisting with integration of pretrained weights, and ensuring the overall training pipeline functioned correctly. Worked on implementing and experimenting with LoRA for parameter-efficient fine-tuning, including running and analyzing LoRA training sweeps, and also helped with SMART. Led the paraphrase detection portion of the project, including implementing the prompted yes/no formulation, running training experiments, and analyzing results, as well as contributed to the sonnet portion code. Contributed to experimental design and evaluation across tasks, including interpreting model behavior and comparing adaptation strategies. Played a major role in writing and editing the final report, particularly the introduction, related work, experiments, paraphrase analysis, and conclusion.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.
- [2] 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 *EMNLP*, 2013.
- [3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [4] 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 *ACL*, 2020.
- [5] Alec Radford. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018.
- [6] 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.
- [7] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *UK Special Interest Group for Computational Linguistics Research Colloquium*, 2008.
- [8] Maja Popović. chrF: Character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015.