

Task-Dependent Effects of Parameter-Efficient Fine-Tuning: A GPT-2-Based Study

Stanford CS224N Default Project

Weiwei Wu

Department of Electrical Engineering
Stanford University
wwwu@stanford.edu

Abstract

We study how a shared GPT-2 backbone adapts to three tasks: sentiment classification, cloze-style paraphrase detection, and Shakespearean sonnet generation [1]. We compare full fine-tuning with parameter-efficient Low-Rank Adaptation (LoRA) applied to the attention query, key, and value projections [2]. Full fine-tuning reproduces strong sentiment baselines, reaches 0.860 development accuracy on paraphrase detection, and provides a strong starting point for sonnet generation. On SST, CFIMDB, and paraphrase detection, LoRA reduces trainable parameters by roughly $280\times$ but consistently underperforms full fine-tuning. Sonnet generation behaves differently. A decode-only sweep increases development chrF from 23.61 to 38.65 without retraining, and the best checkpoint is selected using dev chrF rather than language-model loss. Under matched decoding settings, LoRA achieves 39.99 dev chrF, outperforming full fine-tuning (37.96) while training only 0.35% of the parameters. A controlled top- k plus repetition-penalty refinement does not improve the baseline generator. Overall, parameter-efficient adaptation is not uniformly beneficial across tasks, but can be particularly effective for small-data generation settings.

1 Key Information to include

- Mentor: Tristan Thrush.
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Pretrained decoder-only language models have become a common backbone for both discriminative and generative NLP tasks. A standard adaptation strategy is full fine-tuning, where all model parameters are updated for a new task. While effective, full fine-tuning can be computationally expensive and may overfit when supervision is limited. Parameter-efficient methods such as Low-Rank Adaptation (LoRA) instead update a small set of parameters while keeping the pretrained weights fixed, greatly reducing the number of trainable parameters [2]. However, it remains unclear when such low-rank updates are sufficient for downstream tasks.

We study this question using a shared GPT-2 implementation applied to three tasks with different supervision structures: sentiment classification on SST and CFIMDB, cloze-style paraphrase detection on Quora, and Shakespearean sonnet generation. These tasks span both classification and generation settings and provide a useful test bed for comparing adaptation strategies.

We evaluate full fine-tuning and LoRA under matched training settings across all tasks. For the generation task, we additionally separate decoding effects from training effects through controlled decoding sweeps and checkpoint analysis. Evaluation combines quantitative metrics (accuracy and chrF) with qualitative inspection of generated text.

Our results show that the effectiveness of parameter-efficient adaptation is task-dependent. Full fine-tuning performs better on the classification tasks, sentiment classification and paraphrase detection, where LoRA trades accuracy for efficiency. Sonnet generation exhibits a different pattern: decoding choices strongly influence generation quality, and LoRA achieves the best development chrF among the matched runs. These results highlight where low-rank adaptation can replace full fine-tuning and where broader model updates remain important.

3 Related Work

Our work builds on GPT-2, a decoder-only Transformer that demonstrated how large-scale language-model pretraining can transfer to many downstream tasks through fine-tuning [1]. In this paradigm, a pretrained model provides a shared representation that can be adapted to both discriminative and generative tasks.

For sentiment classification we use the Stanford Sentiment Treebank (SST), a standard benchmark for sentence-level sentiment analysis [3]. For generation evaluation we use chrF, a character-level metric that measures n-gram overlap between generated and reference text [4].

Our main methodological extension is Low-Rank Adaptation (LoRA), a parameter-efficient fine-tuning method that freezes pretrained weights and learns low-rank updates inside selected linear layers [2]. This approach greatly reduces the number of trainable parameters while often retaining much of the performance of full fine-tuning.

Our sonnet-generation analysis also relates to work on neural text degeneration and decoding strategies. Holtzman et al. [5] show that sampling methods such as temperature scaling and nucleus sampling strongly influence generation quality. Motivated by this, we perform controlled decoding sweeps and checkpoint analysis to separate decoding effects from training effects.

4 Approach

4.1 Model Architecture

4.1.1 GPT-2 Backbone

We use GPT-2 small as a decoder-only Transformer language model [1]. The model contains learned token and positional embeddings followed by 12 decoder blocks. Each block contains masked multi-head self-attention, a position-wise feed-forward network, residual connections, layer normalization, and dropout. The causal attention mask prevents each token from attending to future positions, so the hidden state at position t depends only on tokens up to t .

For one attention head, the masked self-attention computation is

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

where M is an upper-triangular mask with large negative values above the diagonal. This backbone is shared across all three tasks. Classification tasks use the final token representation as input to a linear head, while sonnet generation uses the full sequence of hidden states for next-token prediction.

4.1.2 LoRA Architecture

LoRA replaces a full update of a pretrained linear map W with a low-rank update

$$W' = W + \Delta W, \quad \Delta W = \frac{\alpha}{r}BA,$$

where A and B are trainable low-rank matrices and $\text{rank}(\Delta W) = r \ll d$ [2]. In our experiments, the pretrained GPT-2 weights remain frozen and only the adapter weights and task-specific heads are updated. We apply LoRA to the attention query, key, and value projections with rank (r) 8, scaling (α) 16, and zero adapter dropout. This keeps the model interface unchanged while reducing the number of trainable parameters by over two orders of magnitude.

4.2 Generation and Decoding

For sonnet generation, tokens are produced autoregressively using

$$p(x_t \mid x_{<t}) = \text{Softmax}(z_t),$$

where z_t are the logits produced by the GPT-2 backbone.

Temperature scaling. Temperature modifies the sharpness of the distribution:

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}.$$

Lower temperatures make generation more deterministic, while higher temperatures increase randomness. Our default generator uses $T = 1.2$.

Nucleus (top- p) sampling. Nucleus sampling restricts generation to the smallest token set whose cumulative probability exceeds p :

$$\sum_{i \in S} p_i \geq p.$$

Sampling then occurs within this truncated set. Our default generator uses $p = 0.9$.

Top- k filtering. Top- k sampling restricts the candidate set to the k tokens with the highest probability before sampling:

$$S = \text{TopK}(p, k).$$

This further limits low-probability tokens and can sometimes reduce incoherent outputs. In our generator-refinement experiments we evaluate $k = 60$.

Repetition penalty. To discourage the model from repeating previously generated tokens, a repetition penalty modifies the logits of tokens that have already appeared in the sequence. If token i has appeared before, its logit is scaled by a penalty factor $\gamma > 1$, reducing its probability during sampling.

The baseline generation experiments use temperature-scaled nucleus sampling. The generator-refinement experiments introduce top- k filtering and repetition penalties during decoding while leaving the trained model unchanged, following the same general decoding-control motivation discussed by Holtzman et al. [5].

The shared GPT-2 backbone is then adapted to three task-specific prediction settings.

4.3 Sentiment Classification

Baseline. For SST and CFIMDB databases, we use GPT-2 as a sentence encoder and attach a linear classification head to the final token representation. We evaluate two baseline settings: a last-linear-layer model, where GPT-2 is frozen and only the classifier is trained, and a full-model setting, where all GPT-2 parameters are updated.

Extensions. We extend the sentiment task with LoRA adapters applied to the GPT-2 attention query, key, and value projections. The base GPT-2 weights are frozen and only the low-rank adapter weights plus the classifier head are trained. We compare LoRA against full fine-tuning using matched seeds, matched epoch budgets, and matched evaluation metrics. We also report trainable parameter counts and training time so that efficiency comparisons are based on explicit measurements rather than parameter counts alone.

4.4 Paraphrase Detection

Baseline. We cast Quora paraphrase detection as a cloze-style next-token prediction problem. Each pair of questions is converted into a prompt of the form: Is "<sentence 1>" a paraphrase of "<sentence 2>"? Answer "yes" or "no":. GPT-2 predicts the next-token distribution, and the model is trained to prefer the token for yes or no. The baseline model fully fine-tunes GPT-2 under this formulation.

Extensions. We apply the same LoRA design used for sentiment classification to paraphrase detection. This keeps the prompt format, train-dev-test split, and selection rule unchanged while testing whether low-rank adaptation can retain cloze-style accuracy. As in the sentiment extension, we report trainable parameter counts and training time alongside task accuracy to evaluate the efficiency–performance tradeoff.

4.5 Sonnet Generation

Baseline. For sonnet generation, we fine-tune GPT-2 with standard next-token prediction on the Shakespeare corpus and generate the remainder of each held-out sonnet conditioned on its first three lines. The raw baseline uses autoregressive sampling with temperature 1.2 and top-p 0.9. Because generation quality depends on decoding as well as training, we evaluate outputs on a held-out development split using chrF and inspect representative generations qualitatively.

Extensions. The sonnet task receives the largest extension among all three tasks. First, we run a decode-only sweep over temperature and top-p on a fixed checkpoint to isolate generation-time effects. Second, we freeze the best decode setting from that sweep and evaluate checkpoints across training to study the relationship between training loss, development loss, and dev chrF. Third, we test a generator refinement that adds top-k filtering and a repetition penalty; this becomes a controlled

negative result because it does not outperform the baseline generator. Finally, we apply LoRA to the sonnet model with adapters on the attention query, key, and value projections and compare it directly against full fine-tuning using the same decode settings, evaluation metric, and checkpoint-selection rule.

5 Experiments

5.1 Data

5.1.1 Sentiment Classification

Baseline. We use two sentiment datasets. The Stanford Sentiment Treebank five-way classification dataset (SST-5) contains 8,544 training examples, 1,101 development examples, and 2,210 test examples. The CFIMDB dataset is a binary sentiment dataset with 1,701 training examples, 245 development examples, and 488 test examples. In both cases, the model input is a single sentence or review tokenized with the GPT-2 tokenizer, and the output is a sentiment class label.

Extensions. The LoRA sentiment experiments reuse the same train, development, and test splits. Keeping the data and task formulation unchanged isolates the effect of the fine-tuning method.

5.1.2 Paraphrase Detection

Baseline. We use the provided Quora paraphrase dataset with fixed train, development, and test splits. Each example consists of a pair of questions and a binary paraphrase label. Following the formulation described in the Approach section, each pair is converted into a cloze-style prompt and GPT-2 predicts the next-token distribution. The model output is the token yes or no.

Extensions. The LoRA paraphrase experiments use the same prompts and the same provided dataset splits. This ensures that any differences between the baseline and LoRA models reflect the adaptation method rather than changes in data or task formulation.

5.1.3 Sonnet Generation

Baseline. We use the Shakespeare sonnet corpus packaged with the project codebase. The training set consists of the available sonnets in the corpus, and the repository also provides a small development set of 12 held-out sonnets with gold references for local evaluation. During evaluation, the model receives the first three lines of each sonnet as a prompt and generates the remaining lines. Test generation uses a separate held-out prompt file without local gold targets.

Extensions. All sonnet extensions use the same training corpus and the same held-out development and test prompt files. Because the sonnet dataset is relatively small, keeping the dataset fixed ensures that any observed differences are due to modeling or decoding changes rather than data variation.

5.2 Evaluation Method

5.2.1 Sentiment Classification

Baseline. The primary evaluation metric for sentiment classification is development accuracy. Models are selected based on the highest dev accuracy, and test predictions are produced in the required submission format.

Extensions. For the LoRA comparison, dev accuracy remains the primary model-selection metric. We additionally report macro-F1 as a diagnostic metric to assess class-wise performance differences. We also report trainable parameter counts and training time to analyze the efficiency-performance tradeoff between full fine-tuning and parameter-efficient adaptation.

5.2.2 Paraphrase Detection

Baseline. The main evaluation metric for paraphrase detection is development accuracy. Each model predicts either the token yes or no for the cloze prompt described in the Approach section, and predictions are saved in the standard leaderboard submission format.

Extensions. For the LoRA extension, we additionally report macro-F1 as a diagnostic metric. Because the dataset is not perfectly balanced, macro-F1 helps interpret whether changes in accuracy reflect threshold-like behavior or broader shifts in prediction quality. Checkpoints are still selected based on dev accuracy to remain consistent with the baseline formulation.

5.2.3 Sonnet Generation

Baseline. We evaluate sonnet generations using development chrF, which measures character-level n -gram overlap between generated text and held-out reference sonnets [4]. Because chrF can reward surface overlap even when outputs are repetitive, we also inspect representative generations qualitatively.

Extensions. The same dev chrF metric is used for decode sweeps, training-dynamics analysis, generator ablations, and the LoRA comparison. Checkpoints are selected using dev chrF rather than training loss or final epoch, and the same development split is used across all sonnet analyses to ensure comparability.

5.3 Experimental Details

5.3.1 Sentiment Classification

Baseline. For the baseline sentiment experiments, we first performed learning-rate searches. The last-linear-layer model performed best at learning rate 4×10^{-3} , while the full-model baseline performed best at 2×10^{-5} . These baseline sweeps were trained for 15 epochs, with the best development performance typically appearing around epochs 8–10. These runs produce the baseline reproduction reported in Table 1.

Extensions. For the LoRA comparison, we evaluate full fine-tuning and LoRA under matched settings: the same random seed, 12 training epochs, learning rate 2×10^{-5} , and batch size 32. The last-linear-layer result is retained as a reference baseline, while the main comparison focuses on full fine-tuning versus LoRA. We record trainable parameter counts and training time to measure the efficiency–performance tradeoff (see Table 2).

5.3.2 Paraphrase Detection

Baseline. The paraphrase baseline uses full fine-tuning for 4 epochs with learning rate 5×10^{-5} and the standard cloze prompt described in the Approach section. The provided train and development files are kept unchanged, and models are selected using development accuracy. The resulting development performance is reported in Table 3.

Extensions. The LoRA paraphrase run uses the same 4-epoch budget and learning rate as the full-model run. Unless otherwise specified, the batch size is the default value of 8 used by the paraphrase script. LoRA uses rank 8, scaling factor 16, zero adapter dropout, and adapters applied to the attention query, key, and value projections. The matched comparison between full fine-tuning and LoRA is reported in Table 4.

5.3.3 Sonnet Generation

Baseline. Our raw sonnet baseline is trained for 8 epochs with learning rate 10^{-5} and batch size 32. Generation uses temperature 1.2 and top- p 0.9. To separate training effects from decoding effects, we fix one trained checkpoint and perform a decode sweep over five settings labeled D0–D4: D0 = (0.5, 0.8), D1 = (0.6, 0.85), D2 = (0.7, 0.95), D3 = (0.8, 0.9), and D4 = (0.9, 0.95), where each pair denotes temperature and top- p . The results of this decode sweep are reported in Table 5. After selecting the best decode setting, we evaluate checkpoints from a 10-epoch full fine-tuning run at epochs 1, 2, 4, 6, 8, and 10, whose training dynamics are summarized in Table 6.

Extensions. The decode sweep fixes the checkpoint and varies only temperature and top- p . The generator-refinement study keeps training unchanged and introduces top- k filtering and a repetition penalty at evaluation time. The results of this ablation are summarized in Table 7. For the LoRA comparison, we keep the same decode settings as the matched full fine-tuning run and evaluate checkpoints on the same schedule. LoRA again uses rank 8, scaling factor 16, zero adapter dropout, and adapters applied to the attention query, key, and value projections. The matched comparison between full fine-tuning and LoRA is reported in Table 8.

5.4 Results

Across the classification tasks, full fine-tuning consistently outperforms LoRA, while the sonnet generation task exhibits the opposite trend. The tables below report detailed results for each task.

5.4.1 Sentiment Classification

Baseline. Table 1 reports the baseline sentiment results and compares them with the standard reference development accuracies used for verification.

Dataset	Method	Reference Dev	Our Dev
SST	Last Linear Layer	0.462	0.457
SST	Full Model	0.513	0.513
CFIMDB	Last Linear Layer	0.861	0.865
CFIMDB	Full Model	0.976	0.980

Table 1: Baseline sentiment results compared with standard reference development accuracies.

Extensions. Table 2 reports the matched comparison between full fine-tuning and LoRA for SST and CFIMDB.

Dataset	Method	Trainable Params	Best Epoch	Dev Acc	Dev F1	Train Time (s)	Wall Time (s)
SST	Full FT	125,034,245	8	0.513	0.508	427.17	631.82
SST	LoRA	446,213	12	0.431	0.316	289.91	489.79
CFIMDB	Full FT	125,031,938	4	0.980	0.980	740.19	1046.90
CFIMDB	LoRA	443,906	12	0.922	0.922	594.61	929.72

Table 2: Matched sentiment comparison between full fine-tuning and LoRA.

5.4.2 Paraphrase Detection

Baseline. Table 3 reports the full fine-tuning paraphrase baseline.

Method	Best Epoch	Dev Acc	Dev F1
Full FT	4	0.860	0.884

Table 3: Paraphrase full fine-tuning baseline.

The corresponding test accuracy for this baseline model is 0.893.

Extensions. Table 4 reports the matched comparison between full fine-tuning and LoRA for paraphrase detection.

Method	Trainable Params	Best Epoch	Dev Acc	Dev F1	Train Time (s)	Wall Time (s)
Full FT	125,031,938	4	0.860	0.884	10517.92	10916.51
LoRA	443,906	4	0.846	0.840	6842.94	7262.68

Table 4: Paraphrase comparison between full fine-tuning and LoRA.

5.4.3 Sonnet Generation

Baseline. Table 5 reports the decode-only sweep, and Table 6 reports the full fine-tuning training dynamics under the selected decode setting.

Trial	Temperature	Top-p	Dev chrF
Raw baseline	1.2	0.9	23.61
D0	0.5	0.8	33.47
D1	0.6	0.85	34.63
D2	0.7	0.95	38.03
D3	0.8	0.9	38.65
D4	0.9	0.95	37.87

Table 5: Sonnet decode-only sweep on a fixed checkpoint. D0–D4 correspond to the temperature and top- p pairs defined in the experimental details.

Epoch	Train Loss	Dev Loss	Dev chrF
1	4.523630	4.329052	35.528
2	4.387959	4.242442	31.886
4	4.257075	4.127629	31.381
6	4.146423	4.027250	39.280
8	4.078826	3.990640	34.985
10	4.019321	3.953976	37.617

Table 6: Sonnet full fine-tuning training dynamics evaluated using the fixed D3 decoding setting.

Extensions. Table 7 reports the generator-refinement ablation, and Table 8 reports the matched comparison between full fine-tuning and LoRA.

Setting	Best Dev chrF	Delta vs Baseline
Baseline generator (epoch 6)	39.280	0.000
top_k=60, fixed D3 decode	37.004	-2.276
repetition_penalty=1.02, fixed D3 decode	35.994	-3.286
Best modified rescue setting	38.341	-0.939

Table 7: Sonnet generator-refinement summary at the selected epoch-6 checkpoint. The best modified decode (top_k=60 & repetition_penalty=1.02) remains below the baseline generator.

Method	Trainable Params	Best Dev chrF	Wall Time (s)	Train Time (s)
Full FT	125,030,400	37.961	231.58	28.45
LoRA	442,368	39.995	293.90	23.32

Table 8: Matched sonnet comparison between full fine-tuning and LoRA.

Under the same decoding setting, the final model achieves a test chrF of 38.352.

Task	Full FT Best Dev	LoRA Best Dev	Delta	Parameter Reduction
SST	0.513	0.431	-0.083	99.65%
CFIMDB	0.980	0.922	-0.057	99.65%
Paraphrase	0.860	0.846	-0.014	99.65%
Sonnet chrF	37.961	39.995	+2.034	99.65%

Table 9: Cross-task summary comparing the best full fine-tuning and LoRA development results.

6 Analysis

6.1 Sentiment Classification

Baseline. The sentiment baselines are useful because they reproduce the reference numbers closely. SST full fine-tuning matches the reference value exactly, and CFIMDB full fine-tuning improves slightly from 0.976 to 0.980. This supports the correctness of the GPT-2 forward path, masking, and optimization. The earlier 15-epoch sweeps also show mild overfitting: the best development checkpoints appear around epochs 8 to 10 rather than at the final training step.

Extensions. LoRA does not preserve quality on either sentiment dataset in our matched runs. On SST, LoRA is 0.083 absolute accuracy below full fine-tuning. On CFIMDB, the gap is 0.057. These gaps are large enough that they are unlikely to be explained by trivial run-to-run noise. The efficiency gains are real: LoRA trains only about 0.35% of the parameters and reduces full-run training time on both datasets. However, the time-to-best-checkpoint story is weaker than the full-run story. On SST, LoRA reaches its best score only at epoch 12, while full fine-tuning peaks at epoch 8. On CFIMDB, full fine-tuning peaks as early as epoch 4, so LoRA is slower to its own best result even though its full 12-epoch run is shorter. The evidence therefore supports a narrow claim: LoRA is much more parameter-efficient on sentiment classification, but it is not quality-competitive with full fine-tuning in this setup.

6.2 Paraphrase Detection

Baseline. The paraphrase baseline performs well in absolute terms, reaching 0.860 development accuracy and 0.893 test accuracy. Because the reported test score is higher than the development score, we avoid over-interpreting small development-set differences in isolation. The main takeaway is that the full fine-tuning baseline remains strong enough to serve as a meaningful comparison point for the LoRA study.

Extensions. LoRA again produces a quality–efficiency tradeoff rather than a win on both axes. The trainable parameter count drops from 125,031,938 to 443,906, which is approximately a 99.65% reduction. Summed training time drops by about 35%, and wall-clock time drops by about 33%. The cost is accuracy: development accuracy falls from 0.860 to 0.846. This gap is smaller than in the sentiment tasks, but it still favors full fine-tuning. Because both methods peak at epoch 4, this comparison is cleaner than the sentiment time-to-best comparison. The conclusion is therefore modest but direct: on paraphrase detection, the LoRA configuration used here is cheaper, but still slightly worse than full fine-tuning on quality.

6.3 Sonnet Generation

Baseline. The sonnet task illustrates why generation requires more analysis than a single final score. First, decoding alone changes the metric substantially. Moving from the raw baseline decode to D3 raises development chrF from 23.61 to 38.65, a gain of 15.04 points without retraining. Second, the checkpoint that minimizes language-model loss is not the checkpoint that maximizes generation quality. In Table 6, both training loss and development loss decrease steadily from epoch 1 to epoch 10, but development chrF peaks at epoch 6 and then falls. This behavior motivates selecting checkpoints using development chrF rather than the final training epoch or language-model loss alone.

Table 10 provides a qualitative example from the recorded decode sweep. The D0 continuation quickly collapses into rigid repetition, while D3 maintains more variation and longer coherent phrases. This qualitative difference aligns with the large chrF improvement observed between D0 and D3.

Extensions. The sonnet extensions produce one negative result and one positive result. The negative result is the generator refinement. Adding top- k filtering and repetition penalties does not improve the baseline generator. The modified decoder produces more repetitive and less varied continuations than the baseline generator at the selected epoch-6 checkpoint. This kind of sensitivity to decoding controls is consistent with prior work showing that generation quality can change substantially with sampling choices [5]. The tracked prompts from the experiment log show the same qualitative pattern: for the prompt beginning “Love is my sin, and thy dear virtue hate,” the modified decoder falls into stronger template reuse, while for the prompt beginning “Thine eyes I love, and they, as pitying me,” repetition appears earlier and persists for longer spans of text.

The positive result is LoRA. In the matched comparison, LoRA improves development chrF by about 2.03 points over full fine-tuning while using only about 0.35% of the trainable parameters. The timing evidence is mixed. LoRA has lower summed training time but higher wall-clock time in this specific run, so we do not claim a uniform end-to-end speedup. The evidence-backed claim is therefore narrower: on this small-data generation task, LoRA achieves the best development chrF among the matched runs while remaining dramatically more parameter-efficient. Under the same decoding configuration, the final model also reaches a dev chrF of 39.995 and test chrF of 38.352. This pattern suggests that the low-rank updates used by LoRA are sufficient to adapt the pretrained model to the stylistic structure of the sonnet task, even though the same configuration was not competitive for the classification tasks.

7 Conclusion

This study compares full fine-tuning and parameter-efficient fine-tuning across three task types using one shared GPT-2 backbone. The baseline systems are strong on all three tasks: sentiment results match reference numbers closely, the paraphrase baseline reaches 0.860 dev accuracy, and the sonnet baseline becomes much stronger after decode tuning. The extension results are mixed in an informative way. LoRA is consistently more efficient in trainable parameters, but on SST, CFIMDB, and paraphrase detection, that efficiency comes with a clear quality loss. Sonnet generation is the exception. There, decode choice matters strongly, checkpoint selection should be based on dev chrF rather than loss, and LoRA gives the best development chrF among the matched runs.

The main limitation of this study is experimental scale. Each task uses a single backbone size and a limited set of LoRA hyperparameters, so the negative results on sentiment and paraphrase should not be read as universal limits of PEFT. Even so, the task contrast is useful. In this setting, LoRA looks most promising when the downstream problem is a small-data generation task rather than a classification task that benefits from broader full-model adaptation.

Prompt (first 3 lines)	D0 continuation snippet	D3 continuation snippet
<i>Love is my sin, and thy dear virtue hate, Hate of my sin, grounded on sinful loving. O, but with mine compare thou thine own state,</i>	<i>And thy love of thyself, Which thou hast loved in thy own heart, That thou shouldst love thyself more, Than thou shouldst love thyself more, That thou shouldst love thyself more, Than thou shouldst love thyself more, ...</i>	<i>In which thou art changed, yet thou canst see no reason why To thy own ignorance I should give thee such thoughts. Nor can I do it with thy pardon, But I will place thee before the eyes of all men, For my love, my self, is broken. And if I do love thee so that thou canst see My love's beauties, yet thou canst not believe Thy love's love's worth, for it alone is worth.</i>

Table 10: Recorded decode-sweep example for the prompt beginning “Love is my sin, and thy dear virtue hate.” D0 shows early hard repetition, while D3 is more varied but still imperfect.

Team contributions

This was a single-person project from Weiwei Wu.

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] 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. In *International Conference on Learning Representations*, 2022.
- [3] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
- [4] 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.
- [5] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.