

Fine-Tuning GPT-2 for Sentiment, Paraphrase, and Sonnet Tasks

Stanford CS224N Default Project

Walter Lopez Chavez

Department of Computer Science
Stanford University
walterch@stanford.edu

Abstract

This project investigates how a pretrained GPT-2 language model can be adapted to two downstream tasks: paraphrase detection and Shakespearean sonnet generation. For paraphrase detection, we compare full-model fine-tuning with parameter-efficient fine-tuning using LoRA. Our experiments show that LoRA achieves performance close to full fine-tuning (0.867 vs. 0.885 development accuracy) while updating far fewer parameters, although it remains slightly less expressive than full-model updates. For sonnet generation, we evaluate multiple decoding strategies and find that top- k sampling consistently produces the strongest results, outperforming top- p and reranking approaches in our experiments. However, reranking methods did not yield the improvements we initially expected, suggesting that simple sampling strategies may already align well with the model's learned probability distribution. Overall, our results demonstrate that efficient adaptation methods and careful decoding choices can significantly influence the performance of pretrained language models across diverse tasks.

1 Key Information

- Mentor: Wei Liu
- Late Days :

Team member	Late days remaining
Walter	0
Total pooled	0
Team size	1
Late days applied for the report	1

2 Introduction

Large pretrained language models such as GPT-2 have demonstrated strong performance across a wide range of natural language processing tasks through fine-tuning. In this project, we explore how GPT-2 can be adapted to two downstream tasks: paraphrase detection and Shakespearean sonnet generation. We investigate the use of parameter-efficient fine-tuning through LoRA and compare it with full-model fine-tuning for paraphrase detection, while also evaluating several decoding strategies for text generation. Our results show that LoRA can achieve performance close to full fine-tuning with far fewer trainable parameters, and that simple top- k sampling provides the most effective decoding strategy for sonnet generation in our experiments.

3 Related Work

Large language models have become the foundation for many modern NLP systems due to their ability to learn general language representations from large corpora and then adapt to downstream tasks through fine-tuning. In this project we build directly on GPT-2 [1], which demonstrated that a generative language model trained on large amounts of unlabeled text can perform a wide variety of language tasks after fine-tuning. GPT-2 provides both a well-studied architecture and publicly available pretrained weights, making it a practical backbone for experimentation.

One recent direction in adapting large language models is parameter-efficient fine-tuning (PEFT), which aims to achieve competitive performance while updating far fewer parameters than full model fine-tuning. In this work we explore Low-Rank Adaptation (LoRA) [2], a PEFT method that freezes the original pretrained model weights and instead learns small low-rank update matrices that modify the model’s behavior during fine-tuning. This approach significantly reduces the number of trainable parameters while still allowing the model to adapt to new tasks. LoRA was chosen as a primary extension in this project because it provides a practical way to study the trade-off between training efficiency and model performance. In addition to model adaptation, we also study decoding strategies for text generation. While the baseline implementation already included top- p sampling, we extend the system by implementing top- k sampling as well as reranking strategies for both top- k and top- p generation.

4 Approach

Our project builds on a pretrained GPT-2 language model whose weights are loaded from Hugging-Face and used as the backbone architecture for all downstream tasks. GPT-2 is a transformer-based autoregressive language model that produces contextual representations for each token in an input sequence. Given an input sequence of tokens $x_1, x_2, \dots, x_T$, the model computes hidden representations $h_1, h_2, \dots, h_T$ using stacked transformer layers, where each token attends to previous tokens through masked self-attention. During generation tasks, GPT-2 predicts the probability distribution of the next token according to

$$P(x_t \mid x_1, x_2, \dots, x_{t-1}) = \text{softmax}(Wh_{t-1})$$

where h_{t-1} is the hidden representation of the last token and W projects the hidden state to the vocabulary space. These pretrained representations allow the model to be adapted to new tasks through fine-tuning.

4.1 Paraphrase Detection

For paraphrase detection, we fine-tune GPT-2 to determine whether two sentences convey the same meaning. The two input sentences are combined into a single prompt and passed through the GPT-2 encoder to produce contextual token representations. The final token representation h_T is then used as a summary of the input pair and fed into a linear classifier to predict whether the pair is a paraphrase. Formally, the model computes

$$y = \text{softmax}(Wh_T + b)$$

where h_T is the final hidden representation and y represents the probability of the pair being a paraphrase.

To improve training efficiency, we also implement Low-Rank Adaptation (LoRA) [2]. Instead of updating the full GPT-2 weight matrices during fine-tuning, LoRA freezes the pretrained weights and introduces small trainable low-rank update matrices into specific attention layers. If the original projection matrix is W , LoRA modifies it as

$$W' = W + BA$$

where $A \in \mathbb{R}^{r \times d}$ and $B \in \mathbb{R}^{d \times r}$ are low-rank matrices with rank $r \ll d$. In our implementation these matrices are inserted into the attention query and value projections of the transformer layers.

This allows the model to adapt to the paraphrase task while training far fewer parameters than full fine-tuning.

4.2 Sonnet Generation

For the generation task, GPT-2 is fine-tuned on a dataset of Shakespearean sonnets. During training, the model learns using next-token prediction, where the objective is to maximize the likelihood of the next word given the previous context:

$$\mathcal{L} = - \sum_{t=1}^T \log P(x_t \mid x_1, x_2, \dots, x_{t-1})$$

At inference time, the model is conditioned on the first three lines of a sonnet and then generates the remaining lines autoregressively. We evaluate several decoding strategies that determine how tokens are sampled from the predicted probability distribution. In top- p (nucleus) sampling, the model selects the smallest set of tokens whose cumulative probability exceeds a threshold p , and samples from that set:

$$\sum_{x_i \in V_p} P(x_i) \geq p$$

In contrast, top- k sampling restricts generation to the k most probable tokens at each step:

$$V_k = \text{TopK}(P(x))$$

We additionally implement a reranking strategy that generates multiple candidate sequences using these sampling methods and then scores them using a heuristic combining model likelihood and structural properties of the poem. Each candidate sequence s receives a score

$$\text{Score}(s) = \frac{1}{T} \sum_{t=1}^T \log P(x_t) + \lambda \cdot S(s)$$

where the first term measures average token likelihood and $S(s)$ is a structural bonus encouraging outputs closer to the expected 14-line sonnet structure. The candidate with the highest score is selected as the final generated poem.

5 Experiments

5.1 Data

Quora Paraphrase Dataset. For the paraphrase detection task we use a subset of the Quora Question Pairs dataset. This dataset contains pairs of user-submitted questions along with binary labels indicating whether the two questions express the same meaning. The provided subset contains 141,506 training examples, 20,215 development examples, and 40,431 test examples. Each example consists of two questions and a label indicating whether the pair is a paraphrase, making it a standard benchmark for semantic similarity and paraphrase detection.

Shakespeare Sonnet Dataset. For the generation task we use a dataset of Shakespearean sonnets. Each sonnet contains 14 lines following the traditional rhyme structure ABAB CDCD EFEF GG. The dataset includes 143 training sonnets and 12 held-out sonnets for evaluation. During evaluation the model is given the first three lines of each sonnet and must generate the remaining lines, allowing us to measure how well the model captures the structure and language style of Shakespeare.

Both datasets were left completely untouched and used as is for the project.

5.2 Evaluation Method

5.2.1 Paraphrase Detection

For the paraphrase detection task we evaluate performance using classification accuracy. Since the problem is a binary decision task (predicting whether two sentences are paraphrases or not), accuracy provides a straightforward measure of how often the model predicts the correct label. This metric allows us to directly compare the performance of different training approaches, including full fine-tuning and LoRA-based parameter-efficient fine-tuning. Final results from our best-performing models were also compared to the baseline scores provided by the course staff.

5.2.2 Sonnet Generation

For the sonnet generation task we evaluate outputs using the chrF (character n-gram F-score) metric. chrF measures similarity between generated text and reference text based on overlapping character n-grams, making it more appropriate than exact word matching for evaluating generated poetry where multiple valid word choices may exist. In our experiments we compare each generated sonnet to the corresponding original sonnet that was held out from training. Development experiments were evaluated using the sacrebleu implementation of chrF, and the final decoding strategies were compared using the leaderboard scores from the course autograder alongside the course staff baselines.

5.3 Experimental Details

5.3.1 Paraphrase Detection

All paraphrase detection experiments were conducted using the pretrained gpt2 model as the backbone architecture. We explored both full fine-tuning and parameter-efficient fine-tuning using LoRA adapters.

Learning Rate Sweep. To understand the effect of optimization on LoRA performance, we conducted a sweep over learning rates of $1e-5$, $1e-4$, $2e-4$, and $1e-3$, training each configuration for five epochs. These runs were compared against a full-model fine-tuning baseline trained for the same number of epochs. This experiment allowed us to examine whether the performance gap between LoRA and full fine-tuning was due to optimization choices rather than model capacity.

LoRA Rank Sweep. After identifying a strong learning rate ($2e-4$), we performed an additional sweep over LoRA rank values to evaluate how the size of the low-rank updates affected model performance. We tested ranks $r = 8$, $r = 16$, and $r = 32$, again training each configuration for five epochs. This experiment investigates whether increasing the number of trainable parameters in the LoRA adapters provides measurable performance improvements.

5.3.2 Sonnet Generation

For sonnet generation we fine-tuned the gpt2 model on the Shakespeare sonnet dataset using next-token prediction. During inference we evaluated several decoding strategies while keeping the underlying fine-tuned model fixed in order to isolate the effect of the decoding algorithm.

We compared four generation strategies: top- p sampling, top- k sampling, reranked top- k , and reranked top- p . For top- p sampling we used a probability threshold of $p = 0.9$, while top- k sampling restricted the candidate pool to the $k = 50$ most likely tokens at each step. For the reranking methods we generated 10 candidate sequences and selected the highest-scoring output according to the scoring heuristic described in Section 3.

5.4 Results

5.4.1 Paraphrase Detection

As shown in Figure 1, LoRA-based parameter-efficient fine-tuning achieves performance close to full-model fine-tuning despite updating far fewer parameters. Increasing the learning rate significantly improves LoRA performance, allowing it to approach the full fine-tuning baseline. We also examine

the effect of LoRA rank and find that increasing the rank from $r = 8$ to $r = 32$ produces little change in final accuracy. This suggests that relatively small low-rank updates are sufficient to adapt the pretrained GPT-2 representations to the paraphrase detection task.

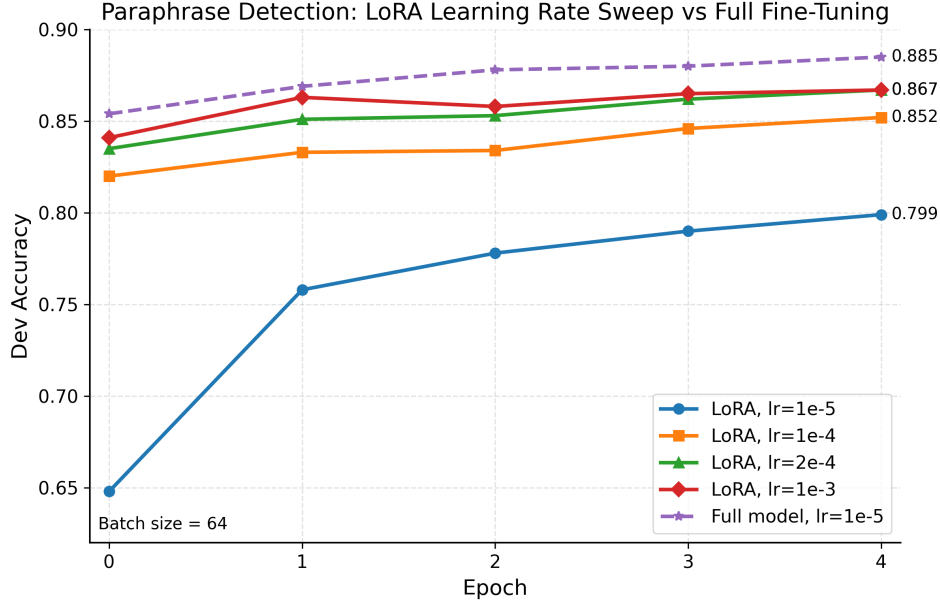


Figure 1: Learning rate sweep comparing LoRA fine-tuning with full model fine-tuning.

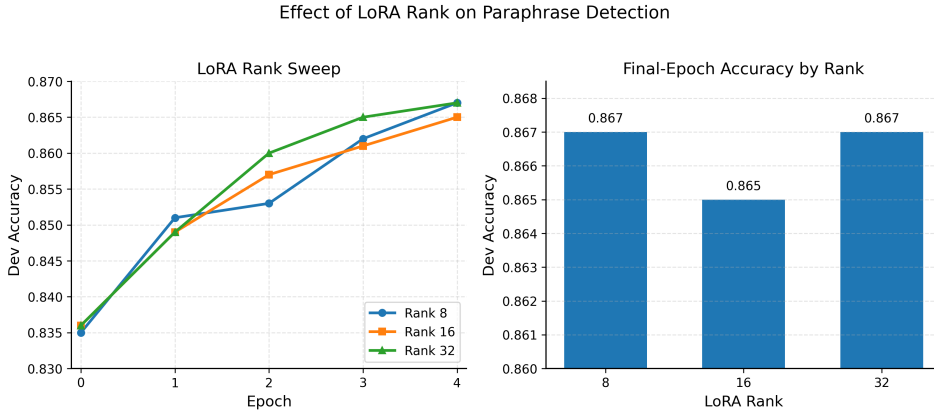


Figure 2: Effect of LoRA rank on paraphrase detection performance.

5.4.2 Sonnet Generation

We compare several decoding strategies while keeping the underlying fine-tuned GPT-2 model fixed. The evaluated methods include top- p sampling, top- k sampling, and reranked variants of both approaches. As shown in Table 1, top- k sampling produced the strongest performance among the tested strategies, suggesting that restricting generation to a fixed set of high-probability tokens provided a good balance between diversity and coherence for this task.

5.4.3 Leaderboard Comparison

We also compare our final models against the course staff baselines using the official leaderboard evaluation. As shown in Table 2, our paraphrase model slightly outperforms the baseline on the

Decoding Method	chrF2 Score
Top- k Sampling	12.86
Reranked Top- k	12.75
Top- p Sampling	12.61
Reranked Top- p	12.20

Table 1: chrF2 scores on dev set

development set while remaining comparable on the test set. For sonnet generation, our model achieves higher chrF scores than the staff baseline on both development and test leaderboards.

Task	Our Score	Staff Baseline
Paraphrase Dev Accuracy	0.885	0.882
Paraphrase Test Accuracy	0.850	0.856
Sonnet Dev chrF	41.790	41.414
Sonnet Test chrF	41.589	41.208

Table 2: Best models vs. Course staff baselines.

6 Analysis

The results from the paraphrase detection experiments highlight the trade-off between full fine-tuning and parameter-efficient methods such as LoRA. While full fine-tuning achieved the highest development accuracy (0.885), the best LoRA configuration achieved a comparable score of 0.867 while updating far fewer parameters. This difference is expected because full fine-tuning allows the model to adjust the entire parameter space, making it more expressive. In contrast, LoRA restricts updates to low-rank matrices inserted into specific attention layers, which limits the model’s ability to fully adapt but significantly reduces training cost.

The learning rate sweep suggests that LoRA benefits from larger learning rates compared to full fine-tuning. Because LoRA updates only a small set of adapter parameters while the base model weights remain frozen, larger learning rates allow these parameters to adapt more quickly without destabilizing the pretrained model. The LoRA rank sweep further shows that increasing the rank from 8 to 32 provides almost no improvement in performance. This suggests that the paraphrase detection task requires only small adjustments to the pretrained GPT-2 representations, and that low-rank updates already capture most of the necessary information. Increasing the rank therefore adds additional parameters without significantly improving model capacity.

For the sonnet generation task, top- k sampling produced the strongest results among the decoding strategies tested. One possible explanation is that top- k provides a stable balance between diversity and coherence by restricting generation to a fixed set of high-probability tokens at each step. In contrast, top- p sampling allows the candidate set to vary dynamically, which can sometimes include lower-probability tokens that introduce additional noise into the generation process. Although reranking was intended to improve generation by selecting the best candidate sequence according to a scoring heuristic, it did not outperform the base top- k strategy. This likely occurred because the GPT-2 probability distribution already serves as a strong signal for selecting coherent tokens, and the additional heuristics used in reranking did not align closely enough with the chrF evaluation metric to consistently improve the final outputs.

7 Conclusion

In this project we explored how GPT-2 can be adapted to multiple downstream tasks through both traditional fine-tuning and more efficient adaptation techniques. We implemented and evaluated a paraphrase detection system using both full-model fine-tuning and LoRA-based parameter-efficient fine-tuning, showing that LoRA can achieve performance close to full fine-tuning while updating far fewer parameters. We also investigated sonnet generation using a fine-tuned GPT-2 model and compared several decoding strategies, finding that simple top- k sampling produced the most consistent results among the methods tested. Overall, these experiments demonstrate that pretrained language

models like GPT-2 remain highly flexible tools for a variety of NLP tasks, and that efficient adaptation methods such as LoRA can provide a practical balance between performance and computational cost.

References

- [1] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- [2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021.