

GPT-2 Default Project with Attention-Only LoRA for Paraphrase Detection

Stanford CS224N Default Project

Yiqing Liu

Graduate School of Education
Stanford University
yiqingl@stanford.edu

Abstract

We complete and verify the CS224N default GPT-2 project pipeline, then focus our main extension on parameter-efficient adaptation for paraphrase detection. We compare standard full fine-tuning against an in-stack Low-Rank Adaptation (LoRA) variant that inserts low-rank updates only into attention projections while freezing the base GPT-2 weights. On Quora paraphrase detection, full fine-tuning reaches 0.8675 dev accuracy and 0.850 test accuracy on the course leaderboard. A naive attention-only LoRA setting underperforms at 0.7887 dev accuracy, while a better-tuned setting improves to 0.8438 dev accuracy, recovering most of the baseline while training far fewer parameters. To understand where the remaining gap comes from, we perform a lightweight diagnostic analysis motivated by psychometric ideas about heterogeneous item difficulty; the remaining gap is most visible in lower-margin regions, where each model is less confident. For sonnet generation, we report a working baseline with local dev chrF 19.08 and leaderboard chrF scores of 34.405 on dev and 34.798 on test. Overall, our results suggest that attention-only LoRA is a strong but not lossless substitute for full fine-tuning in this GPT-2 paraphrase setting, especially in more ambiguous, lower-confidence regions.

1 Key Information

- Mentor: Simon Kim
- External Collaborators (if you have any): None
- Sharing project: None
- Late Days (if you are using any): None

2 Introduction

The CS224N default final project asks us to do more than call an existing library model: it requires implementing core GPT-2 components, validating them against reference behavior, and then studying how the resulting model adapts to downstream tasks. This setup is pedagogically valuable because it connects architecture-level understanding to empirical model behavior. At the same time, it creates a practical research question: once the GPT-2 stack is working, how much task performance actually requires updating the full model, and how much can be recovered with cheaper parameter-efficient adaptation?

We study this question in the default-project setting, with the main emphasis on Quora paraphrase detection. Full fine-tuning is a strong baseline, but it updates all GPT-2 parameters and therefore represents the most expensive adaptation regime. Parameter-efficient fine-tuning methods aim to capture most of the task-specific gain while training only a small number of added parameters.

Among these methods, LoRA is especially appealing because it can be implemented as a simple low-rank update to existing linear projections rather than a full architectural rewrite [Hu et al., 2021].

Our project has two linked goals. First, we complete and verify the required default-project implementation: GPT-2 attention, transformer-layer wiring, embeddings, AdamW, sentiment classification, paraphrase detection, and sonnet generation. Second, we develop a controlled extension for paraphrase detection by inserting attention-only LoRA modules directly into the provided GPT-2 stack, keeping the tokenizer, prompt format, data splits, and evaluation procedure fixed. This allows an apples-to-apples comparison between full fine-tuning and parameter-efficient adaptation.

We also wanted the analysis to go beyond a single leaderboard number. Inspired by psychometric perspectives on item-level heterogeneity, our original proposal considered fuller item-response-style analyses. Recent work on psychometric perspectives for LLM evaluation motivated our interest in going beyond top-line accuracy and asking where different adaptation strategies succeed or fail across examples [Zhou et al., 2025]. For the final report, we keep the analysis lightweight and robust: instead of fitting a formal IRT model, we use prediction margin as a practical proxy for ambiguity and examine which kinds of examples separate full fine-tuning from LoRA. This yields a clear empirical story without introducing fragile auxiliary modeling assumptions.

Our main findings are straightforward. Full fine-tuning remains best on paraphrase detection, but a tuned attention-only LoRA configuration recovers most of that performance. The remaining gap is not uniform: it is concentrated in examples where the model itself is uncertain. This suggests that parameter-efficient updates may be sufficient for easier, higher-margin cases, while full adaptation better handles ambiguous decisions. We therefore center the extension analysis on paraphrase detection, which provided the clearest setting for comparing full fine-tuning and parameter-efficient adaptation.

3 Related work

Our work sits at the intersection of three lines of literature: decoder-only transformer language models, parameter-efficient fine-tuning, and task-specific evaluation for generation and classification.

GPT-style decoder-only transformers established the now-standard pretrain-then-adapt paradigm for NLP. GPT-2 in particular showed that a unidirectional transformer trained on large-scale next-token prediction can support a wide range of downstream tasks after fine-tuning [Radford et al., 2019]. Architecturally, our implementation follows the standard transformer design with masked self-attention and position-wise feed-forward layers [Vaswani et al., 2017]. Because the default project provides a minimal GPT-2 codebase rather than a turn-key framework, part of our contribution is implementing and validating these components inside the course scaffold.

Our extension is most directly motivated by LoRA, which adapts pretrained models by learning low-rank updates to frozen linear layers [Hu et al., 2021]. The appeal of LoRA is that it reduces the number of trainable parameters and the optimizer state footprint while often preserving much of the benefit of full fine-tuning. The default-project handout explicitly points to PEFT methods such as LoRA as a promising extension direction, so our work can be viewed as a focused replication and stress test of that idea in the course GPT-2 stack rather than in a large external framework [Radford et al., 2019, Hu et al., 2021]. Our implementation is deliberately narrow: we apply LoRA only to attention projections and keep all other aspects of the pipeline fixed.

Paraphrase detection has long been a standard sentence-pair classification setting, and the Quora Question Pairs (QQP) task is a widely used benchmark for question equivalence. In our project, the paraphrase component follows this general QQP-style formulation, where the goal is to predict whether two questions express the same underlying intent [Wang et al., 2018, Quora, 2017].

For the task setting, we follow the default project formulation of paraphrase detection as a cloze-style yes/no decision on Quora question pairs and sonnet generation as conditional continuation evaluated with chrF. chrF is a standard character-level overlap metric that is often more forgiving than word-level metrics for generation with stylistic variation [Popović, 2015]. We treat leaderboard scores as useful external reference points, but we do not rely on them alone.

Finally, our analysis is informed by psychometric thinking, though not by a formal psychometric model in the final version. The guiding idea is that aggregate accuracy can hide meaningful hetero-

generality across examples. We therefore stratify performance by prediction margin, using the model’s own confidence separation as a proxy for item ambiguity. This is closer in spirit to diagnostic item analysis than to fitting a full item response model, but it preserves the proposal’s original motivation: understand *where* model differences arise, not just whether one average score is larger than another.

4 Approach

4.1 Required default-project implementation

We first completed the course-provided GPT-2 scaffold. Concretely, this included: (1) masked multi-head self-attention, (2) GPT-2 layer residual and feed-forward wiring, (3) token and positional embedding lookup, (4) the AdamW-style optimizer step used in the starter code, and (5) the sentiment-classification head based on the last-token representation. We validated the model and optimizer with the provided sanity checks before running downstream experiments. This implementation work mattered for the report because every extension result depends on the correctness of this base stack.

4.2 Baseline task pipelines

For sentiment analysis, we used the provided setup in which GPT-2 encodes the input and a classification head maps the final non-padding token representation to label logits. For paraphrase detection, we formatted each question pair with a natural-language prompt and fed it through GPT-2. We then took the final non-padding token representation and passed it to a lightweight two-way classification head predicting paraphrase versus non-paraphrase. In the baseline setting, all GPT-2 parameters were fine-tuned jointly with the classifier. In the LoRA setting, the base GPT-2 weights were frozen and low-rank adapters were injected into the attention modules, while training only the adapters and the classification head. For sonnet generation, we fine-tuned GPT-2 on the provided sonnet corpus and generated continuations conditioned on the first three lines.

4.3 Attention-only LoRA extension

Our main extension inserts LoRA modules into GPT-2 attention projections while freezing the base GPT-2 weights. For a linear map with pretrained weight matrix W , LoRA keeps W frozen and parameterizes the trainable update as a low-rank residual:

$$Wx \mapsto Wx + \frac{\alpha}{r}B(Ax),$$

where $A \in \mathbb{R}^{r \times d}$, $B \in \mathbb{R}^{d' \times r}$, r is the LoRA rank, and α is a scaling factor. We apply this update to the query, key, value, and attention-output projections in each transformer block. During LoRA training, the pretrained GPT-2 parameters remain frozen, and only the LoRA matrices together with the task head are updated.

This design choice was intentional. By limiting LoRA to attention projections, we created a constrained parameter-efficient variant rather than a nearly-full reimplementation of fine-tuning. That makes the comparison cleaner but also more demanding: any performance gap can plausibly be attributed to the reduced adaptation capacity rather than to changes elsewhere in the training stack.

4.4 Diagnostic analysis by prediction margin

Our original proposal considered more formal IRT-style diagnostics, but in the final report we use a simpler and more defensible measure. For each paraphrase example, we compute the margin between the top-1 and top-2 class probabilities. Small margins indicate uncertain decisions, while large margins indicate confident ones. We bucket dev-set examples by margin and compare accuracy across methods within each bucket. This is not a formal item response model; rather, it is a practical diagnostic for locating where one training regime gains over another.

5 Experiments

5.1 Data

We follow the datasets provided in the CS224N default final project handout. The sentiment component uses SST and CFIMDB. The main extension task uses the Quora paraphrase dataset, with the course-provided train, dev, and test splits. The generation component uses the Shakespeare sonnet dataset, where the model is conditioned on the first three lines and must generate the remainder. We do not use external training data.

5.2 Evaluation methods

For paraphrase detection, the main metric is accuracy, matching the course leaderboard. We report dev accuracy for controlled model comparisons and our selected test leaderboard score as the external evaluation result required by the default-project instructions. For sonnet generation, we report chrF both from our local evaluation pipeline and from the course leaderboards. We treat sentiment analysis primarily as a verification task for the required pipeline rather than as the focus of the extension study.

Beyond topline metrics, we include two qualitative forms of evaluation. First, for paraphrase detection we examine accuracy across prediction-margin buckets. Second, for sonnet generation we inspect representative outputs to identify strengths and failure modes not captured by chrF alone.

5.3 Experimental details

All experiments use the course GPT-2 stack with the same tokenizer and task formatting conventions. Our baseline paraphrase model is full fine-tuning of GPT-2 plus the task head. For LoRA, we freeze the GPT-2 backbone and train only the LoRA parameters and the classification head.

We evaluate two attention-only LoRA settings, both with rank $r = 8$ and scaling $\alpha = 16$:

- **LoRA v1:** dropout = 0.05, learning rate = 10^{-5} .
- **LoRA v2:** dropout = 0.00, learning rate = 10^{-4} .

The contrast between these two settings lets us test whether the poor initial LoRA result reflected the method itself or an overly conservative configuration.

For sonnet generation, we fine-tune GPT-2 on the sonnet corpus and generate continuations for held-out prompts. Because the sonnet dataset is small and the project timeline limited, we treat this as a functioning baseline rather than a fully optimized generation study.

5.4 Results

5.4.1 Sentiment classification results

As part of the default project requirements, we trained GPT-2 sentiment classifiers on SST and CFIMDB under both required fine-tuning settings: updating only the last linear layer and full-model fine-tuning. On SST, the last-linear-layer model achieved a best development accuracy of 0.454, while full-model fine-tuning improved this to 0.497. On CFIMDB, the corresponding best development accuracies were 0.873 and 0.980. These results confirm that the required sentiment pipeline ran successfully under both training settings. Full-model fine-tuning outperformed last-linear-layer training on both datasets, with a particularly large gain on CFIMDB.

Task	Fine-tuning mode	Best dev accuracy
SST	Last linear layer	0.454
SST	Full model	0.497
CFIMDB	Last linear layer	0.873
CFIMDB	Full model	0.980

Table 1: Sentiment classification results for the required default-project settings.

5.4.2 Paraphrase detection: full fine-tuning vs. LoRA

Table 2 shows the main paraphrase results. Full fine-tuning reaches 0.8675 dev accuracy. LoRA v1 performs much worse at 0.7887, indicating that a naive attention-only configuration can under-adapt. However, LoRA v2 improves sharply to 0.8438 dev accuracy, recovering most of the baseline. The remaining gap between full fine-tuning and tuned LoRA is 2.37 accuracy points.

Our selected paraphrase test leaderboard submission achieves 0.850 accuracy. Because the course leaderboard permits only a limited number of test submissions, we use this score mainly as an external reference point for our finalized baseline rather than as a primary hyperparameter-selection signal.

Method	Trainable components	Configuration	Dev accuracy
Full fine-tuning	GPT-2 + head	–	0.8675
LoRA v1	LoRA + head (frozen GPT-2)	$r = 8, \alpha = 16, dropout = 0.05, lr 10^{-5}$	0.7887
LoRA v2	LoRA + head (frozen GPT-2)	$r = 8, \alpha = 16, dropout = 0.00, lr 10^{-4}$	0.8438

Table 2: Paraphrase dev accuracy for the full fine-tuning baseline and two attention-only LoRA variants. All runs use the same tokenizer, prompt format, loss, and evaluation pipeline.

5.4.3 Sonnet generation baseline

For sonnet generation, our local held-out dev evaluation gives chrF = 19.08. On the course leaderboards, our sonnet model achieves chrF = 34.405 on dev and 34.798 on test. We report these as baseline results rather than as evidence of exhaustive tuning. In creative generation, overlap metrics can be informative but incomplete, so we supplement them with qualitative inspection below. Because the local sonnet evaluation uses our held-out prompts rather than the official leaderboard split, we treat the local chrF and leaderboard chrF as complementary indicators rather than directly comparable scores.

6 Analysis

Although the sentiment experiments confirm that the required classification pipeline worked under both training settings, our main extension question concerns full fine-tuning versus attention-only LoRA on paraphrase detection. We therefore focus the detailed analysis on paraphrase detection, where the comparison is most central and interpretable.

6.1 Where does full fine-tuning beat LoRA?

Beyond average accuracy, we want to know where full fine-tuning gains over LoRA. To obtain a lightweight difficulty-oriented diagnostic, we bucketed paraphrase development examples by each model’s prediction-margin quantiles. Here, margin is defined as the difference between the top two predicted class probabilities, so lower-margin examples are more ambiguous under that model and higher-margin examples are more confidently classified. Because the buckets are constructed from quantiles, they contain approximately equal numbers of examples, although ties can make the counts not perfectly identical. Importantly, these buckets are defined separately for each model using its own margin distribution, so the curves compare model-specific confidence profiles rather than accuracy on one shared partition of examples.

Several patterns are visible in Figure 1. First, accuracy rises monotonically with margin for all three methods, supporting the use of prediction margin as a simple ambiguity indicator. Second, full fine-tuning stays above both LoRA variants across most of the bucket range, suggesting that its average advantage is not driven by only one narrow region. Third, the gaps narrow in the highest-margin buckets, where all methods approach near-ceiling accuracy. This suggests that the main trade-off is not on easy examples, but in lower-confidence regions where each model is less certain.

6.2 Why did the naive LoRA setting fail?

The simplest explanation is that the first configuration was too conservative. Because GPT-2 is frozen in LoRA mode, the optimization burden shifts to a much smaller parameter set. A low learning

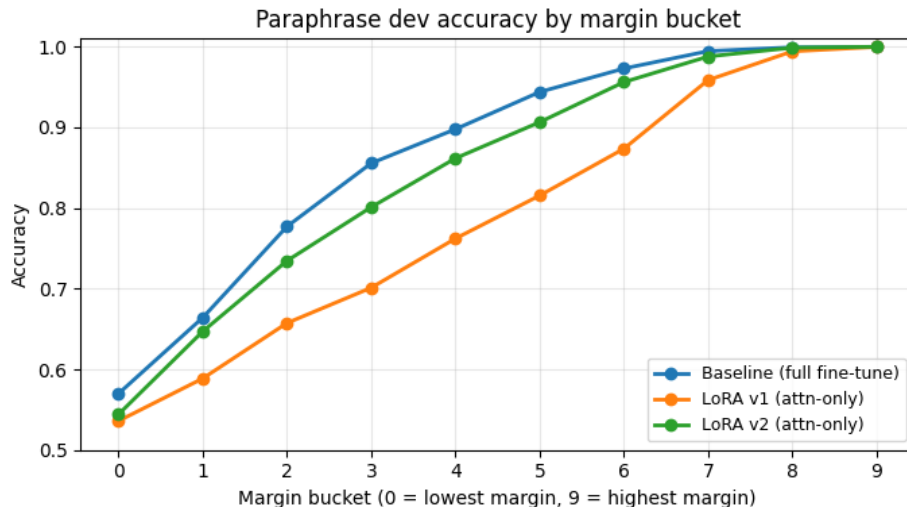


Figure 1: Paraphrase development accuracy by model-specific prediction-margin bucket. For each model, examples are partitioned into approximately equal-count quantile buckets using that model’s own prediction-margin distribution, where margin is defined as the difference between the top two predicted class probabilities. Accuracy increases monotonically with margin for all methods, while full fine-tuning remains stronger across most of the range.

rate together with nonzero LoRA dropout appears to have limited how quickly the adapters could specialize to the paraphrase task. Once we increased the learning rate and removed dropout, dev accuracy improved substantially. This result is useful in its own right: it shows that when parameter-efficient methods underperform, the gap may reflect training configuration choices rather than the underlying idea alone.

6.3 Qualitative inspection of sonnet outputs

The sonnet pipeline produces recognizable Shakespeare-like surface patterns but with uneven quality. Stronger generations maintain an archaic register and local syntactic coherence for several lines. Weaker generations drift semantically, break poetic flow, or introduce clearly spurious tokens and metadata-like fragments. This failure pattern is unsurprising given the tiny dataset and the difficulty of learning sustained poetic structure from limited supervision. It also reinforces why chrF should be interpreted cautiously: a model can obtain a reasonable overlap score while still generating awkward or visibly broken text.

7 Conclusion

We completed the CS224N default GPT-2 project pipeline and used it to study a focused extension: attention-only LoRA for paraphrase detection. The implementation portion includes the core GPT-2 and optimizer components required by the assignment, along with working sentiment, paraphrase, and sonnet pipelines. On Quora paraphrase detection, full fine-tuning remains strongest at 0.8675 dev accuracy and 0.850 test leaderboard accuracy. However, a tuned attention-only LoRA variant reaches 0.8438 dev accuracy, showing that substantial task performance can be recovered while updating far fewer parameters.

The most informative result is not just the average gap, but its location. Our margin-based diagnostic indicates that tuned LoRA and full fine-tuning are more similar in their respective high-margin regions, while the performance gap is larger in lower-margin regions where each model is less certain. This suggests a practical takeaway: in a constrained GPT-2 setting, parameter-efficient adaptation can be an attractive tradeoff, but full-model adaptation still matters when decisions are hardest. A main limitation of our report is that the extension study is much deeper for paraphrase detection than for sonnet generation or sentiment classification. A natural next step would be to compare broader PEFT

variants or layer-selection strategies, and to revisit more formal psychometric analyses once the core empirical story is firmly established.

Team contributions

Single-person project.

References

- 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*, 2021.
- Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.
- Quora. First quora dataset release: Question pairs, 2017. URL <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Openai technical report, OpenAI, 2019.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.
- Hongli Zhou, Hui Huang, Ziqing Zhao, Lvyuan Han, Huicheng Wang, Kehai Chen, Muyun Yang, Wei Bao, Jian Dong, Bing Xu, Conghui Zhu, Hailong Cao, and Tiejun Zhao. Lost in benchmarks? rethinking large language model benchmarking with item response theory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

A Representative sonnet outputs

We include brief excerpts to illustrate the kinds of behavior discussed in the main text.

A.1 Stronger example (excerpt)

Prompt (first 3 lines):

Those lips that Love’s own hand did make
Breathed forth the sound that said “I hate”
To me that languished for her sake;

Generated continuation (excerpt):

That though to love found it always,
Though at her breast love grow too forsaken too
Relating recently fond pleas on my lips someday
Till Thu i passest Moon’s hand somehow wetting it do.

A.2 Failure case (excerpt)

Prompt (first 3 lines):

Canst thou, O cruel, say I love thee not
When I against myself with thee partake?
Do I not think on thee when I forgot

Generated continuation (excerpt):

A space of days, to tear er long and countless mad lengths I slain
At peak-doors charged in travail, counteramped in sudden wonky bleeds
|| 481.068 18 CHRISTIAN VICTIMAL.