

Task-Specific Fine-Tuning Strategies for Improving GPT-2 Across Classification and Generative NLP Tasks

Stanford CS224N Default Project

Cheney Sang cheney@stanford.edu

Harris Alan Lee harrisli@stanford.edu

Austin Chen austin05@stanford.edu

Abstract

We study two GPT-2-based NLP tasks with different objectives: Quora paraphrase detection and sonnet generation. For paraphrase detection, we compare prompt/scoring ablations, LoRA, model scaling, and inference-time aggregation, and find that GPT-2 medium with bidirectional checkpoint ensembling gives the strongest retained dev performance. For sonnet generation, we evaluate reranking and structure-aware prompting, showing that better candidate selection matters more than simply generating more outputs. Overall, our results suggest that adaptation and inference strategies should be matched to the structure of the downstream task rather than treated as one-size-fits-all improvements.

1 Key Information

- Mentor: Shicheng Liu
- Late Days:

Team member	Late days remaining
Cheney Sang	2
Harris Alan Lee	0
Austin Chen	6
Total pooled	8
Team size	3
Late days applied for the report	6

2 Introduction

Large pretrained language models can support both discriminative and generative NLP tasks, but the most effective adaptation strategy depends on the task objective. In this project, we study two GPT-2-based settings: Quora paraphrase detection, framed as binary semantic-equivalence classification, and sonnet generation, framed as constrained poetic continuation under the chrF metric. We ask how far task-specific fine-tuning and inference choices can push GPT-2 beyond the default baseline in these settings. For paraphrase detection, we compare prompt and scoring ablations, LoRA-based fine-tuning, model scaling, and inference-time aggregation, finding that GPT-2 medium with bidirectional checkpoint ensembling gives the strongest retained dev performance. For sonnet generation, we examine reranking and structure-aware prompting, and find that larger candidate pools help only when paired with selection criteria that better reflect sonnet form. Overall, our results suggest that improving a general-purpose language model is less about one universally superior method than about matching adaptation and inference strategies to the structure of the downstream task.

3 Related Work

Our project builds on GPT-2, a decoder-only Transformer language model that demonstrated the strength of large-scale generative pretraining for downstream adaptation [1]. For paraphrase detection, our setting is based on Quora question pairs, a benchmark centered on semantic equivalence between naturally occurring user questions [2]. This task is closely related to broader work on sentence-pair modeling and natural language inference, where the central challenge is to distinguish semantic similarity from superficial lexical overlap.

Methodologically, our experiments are also connected to prior work on efficient adaptation and inference-time improvement. In particular, we explore LoRA, which injects low-rank trainable adapters into pretrained models and has become a standard parameter-efficient fine-tuning method for large language models [3]. We also examine ensembling, bidirectional scoring, and reranking, which follow the broader intuition that inference-time aggregation can improve robustness even when the base model

family is fixed. For sonnet generation, our structure-aware reranking approach is motivated by the fact that poetic generation is not judged only by local fluency, but also by higher-level formal constraints such as line count and continuation shape. Our work is therefore best understood as an empirical study of how classic adaptation ideas—fine-tuning, PEFT, ensembling, and reranking—behave differently across a classification task and a constrained generation task.

4 Task 1: Paraphrase Detection

4.1 Task and Baseline

We study Quora paraphrase detection in the default final-project setting, where a GPT-2 family model receives two questions in a yes/no prompt and predicts whether they are paraphrases. Our retained experiments used 283,003 training pairs and 40,429 development pairs, and we selected models primarily by development accuracy because that metric matches the official leaderboard objective. We also tracked macro-F1 whenever available to distinguish genuine ranking gains from purely threshold-driven effects.

Our starting point was a GPT-2 small full-model baseline trained for 10 epochs with learning rate 1×10^{-5} . That run reached about 0.893 dev accuracy, which turned out to be a strong reference rather than a trivial starter model. This mattered for the rest of the project: any extension had to beat a fully tuned small-model baseline, not just a weak off-the-shelf system. We therefore organized the write-up around a small number of experiments that actually changed the project decision boundary: prompt/scoring calibration on GPT-2 small, LoRA-based parameter-efficient tuning, scaling to GPT-2 medium, and final inference-time aggregation.

4.2 Method / Extensions

Our extensions were cumulative rather than independent. First, we ran controlled ablations on the GPT-2 small setup to determine whether prompt wording or the scoring rule mattered enough to justify keeping them in later experiments. Prompt-template changes produced only tiny differences: the best prompt reached 0.8908 dev accuracy, essentially tied with the strongest alternatives. By contrast, using a hybrid scorer that combined the language-model yes/no logits with the learned classification head was a small but consistent win over LM-only or head-only scoring, so we promoted hybrid scoring as the default recipe.

Second, we added post-hoc threshold tuning on the yes-minus-no margin. This did not change the model itself, but it improved calibration enough to raise dev accuracy from 0.8908 to 0.8929 on the best GPT-2 small checkpoint. We kept this result because it established an important pattern that reappeared later: calibration alone gives only modest gains, but it becomes much more useful once paired with stronger underlying checkpoints.

Third, we explored LoRA as a parameter-efficient alternative to full fine-tuning. Minimal LoRA placements on only a few attention projections were clearly weaker than the full-model baseline, but broader placement over q,k,v,o and MLP layers made LoRA competitive. The best retained LoRA run used rank 32 and reached 0.8911 dev accuracy while training only about 4.07% of parameters. This was a meaningful efficiency result, but it still did not create a new project frontier.

Finally, we changed both scale and inference strategy. Scaling from GPT-2 small to GPT-2 medium with full-model hybrid training created the first clear jump, reaching 0.9074 dev accuracy after a 10-epoch continuation. On top of that stronger checkpoint family, we tested inference-time aggregation. Bidirectional scoring helped, checkpoint ensembling helped more, and combining both produced our best retained result: a bidirectional two-checkpoint ensemble with threshold tuning that reached 0.9140 dev accuracy and 0.9083 macro-F1 on the dev set. We briefly explored larger-backbone LoRA and template-heavy ensembling, but both underperformed the medium-hybrid checkpoint-ensemble line and are omitted from the main table for space.

4.3 Experimental Setup

All retained runs used the same Quora train/dev split and the same cloze-style prompt format, with predictions derived from the margin between the yes and no scores. For the early controlled ablations, we fixed GPT-2 small, full-model fine-tuning, 5 epochs, batch size 32, and learning rate 1×10^{-5} so that only one factor changed at a time. This let us interpret the prompt and scoring results causally rather than as confounded differences between training recipes.

For calibration experiments, we tuned a scalar threshold on the dev margin to maximize macro-F1, then reported both raw and tuned metrics. For LoRA, we varied the target modules and rank while keeping the core paraphrase formulation unchanged. The strongest retained LoRA setting used q,k,v,o and MLP adapters with rank 32, alpha 64, dropout 0.05, batch size 32, and learning rate 5×10^{-5} . For model scaling, we switched to GPT-2 medium with full-model hybrid scoring and ran both a 5-epoch control and a 10-epoch continuation. We also evaluated a bf16 batch-size-16 variant as a systems optimization; it nearly matched the best fp32 checkpoint while cutting per-epoch runtime substantially, so it became our preferred fast backup rather than our primary submission model.

Our final inference experiments reused the strongest GPT-2 medium checkpoints and added no new training. Instead, we averaged scores across checkpoints and/or sentence orderings, then tuned the final threshold on the dev set. This design was

Table 1: Main retained paraphrase results on the dev set

Stage	Method	Dev acc	Takeaway
Baseline	GPT-2 small full fine-tuning	0.8930	Strong starter reference
Small-model ablations	Hybrid scoring + threshold tuning	0.8929	Calibration helps, but only incrementally
PEFT	LoRA on q,k,v,o,mlp, rank 32	0.8911	Efficient and competitive, but not better than full FT
Scaling	GPT-2 medium hybrid, 10 epochs	0.9074	First clear project-level jump
Final system	2 checkpoints + bidirectional + threshold tuning	0.9140	Best retained result

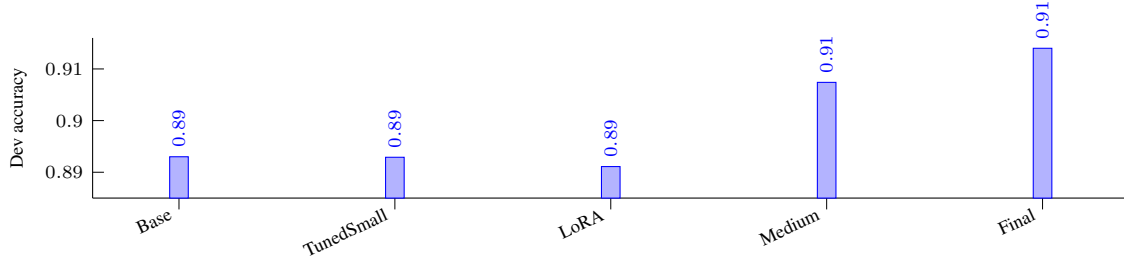


Figure 1: Project progression across the main retained paraphrase systems. The largest gains came late: first from scaling to GPT-2 medium, then from inference-time aggregation.

attractive under deadline pressure because it converted extra test-time compute into measurable gains without paying for another full training run.

4.4 Results

Table 1 summarizes the retained paraphrase results. The overall pattern is clear. Small-model prompt and scoring changes were useful mainly because they removed uncertainty, not because they produced large jumps. LoRA became competitive only after broader adapter placement, but still plateaued around the small full-model baseline. The first large gain came from moving to GPT-2 medium, and the final best result came from inference-time aggregation on top of the medium checkpoints.

In absolute terms, the strongest GPT-2 medium checkpoint improved by about +0.0145 accuracy over the tuned small-model recipe, which is much larger than any gain from prompt engineering, scoring-rule choice, or threshold tuning alone. The final inference-time ensemble then added another +0.0066 over that strong trained checkpoint. We therefore treat the GPT-2 medium family as the main modeling breakthrough and the final ensemble as the last-mile optimization that converted a strong model into the best submission candidate.

We do not report a full per-experiment test-leaderboard table here because the uploaded materials did not contain a trustworthy official leaderboard score for every intermediate stage. Instead, we use dev accuracy and macro-F1 consistently for model selection, and we map the final test submission to the same recipe that won on dev: the bidirectional two-checkpoint ensemble with threshold tuning.

4.5 Analysis

Three lessons mattered most. First, some intuitively plausible extensions were simply too weak to move the frontier. Prompt wording changed results only marginally, and minimal LoRA placements were clearly below the best full-model line. These experiments were still useful because they narrowed the search space and let us stop investing in directions that were unlikely to win under a tight deadline.

Second, stronger ranking signals mattered more than calibration tricks. Threshold tuning was consistently useful, but only by a few tenths of a point. When the underlying checkpoints were weak or poorly centered, as in the template-heavy inference family, calibration could not fully rescue them. By contrast, when we started from strong medium-family checkpoints, the same threshold-tuning step became a clean final boost rather than a crutch.

Third, the biggest gains came from matching compute to the right stage of the project. During training, the key move was scaling to GPT-2 medium while keeping the successful hybrid scoring recipe. After that, the marginal value of more training looked small, so extra compute was better spent at inference time through checkpoint ensembling and bidirectional scoring. This combination reduced checkpoint-specific boundary noise and sentence-order sensitivity at the same time, which explains why the two methods were additive.

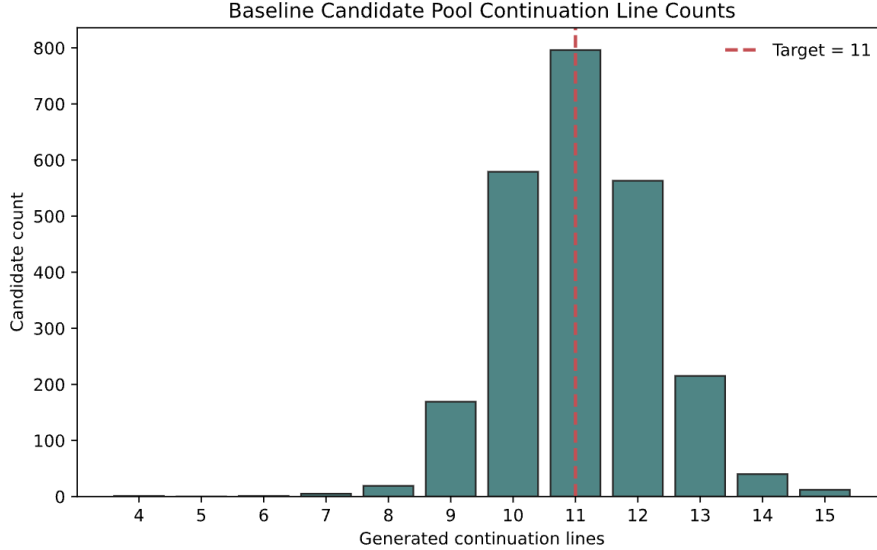


Figure 2: Distribution of continuation line counts for the LoRA baseline. Although the target continuation length is 11 lines, the baseline produces a much wider distribution.

5 Task 2: Sonnet Generation

5.1 Task and Baseline

In the sonnet generation task, the goal is to generate a continuation that matches the target sonnet as closely as possible under the course evaluation metric, chrF. Unlike paraphrase detection, this task is generative rather than discriminative, so performance depends not only on the trained model but also on the inference-time procedure used to sample and select candidate outputs.

Our starting point was the default GPT-2-XL model. We also tested a LoRA-based variant that updated only the query and value projections, using rank $r = 8$ and $\alpha = 16$. The two approaches performed very similarly: the LoRA model achieved a chrF of roughly 42.24, while full GPT-2-XL achieved about 42.13. Given the very small development set size of 10, this difference is likely at least partly due to noise, although LoRA may also provide a mild regularization benefit because the training set is extremely small, with only 131 examples. Since performance was nearly identical and LoRA was more parameter-efficient, we treat the LoRA version as our baseline in the remainder of this section.

5.2 Method / Extensions

We explored two main extensions for sonnet generation.

First, we observed that the baseline model did not reliably preserve the expected sonnet structure. In the training data, 129 out of 131 sonnets have a total of 14 lines, so the continuation should almost always contain 11 lines. However, the baseline generated a much wider distribution of continuation lengths, as shown in Figure 2. This suggested that the model was not strongly capturing the structural constraints of the task.

To address this, we introduced a *countdown* model. We augmented the training data by prepending each line with a countdown token indicating how many lines remained, such as <113> before the first line and <10> before the final line. The goal was to make the model more aware of sonnet structure and encourage it to learn the desired continuation length more explicitly.

Second, we varied the *best-of- n* candidate pool size at inference time. Rather than generating a single continuation, we sampled n candidate continuations and selected the one with the highest average log-likelihood. However, we found that as n increased, this selection rule introduced a strong bias toward shorter outputs. Figure 3 shows that the average selected line count drifted away from the target as the candidate pool grew.

To counteract this effect, we introduced a penalty-based reranking rule with strength parameter λ . The final score for a candidate continuation was

$$s = \bar{\ell} - \lambda|11 - c| \quad (1)$$

where $\bar{\ell}$ is the average log-likelihood of the candidate continuation and c is its continuation line count.

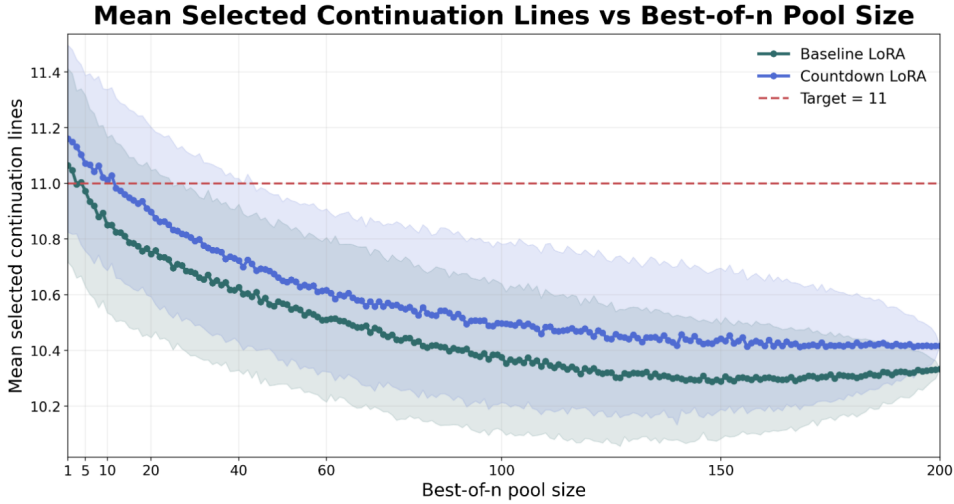


Figure 3: Mean selected continuation length as a function of best-of- n pool size for the baseline and countdown models. The dashed line marks the target continuation length of 11 lines.

5.3 Experimental Setup

We evaluated sonnet-generation quality on the development set using chrF. We varied the penalty parameter over

$$\{0, 0.01, 0.03, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5\}.$$

For each prompt, we first generated a pool of 200 candidate continuations. Then, for each value of n , we repeatedly sampled n candidates uniformly at random from this pool up to 1000 times and averaged the resulting statistics to obtain smoother curves. For each value of n , we recorded metrics including mean selected line count and best chrF. This procedure allowed us to study how candidate-pool size and reranking strength affected both structural properties and evaluation performance.

5.4 Results

Figure 4 shows the effect of best-of- n pool size on chrF for the baseline and countdown methods. The baseline improves slightly for small n but then plateaus and eventually declines as n increases. In contrast, the countdown method continues to improve as n grows and eventually outperforms the baseline by a substantial margin. Interestingly, the countdown curve appears to scale almost linearly over much of this range.

These results suggest that the structural penalty successfully keeps the mean selected line count close to 11 and prevents the drift seen under pure likelihood-based selection. However, its effect on chrF differs across models: it appears to help the countdown model while slightly harming the baseline.

Figure 5 shows that after adding the penalty term, the mean selected continuation length remains close to the target across values of n , rather than drifting downward as before.

Moreover, for the countdown model, the penalty-based reranking substantially narrows the selected line-count distribution, and for larger values of n , almost all selected outputs contain exactly 11 lines, as shown in Figure 6.

5.5 Analysis

The main question raised by the results is why increasing best-of- n helps the countdown model so much more than the baseline. The answer appears to be structural rather than purely probabilistic.

Under baseline reranking, increasing n does not simply provide more chances to find a better continuation. Instead, it increasingly favors shorter outputs. Since candidates are selected by average log-likelihood, shorter continuations often receive artificially favorable scores. As the pool grows, the chance of observing one of these short but highly scored continuations increases, so the selected output drifts farther from the target 11-line structure. This helps explain why the baseline does not continue improving with larger n : the larger pool amplifies a bias in the selection rule rather than improving alignment with the reference.

The countdown model changes this behavior in two ways. First, by inserting explicit countdown tokens during training, it gives the model a much stronger representation of how many lines remain in the sonnet. As a result, even before reranking, its candidate pool contains outputs whose structure is more consistent with the task. Second, once the penalty term is added,

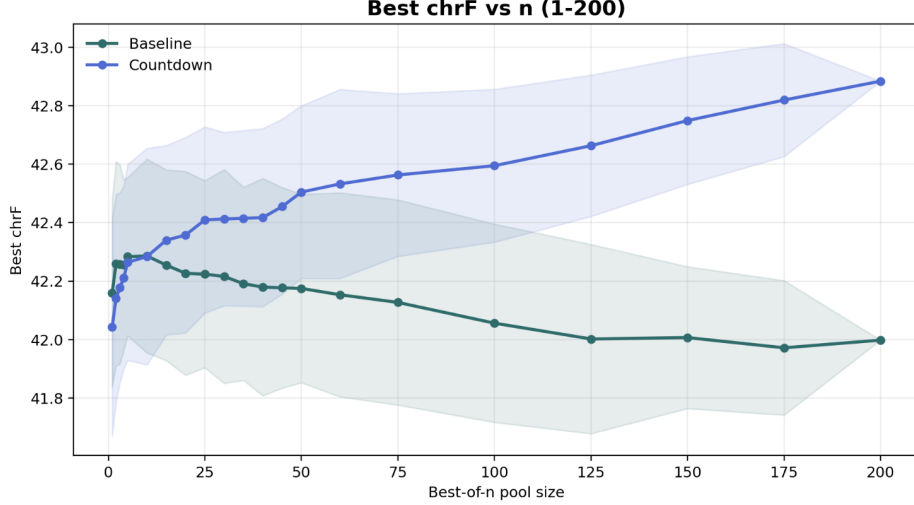


Figure 4: Best chrF as a function of best-of- n pool size for the baseline and countdown models. Larger candidate pools help the countdown model substantially more than the baseline.

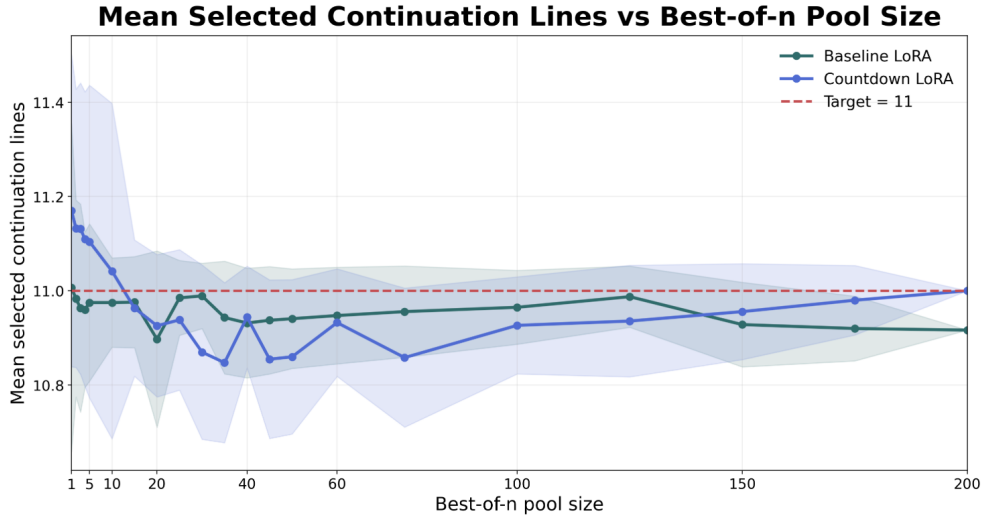


Figure 5: Mean selected continuation length after applying the penalty-based reranking criterion. The penalty largely eliminates the drift away from the 11-line target.

the reranker can exploit larger candidate pools more effectively because it is selecting among candidates that are both more structurally plausible and less likely to benefit from the short-output bias. In other words, larger pools help the countdown model because they provide more opportunities to find continuations that are simultaneously close to the target structure and high-quality under chrF.

The λ -sweep also shows that structural conformity alone is not enough. Keeping the selected continuation length close to 11 is clearly beneficial, but chrF does not improve monotonically with larger penalties. If λ is too small, reranking does not adequately correct the short-output bias; if it is too large, the reranker may over-prioritize line-count matching at the expense of textual quality. This indicates that line-count control is an important part of good sonnet generation under chrF, but it is not the entire story.

Overall, these experiments show that inference-time selection is a major component of sonnet-generation performance. The key improvement does not come merely from generating more candidates. Rather, it comes from combining larger candidate pools with a reranking strategy that better matches the structural properties of the task. The countdown model benefits from best-of- n because it makes larger pools genuinely useful, whereas under the baseline model, larger pools mainly strengthen an undesirable preference for shorter continuations.

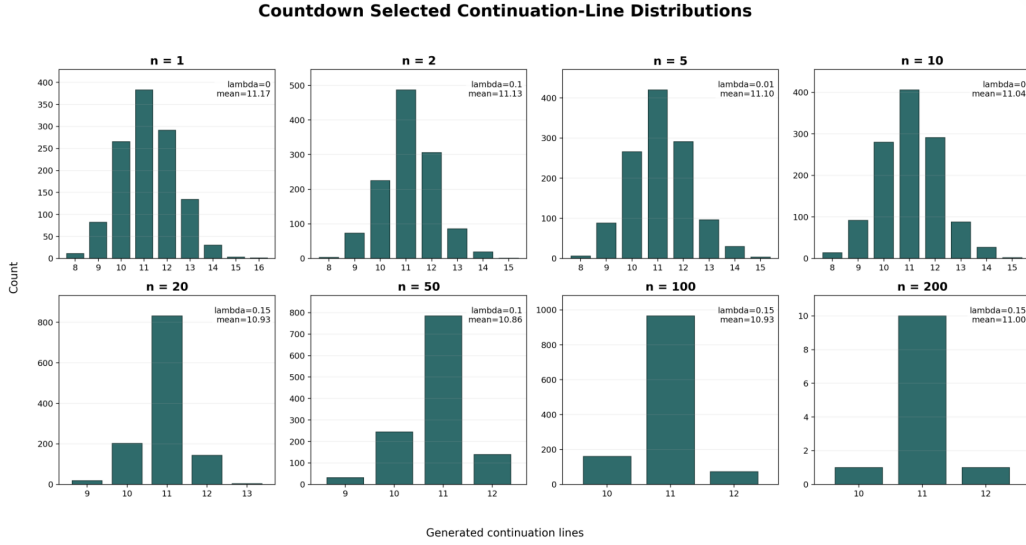


Figure 6: Distribution of selected continuation lengths for the countdown model under the best λ for each value of n . As n increases, the distribution becomes increasingly concentrated at the target length of 11 lines.

6 Conclusion

We studied task-specific strategies for improving GPT-2 across the two default CS224N project tasks: Quora paraphrase detection and sonnet generation. Although both tasks begin from the same pretrained transformer family, our results show that the most effective improvements were highly task-dependent.

For paraphrase detection, the largest gains did not come from prompt wording or lightweight parameter-efficient tuning alone. Instead, the main project-level improvement came from scaling to GPT-2 medium while preserving the strongest scoring recipe, followed by inference-time aggregation through checkpoint ensembling, bidirectional scoring, and threshold tuning. This progression improved development accuracy from a strong GPT-2 small baseline of about 0.893 to a final retained result of 0.914, showing that careful ranking design and inference strategy can add substantial value once the underlying checkpoint family is strong enough.

For sonnet generation, the central challenge was not just producing fluent text, but generating continuations whose structure aligned with the target poems under chrF evaluation. We found that the baseline model exhibited a wide continuation-length distribution and that likelihood-based best-of- n selection introduced a growing bias toward shorter outputs as n increased. Our countdown augmentation and penalty-based reranking addressed this problem by making structural information explicit during training and by discouraging deviations from the target line count at inference time. As a result, the countdown model benefited much more from larger candidate pools and achieved steadily stronger chrF performance than the baseline.

Taken together, these results support a simple conclusion: sharing a pretrained backbone does not imply that the same extension strategy will work across tasks. In the classification setting, the key gains came from stronger checkpoints and better inference-time aggregation. In the generative setting, the key gains came from making the model and reranker more aware of task structure. More broadly, our project suggests that adapting large language models successfully requires matching the method to the failure mode of the task rather than searching for a single universally effective modification.

Team Contributions

Cheney Sang led much of the experimental design and analysis across both tasks, with particular responsibility for the sonnet-generation extensions, including the countdown formulation, best-of- n analysis, structural reranking experiments, and figure preparation. Harris Alan Lee focused primarily on the paraphrase-detection pipeline, including prompt/scoring ablations, LoRA experiments, checkpoint comparisons, and development-set evaluation. Austin Chen contributed to implementation, training infrastructure, debugging, and experiment execution across both tasks, and also helped consolidate results into the final retained systems. All three team members discussed project direction, interpreted results, and contributed to writing and revising the final report.

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] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. First quora dataset release: Question pairs. In *Quora*, 2017.
- [3] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.