

Adapting GPT-2 for Sentiment Analysis, Paraphrase Detection, and Sonnet Generation

CS224N Default Final Project

Fiona Han
Stanford University
hanf@stanford.edu

Samih Qureshi
Stanford University
samih@stanford.edu

William Rose
Stanford University
wrose03@stanford.edu

Abstract

GPT-2 learns strong linguistic representations through next-token prediction, but its pretraining objective does not directly optimize for semantic equivalence judgments or globally coherent structured generation, making downstream adaptation to tasks like paraphrase detection and sonnet completion difficult. We implement GPT-2 from scratch, including masked multi-head self-attention, the GPT transformer layers, token and positional embedding extraction, and the AdamW optimizer. We deploy our model for sentiment analysis on the SST and CFIMDB datasets, paraphrase detection on the Quora dataset, and sonnet generation on the Shakespeare sonnet dataset. To improve performance, we explore several extensions: LoRA for parameter-efficient fine-tuning, data augmentation via word dropout and synonym replacement for paraphrase detection, discriminative fine-tuning with separate learning rates for the pretrained backbone and classifier head, label smoothing on augmented positive pairs, multi-task training combining paraphrase classification and sonnet generation, CHRF-based early stopping, and scaling to GPT-2 Medium. We find that full fine-tuning consistently outperforms parameter-efficient alternatives, and that increased model capacity yields the largest gains for generation. Our best paraphrase model achieves 88.9% test accuracy (89.0% dev), and our best sonnet model achieves 42.36 CHRF on the test set using GPT-2 Medium with nucleus sampling.

1 Key Information

- TA mentor: Ahmed Ahmed (ahmedah@cs.stanford.edu)
- External Collaborator, External Mentor, Sharing Project: **No**
- Late Days (if you are using any):

Team member	Late days remaining
Fiona Han	4
Samih Qureshi	5
William Rose	1
Total pooled	10
Team size	3
Late days applied for the report	3 (for 1 day)

2 Introduction

In this paper, we study how to adapt a pretrained GPT-2 style model to downstream tasks, focusing our extensions on paraphrase detection and sonnet generation.

Both tasks highlight weaknesses of pretrained language models when used “out of the box.” Paraphrase detection requires fine-grained semantic equivalence rather than surface similarity, and sonnet generation asks for global coherence and stylistic constraints across lines, where small early errors can cascade. While GPT-2 learns strong linguistic priors during pretraining, its objective of next-token prediction does not directly optimize for tasks requiring semantic equivalence judgments or structured poetic generation. This motivates our targeted fine-tuning.

For paraphrase detection, we explored several strategies to improve robustness, including data augmentation through synonym replacement on positive paraphrase pairs, as well as word dropout accompanied by label smoothing. For sonnet generation, we trained a multitask model on both paraphrase and generation tasks. We also evaluated parameter-efficient fine-tuning (PEFT) using LoRA across both tasks. Our experiments show that full fine-tuning of GPT-2 base achieved the highest paraphrase accuracy (89.0%), outperforming LoRA (87.0%) and data augmentation with label smoothing (88.0%). Sonnet generation benefited most from increased model capacity and careful regularization, with GPT-2 medium and CHRF-based early stopping producing the best dev score (41.93). We also found that multitask training with paraphrase detection improved sonnet generation for GPT-2 base (41.26), while LoRA consistently underperformed full fine-tuning across both tasks (39.00).

3 Related Work

The GPT series of models has established a strong foundation for transfer learning in NLP. GPT-1 [1] introduced a decoder-only transformer trained on large amounts of unlabeled text, showing that unsupervised pretraining can improve performance on downstream tasks such as question answering or sentiment analysis. GPT-2 [2] scaled this approach significantly, demonstrating that larger models trained on more data capture richer contextual patterns and improve generalization. These works provide the basis for adapting pretrained language models to a variety of tasks.

In the context of paraphrase detection, prior work often uses sentence embedding or siamese network architectures like BERT to directly align semantically equivalent sentences [3]. These approaches highlight a limitation of autoregressive language models like GPT for classification tasks, since they are trained on next-token prediction rather than pairwise similarity, highlighting the need to explore adaptation strategies.

Data augmentation has been widely explored as a way to improve robustness and generalization in NLP models. Techniques such as synonym replacement and word dropout introduce controlled perturbations to training examples, encouraging models to rely less on exact lexical overlap and more on underlying semantic meaning [4]. Parameter-efficient fine-tuning methods such as LoRA [5] adapt large pretrained models while updating only a small number of additional parameters. By injecting low-rank trainable matrices into existing layers, LoRA can retain the general knowledge encoded

in the pretrained model while still allowing task-specific adjustments. Finally, multitask learning leverages shared representations across related tasks, allowing knowledge learned from one objective to benefit another. This method encourages the model to learn more generalizable features and reduce reliance on task-specific patterns, potentially improving performance on each individual task [6]. These techniques have inspired our approach, providing methods to improve model adaptation while reducing training cost.

4 Approach

The project builds on a starter codebase provided by the CS224N course staff, which includes the data loading utilities (`datasets.py`), evaluation scaffolding (`evaluation.py`), model configuration (`config.py`), and the skeleton class structure for each task. We implemented the following components from scratch: causal multi-head self-attention (`attention.py`), GPT-2 transformer layers (`gpt2_layer.py`), the full GPT-2 model with pretrained weight loading (`gpt2.py`), the AdamW optimizer (`optimizer.py`), the sentiment classification head (`classifier.py`), and the LoRA adapter module (`modules/lora.py`). All extensions described below (data augmentation, label smoothing, discriminative fine-tuning, multi-task training, CHRF early stopping, model scaling) are our own additions.

4.1 GPT-2 Architecture

For GPT-2, we implemented masked multi-head self-attention, GPT-2 transformer layers (small implementation / 12 layers), and token/positional embedding extraction. We use causal masking to restrict each token’s attention to only its preceding context. Each transformer layer also includes a position-wise feed-forward network with two linear transformations and a GeLU activation function. We use dropout with $p = 0.1$ throughout the network for regularization.

In order to train our model, we implemented the AdamW optimizer with decoupled weight decay regularization [7]. At each step t , the first and second moment estimates are updated as $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ and $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$. We apply the efficient bias-corrected formulation $\tilde{\alpha}_t = \alpha \cdot \sqrt{1 - \beta_2^t} / (1 - \beta_1^t)$ and update parameters as:

$$\theta_t = \theta_{t-1} - \tilde{\alpha}_t \cdot \frac{m_t}{\sqrt{v_t} + \epsilon} - \alpha \lambda \theta_{t-1}$$

where the weight decay term $\alpha \lambda \theta_{t-1}$ is applied directly to the parameters rather than added to the gradient, decoupling regularization from the adaptive learning rate.

4.2 Sentiment Analysis

We implemented a `GPT2SentimentClassifier` for downstream sentiment classification. The model extracts the last token’s hidden state (dim 768) from GPT-2, applies dropout ($p = 0.3$), and projects through a linear layer to produce logits over label classes. We evaluate two fine-tuning regimes: *last-linear-layer*, where GPT-2 weights are frozen and only the classification head is trained, and *full-model*, where all parameters are updated end-to-end. Using the last token is the natural GPT-2 analog of BERT’s [CLS] token, since GPT-2 is a causal (left-to-right) model.

4.3 Paraphrase Detection

We built our paraphrase detection system on top of the starter implementation provided in the assignment, which adapts GPT-2 for binary classification through a cloze-style prompting formulation. The input is formatted as `Is “{s1}” a paraphrase of “{s2}”? Answer “yes” or “no”:`, and the last token’s hidden state is passed through a two-class linear head to produce logits corresponding to “yes” and “no.” For the autograder, predicted classes are mapped to GPT-2 token IDs (“yes” $\rightarrow$ 8505, “no” $\rightarrow$ 3919). We used this starter implementation as our baseline and extended it with two methods of data augmentation to increase the diversity of training examples: synonym replacement using WordNet and word dropout accompanied by label smoothing.

The first method uses synonym replacement based on WordNet [8], where words sharing a synset are treated as mutually replaceable. For each sentence, we consider only content tokens (excluding

stopwords and punctuation), and replace each candidate with probability 0.1, up to two replacements per sentence. This augmentation is applied only to positive pairs (label = 1), and we generate three variants per pair: (s1_aug, s2), (s1, s2_aug), and (s1_aug, s2_aug) to increase diversity while preserving the paraphrase label.

The second augmentation method uses word dropout, where tokens in the input sequence are randomly removed with probability $p_{\text{drop}} = 0.1$ during training. This encourages the model to rely less on specific lexical cues and learn more robust semantic representations. To stabilize training under this noisy input setting, we apply label smoothing: augmented positive pairs receive soft labels $y = 1 - p_{\text{drop}}$ instead of hard labels $y = 1$. The training loss uses KL divergence between the model’s output distribution and the smoothed target:

$$\mathcal{L} = \text{KL}([y, 1-y] \parallel \text{softmax}(z)) \quad (1)$$

where z are the logits and $y = 0.9$ for augmented positives. This reflects the intuition that a dropout-corrupted pair is slightly less confident as a paraphrase. Lastly, we also implemented parameter-efficient fine-tuning with LoRA to compare against full fine-tuning.

4.4 Sonnet Generation

Our approach in sonnet generation differs from the previous downstream applications of the GPT-2 model. This task is generative, rather than based in classification. Therefore, we train the model with a cross-entropy loss objective for next-token prediction. To bolster the performance of the model, we implemented several extensions. We use Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning [5]. LoRA freezes the pretrained weight matrix $W_0 \in \mathbb{R}^{d \times k}$ and adds a low-rank decomposition, so the forward pass becomes:

$$h = W_0 x + \frac{\alpha}{r} (x A^T) B^T \quad (2)$$

where $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ are the only trainable parameters, with rank $r \ll \min(d, k)$ and scaling factor α/r . We apply LoRA to the Q, K, V, and output projections in every attention layer, reducing trainable parameters from 117M to approximately 3.5M.

Second, we trained a multitask model jointly on paraphrase detection and sonnet generation, sharing a single GPT-2 backbone while using task-specific output heads and loss functions for each task.

5 Experiments

5.1 Data

5.1.1 Sentiment Analysis

We evaluate on SST [9] (8,544 train / 1,101 dev, 5-class sentiment over movie review sentences) and CFIMDB (1,701 train / 245 dev, binary sentiment over full movie reviews).

5.1.2 Paraphrase Detection

We evaluate our model on the Quora dataset[10]. The training set contains 141,506 examples, the development set contains 20,215 examples, and the test set contains 40,431 examples.

5.1.3 Sonnet Generation

We draw from the Shakespeare Sonnet Dataset, which contains 154 Shakespearean sonnets. We use cross-entropy loss on 143 of the sonnets to train our model. The other 12 sonnets are held out, where our evaluation task involves generating a complete sonnet, given the first 3 lines.

5.2 Evaluation method

5.2.1 Sentiment Analysis

We evaluate the accuracy of our model on both datasets (SST, CFIMDB). In the SST dataset, we measure the percent of correctly predicted polarities out of the five possible labels. In the CFIMDB

dataset, labels are binary, and we similarly measure the percent of correctly predicted labels in the evaluation set.

5.2.2 Paraphrase Detection

Our model is evaluated for accuracy on the Quora dataset. We compare the predictions to the true binary (yes/no) labels. The accuracy of our model is measured as the percent of correctly predicted labels out of the evaluation set.

5.2.3 Sonnet Generation

Our model is evaluated on 12 held-out sonnets from the Shakespeare Sonnet Dataset. After training on the other 143, we present our model with the first 3 lines of the held-out sonnets and prompt it to complete the remaining lines. The generated sonnet is then evaluated with the CHRF metric, which ranges from 0 to 100. This measures how similar the generated sonnet is to the true original, using a character-level n-gram comparison metric. All reported scores use nucleus sampling ($p = 0.9$, $T = 1.2$).

5.3 Experimental details

Training infrastructure. All models were trained on a single NVIDIA L4 GPU (24 GB VRAM) provisioned via Google Cloud Platform (g2-standard-4 instance). We use mixed-precision training (`torch.cuda.amp`) for paraphrase detection to reduce memory footprint and accelerate training.

Optimizer. All tasks use our AdamW implementation with decoupled weight decay ($\lambda = 0.01$), $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and bias correction for both moment estimates.

Sentiment analysis. We train on SST with learning rate 1×10^{-5} , batch size 64, and dropout $p = 0.3$ on the classification head for 10 epochs under both fine-tuning regimes.

Paraphrase detection. Our baseline uses GPT-2 base (117M parameters) with full fine-tuning, learning rate 1×10^{-4} , batch size 32, and 5 training epochs. We implemented and explored the following extensions:

- **LoRA** [5]: rank $r=8$, scaling $\alpha=16$, learning rate 1×10^{-4} , applied to Q, K, V, and output projections. Base weights are frozen; only LoRA adapters and the classification head are trained.
- **Word dropout with label smoothing**: word dropout ($p=0.1$) on positive pairs only. Augmented pairs receive soft labels ($y = 0.9$) to reflect degraded paraphrase quality.
- **WordNet synonym augmentation**: synonym lookup via NLTK WordNet with a max of two replacements per pair, each with 0.1 probability. Same positive-only and label-smoothing strategy.
- **Discriminative fine-tuning**: separate learning rates for the pretrained backbone (1×10^{-5}) and the classification head (1×10^{-3}), allowing the task-specific layer to adapt more aggressively while preserving pretrained representations.

Sonnet generation. The baseline uses GPT-2 base with learning rate 1×10^{-5} , batch size 8, and 10 epochs. We train with cross-entropy loss for next-token prediction on 143 sonnets and generate completions from 3-line prompts using nucleus sampling ($p = 0.9$, $T = 1.2$). Extensions:

- **LoRA**: same configuration as paraphrase ($r=8$, $\alpha=16$), learning rate 1×10^{-4} .
- **Multi-task training**: alternating batches of paraphrase classification loss and sonnet language-modeling loss within each epoch, sharing the GPT-2 backbone while maintaining separate task heads.
- **CHRF early stopping (CHRF-ES)**: we evaluate dev CHRF after each epoch and retain the best checkpoint with patience 8, preventing overfitting on the small sonnet corpus.
- **Model scaling**: GPT-2 medium (345M, 1024d, 24 layers) with $\text{lr} = 1 \times 10^{-5}$, up to 30 epochs with CHRF early stopping.

5.4 Results

5.4.1 Sentiment Analysis

Table 1 reports dev accuracy for both fine-tuning regimes. Full-model fine-tuning achieves 50.8% on the 5-class SST task and 96.3% on binary CFIMDB. The last-linear-layer regime, which freezes all GPT-2 weights and trains only the classification head, reaches 45.1% on SST and 82.9% on CFIMDB. The gap is especially large on CFIMDB, where updating the full model allows it to adapt its representations to longer review texts. On SST, the modest accuracy under both types reflects the difficulty of fine-grained 5-class sentiment prediction with a causal language model.

Table 1: Sentiment analysis dev accuracy (%)

Fine-tuning regime	SST (5-class)	CFIMDB (binary)
Last-linear-layer	45.1	82.9
Full-model	50.8	96.3

5.4.2 Paraphrase Detection

Table 2 summarizes dev and test accuracy for all paraphrase configurations. Our full fine-tuning baseline achieves 89.0% dev accuracy. Among our extensions, WordNet synonym augmentation comes closest at 88.9%, while word dropout with label smoothing (88.0%) and LoRA (87.0%) both fall short. Discriminative fine-tuning matches the baseline on dev (89.0%) and achieves 88.9% on the test set.

Table 2: Paraphrase detection results (dev and test accuracy)

Method	Dev Acc (%)	Test Acc (%)	Best Epoch
Full fine-tuning (baseline)	89.0	—	2
+ Discriminative fine-tuning	89.0	88.9	3
+ WordNet augmentation	88.9	—	8
+ Word dropout + label smoothing	88.0	—	4
LoRA ($r=8$, $\alpha=16$)	87.0	—	4

5.4.3 Sonnet Generation

Table 3 reports CHRF scores across sonnet configurations. Our best model, GPT-2 medium with CHRF early stopping, achieves 41.93 on dev and 42.36 on the test dataset. Among GPT-2 base models, multi-task training scores 41.26 and a higher learning rate ($lr = 5 \times 10^{-5}$) yields 41.22. LoRA underperforms at 39.00, and combining multi-task training with LoRA produces the lowest score (37.72).

Table 3: Sonnet generation results (dev and test CHRF)

Configuration	Model	CHRF-ES	Dev CHRF	Test CHRF
Medium, 30 epochs	GPT-2 medium	✓	41.93	42.36
Multi-task	GPT-2 base		41.26	—
$lr = 5 \times 10^{-5}$, 20 ep	GPT-2 base	✓	41.22	—
$lr = 1 \times 10^{-4}$, 20 ep	GPT-2 base	✓	40.87	—
LoRA ($r=8$)	GPT-2 base		39.00	—
Multi-task + LoRA	GPT-2 base		37.72	—

6 Analysis

Both tasks reveal a consistent pattern: the simplest approaches tend to work best, and the primary challenge is managing overfitting and noise rather than increasing model expressiveness.

6.1 Paraphrase Detection

Extensions. With approximately 141k training pairs, GPT-2 base has enough data to adapt without overfitting, and our baseline converges by epoch 2 (Figure 1). Every extension we tried either matched or underperformed this result. LoRA ($r=8$) reaches only 87.0%, suggesting that the rank-8 bottleneck is too restrictive for a task that requires updating the full representation space. Discriminative fine-tuning (head $lr = 10^{-3}$, backbone $lr = 10^{-5}$) matches the baseline at 89.0% but does not improve on it, indicating that the backbone already adapts well at a uniform learning rate. These findings are consistent with prior work showing LoRA is most beneficial when labeled data is scarce or the model is very large relative to the task [5].

Data augmentation. WordNet augmentation (88.9%) nearly matches the unaugmented baseline, while word dropout with label smoothing drops to 88.0%. The training set is large enough that augmentation introduces more noise than new signal. Word dropout in particular can distort semantic meaning: dropping a content word from “*Why is computer vision computationally hard?*” changes the question entirely, and even soft labels ($y = 0.9$) cannot fully compensate. WordNet replacement fares better because its substitutions preserve meaning more faithfully, but the marginal gain confirms that the model already generalizes well from the original data.

Error analysis. To understand the remaining 11% of errors, we examined false positives and false negatives on the dev set. Our best model produces 2,434 false positives and 1,999 false negatives, revealing two systematic failure modes. False positives tend to involve pairs with high word overlap but a subtle semantic difference, such as “*How long can you leave cooked chicken wings out at room temp?*” vs. “*Can you leave raw chicken out all night?*”, which share a topic but ask fundamentally different things. The cloze formulation, which concatenates both sentences into a single sequence, may bias the model toward surface-level token overlap. False negatives involve semantically equivalent sentences with different wording, such as “*Who are the people with the most followers on Quora?*” vs. “*Who is the most popular writer on Quora?*”. These require reasoning about synonymy that the model’s left-to-right attention cannot easily capture. The slight asymmetry (more false positives than false negatives) suggests the model is biased toward predicting “paraphrase,” likely because the cloze prompt places both sentences in close proximity, making surface similarity a strong confounding factor for the task.

6.2 Sonnet Generation

Overfitting and early stopping. With only 143 training sonnets, we observed early in our experiments that models with lower training loss often scored *worse* on CHRF. A model trained for 20 epochs at $lr = 10^{-4}$ reached a training loss near 0.5, effectively memorizing training sonnets verbatim, and produced fluent but non-generalizable completions. This motivated our implementation of CHRF-based early stopping, which evaluates dev CHRF after each epoch and retains the best checkpoint with patience 8. Figure 2 illustrates this for our best model (GPT-2 medium): training loss decreases monotonically throughout all 25 epochs, but dev CHRF plateaus after epoch 2 and oscillates around 40–42, with the peak at epoch 16. This confirms that training loss is a poor proxy for generation quality on very small datasets, and that task-specific evaluation metrics are essential for proper checkpoint selection.

Model capacity. Scaling from GPT-2 base (117M parameters) to GPT-2 medium (345M) yields the largest improvement across all experiments: 41.93 vs. 40.87 CHRF for the best base-model variant. The additional 12 transformer layers and wider hidden dimension (1024 vs. 768) allow the model to capture longer-range dependencies and richer stylistic patterns. We also attempted GPT-2 large (774M parameters), but it exceeded the L4 GPU’s 22 GB VRAM even at batch size 1, so medium was our practical ceiling.

Multi-task training and LoRA. Multi-task training achieves the best GPT-2 base score (41.26), outperforming all single-task base-model variants. The paraphrase classification objective encourages the shared backbone to learn more robust semantic representations, which appear to benefit sonnet generation by improving coherence across lines. However, combining multi-task training with LoRA (37.72) degrades performance below either method alone, indicating that the rank-8 bottleneck cannot support two objectives simultaneously. More broadly, LoRA consistently underperforms

full fine-tuning across all sonnet experiments (39.00 vs. 40.87+ for base models), suggesting that parameter-efficient methods are less suited to open-ended generation on small corpora where the model needs maximum flexibility to learn subtle stylistic patterns.

Qualitative examination. Table 4 in the Appendix shows the full generated output for Sonnet 132 alongside Shakespeare’s original (lines 1–3 are the given prompt). The model maintains Shakespearean diction throughout, using words like “divining,” “hearsay,” and “o’erlooks,” and sustains the mourning theme through the early generated lines. However, three recurring failure modes emerge:

- *Semantic drift*: by line 9 the imagery shifts from mourning to unrelated scenes (“*Who linger at tables their manners forbid*”), and by line 11 the text becomes nonsensical (“*peep with gentle paws*”). The model generates locally plausible n-grams but lacks a global plan for the idea of the sonnet.
- *Rhyme scheme loss*: Shakespeare’s sonnets follow a strict ABAB CDCD EFEF GG pattern, but the generated text shows no consistent end rhyme. The autoregressive generation process has no mechanism to plan future line endings in advance.
- *Structural violations*: the model produces 15 lines instead of 14, with a final couplet that lacks the epigrammatic closure characteristic of Shakespearean sonnets.

These patterns are consistent across all 12 dev sonnets. The model captures local style effectively but struggles with global structural constraints. Improving generation further would likely require structural constraints during decoding (such as enforcing 14-line output and end-rhyme) or training objectives that explicitly reward formal properties of the sonnet.

7 Conclusion

This project explored adapting GPT-2 to two downstream tasks: paraphrase detection and sonnet generation. For paraphrase detection on the Quora Question Pairs dataset, full fine-tuning of GPT-2 base achieved the best dev accuracy (89.0%), outperforming LoRA (87.0%) and data augmentation with label smoothing (88.0%). WordNet-based synonym replacement performed comparably (88.9%), indicating that the large training set provides sufficient signal for full fine-tuning, while low-rank or noisy updates offer limited benefit.

For sonnet generation, scaling to GPT-2 medium and using CHRF-based early stopping produced the best dev score (41.93%). Multi-task training with paraphrase detection improved GPT-2 base outputs (41.26%), suggesting semantic supervision aids coherence. LoRA consistently underperformed (39.00–37.72%), highlighting the challenge of low-rank adaptation for open-ended creative generation.

These results show that adaptation strategies must align with what the task demands: full fine-tuning performed best for the large paraphrase dataset, while generation quality benefited more from increased model capacity. Parameter-efficient methods such as LoRA, while attractive for reducing training cost, may limit the model’s ability to adapt flexibly for both semantic classification and open-ended generation tasks.

Our work has a few limitations. The paraphrase model relies on a cloze-style prompt that concatenates both sentences into a single sequence, biasing predictions toward surface level overlap rather than deeper semantic reasoning. For sonnet generation, the autoregressive decoding process has no mechanism to plan ahead for rhyme scheme or line count, leading to structural violations in nearly all generated outputs. Future work could address these gaps by incorporating bidirectional context for paraphrase detection, and by adding constrained decoding strategies that enforce sonnet structure, such as fixed 14-line output and end-rhyme scoring during generation. Exploring larger-scale PEFT methods with higher rank or alternative adapters may also close the gap between parameter-efficient and full fine-tuning.

Overall, our experiments demonstrate that full fine-tuning and model scaling are the most reliable strategies for adapting pretrained language models, while parameter-efficient methods and data augmentation offer limited benefit when sufficient data or model capacity is already available.

Team contributions (Required for multi-person team)

- **Fiona Han:** Implementation of paraphrase detection task + extensions.
- **Samih Qureshi:** Sonnet extensions. Training, which included integrating Fiona’s and William’s changes, making bug-fixes, and monitoring GPUs
- **William Rose:** GPT-2 Layers. Adam Optimizer. Implementation of sonnet generation + extensions.

References

- [1] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf, 2019.
- [3] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.
- [4] Jason Wei and Kai Zou. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China, November 2019. Association for Computational Linguistics.
- [5] 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. *arXiv preprint arXiv:2106.09685*, 2021.
- [6] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, 2018.
- [7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2019.
- [8] George A Miller. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.
- [9] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.
- [10] Shankar Iyer, Nikhil Dandekar, and Kornél Csernai. Quora question pairs dataset. <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>, 2017. Dataset release announcement, Quora Inc.

A Appendix

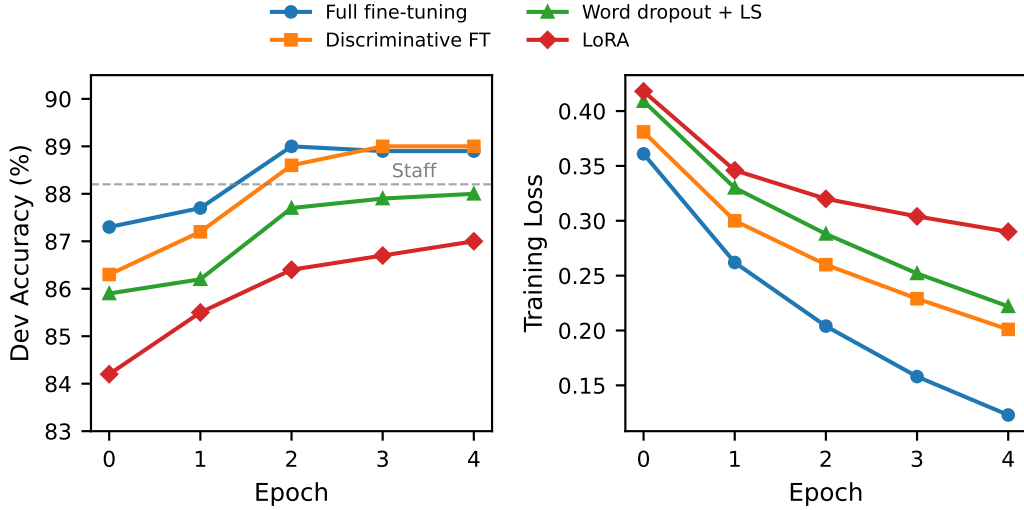


Figure 1: *Paraphrase detection training curves.*

Left: dev accuracy across 5 epochs for all methods. Full fine-tuning converges fastest (epoch 2), while LoRA and augmentation continue improving through epoch 4 but remain below baseline.

Right: training loss.

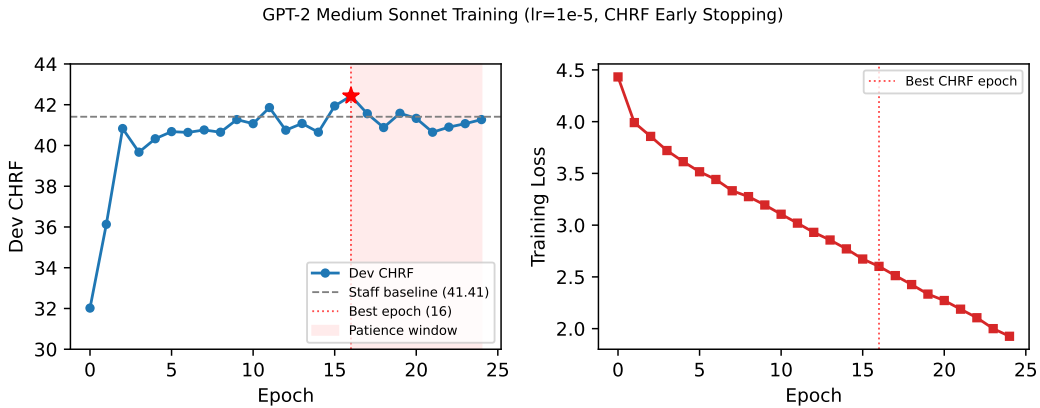


Figure 2: *GPT-2 Medium sonnet training with CHRF early stopping ($lr = 10^{-5}$, patience = 8).*

Left: dev CHRF plateaus after epoch 2 and oscillates around 40–42 despite continued training, with the best checkpoint at epoch 16. The red shaded region shows the patience window after which training stops.

Right: training loss decreases monotonically throughout, demonstrating that lower loss does not correspond to better generation quality on this small corpus.

Table 4: Generated vs. reference completion for Sonnet 132. Lines 1–3 are the given prompt. Generated lines that we determined are incoherent or break sonnet form are marked in **bold**.

	Reference (Shakespeare)	Generated (GPT-2 Medium)
1	<i>Thine eyes I love, and they, as pitying me,</i>	
2	<i>Knowing thy heart torment me with disdain,</i>	
3	<i>Have put on black, and loving mourners be,</i>	
4	Looking with pretty ruth upon my pain.	In tombs reserved for widows and orphans,
5	And truly not the morning sun of heaven	Full divining hearts on hearsay fears,
6	Better becomes the gray cheeks of the east,	Whilst hearts itself pursuing cause confined
7	Nor that full star that ushers in the even	Without eyes like determined maidens remain.
8	Doth half that glory to the sober west	Nay, but for their faults the world o'erlooks
9	As those two mourning eyes become thy face.	Who linger at tables their manners forbid;
10	O, let it then as well beseem thy heart	For unseen sorrowful faces and eyes cannot
11	To mourn for me, since mourning doth thee grace,	Mark truth, but peep with gentle paws
12	And suit thy pity like in every part.	To dark thought with knowing lips or to fair fingers,
13	Then will I swear beauty herself is black,	And know all things from thence by smell.
14	And all they foul that thy complexion lack.	So the sour many mourners are,
15		And sour hope their mourners are too.