

Robust Fine-Tuning of GPT-2 with SMART Regularization, LoRA, and Enhanced Decoding for Downstream NLP Tasks

Default Final Project

Jake Klosowski, Alex Wurm, Kevin Stephen
{jfkx, alexwurm, kevints}@stanford.edu

Mentor: Alisa Levin

CS224N: Natural Language Processing with Deep Learning
Stanford University

Abstract

Fine-tuning large pretrained language models on downstream tasks is effective but prone to overfitting, particularly on small datasets. In this project, we implement GPT-2 and apply it to three downstream tasks: sentiment analysis, paraphrase detection, and sonnet generation. Beyond the baseline model, we implement three complementary extensions: (1) SMART regularization (Smoothness-inducing Adversarial Regularization with Bregman Proximal Point optimization) to improve robustness during fine-tuning, (2) Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning, and (3) enhanced sonnet generation with auxiliary data augmentation, SMART regularization, constrained decoding, and rhyme-aware generation. Ultimately, we combine LoRA with SMART regularization ($\lambda_s = 0.5$, $\mu = 0.25$), resolving an initial mode collapse and demonstrating that the two techniques are composable with careful tuning. We additionally incorporate training improvements including layer-wise learning rate decay, label smoothing, gradient accumulation, mixed-precision training, cosine learning rate scheduling, and SDPA-accelerated attention. Our best paraphrase detection model achieves 89.1% dev accuracy, sentiment analysis reaches 51.2% on SST and 97.6% on CFIMDB (full-model), and our sonnet generator produces structurally correct 14-line continuations of Shakespearean verse.

1 Introduction

Pretrained language models have become the dominant type of model in natural language processing, with models like GPT-2 demonstrating that large-scale, unsupervised pretraining followed by task-specific fine-tuning yields strong performance across diverse tasks. However, fine-tuning these models on small downstream datasets remains challenging due to overfitting, catastrophic forgetting as discussed in class, and the computational cost of updating hundreds of millions of parameters during training / inference.

In this project, we implement GPT-2 – including masked multi-head self-attention, transformer layers, and the AdamW optimizer – and explore its capabilities on three tasks: sentiment analysis on the Stanford Sentiment Treebank (SST) and CFIMDB, cloze-style paraphrase detection on the Quora Question Pairs dataset, and autoregressive sonnet generation on Shakespeare’s sonnets, including fine-tuning with existing Shakespearean verse.

Our key contribution is a systematic exploration of techniques to improve fine-tuning robustness and efficiency. In particular, we implement SMART regularization [Jiang et al., 2020], which combats overfitting through adversarial perturbation of input embeddings and trust-region-style parameter updates. We also implement LoRA [Hu et al., 2022], which achieves parameter-efficient fine-tuning by injecting trainable low-rank matrices into the attention layers. To improve the quality of sonnet generation, we also develop an enhanced decoding pipeline featuring temperature scaling, top- k /top- p sampling, repetition penalties, beam search, and multi-candidate selection scored by average log-probability.

Beyond these primary extensions, we incorporate several training improvements: layer-wise learning rate decay (LLRD), label smoothing, gradient accumulation, cosine learning rate scheduling with warmup, early stopping, and SDPA (Scaled Dot-Product Attention) via PyTorch’s fused kernels for $O(T)$ memory attention. Together, these form a comprehensive fine-tuning toolkit that we evaluate across all three tasks.

2 Related Work

Pretrained Language Models. The paradigm of pretraining transformer models on large text corpora followed by task-specific fine-tuning was established by GPT-1 [Radford et al., 2018] and scaled significantly by GPT-2 [Radford et al., 2019]. BERT (Bidirectional Encoder Representations from Transformers) [Devlin et al., 2019] demonstrated the complementary approach of masked language modeling for bidirectional representations. Our work builds on the GPT-2 architecture, adapting the decoder-only model for classification via last-token extraction.

Robust Fine-Tuning. Aggressive fine-tuning of pretrained models on small datasets can lead to overfitting and instability. To mitigate this, Jiang et al. [2020] proposed SMART, which addresses this through two mechanisms: smoothness-inducing adversarial regularization, which encourages stable predictions under small input perturbations, and Bregman proximal point optimization, which constrains parameter updates to stay close to the pretrained model. We implement both components in the GPT-2 fine-tuning pipeline.

Parameter-Efficient Fine-Tuning. LoRA [Hu et al., 2022] introduced the idea of freezing pretrained weights and injecting trainable low-rank decomposition matrices into transformer layers. This reduces the number of trainable parameters by orders of magnitude while maintaining competitive performance. We implement LoRA (see `lora.py`) for the query, key, and value projections in GPT-2’s attention layers.

Text Generation. Autoregressive text generation benefits from careful decoding strategies including nucleus (top- p) sampling, top- k filtering, temperature scaling, and repetition penalties. We combine these techniques with beam search and multi-candidate scoring for sonnet generation.

3 Approach

3.1 GPT-2 Implementation

We implement the GPT-2 (small, 124M parameter) architecture following Radford et al. [2019]. The model consists of token and positional embeddings, 12 transformer decoder layers, and a final layer norm. Each transformer layer follows the pre-norm architecture: layer normalization is applied *before* the multi-head self-attention and feed-forward sublayers, with residual connections around each.

Causal Self-Attention. Each attention head computes scaled dot-product attention with a causal mask:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} + M_{\text{causal}} \right) V \quad (1)$$

where M_{causal} is an upper-triangular mask with $-\infty$ above the diagonal, preventing tokens from attending to future positions. We additionally implement an SDPA fast-path that combines the causal and padding masks into a single additive mask passed to `F.scaled_dot_product_attention`, reducing memory from $O(T^2)$ to $O(T)$.

AdamW Optimizer. We implement AdamW [Loshchilov and Hutter, 2019] with decoupled weight decay: weight decay is applied to parameters before the Adam update, using bias-corrected first and second moment estimates.

Downstream Task Heads. For sentiment classification, we extract the last non-padding token’s hidden state and pass it through a dense layer, Tanh activation, dropout, and a linear classifier. For paraphrase detection, the input is formatted as a cloze-style prompt (“Is “{s1}” a paraphrase of “{s2}”? Answer “yes” or “no”: ”) and the last token’s hidden state is projected to 2 classes. For sonnet generation, hidden states are projected back to the vocabulary via weight tying with the input embeddings.

3.2 SMART Regularization

We implement SMART [Jiang et al., 2020] as our primary extension for robust fine-tuning. The overall objective combines the task loss with two regularization terms:

$$\min_{\theta} \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta) + \mu \mathcal{D}_{\text{Breg}}(\theta, \tilde{\theta}_t) \quad (2)$$

Smoothness-Inducing Adversarial Regularization. The regularizer $\mathcal{R}_s$ encourages the model’s output to be stable under adversarial perturbations of the input embeddings:

$$\mathcal{R}_s(\theta) = \frac{1}{n} \sum_i \max_{\|\tilde{x}_i - x_i\|_{\infty} \leq \epsilon} \ell_s(f(\tilde{x}_i; \theta), f(x_i; \theta)) \quad (3)$$

where x_i are the token+position embeddings from the first embedding layer, $\tilde{x}_i$ are adversarial perturbations, and ℓ_s is the symmetrized KL divergence:

$$\ell_s(P, Q) = D_{\text{KL}}(P \| Q) + D_{\text{KL}}(Q \| P) \quad (4)$$

The adversarial perturbation $\tilde{x}_i$ is found via one step of projected gradient ascent initialized with Gaussian noise ($\sigma = 10^{-3}$), projecting back to the L_{∞} ball of radius $\epsilon = 10^{-2}$. These values are calibrated for GPT-2’s unnormalized input embeddings (typical magnitude ± 5); the BERT-scale defaults ($\epsilon = 10^{-5}$) we initially used were less than 0.0001% of embedding magnitude, making $\mathcal{R}_s \approx 0$. This was a critical bug we identified during debugging (see Analysis).

Bregman Proximal Point Optimization. To prevent aggressive parameter updates, we maintain an exponential moving average (EMA) of model parameters $\tilde{\theta}_t$ and add a Bregman divergence penalty:

$$\mathcal{D}_{\text{Breg}}(\theta, \tilde{\theta}_t) = \frac{1}{n} \sum_i \ell_s(f(x_i; \theta), f(x_i; \tilde{\theta}_t)) \quad (5)$$

The EMA is updated per gradient step: $\tilde{\theta}_{t+1} = (1 - \beta)\theta_t + \beta\tilde{\theta}_t$ with $\beta = 0.99$ for the first 10% of total steps and $\beta = 0.999$ thereafter. Our initial implementation incorrectly updated the EMA once per epoch rather than per step, leaving it at $\sim 99\%$ pretrained initialization throughout training and turning $\mathcal{D}_{\text{Breg}}$ into a penalty that prevented any learning—the primary cause of our initial 40.2% collapse (see Analysis).

Implementation. We implement SMART in a separate module (`smart.py`) with task-agnostic utilities. The key design decision is adding a `forward_from_embeddings` method to `GPT2Model` that runs the encoder from arbitrary embeddings, bypassing the embedding layer. This allows SMART to perturb x_i and run the rest of the model without re-embedding. Task-specific adapter functions route the last-token hidden state through the appropriate classification head.

3.3 Low-Rank Adaptation (LoRA)

We implement LoRA [Hu et al., 2022] by replacing targeted `nn.Linear` layers with `LoRALinear` wrappers. Each wrapper freezes the original weight matrix W and adds a trainable low-rank decomposition:

$$h = Wx + \frac{\alpha}{r}(xA)B \quad (6)$$

where $A \in \mathbb{R}^{d_{\text{in}} \times r}$ is initialized with small random values, $B \in \mathbb{R}^{r \times d_{\text{out}}}$ is initialized to zero, r is the rank (default 16), and α is a scaling factor (default 32). We target the query, key, value, attention output, intermediate dense, and output dense projection layers in all attention heads, reducing trainable parameters from $\sim 124\text{M}$ to $\sim 590\text{K}$ (0.48% of total).

We also combine LoRA with SMART regularization. This combination initially exhibited mode collapse (100% “yes” predictions, $\approx 37\%$ accuracy on the imbalanced Quora dev set). A targeted hyperparameter search revealed that the default regularization strengths ($\lambda_s = 1.0$, $\mu = 0.5$) over-constrain LoRA’s limited parameter set. Reducing to $\lambda_s = 0.5$, $\mu = 0.25$ yields a balanced prediction distribution (61% no, 39% yes) comparable to the full-model baseline, while retaining SMART’s robustness properties with $< 1\%$ of trainable parameters.

3.4 Enhanced Sonnet Generation

Beyond the baseline autoregressive generation, we implement improvements across data augmentation, training, decoding, and structural enforcement.

Auxiliary Data Augmentation. The 143 training sonnets provide limited exposure to Shakespearean verse. We augmented training with $\sim 2,000$ 14-line verse chunks extracted from Shakespeare’s complete plays via Project Gutenberg (`fetch_shakespeare.py`). These auxiliary examples are interleaved with sonnet batches during training at a reduced weight of $0.3\times$, giving the model broader exposure to iambic pentameter verse patterns without overpowering the sonnet-specific signal.

SMART Regularization for Sonnet Training. We integrate SMART into the sonnet fine-tuning loop using the same adversarial smoothness and Bregman proximal framework as the classification tasks, with a `forward_from_embeddings` method on the sonnet model to support the adversarial pass. An EMA model provides the proximal anchor.

Decoding Improvements. We implement several complementary decoding strategies:

- **Temperature scaling:** Logits are divided by temperature $\tau = 0.75$ before softmax.
- **Top- k /top- p sampling:** Top- $k = 50$ filtering followed by nucleus sampling with $p = 0.9$.
- **Token-level repetition penalty:** Already-generated tokens have their logits divided (if positive) or multiplied (if negative) by a factor of 1.2.
- **N -gram repetition ban:** Any token that would continue a trigram already seen in the generated text is hard-blocked (logit set to $-\infty$), preventing looping.
- **EOS suppression:** The end-of-sequence token is blocked until the target number of continuation lines has been reached, preventing the model from terminating sonnets prematurely at ~ 10 – 13 lines.
- **Rhyme-aware boosting:** When generating the final word of a line that should rhyme (per the ABAB CDCD EFEF GG scheme), tokens that produce rhyming words (via the CMU Pronouncing Dictionary) receive a gentle logit boost of $+1.0$. The boost is kept mild to nudge rhyme without overriding coherence.

- **Multi-candidate selection:** For submission, we generate 8 candidates per prompt and select the best using a composite score of average log-probability, line-count correctness, malformation penalties, and rhyme reward.
- **Flash attention:** We implement an SDPA fast-path in our attention module using PyTorch’s scaled-dot-product-attention, which can leverage fused kernels (including FlashAttention) when running on supported GPUs, enabling training on longer sequences with reduced memory overhead.

Structural Enforcement. We enforce minimum words per line (7) via logit penalties on the newline token, and boost the newline token when approaching the target line count to encourage clean line breaks. The 14-line structure is enforced by the EOS suppression and line-count-based stopping condition.

Scaling to Larger Models. To improve performance, we add support for GPT-2 Large (774M, 1280-dim, 36 layers) and GPT-2 XL (1.5B, 1600-dim, 48 layers) for sonnet generation, enabled by LoRA (rank 16, $\alpha = 32$), gradient accumulation (effective batch size $2 \times 4 = 8$), and mixed-precision training.

3.5 Training Improvements

We incorporate several training techniques across all tasks:

- **Layer-wise learning rate decay (LLRD):** Lower transformer layers receive exponentially smaller learning rates ($\text{lr}_i = \text{lr} \cdot 0.9^{L-i}$), preserving low-level features while allowing upper layers and the task head to adapt more aggressively.
- **Label smoothing:** Cross-entropy loss uses smoothing factor 0.1 to improve calibration and reduce overfitting to hard labels.
- **Cosine LR schedule with warmup:** Linear warmup for the first 10% of steps, followed by cosine decay to 10% of peak LR.
- **Gradient accumulation:** Effective batch sizes are increased by accumulating gradients over multiple mini-batches before stepping.
- **Gradient clipping:** Maximum gradient norm of 1.0 across all tasks.
- **Early stopping:** Training halts after 3 epochs without dev accuracy improvement.
- **Automatic Mixed Precision (AMP):** Float16 forward passes with dynamic loss scaling for $\sim 2\text{--}3\times$ GPU speedup on A10G [Micikevicius et al., 2018].

4 Experiments

4.1 Data

We use three datasets as specified in the project handout:

- **Stanford Sentiment Treebank (SST):** 11,855 sentences with 5-class sentiment labels (negative to positive). Split into train/dev/test.
- **CFIMDB:** Movie reviews with binary sentiment. Split into train/dev/test.
- **Quora Question Pairs:** 141,506 training pairs, 20,215 dev pairs, 40,431 test pairs. Binary paraphrase labels. Evaluated by accuracy.
- **Shakespeare Sonnets:** 143 training sonnets and 12 held-out test sonnets. Each sonnet consists of 14 lines following the ABAB CDCD EFEF GG rhyme scheme. The model is conditioned on the first 3 lines and generates the rest. We additionally extract $\sim 2,000$ verse chunks (14 lines each) from Shakespeare’s complete plays via Project Gutenberg as auxiliary training data. This was evaluated by chrF score on the leaderboard; our final score was 41.8.

4.2 Evaluation Metrics

For sentiment analysis and paraphrase detection, we use **accuracy** as the primary metric, consistent with the leaderboard evaluation. For sonnet generation, the leaderboard uses **chrF** [Popović, 2015], a character-level n -gram F-score measuring closeness to Shakespeare’s original language.

4.3 Experimental Details

Classification experiments use GPT-2 Small (124M, 768-dim, 12 layers, 12 heads); sonnet generation uses GPT-2 Large (774M, 1280-dim, 36 layers) with LoRA. All pretrained weights are loaded from HuggingFace. GPU training is performed on NVIDIA A10G (24GB) via Modal cloud compute.

Sentiment Analysis. Last-linear-layer: $\text{lr}=10^{-3}$, 10 epochs, batch size 32 (SST) / 8 (CFIMDB), dropout 0.3. Full-model: $\text{lr}=10^{-5}$, 5 epochs. Both use LLRD with decay 0.9 and label smoothing 0.1.

Paraphrase Detection. Full-model fine-tuning: $\text{lr}=10^{-5}$, 10 epochs, batch size 64, weight decay 0.01, patience 3. Cosine schedule with 10% warmup. SDPA fast attention enabled. SMART (full-model): $\lambda_s = 1.0$, $\epsilon = 10^{-2}$, $\mu = 0.5$, $\sigma = 10^{-3}$, $\eta = 0.1$, batch size 32 (auto-capped from 64 to prevent OOM from 4 forward passes per step). LoRA+SMART (best config from hyperparameter search): $\lambda_s = 0.5$, $\mu = 0.25$, batch size 64, all other hyperparameters identical to full-model SMART.

Sonnet Generation. LoRA fine-tuning of GPT-2 Large: $\text{lr}=10^{-5}$, up to 20 epochs, batch size 2 with gradient accumulation over 4 steps (effective batch 8), weight decay 0.01, patience 10, mixed-precision (fp16). LoRA rank 16, $\alpha = 32$, targeting query, key, value, attention output, intermediate dense, and output dense layers. SMART regularization: $\lambda_s = 1.0$, $\mu = 0.5$. Auxiliary Shakespeare verse weighted at $0.3\times$. Cosine schedule with 10% warmup. Generation: temperature 0.75, top- p 0.9, top- k 50, repetition penalty 1.2, 8 candidates per prompt.

4.4 Results

Table 1: Sentiment analysis results (dev accuracy). DFP baselines from the project handout.

Mode	SST Test Acc	CFIMDB Dev Acc	DFP Baseline
Last-linear-layer	46.6%	84.5%	46.2% / 86.1%
Full-model	54.1%	97.6%	51.3% / 97.6%

Table 2: Paraphrase detection results (dev accuracy).

Method	Test Accuracy	Notes
Full model	89.1%	Best result

For paraphrase detection, our baseline model with LLRD, cosine scheduling, and label smoothing achieves 89.1% dev accuracy. This substantially exceeds a naive majority-class baseline and demonstrates that the cloze-style formulation is effective.

Table 3: Sonnet generation results (test accuracy).

Method	Test Accuracy	Notes
Full model (LoRA + SMART)	41.160	Best result

5 Analysis

5.1 Sentiment Analysis: Per-Class Error Analysis

When looking at our sentiment analysis results, we found that our full-model SST results involved substantial variation across the five sentiment classes. We computed a per-class confusion matrix on the dev set (1,101 examples) and found that the model performs well on classes 1 (negative, 87.5% accuracy) and 4 (very positive, 61.2%), but struggles severely with class 2 (neutral, 9.2%) and class 0 (very negative, 18.7%).

Table 4: SST 5-class confusion matrix (full-model fine-tuning, dev set). Rows are gold labels, columns are predictions. Per-class accuracy on the diagonal.

Gold ↓ / Pred →	0	1	2	3	4	Acc.
0 (very neg.)	26	111	0	2	0	18.7%
1 (negative)	12	253	8	16	0	87.5%
2 (neutral)	6	117	21	78	7	9.2%
3 (positive)	2	47	8	163	59	58.4%
4 (very pos.)	0	8	0	56	101	61.2%

The neutral class is the most interesting failure mode: of 229 neutral examples, only 21 are correctly identified. This suggests the model has not learned a robust representation of neutrality and instead treats the sentiment space as roughly bipolar, absorbing ambiguous examples into the adjacent negative or positive categories. On the binary CFIMDB task, by contrast, the full model reaches 97.6%, suggesting that GPT-2’s pretrained representations handle coarse polarity well compared to more fine-grained sentiment distinctions.

5.2 Paraphrase Detection: Error Analysis by Lexical Overlap

Our best model achieves 89.1% dev accuracy with precision 0.83 and recall 0.88 ($F1 = 0.86$) on the Quora paraphrase dev set. Recall that the Quora dataset is made up of question pairs, and our task is to predict whether they’re asking the same thing. Therefore, to understand where the model succeeds and fails, we stratified errors by the word-level overlap Jaccard similarity between each sentence pair. We found that pairs with low lexical overlap ($\text{Jaccard} < 0.3$) achieved 92.6% accuracy across 21,563 examples. At the other extreme, high-overlap pairs ($\text{Jaccard} \geq 0.6$) achieved 87.7% (6,240 examples), suggesting that when sentences share most words, the model can rely on near-identical phrasing. The hardest region was medium overlap ($\text{Jaccard } 0.3\text{--}0.6$), where accuracy drops to 83.8% across 12,626 examples. Inspecting false positives in this range reveals a consistent pattern: the model is fooled by pairs that ask about the same topic with slightly different intent (e.g., “How can I start writing short stories?” vs. “How can I write easy short stories?”—labeled non-paraphrase but predicted paraphrase). This suggests the cloze-style approach relies partly on lexical cues and struggles with semantic equivalence requiring deeper reasoning.

5.3 SMART Regularization: Porting from BERT to GPT-2

Porting SMART to GPT-2 required two key recalibrations. The perturbation radius $\epsilon = 10^{-5}$ is negligible for GPT-2’s unnormalized embeddings (magnitude ± 5); increasing to $\epsilon = 10^{-2}$ restored meaningful adversarial regularization. The Bregman EMA anchor also needs per-step updates rather than per-epoch—per-epoch updates left the anchor near the pretrained initialization throughout training, penalizing all learning and causing collapse to 40.2%. Combining LoRA with SMART required further reducing regularization strengths ($\lambda_s = 0.5$, $\mu = 0.25$); the original values caused mode collapse (100% “yes” predictions). The fix reflects a general principle: LoRA’s limited parameter budget ($<1\%$ of total) is more easily over-constrained, so regularization must be proportionally weaker.

5.4 Sonnet Generation: Qualitative Evaluation of Selected Examples

We qualitatively examined generated sonnet continuations across the dev set (12 sonnets). The model reliably produces the target 14-line structure and maintains Shakespearean vocabulary (“thee,” “thy,” “dost,” “hath”) in the first 4–6 continuation lines. However, after line 7 or 8, outputs tend to drift from verse into prose-like commentary, sometimes producing text such as stage directions or critical annotations rather than verse.

This degradation is consistent with a known limitation of autoregressive generation: as the model generates further from the conditioning prompt, it accumulates distributional drift. Data augmentation with $\sim 2,000$ auxiliary verse chunks from Shakespeare’s plays mitigated this but did not eliminate it entirely. The rhyme-aware decoding produces end-rhymes on 60–70% of lines where the ABAB/GG scheme requires them, based on manual inspection. Multi-candidate selection (8 candidates scored by log-probability, line-count, and rhyme) substantially improves output quality by filtering degenerate samples, but the fundamental limitation remains the model’s inability to plan ahead for rhyme and meter across multiple lines.

6 Conclusion

We implemented GPT-2 from scratch and explored its application to sentiment analysis, paraphrase detection, and sonnet generation. Our main findings are:

1. **Training procedure matters.** Our strongest paraphrase result (89.1%) came from careful training including: LLRD, label smoothing, cosine scheduling, and SDPA-enabled larger batches.
2. **LoRA and SMART are composable with careful tuning.** Combining LoRA ($<0.5\%$ trainable parameters) with SMART initially caused mode collapse, but reducing regularization strengths ($\lambda_s = 0.5$, $\mu = 0.25$) restored balanced predictions. The key insight is that LoRA’s limited capacity requires proportionally weaker regularization.
3. **Sonnet generation benefits from layered constraints.** No single decoding technique suffices; the combination of EOS suppression, n -gram bans, rhyme boosting, minimum word enforcement, and multi-candidate selection addresses complementary failure modes. Data augmentation from auxiliary Shakespeare verse and scaling to GPT-2 Large via LoRA further improved quality.

Limitations: Our main limitations are that the SST sentiment classifier struggles with fine-grained distinctions—neutral and very negative examples are frequently miscategorized—and the sonnet generator reliably maintains verse register for only the first half of each continuation before drifting into prose. Both suggest directions for future work: ordinal-aware loss functions for sentiment, and longer-context or planning-based decoding for generation.

Team Contributions

Jake Klosowski built the paraphrase component, conducted iterative optimization passes across the core training modules, built the initial sonnet generation pipeline, integrated team contributions in a major consolidation commit, and produced the project poster and presentation assets. **Kevin Stephen** implemented LoRA, iterated on the sonnet generation pipeline including rhyme-aware boosting, auxiliary Shakespeare data augmentation, repetition penalties, multi-candidate selection, and EOS suppression, and diagnosed and fixed the 14-line sonnet bug. **Alex Wurm** implemented SMART regularization end-to-end, identified and fixed the EMA per-step and perturbation scale bugs causing training collapse, ran LoRA+SMART hyperparameter search, and consolidated the final sonnet generation pipeline.

References

- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shanen Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.
- Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, 2020.
- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2019.
- Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations*, 2018.
- Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Preprint*, 2018.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.