

Efficient Fine-Tuning of GPT-2 via Low-Rank Adaptation (LoRA)

Stanford CS224N Default Project

Shang Gao

Civil and Environmental Engineering
Stanford University
shanggao@stanford.edu

Min Zhang

Civil and Environmental Engineering
Stanford University
mzhang76@stanford.edu

Shang Gao

Civil and Environmental Engineering
Stanford University
shang698@stanford.edu

Abstract

We study parameter-efficient fine-tuning of GPT-2 using Low-Rank Adaptation (LoRA) across both classification and generation tasks. We compare last-layer tuning, full-model fine-tuning, and LoRA-based adaptation on sentiment analysis, paraphrase detection, and Shakespearean sonnet generation. Our experiments show that LoRA achieves the best overall performance while using significantly fewer trainable parameters. On paraphrase detection, our model reaches an accuracy of 0.911 on the test set. For structured text generation, combining LoRA with data augmentation and model scaling achieves a chrF score of 44.571 on the test set for Shakespearean sonnet generation. These results demonstrate that LoRA provides an effective and efficient approach for adapting pretrained language models across diverse NLP tasks.

1 Key Information to include

- Mentor: Alisa Levin.
- External Collaborators (if you have any): None.
- Sharing project: No.
- Late Days (if you are using any):

Team member	Late days remaining
Shang Gao	6
Min Zhang	6
Shang Gao	6
Total pooled	18
Team size	3
Total late days applied for the report	9

2 Introduction

Large pretrained language models have become a core paradigm in natural language processing [1, 2, 3]. However, full fine-tuning is computationally expensive for large models, motivating parameter-efficient adaptation methods.

Low-Rank Adaptation (LoRA) freezes the original model weights and injects trainable low-rank updates into Transformer layers [4]. This greatly reduces the number of trainable parameters while maintaining strong performance.

In this project, we study efficient adaptation of GPT-2 on three tasks: sentiment analysis, paraphrase detection, and sonnet generation. We compare last-layer tuning, full-model fine-tuning, and LoRA-based adaptation, and analyze the trade-off between parameter efficiency and task performance. In particular, we examine whether LoRA can achieve competitive or better results while updating only a small fraction of the model parameters.

3 Related Work

3.1 Pretrained Language Models

Pretrained language models based on the Transformer architecture have achieved strong performance across a wide range of NLP tasks [1, 2, 3]. Encoder-based models such as BERT and subsequent improvements including RoBERTa and DeBERTa further demonstrated the importance of large-scale pretraining and improved contextual representations for language understanding tasks [5, 6, 7].

3.2 Parameter-Efficient Fine-Tuning

Parameter-efficient fine-tuning methods adapt large language models by updating only a small subset of parameters while keeping most pretrained weights frozen. Representative approaches include adapters, prompt tuning, and LoRA [8, 9, 4]. For generation tasks, Direct Preference Optimization (DPO) provides an efficient objective for aligning language models with desired outputs [10].

3.3 Sentence Understanding and Text Generation Tasks

Sentence-level benchmarks such as sentiment analysis and paraphrase detection are widely used to evaluate semantic understanding in language models [11, 12]. In addition, autoregressive language models have shown strong capabilities in text generation tasks. Poetry generation is particularly challenging because it requires both semantic coherence and adherence to structural constraints such as rhyme schemes and line patterns [13]. This motivates our investigation of efficient adaptation methods across both classification and structured generation tasks.

4 Approach

We fine-tune a pre-trained GPT-2–style Transformer for multiple NLP tasks, including sentiment classification, paraphrase detection, and sonnet generation.

Classification Head Mode: We implement 4 modes for classifications: (1) *Last-linear*: Directly maps the final token to the output logits using a single linear layer, producing the final yes/no classification result. (2) *Last-linear (pooling)*: Applies a mean pooling operation over all token representations and then feeds the pooled vector into a linear classifier [14]. (3) *MLP-head*: Replaces the single linear layer with a multi-layer perceptron to provide greater expressive capacity for classification. (4) *Full-head*: Uses the pretrained embedding matrix as the output head, assigning a probability to every token by similarity, and predicts “yes” or “no” based on their respective probabilities [15].

Low-Rank Adaptation (LoRA): To enable parameter-efficient adaptation, we implement LoRA [4] by freezing the pre-trained Multi-Head Attention weights and injecting trainable low-rank matrices into the Query and Value projections. The forward pass is:

$$h = W_0x + \frac{\alpha}{r}BAx \quad (1)$$

where W_0 is frozen, $A \in \mathbb{R}^{r \times d}$ is Kaiming-initialized, and $B \in \mathbb{R}^{d \times r}$ is zero-initialized. This greatly reduces parameters while keeping adaptability. We experimented with three variants:

1. *QV LoRA*: Applies LoRA only to Query and Value projections.
2. *MLP LoRA*: Applies LoRA to the feed-forward MLP layers.

3. *Full LoRA*: Applies LoRA to both attention (Q/V) and MLP layers for maximal flexibility.

Prompt Engineering: We design task-specific prompts to guide the model’s predictions, helping it focus on relevant information and reduce ambiguity [16]. We unify the training and testing prompt and ask the model to explicitly output only yes or no. The unified prompt is "*Question 1: {s1}*"
"*Question 2: {s2}*"
"*Are these questions asking the same thing? Answer yes or no:*"

Learning Rate Scheduling: We use a cosine decay schedule, where the learning rate starts at a maximum value, gradually decreases following a cosine curve, and approaches zero at the end of training to stabilize updates and improve convergence [17].

Label Smoothing: We apply label smoothing with a factor of 0.1, assigning 0.9 probability to the correct class and distributing 0.1 among others, which prevents overconfidence and improves generalization [18].

Data Augmentation: For the paraphrase classification task, we swap the order of sentences A and B during training. During inference, we compute predictions for both AB and BA and take the average to enhance robustness and accuracy [19].

Direct Preference Optimization (DPO): To improve generation quality, we apply DPO [10], which optimizes the policy π_θ against a reference model π_{ref} without a separate reward model:

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)} \right) \right] \quad (2)$$

where y_w, y_l are preferred and dispreferred completions. This offers a stable and efficient objective for fine-tuning on preference-guided tasks like sonnet generation.

5 Experiments

We evaluate LoRA and other different extensions on the GP2-based model with different datasets. We ran our experiments on one Nvidia RTX5090 GPU and one Nvidia L4 GPU.

5.1 Sentiment Analysis

5.1.1 Data and Evaluation Method

We use the Stanford Sentiment Treebank (SST) and CFIMDB datasets. SST contains 11,855 sentences, partitioned into train (8,544), dev (1,101), and test (2,210) sets. CFIMDB consists of 2,434 movie reviews, with train (1,701), dev (245), and test (488) examples.

The models are evaluated based on classification accuracy on both the SST and CFIMDB datasets. For SST, we measure the granular sentiment prediction performance across various sentence lengths, while for CFIMDB, we assess the robustness of binary sentiment classification on longer movie reviews.

5.1.2 Experimental Details and Results

The parameters and accuracy results for sentiment analysis are summarized in Table 1.

Table 1: Summary of Parameters and Results for Sentiment Analysis Task

Method	Learning Rate	Dropout Prob	Batch Size	Epoch	Trainable Parameters	SST Accuracy	CFIMDB Accuracy
Last Linear	1e-3	0.1	32	10	3.85k	0.461	0.886
Full Model	1e-5	0.1	32	10	125M	0.511	0.984

5.2 Paraphrase Detection

5.2.1 Data and Evaluation Method

We utilize the Quora Question Pairs dataset, containing over 400,000 question pairs. For our study, we use the provided split with train (141,506), dev (20,215), and test (40,431) examples.

We evaluate the paraphrase detection task using classification accuracy on the Quora Question Pairs dataset. We compare the accuracy of different fine-tuning strategies, including last-linear-layer tuning, full-model fine-tuning, and LoRA adaptation.

5.2.2 Experimental details and results

Our evaluation process for sonnet generation was conducted in 5 progressive stages.

Stage 1: Baseline LoRA and Prompt Engineering. We compare different types of LoRA configurations and prompt designs using GPT-2, including applying LoRA to different components of the model as well as full LoRA. We also compare the original prompt with a unified prompt. The results show that full LoRA and QV LoRA both perform well, and the unified prompt is beneficial for training.

Table 2: Stage 1: Comparison of LoRA and Prompt
(Base mode: gpt2, LR: 1e-3, Epoch: 5)

	Mode	LoRA	Prompt	Params	Acc
base	last-linear	none	original		
A1	full-model	MLP	original	1.54k	0.721
A2				738k	0.823
A3				296k	0.858
A4		QV	unified	125M	0.831
A5				296k	0.869
A6				1.33M	0.866

Stage 2: Learning Rate scheduling. We compare training with and without learning rate scheduling under the same model configuration. The results show that using a cosine learning rate scheduling is beneficial for training.

Table 3: Stage 2: Comparison of Learning Rate scheduling
(Base mode: gpt2, Mode: last-linear, LoRA: QV, Prompt: unified, LR: 1e-3, Epoch: 5)

	LR scheduling	Params	Acc
B1	none	296k	0.869
B2	cosine	296k	0.878

Stage 3: Modes. We compare several model of classification head designs, including pure last-linear, last-linear with pooling, MLP head, and full head architecture. The results show that the full head is the most effective design among the tested options.

Table 4: Stage 3: Comparison of Modes
(Base mode: gpt2, Prompt: unified, LR: 1e-3, Epoch: 5)

	Mode	LoRA	LR scheduling	Params	Acc
base	last-linear	QV	none		
C1	last-linear(pooling)	full	cosine	296k	0.869
C2				296k	0.872
C3				590k	0.869
C4				294k	0.871
C5	last-linear(pooling)	full	cosine	1.33M	0.890
C6	full-head	full	cosine	1.33M	0.892

Stage 4: Label Smoothing. We evaluate the effect of label smoothing under the selected model configuration. The results show that label smoothing does not bring performance gains and slightly degrades the results.

Table 5: Stage 4: Label Smoothing
(Base mode: gpt2, Prompt: unified, LR: 1e-3, Epoch: 5)

	Mode	LoRA	Smooth	LR scheduling	Params	Acc
base	last-linear	QV	0.0	none		
D1					296k	0.869
D2			0.1		296k	0.869
D3	last-linear(pooling)	full		cosine	1.33M	0.890
D4	last-linear(pooling)	full	0.1	cosine	1.33M	0.888

Stage 5: Data Augmentation. We evaluate the effect of applying data augmentation during training while keeping the model configuration unchanged. The result indicates that data augmentation helps enhance the model’s generalization ability. We conduct the final experiments using the larger backbone model.

Table 6: Stage 5: Comparison of Data Augmentation
(Mode: full-head, LoRA: full, Prompt: unified, LR scheduled: consine)

	Data Augmentation		Learning Rate	Epoch	Params	Acc
E1	gpt2	none	1e-3	5	1.33M	0.892
E2	gpt2	yes	1e-3	5	1.33M	0.901
E3	gpt2-large	none	2e-4	10	6.64M	0.911
E4	gpt2-large	yes	2e-4	10	6.64M	0.914

On the leaderboard, the best accuracy is **0.914** on the development dataset and **0.911** on the test dataset.

5.3 Sonnet Generation

5.3.1 Data

Our foundational dataset consists of 131 original Shakespearean sonnets. The generation task requires the model to produce a 14-line continuation adhering to the English sonnet schema: iambic pentameter, ABAB CDCD EFEF GG rhyme scheme, and two-space indentation for the final couplet.

To address data scarcity, we augmented the training set to 417 sonnets using LLM-generated synthetic data in two stages: rule-based prompts (Stage 1, +100 sonnets) and few-shot prompting with dev exemplars (Stage 2, +186 sonnets). For DPO, we constructed 4,150 preference pairs using true continuations as positives and structurally corrupted samples as negatives.

5.3.2 Evaluation Method

We evaluate against a held-out dev set of 12 sonnets using chrF [20], which computes an F-score over character n -gram matches. chrF is better suited than word-level metrics (e.g., BLEU) for Shakespearean poetry, where archaic vocabulary and spelling variants cause low word-level overlap even for high-quality outputs. However, as a surface-form metric it cannot assess structural compliance (iambic pentameter, rhyme scheme) or semantic coherence, so we complement it with qualitative inspection of generated samples.

5.3.3 Experimental Details and Results

Our evaluation process for sonnet generation was conducted in 4 progressive stages.

Stage 1: Baseline LoRA. We initially applied LoRA to GPT-2 Small using the original 131 Shakespearean sonnets. Regardless of the rank or learning rate applied to the Attention layers, all configurations converged to a ceiling of ~ 42 chrF, indicating that data scarcity was the primary bottleneck.

Table 7: Stage 1: LoRA Baseline (GPT-2 Small, 131 Original Sonnets)

Exp	Rank (r)	Learning Rate	Dev chrF
Exp 3	8	1e-4	42.05
Exp 2	8	1e-3	42.18
Exp 7	32	5e-4	42.04

Stage 2: Generation-Stage Optimization. A grid search over 72 decoding configurations showed that N-gram repetition penalty is critical, while keeping temperature=1.0 (no probability mass restriction).

Table 8: Stage 2: Best Decoding Parameter Configurations

Temperature	Nucleus (Top- p)	N-gram Penalty	Dev chrF
1.0	0.85	3	42.18
1.0	0.95	3	42.15
1.0	0.90	3	42.06

Stage 3: Data Augmentation & Model Scaling. To break the baseline ceiling, we augmented our training set up to 417 sonnets using LLMs. This additional data delayed overfitting, enabling successful scaling to GPT-2 Large and pushing peak performance past 44 chrF.

Table 9: Stage 3: Scaling Data Size (GPT-2 Large, LoRA Attention $r = 32$)

Exp	Dataset Size	Best Epoch	Dev chrF
Exp 29	131 sonnets (Original)	12	43.53
Exp 19	181 sonnets (Rule Prompt)	12	43.89
Exp 28	417 sonnets (Few-shot)	21	44.201

Stage 4: Expanded LoRA (Attention vs. Full-Layer Adaptation). We extended LoRA from *Attention-only* (Q, V) to *Full-Layer* (Q, K, V, output, MLP). High rank ($r = 32$) across all layers caused overfitting; reducing to $r = 8$ with stronger weight decay acted as a regularizer, yielding 43.94 chrF—competitive with but not exceeding Attention-only ($r = 32$) at 44.201 chrF.

Table 10: Stage 4: Attention-only vs Full-Layer LoRA on GPT-2 Large (417 Sonnets)

Exp	LoRA Target	Rank (r)	Learning Rate	Dev chrF
Exp 28	Attention-only	32	5e-5	44.201
Exp 26	All Layers	32	5e-5	42.63
Exp 30	All Layers	8	1e-5	43.94

Stage 5: Structural Alignment via DPO. Despite high chrF scores (lexical overlap), pure SFT models frequently ignored sonnet formatting constraints (e.g., overgeneration, missing indentations). Applying DPO over our SFT models penalized these formatting errors. While it incurred an “alignment tax” on vocabulary mimicry, DPO successfully reduced overgeneration and structural repetition loops.

Table 11: Stage 5: DPO Alignment Trade-offs (SFT vs. Post-DPO Checkpoints)

Exp	Training Stage	Base Model Type	Dev chrF
Exp 28	Pure SFT Baseline	GPT-2 Large (Attn $r = 32$)	44.201
Exp 31	Style DPO	Exp 28 Checkpoint	43.60

Summary of Best Results. The best configuration (Exp 28) uses GPT-2 Large with Attention-only LoRA ($r = 32$) on 417 sonnets, trained with $LR=5 \times 10^{-5}$, weight decay 0.01, and decoded with temperature=1.0, top- $p=0.85$, N-gram penalty=3. This achieves a dev chrF of **44.201** and a test chrF of **43.13**.

On the leaderboard, the best chrF score is **44.201** on the dev dataset and **44.571** on the test dataset.

6 Analysis

6.1 Paraphrase Detection

The experiments provide several insights into factors affecting paraphrase detection performance. As shown in Table 2, parameter-efficient fine-tuning with LoRA significantly improves the baseline, achieving higher accuracy than both last-layer tuning and full model fine-tuning while using far fewer trainable parameters. In addition, the unified prompt helps by providing a consistent task format for the model.

To better understand why the full mode performs worse than the LoRA method, we examine the loss curve of the full mode training. We observe that after several training steps, the dev loss exceeds the training loss, indicating overfitting which negatively impacts dev accuracy (Figure 1).

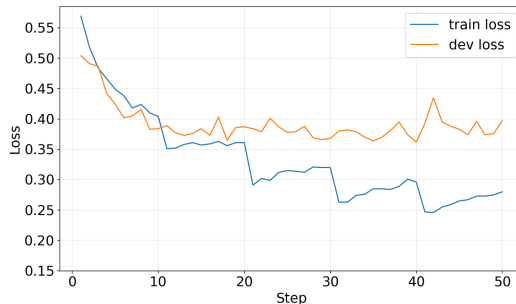


Figure 1: Loss curve of full model (gpt2)

Training strategy and model architecture also influence performance. As shown in Table 3 and Table 4, applying cosine learning rate scheduling helps stabilize training, while the full-head architecture performs better than the simple last-linear head. One possible reason is that the full head can better utilize the rich semantic features already encoded in the pretrained embeddings. Instead of relying on a single linear projection, the deeper head introduces additional transformations that may help capture more complex interactions between the two sentence representations, making it easier for the model to compute semantic similarity.

However, label smoothing does not improve results in this task. As indicated in Table 5, smoothing the labels slightly weakens performance, possibly because paraphrase detection is a clear binary classification problem.

Finally, Table 6 suggests that both model scaling and data augmentation improve generalization. A larger backbone model provides stronger representation capacity, while data augmentation exposes the model to more diverse training examples.

6.2 Sonnet Generation

The 131-sonnet dataset created a hard ceiling of 42.0 chrF for GPT-2 Small regardless of LoRA rank or learning rate, confirming data scarcity as the primary bottleneck. Synthetic augmentation and model scaling broke this ceiling: as shown in Figures 2 and 3, both larger models and larger datasets yield consistent chrF gains, with the optimal supervised configuration (GPT-2 Large, Attention-only LoRA $r = 32$, 417 sonnets) peaking at 44.201 chrF.

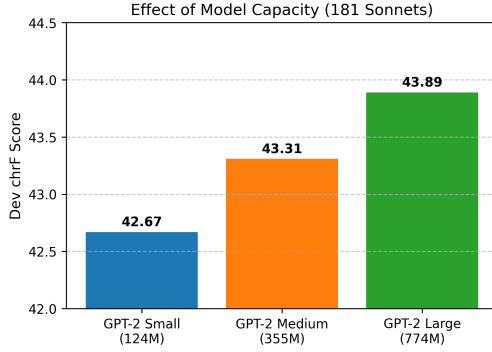


Figure 2: Comparison of GPT-2 base models

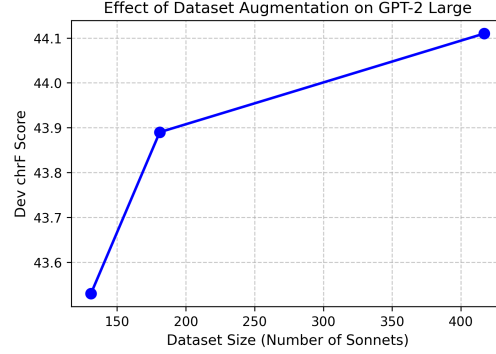


Figure 3: Effect of Dataset Augmentation

For Full-Layer LoRA, high rank ($r = 32$) caused overfitting and degraded performance (42.63 chrF), while low rank ($r = 8$) with strong weight decay acted as a regularizer and recovered quality (43.94 chrF), illustrating the importance of bottleneck constraints when expanding LoRA coverage.

While pure SFT achieved the highest raw chrF, qualitative inspection revealed structural failures—overgeneration (15–20 lines), repetitive loops, and missing couplet indentation. DPO alignment traded a small lexical overlap penalty for structural reliability: Exp 31 (DPO on Exp 28) scored 43.60 chrF with consistent 14-line output and correct couplet formatting.

7 Conclusion

In this work, we investigated parameter-efficient fine-tuning of GPT-2 using Low-Rank Adaptation (LoRA) across both classification and generation tasks. We compared last-layer tuning, full-model fine-tuning, and LoRA-based adaptation on sentiment analysis, paraphrase detection, and Shakespearean sonnet generation. Our experiments show that LoRA achieves the best overall performance while using significantly fewer trainable parameters. For structured text generation, model scaling and data augmentation further improve generation quality. Overall, our results demonstrate that LoRA provides an effective and efficient approach for adapting pretrained language models to diverse NLP tasks.

Team contributions

Shang Gao (shanggao@stanford.edu): I completed the implementation and training of the sonnet generation.

Min Zhang (mzhang76@stanford.edu): I completed the implementation and training of the base model for sentiment analysis, as well as the implementation and training for paraphrase detection.

Shang Gao (shang698@stanford.edu): I completed the synthetic data generation and analyze the impact of generated data on model performance.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- [4] 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, 2021.
- [5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [6] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.
- [7] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention, 2021.
- [8] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR, 2019.
- [9] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059. Association for Computational Linguistics, 2021.
- [10] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024.
- [11] 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 (EMNLP)*, pages 1631–1642, 2013.
- [12] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3982–3992. Association for Computational Linguistics, 2019.
- [13] Yufei Tian and Nanyun Peng. Zero-shot sonnet generation with discourse-level planning and aesthetics features. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3587–3597. Association for Computational Linguistics, 2022.
- [14] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.
- [15] Ofir Press and Lior Wolf. Using the output embedding to improve language models. *arXiv preprint arXiv:1608.05859*, 2017.
- [16] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners, 2020.

- [17] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.
- [18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2016.
- [19] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2020.
- [20] 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. Association for Computational Linguistics, 2015.