

Parameter-Efficient Adaptation of GPT-2 across Discriminative and Generative Tasks

Stanford CS224N Default Project

Junran Jia

Department of Statistics
Stanford University
junran@stanford.edu

Xianya Fu

Department of Statistics
Stanford University
xianya@stanford.edu

Abstract

Adapting large pretrained language models to downstream tasks often requires full fine-tuning, which updates all model parameters and can be computationally expensive. Parameter-efficient adaptation methods aim to reduce this cost by modifying only a small subset of parameters while preserving performance. In this work, we systematically evaluate adaptation strategies for GPT-2 across both discriminative and generative tasks, including sentiment analysis, paraphrase detection, and Shakespearean sonnet generation. We compare full fine-tuning with parameter-efficient methods such as LoRA, ReFT, and BitFit, and further explore combining LoRA with Direct Preference Optimization (DPO) for generation. Our results show that parameter-efficient approaches can achieve performance comparable to full fine-tuning on classification tasks while updating less than 1% of model parameters. In paraphrase detection, LoRA reaches 0.901 development accuracy on GPT2-Medium and achieves 0.90 test accuracy while updating only 0.44% of parameters. For sonnet generation, LoRA-based models substantially outperform the zero-shot baseline, while ReFT achieves the best performance with 43.27 dev chrF while updating only 0.04% of model parameters. Overall, these results demonstrate that parameter-efficient adaptation can deliver strong performance across both discriminative and generative tasks while substantially reducing training cost.

1 Key Information

- Mentor: Chenglei Si

2 Introduction

Pretrained language models such as GPT-2 achieve strong performance across many natural language processing tasks. However, adapting these models to downstream tasks often requires full fine-tuning, which updates all model parameters and can be computationally expensive. Parameter-efficient fine-tuning methods aim to reduce this cost by updating only a small subset of parameters while preserving performance.

In this project, we study adaptation strategies for GPT-2 across both discriminative and generative tasks. We compare last-linear-layer and full-model fine-tuning for sentiment analysis, and evaluate full fine-tuning, LoRA, ReFT, BitFit on paraphrase detection and Shakespearean sonnet generation. We further explore LoRA combined with Direct Preference Optimization (DPO) for sonnet generation.

These tasks provide complementary evaluation settings: sentiment analysis tests sentence-level classification, paraphrase detection requires semantic reasoning between sentence pairs, and sonnet generation evaluates structured text generation. Our experiments show that full fine-tuning provides a strong baseline for classification tasks, but parameter-efficient methods such as LoRA achieve

comparable performance while updating less than 1% of model parameters. In the paraphrase detection task, LoRA nearly matches full fine-tuning on GPT2-Small and slightly surpasses it when applied to GPT2-Medium. For sonnet generation, parameter-efficient adaptation methods also perform competitively, with ReFT achieving the best results. Overall, these findings demonstrate that parameter-efficient adaptation can deliver strong performance across both discriminative and generative tasks while substantially reducing training cost.

3 Related Work

Large pretrained language models such as GPT-2 are commonly adapted to a wide range of downstream tasks after pretraining [1]. Among these tasks, text classification problems such as sentiment analysis and paraphrase detection have been widely studied. For example, sentiment analysis benchmarks such as the Stanford Sentiment Treebank (SST) are commonly used to evaluate how pretrained language models adapt to downstream classification tasks [2].

Paraphrase detection is an important classification task used in applications such as duplicate question identification and information retrieval. The Quora Question Pairs (QQP) dataset is a widely used benchmark for this problem and is also included in the GLUE benchmark suite [3, 4, 5]. Although most approaches rely on encoder-based architectures such as BERT on QQP [6], earlier work also explored neural sentence-pair models for semantic matching [7]. More recently, researchers have begun studying decoder-style language models for this task. For example, Han et al. show that a LLaMA-based model can achieve higher accuracy on QQP [8]. However, existing studies mainly focus on encoder models or large decoder architectures, and relatively little work has examined how smaller autoregressive models such as GPT-2 perform on paraphrase detection under parameter-efficient adaptation.

Beyond classification tasks, large language models have also been applied to generative tasks such as poetry generation. Early approaches relied on rule-based or template-driven systems that explicitly encoded constraints such as rhyme and meter [9, 10, 11, 12]. Later work combined neural language models with structural constraints by generating candidate lines that satisfy rhyme or metrical patterns and ranking them using neural models [13, 14, 15]. More recent studies use pretrained language models fine-tuned on poetic corpora to imitate specific poetic styles [16], sometimes guided by multimodal prompts derived from image captioning systems [17]. Despite these advances, generating structured poetry such as Shakespearean sonnets remains challenging because models must maintain long-range coherence while satisfying strict constraints such as rhyme schemes and meter.

4 Approach

We evaluate several fine-tuning strategies for GPT-2 across three tasks: sentiment analysis, paraphrase detection, and sonnet generation. For sentiment analysis, we compare full-model fine-tuning with last-layer fine-tuning, which updates only the classification head. For paraphrase detection, we evaluate zero-shot inference, full fine-tuning, and several parameter-efficient methods, including LoRA, BitFit, and ReFT. For the sonnet generation task, we compare zero-shot generation, full fine-tuning, LoRA, and ReFT, and further investigate the effect of combining LoRA with Direct Preference Optimization (DPO). Our implementation builds on the GPT-2 training framework provided in the CS224N Default Final Project instructions [18], with additional modules implemented for LoRA, ReFT, BitFit, and DPO experiments.

4.1 Base Model and Baseline Methods

4.1.1 Base Model

We use GPT-2 [1], a decoder-only Transformer language model trained with an autoregressive objective:

$$P(x) = \prod_{t=1}^T P(x_t \mid x_{<t}; \theta),$$

where θ denotes the model parameters and each token is predicted conditioned on previous tokens. GPT-2 consists of stacked Transformer decoder layers with masked multi-head self-attention and

feed-forward networks. Tokens are represented using learned token and positional embeddings, which are processed through the Transformer layers to produce contextualized hidden states. The final hidden state is projected to the vocabulary space to obtain next-token probabilities for generation or downstream prediction.

4.1.2 Baselines

We consider two reference baselines for comparison:

1. Zero-shot (Pretrained) Model. In this setting, the pretrained GPT-2 model is applied directly to downstream tasks without any task-specific training or parameter updates. Predictions are produced using the representations and language patterns learned during large-scale pretraining. This baseline measures the inherent capability of the pretrained model to generalize to new tasks without adaptation.

2. Full Fine-Tuning. In this setting, all pretrained parameters θ are updated during downstream training using task-specific objectives. Fine-tuning allows the model to adapt its internal representations to the target task and dataset. This represents the standard approach for adapting pretrained language models and serves as the primary supervised baseline in our experiments.

4.2 Model Adaptation Methods

In addition to full fine-tuning, we evaluate parameter-efficient approaches that adapt a pretrained model by introducing a small number of trainable parameters while keeping most weights frozen.

4.2.1 LoRA (Low-Rank Adaptation)

LoRA [19] injects trainable low-rank updates into existing linear layers. For a weight matrix $W \in \mathbb{R}^{d \times k}$, LoRA replaces the update with a low-rank decomposition:

$$W' = W + BA$$

where $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ with rank $r \ll \min(d, k)$. During training, the original weights W remain frozen and only A and B are learned. This significantly reduces the number of trainable parameters while preserving model capacity.

4.2.2 ReFT (Representation Fine-Tuning)

ReFT [20] modifies internal representations instead of weight matrices. Given a hidden representation h_l at layer l , ReFT applies a learnable low-rank intervention:

$$h'_l = h_l + UV^\top h_l$$

where $U, V \in \mathbb{R}^{d \times r}$. This allows the model to adapt task-specific representations without directly updating the original Transformer parameters.

4.2.3 BitFit (Bias Fine-Tuning)

BitFit [6] adapts a pretrained model by updating only the bias parameters while keeping all other weights frozen. Given a linear layer with parameters

$$y = Wx + b,$$

BitFit keeps the weight matrix W fixed and only learns updates to the bias term b . This results in a very small number of trainable parameters while still allowing the model to adjust its activations for the downstream task. Despite its simplicity, prior work has shown that updating only bias parameters can achieve competitive performance on several NLP tasks.

Model	SST Dev Accuracy	CFIMDB Dev Accuracy
Last Linear Layer	0.464	0.878
Full Model	0.501	0.967

Table 1: Accuracy (%) for sentiment classification on SST and CFIMDB.

4.2.4 Low-Rank Adaptation (LoRA) with Direct Preference Optimization (DPO)

For the sonnet generation task, we further apply Direct Preference Optimization (DPO) [21] on top of a LoRA-adapted model. DPO trains the model using preference pairs (y^+, y^-) where y^+ is a preferred output and y^- is a rejected one. The objective encourages higher probability for preferred outputs relative to a frozen reference model:

$$\mathcal{L}_{DPO} = -\log \sigma \left(\beta \left(\log \frac{\pi_{\theta}(y^+|x)}{\pi_{ref}(y^+|x)} - \log \frac{\pi_{\theta}(y^-|x)}{\pi_{ref}(y^-|x)} \right) \right)$$

where β controls the strength of preference optimization. This method improves generation quality without requiring reinforcement learning.

5 Experiments

We evaluate different GPT-2 adaptation methods on three tasks from DFP_Instructions [18]: sentiment analysis, paraphrase detection, and sonnet generation. These tasks span both discriminative and generative settings, allowing comparison of fine-tuning strategies across diverse task types. We briefly describe the setup for each task and present the results below.

5.1 Datasets

For sentiment analysis, we evaluate on the Stanford Sentiment Treebank (SST) [2] and the CFIMDB dataset. For paraphrase detection, we use the Quora Question Pairs (QQP) dataset [3]. For sonnet generation, we train and evaluate on a corpus of Shakespearean sonnets provided in DFP_Instructions [18].

5.2 Evaluation Metrics

For classification tasks (sentiment analysis and paraphrase detection), we report classification accuracy, which measures the proportion of correctly predicted labels.

For sonnet generation, we use the chrF metric, which measures character-level similarity between the generated continuation and the reference sonnet. Following the evaluation protocol in DFP_Instructions [18], the model is conditioned on the first three lines of a sonnet and the generated continuation is compared with the ground-truth text.

5.3 Sentiment Analysis

5.3.1 Experimental details

We predict the sentiment polarity of a sentence or review. GPT-2 is fine-tuned on labeled datasets (SST and CFIMDB) using the final hidden state of the last token as a sentence representation, followed by a linear classification head.

5.3.2 Results

We compare *Last Linear Layer Fine-Tuning*, which trains only the classification head, with *Full Model Fine-Tuning*, which updates all GPT-2 parameters. Performance is measured by development accuracy. Full fine-tuning achieves higher accuracy on both datasets (Table 1).

5.4 Paraphrase Detection

5.4.1 Experimental Details

In the paraphrase detection task, we reformulate the original binary classification problem as a cloze-style language modeling task using the GPT-2 framework provided by the instructions. Specifically, each sentence pair (s_1, s_2) is converted into a natural language prompt. The model receives the full prompt as input and predicts the probability distribution of the next token. Instead of adding an additional classification head or MLP, we directly use the GPT-2 language modeling output layer for prediction.

For each prompt, the model produces the hidden state of the final position, which is projected to the vocabulary space to obtain logits over all tokens. We then extract the logits corresponding to the label tokens “yes” and “no” as candidate scores. The label with the higher score is selected as the prediction, and cross-entropy loss is computed over these two logits during training.

To ensure consistency across training and evaluation, we keep the prompt template identical for the train, dev, and test sets. Each sentence pair is formatted as: Is “s1” a paraphrase of “s2”? Answer “yes” or “no”: . During data loading, we apply light preprocessing (lowercasing and punctuation normalization). Sentence pairs are then combined into this prompt format in `collate_fn` and encoded using the GPT-2 tokenizer.

We also exploit the inherent symmetry of the paraphrase task for simple data augmentation. Since paraphrase relations are symmetric, if (s_1, s_2) is a paraphrase, (s_2, s_1) should share the same label. We randomly swap 50% of training pairs and merge them with the original dataset to construct a symmetric augmented dataset, which helps reduce potential bias toward sentence order. An additional experiment demonstrating the effectiveness of this augmentation strategy is provided in Appendix A.

For fine-tuning, we implement and compare several training strategies within the same GPT-2 backbone. Full-model fine-tuning serves as the primary baseline by updating all GPT-2 parameters end-to-end. To reduce training cost, we further explore parameter-efficient fine-tuning methods including LoRA, ReFT, and BitFit, aiming to maintain performance while significantly reducing the number of trainable parameters. We train all models with AdamW using a batch size of 64, and all experiments are conducted on a single NVIDIA A100 GPU.

5.4.2 Results

Table 2 presents a comparison of different fine-tuning methods on the development set for the paraphrase detection task. As a baseline, GPT2-Small without any fine-tuning (zero-shot) achieves an accuracy of only 0.368, indicating that a pretrained language model cannot directly solve this task without task-specific adaptation. In contrast, full model fine-tuning significantly improves performance, reaching a development accuracy of 0.900. This result highlights the importance of updating model parameters to adapt the pretrained model to the downstream task.

Among parameter-efficient fine-tuning methods, LoRA achieves a development accuracy of 0.894 while using only 1,179K trainable parameters, accounting for approximately 0.94% of the total model parameters. This performance is close to that of full fine-tuning, while also reducing the training time per epoch. BitFit further reduces the number of trainable parameters to 103K (0.082%), but its accuracy drops to 0.852. ReFT uses only 0.025% of the model parameters, achieving the highest training efficiency, but its accuracy is lower at 0.804. Finally, when the model size is increased to GPT2-Medium, LoRA reaches a development accuracy of 0.901. Despite training only 0.44% of the parameters, it slightly surpasses the full fine-tuning result of GPT2-Small. The GPT2-Medium (LoRA) model achieves a test accuracy of 0.90 on the official leaderboard, consistent with its development performance.

5.5 Sonnet Generation

5.5.1 Experimental details

In the sonnet generation task, we evaluate the model’s ability to generate Shakespearean-style poetry. GPT-2 is fine-tuned on a corpus of Shakespeare’s sonnets using the autoregressive language modeling objective and prompted with the opening lines of a sonnet to generate the remaining lines.

Model & Method	# Trainable Params	% of Model	Best Epoch	Time/Epoch	Dev Accuracy
GPT2-Small (Zero-shot)	/	/	/	/	0.368
GPT2-Small (Full FT)	125,030K	100	9	24m55s	0.9
GPT2-Small (LoRA)	1,179K	0.94	15	18m54s	0.894
GPT2-Small (ReFT)	30K	0.025	15	10m57s	0.804
GPT2-Small (BitFit)	103K	0.082	8	17m35s	0.852
GPT2-Medium (Zero-shot)	/	/	/	/	0.37
GPT2-Medium (LoRA)	1,572K (/357,445K)	0.44	15	37m30s	0.901

Table 2: Comparison of fine-tuning methods for paraphrase detection on the development set. *Hyperparameters.* Full-model fine-tuning: $\text{lr} = 5\text{e-}5$. LoRA: $\text{lr} = 1\text{e-}4$, rank $r = 8$, and $\alpha = 16$. ReFT: $\text{lr} = 5\text{e-}5$, intervention layer = 8 to 12, and rank $r = 4$. BitFit: $\text{lr} = 1\text{e-}3$. GPT2-Medium with LoRA: $\text{lr} = 1\text{e-}4$, rank $r = 8$, and $\alpha = 16$. The best checkpoint is selected based on development accuracy.

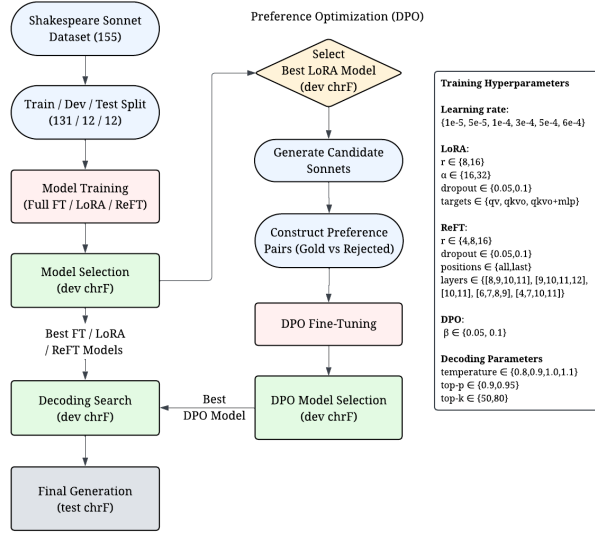


Figure 1: Experimental pipeline for the sonnet generation task

Figure 1 illustrates the training and evaluation pipeline. All experiments are conducted on a single NVIDIA T4 GPU. The Shakespeare sonnet dataset (155 poems) is split into train (131), development (12), and test (12) sets. Models are trained on the training split using full fine-tuning, LoRA, or ReFT, and the best checkpoint for each method is selected based on chrF performance on the development set. After model selection, we perform a grid search over decoding hyperparameters on the development set to determine the best generation settings before producing final outputs on the test set.

For preference-based optimization, we first generate candidate sonnets using the best LoRA model. These samples are paired with ground-truth sonnets from the training data to form preference pairs, where the gold sonnets serve as preferred outputs and generated samples with low chrF scores (typically 20-35) are used as rejected outputs. The model is then further fine-tuned using the Direct Preference Optimization (DPO) objective. The best DPO checkpoint is again selected based on development chrF before decoding hyperparameter search. During decoding, we enforce the 14-line sonnet structure by conditioning on the first three lines and generating the remaining 11 lines. We tune temperature, top- p , and top- k sampling parameters. Repetition penalties were also explored but did not improve development chrF.

5.5.2 Results

Table 3 compares different adaptation methods for Shakespearean sonnet generation across three GPT-2 model sizes. Overall, parameter-efficient adaptation substantially improves performance relative to zero-shot prompting. For GPT2-small, LoRA+DPO achieves the best development score (43.13 chrF), outperforming both LoRA (42.63) and ReFT (41.07) while updating only 0.47% of

Model & Method	# Trainable Params	% of Model	Train Time / Epoch (s)	dev chrF
GPT2-small (ZS)	/	/	0	40.75
GPT2-small (FT)	125,719K	100	7	42.02
GPT2-small (ReFT)	98K	0.08	4	41.07
GPT2-small (LoRA)	590K	0.47	5	42.63
GPT2-small (LoRA + DPO)	590K	0.47	23	43.13
GPT2-med (ZS)	/	/	0	38.20
GPT2-med (FT)	357,577K	100	23	42.02
GPT2-med (ReFT)	131K	0.04	12	43.27
GPT2-med (LoRA)	1,573K	0.44	15	43.13
GPT2-med (LoRA + DPO)	1,573K	0.44	54	43.04
GPT2-large (ZS)	/	/	0	40.55
GPT2-large (FT)	778,619K	100	53	42.21
GPT2-large (ReFT)	164K	0.02	29	42.58
GPT2-large (LoRA)	2,949K	0.38	35	42.84
GPT2-large (LoRA + DPO)	2,949K	0.38	94	42.53

Table 3: Comparison of fine-tuning methods for sonnet generation on the development set.

model parameters. For GPT2-medium, however, ReFT performs best, reaching 43.27 chrF while updating only 131K parameters (0.04% of the model). This configuration also achieves the strongest test performance on the official leaderboard (42.62 chrF), compared with 40.78 for the zero-shot model.

Scaling the base model does not consistently improve performance after adaptation. Although GPT2-large has substantially more parameters, its best development result (42.84 with LoRA) remains slightly below GPT2-medium ReFT. These results suggest that, under the small training set of 131 sonnets, the choice of adaptation method may matter more than raw model size.

Finally, preference-based optimization incurs significantly higher computational cost. LoRA+DPO increases training time per epoch (e.g., 23s vs. 5s for GPT2-small and 54s vs. 15s for GPT2-medium) while providing only marginal improvements over standard LoRA.

6 Analysis

Paraphrase detection. To better understand where the models perform well and where they struggle, we first analyze their confusion matrix. As shown in Table 4, Full FT and GPT2-Medium(LoRA) exhibit nearly identical distributions of true positives and true negatives, indicating very similar decision patterns. In contrast, the weaker models perform worse on both positive and negative predictions. Overall, the models show a slight tendency to predict no, although the predictions remain relatively balanced. Therefore, a more fine-grained subgroup analysis is needed to better understand the sources of model errors, and lexical overlap provides a natural perspective for further investigation.

To analyze whether the models rely excessively on surface lexical similarity, we select three representative models and evaluate their performance across subsets grouped by lexical overlap between sentence pairs. Specifically, we tokenize each sentence and remove common stopwords, then compute the Jaccard overlap between the sets of remaining content words as a measure of surface similarity. Based on this score, the development set is divided into three subsets: low, mid, and high overlap.

As shown in Table 5, all models perform best on the low-overlap subset, while accuracy drops noticeably on high-overlap examples. For instance, Full FT achieves an accuracy of approximately 0.967 on the low-overlap subset but only 0.852 on the high-overlap subset, while ReFT shows an even larger decrease of about 26% from low to high overlap. This suggests that the most challenging cases are not sentence pairs with very different wording, but those with high lexical similarity yet subtle semantic differences. In addition, the Medium+LoRA model achieves the best overall performance among the compared models, meaning that a larger backbone can provide advantages when distinguishing highly similar sentence pairs.

Model	TP	TN	FP	FN
Full FT	0.323	0.577	0.055	0.045
LoRA	0.326	0.568	0.063	0.043
ReFT	0.287	0.518	0.114	0.081
BitFit	0.315	0.537	0.094	0.053
GPT2-M	0.324	0.577	0.054	0.044

Table 4: Normalized confusion statistics.

Overlap	FT	ReFT	GPT2-M
Low	0.967	0.911	0.965
Mid	0.871	0.690	0.873
High	0.852	0.672	0.858

Table 5: Accuracy by lexical overlap.

Sonnet generation. One notable pattern is the strong performance of ReFT at medium and large model scales. Unlike LoRA, which modifies weight matrices through low-rank updates, ReFT directly intervenes on intermediate representations. For structured generation tasks such as Shakespearean sonnets—where models must maintain stylistic consistency across multiple lines—representation-level interventions may provide more targeted control over the generation process. In contrast, Direct Preference Optimization (DPO) does not consistently improve performance. Although LoRA+DPO yields the best result for GPT2-small, it slightly underperforms LoRA for GPT2-medium and GPT2-large. This may be partly due to the construction of the preference data: rejected samples are generated by the LoRA model itself and often have chrF scores around 20-35. As a result, preference pairs may emphasize large lexical mismatches rather than subtle stylistic improvements.

Finally, the small performance differences between methods suggest that chrF alone may not fully capture sonnet quality. While chrF measures lexical similarity to reference continuations, it does not evaluate key poetic properties such as rhyme scheme, meter, or thematic coherence. Future evaluations could incorporate metrics for rhyme accuracy, syllable or stress-pattern consistency (approximating iambic pentameter), and semantic coherence across lines, along with human judgments of poetic quality.

7 Conclusion & Future Work

In this work, we systematically study adaptation strategies for GPT-2 across three tasks: sentiment analysis, paraphrase detection, and Shakespearean sonnet generation. Our results show that parameter-efficient fine-tuning methods such as LoRA can achieve performance comparable to full fine-tuning on classification tasks while updating less than 1% of model parameters. For generative tasks, parameter-efficient approaches also perform competitively, with ReFT achieving the best results for sonnet generation. Overall, these findings demonstrate that carefully designed parameter-efficient adaptation methods can provide strong performance across both discriminative and generative tasks while substantially reducing training cost.

Several directions remain for future work. First, exploring larger backbone models (e.g., GPT-2 Large) could further clarify how model scale interacts with parameter-efficient adaptation methods on tasks such as paraphrase detection. Second, alternative preference construction strategies or more informative negative samples may improve the effectiveness of preference-based optimization methods such as DPO for generation tasks. Third, more systematic exploration of prompt design and decoding strategies may improve performance for both classification and generation settings. Finally, evaluation of creative generation tasks could be strengthened through additional metrics—such as rhyme accuracy, meter consistency, and semantic coherence—as well as human evaluation to better capture poetic quality beyond surface-level lexical similarity.

Team contributions

Both members collaborated on implementing the base GPT-2 model and conducting experiments for the sentiment analysis task.

Junran Jia. Led the sonnet generation task, including training and fine-tuning models across all methods, designing the experimental pipeline, and writing the corresponding sections of this report. **Xianya Fu.** Led the paraphrase detection task, including running and analyzing experiments across different fine-tuning methods on GPT2, and writing the corresponding sections of this report.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019. Accessed: 2024-11-15.
- [2] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew 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*, 2013.
- [3] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *EMNLP*, 2018.
- [4] Tanja Vrbancic and Ana Meštrović. A corpus for paraphrase identification from quora. *Information*, 11(5):241, 2020.
- [5] Ankit Sharma et al. Quora question pairs dataset: A benchmark for duplicate question detection. *PLOS ONE*, 2019.
- [6] Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *ACL*, 2022.
- [7] A. Mishra and A. Jain. Duplicate question detection using deep neural networks. *IEEE Access*, 10:110000–110010, 2022.
- [8] Yu Han et al. Exploring llms for quora question pair detection. *PLOS ONE*, 2025.
- [9] Pablo Gervás. Wasp: Evaluation of different strategies for the automatic generation of spanish verse. 2002.
- [10] Ruli Manurung, Graeme Ritchie, and Henry Thompson. Towards a computational model of poetry generation. Technical report, University of Edinburgh, School of Informatics, 2000.
- [11] Hugo Gonçalo Oliveira. Poetryme : a versatile platform for poetry generation. 2012.
- [12] Tony Veale. Less rhyme, more reason: Knowledge-based poetry generation with feeling, insight and wit. In *Proceedings of the Fourth International Conference on Computational Creativity (ICCC 2013)*, pages 152–159, Sydney, Australia, 2013. Computational Creativity Group.
- [13] Marjan Ghazvininejad, Xing Shi, Yejin Choi, and Kevin Knight. Generating topical poetry. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1191, Austin, Texas, November 2016. Association for Computational Linguistics.
- [14] Jey Han Lau, Trevor Cohn, Timothy Baldwin, Julian Brooke, and Adam Hammond. Deep-speare: A joint neural model of poetic language, meter and rhyme. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1948–1958, Melbourne, Australia, July 2018. Association for Computational Linguistics.
- [15] John Benhardt, Peter Hase, Liuyi Zhu, and Cynthia Rudin. Shall i compare thee to a machine-written sonnet? an approach to algorithmic sonnet generation, 2019.
- [16] Stanley Xie. Deep poetry: Word-level and character-level language models for shakespearean sonnet generation, 2017.
- [17] Doga Nalci, Zehra Bilici, and Saliha Yesilyurt. William shakesblake 2.0: Fine-tuned language models for hybrid poetic style generation, 2025. Technical report.
- [18] Stanford CS224N Course Staff. Cs224n default final project instructions. <link-if-available>, 2025. Stanford University.
- [19] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

- [20] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: Representation finetuning for language models, 2024.
- [21] 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.

A Effect of Data Augmentation

To evaluate the impact of the proposed symmetric data augmentation, we compare full model fine-tuning with and without augmentation. The augmentation is constructed by swapping the order of the two sentences in each paraphrase pair, and it is applied randomly to 50% of the training data.

Figure 2 shows the development accuracy across training epochs for full model fine-tuning with the two settings. The model trained with augmentation consistently achieves slightly higher accuracy in the early and middle stages of training. It also reaches a higher peak dev accuracy (0.900) compared to the model trained without augmentation (0.896).

Although the improvement is modest, the results suggest that exposing the model to both sentence orders helps it better capture the symmetric nature of paraphrase relations. Importantly, the augmentation does not introduce label noise and maintains stable training behavior.

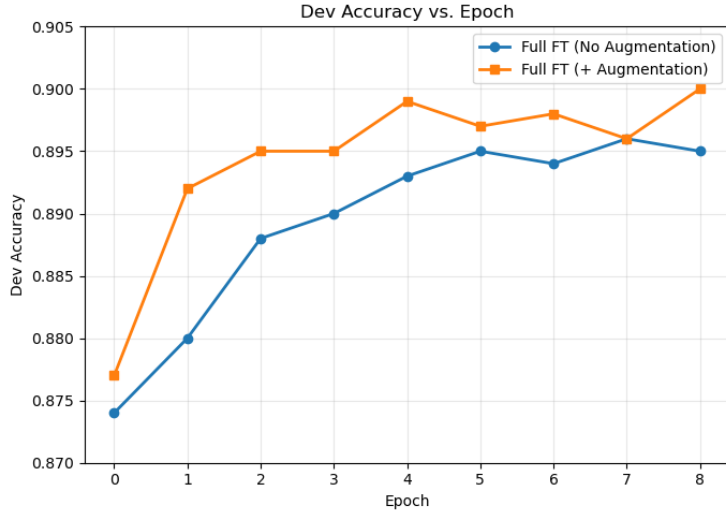


Figure 2: Development accuracy across epochs with and without symmetric data augmentation.