

Leveraging GPT-2 for Multiple Downstream NLP Tasks: Classification and Generation

Stanford CS224N Default Project

Davi Ferreira Veronese
Department of Political Science
Stanford University
davifv@stanford.edu

Abstract

Fine-tuning language models offers a promising avenue for transferring pretrained representations to new tasks, though performance can depend strongly on training choices, such as hyperparameters and decoding strategies. In this project, I fine-tune GPT-2 for several tasks: sentiment classification, paraphrase detection using the Quora dataset, and Shakespearean sonnet generation. In addition to standard fine-tuning, I experiment with several extensions, including smoothness regularization, Low-Rank Adaptation (LoRA), and sampling methods for generation, and I also study the impact of key hyperparameters such as the learning rate and batch size. The results demonstrate that full-model fine-tuning performs well across the tasks considered, indicating that GPT-2 representations transfer effectively even with relatively limited task-specific training data. The model achieves strong performance on the classification tasks, and my error analysis shows that most remaining mistakes arise in cases involving subtle semantic distinctions or implicit paraphrases. For the generation task, the model produces text that broadly follows the structure and stylistic patterns of Shakespearean sonnets. While more complex decoding strategies introduced instability and repetition, reverting to a simpler decoding procedure produced more stable generations. These findings suggest that pretrained language models can adapt effectively to both discriminative and generative tasks through fine-tuning, though generation quality remains sensitive to training and decoding choices.

1 Key Information

- Mentor: Nevin George
- External Collaborators: No external collaborators.
- Sharing project: No.
- Late Days: No.

2 Introduction

Recent progress in natural language processing has shown that large-scale pretrained language models can learn rich representations of linguistic structure that transfer effectively to downstream tasks. One widely adopted strategy is fine-tuning, where a pretrained model is adapted to a specific task by updating some or all of its parameters using labeled data.

Earlier approaches relied on feature-based transfer learning, where a pretrained model produced fixed representations that were then used as inputs to task-specific architectures. While effective, this approach limits the ability of the model to adapt its internal representations to the new task. Fine-tuning addresses this limitation by allowing the parameters of the pretrained model to be optimized

jointly for the downstream objective. This paradigm has proven highly successful in modern NLP systems [1].

In this project, I explore the effectiveness of fine-tuning large transformer-based language models across several tasks. Specifically, I fine-tune GPT-2 for multiple tasks included in the CS224N default project, which include sentiment classification, paraphrase detection, and Shakespearean sonnet generation.

The base model used in all experiments is GPT-2, a decoder-only transformer architecture based on the self-attention framework [2]. GPT-2 is pretrained on large-scale web text using a language modeling objective, enabling it to learn general-purpose linguistic representations that transfer effectively across multiple NLP tasks [3, 4].

Building on this pretrained model, I fine-tune GPT-2 for the tasks included in the CS224N default project. In addition to implementing the baseline fine-tuning procedures, I explore several extensions designed to improve model robustness and generation quality. These extensions include adversarial smoothness regularization for classification tasks, parameter-efficient fine-tuning using LoRA adapters, and improved decoding strategies for text generation.

The idea of this approach is to evaluate how a single pretrained transformer model can be adapted to both discriminative and generative NLP tasks through task-specific fine-tuning and targeted modifications to the training and decoding procedures. For classification tasks such as paraphrase detection, GPT-2 is augmented with a classification head and trained using supervised learning. For text generation, the model is fine-tuned as an autoregressive language model to produce Shakespearean-style sonnets. Finally, I analyze the behavior of the resulting models through both quantitative results and qualitative inspection of generated outputs, highlighting which extensions improve performance and which fail to provide meaningful gains.

3 Related Work

The architecture underlying the models is the transformer, which uses self-attention mechanisms proposed by Vaswani et al.[2] to model dependencies between tokens in a sequence. GPT-2 is a decoder-only transformer architecture with masked self-attention heads trained using large-scale language modeling objectives [3, 4].

These papers demonstrate that large-scale unsupervised pretraining can produce representations that transfer effectively to a wide range of downstream tasks via fine-tuning. Fine-tuning approaches were further popularized by models such as BERT, which uses bidirectional pretraining that allows task-specific fine-tuning for multiple NLP tasks [1].

My extensions are also motivated by recent work on improving the robustness of fine-tuning for pretrained language models. Because fine-tuning can lead to overfitting, producing models that do not generalize well to new data. I follow the work by Jiang et al.[5], who propose a smoothness-inducing adversarial regularization method that encourages models to produce stable predictions under small perturbations of the input representations. The method introduces perturbations during training and penalizes large changes in the model’s predictions. This regularization is intended to promote smoother decision boundaries and can improve the robustness and generalization performance of fine-tuned models.

Additionally, I explore parameter-efficient fine-tuning techniques, specifically Low-Rank Adaptation (LoRA), which allows large pretrained models to be adapted to new tasks by training small low-rank matrices instead of updating the entire set of model parameters [6].

Finally, for the sonnet generation task, the decoding strategy is relevant. Beam search may produce repetitive and degenerate outputs in neural language models [7]. To mitigate this issue, I experiment with a sampling-based decoding strategy that combines temperature scaling with top- k and nucleus (top- p) sampling as proposed by Holtzman et al.[7]. Temperature skews the probability distribution towards high probability events, while top- k sampling restricts the candidate vocabulary to the k most probable tokens. Nucleus sampling selects the smallest set of tokens whose cumulative probability exceeds a threshold p .

4 Approach

This section describes how GPT-2 was fine-tuned for the tasks in the CS224N default project and presents the extensions implemented to improve model performance. The approach combines standard fine-tuning of a pretrained transformer with additional techniques designed to improve robustness and generation quality.

4.1 Base GPT-2 Model

All experiments are built on the pretrained GPT-2 architecture [3, 4]. GPT-2 is a decoder-only transformer trained with self-attention for next-token prediction. Given an input token sequence

$$x = (x_1, \dots, x_T),$$

the model produces contextualized hidden representations

$$H = (h_1, \dots, h_T), \quad h_t \in \mathbb{R}^d,$$

where h_t denotes the contextual embedding of token x_t after the transformer layers. For classification tasks, I use the representation of the last token h_T as the embedding.

4.2 Fine-Tuning for Classification

To adapt GPT-2 to classification tasks, I add a linear prediction head on top of the final hidden representation:

$$z = Wh_T + b,$$

where $W \in \mathbb{R}^{K \times d}$ and K is the number of classes. Model parameters are optimized using the cross-entropy loss

$$\mathcal{L}_{CE} = -\frac{1}{B} \sum_{i=1}^B \log \frac{\exp(z_{i,y_i})}{\sum_{k=1}^K \exp(z_{i,k})}.$$

I employ the Adam optimizer for optimization and evaluate two fine-tuning options. First, only the final classification layer is trained while keeping the GPT-2 parameters fixed, providing a baseline. Second, the entire GPT-2 model is fine-tuned. Additionally, since this is the default project, the baseline results provided in the assignment are used for comparison.

4.3 Paraphrase Detection

For paraphrase detection I fine-tune GPT-2 on the Quora Question Pairs dataset [8]. Each example consists of two questions (s_1, s_2) and a binary label indicating whether the questions are paraphrases. The two questions are concatenated and passed to GPT-2, and the final token representation is used by a linear classification head to predict the binary label.

4.4 Smoothness Regularization Extension

To improve robustness, I implement a smoothness-inducing regularization method inspired by Jiang et al. [5]. The objective augments the standard cross-entropy loss with a smoothness regularization term

$$\min_{\theta} L_{CE}(\theta) + \lambda_s R_s(\theta),$$

The smoothness term measures the discrepancy between the model's predictions on the original input and on a perturbed version of the input representation.

$$R_s(\theta) = \frac{1}{n} \sum_{i=1}^n \max_{\|\delta\| \leq \epsilon} l_s(f(x_i + \delta; \theta), f(x_i; \theta)).$$

Here δ denotes a small adversarial perturbation applied to the input embeddings, bounded by $\|\delta\| \leq \epsilon$. As discrepancy function $l_s(\cdot, \cdot)$, I use the symmetric KL divergence

$$l_s(P, Q) = D_{KL}(P\|Q) + D_{KL}(Q\|P).$$

This regularization penalizes models whose predictions change significantly under small perturbations of the input representation, encouraging smoother decision boundaries to improve generalization.

4.5 Sonnet Generation

For the sonnet generation task I fine-tune GPT-2 as an autoregressive language model using a corpus of Shakespearean sonnets. Given a prompt consisting of the beginning of a sonnet, the model generates the remaining text token by token according to the conditional distribution

$$p(x_t \mid x_1, \dots, x_{t-1}).$$

Training is performed using next-token prediction with teacher forcing, where the model learns to predict the next token in the sequence given the previous tokens.

4.6 LoRA Fine-Tuning Extension

As an extension, I explore Low-Rank Adaptation (LoRA) as a parameter-efficient alternative to full fine-tuning [6] using PyTorch. LoRA freezes the pretrained model weights and trains only small low-rank adapter matrices added to the transformer layers, reducing the number of trainable parameters.

4.7 Decoding Strategies for Text Generation

Attempting to improve generation quality, I implement several sampling-based decoding techniques designed to balance fluency and diversity.

I first apply **temperature scaling**. Before sampling the next token, the logits are scaled by a temperature parameter τ :

$$p'(w) = \frac{\exp(z_w/\tau)}{\sum_u \exp(z_u/\tau)}.$$

Because $t \in [0, 1)$ skews the distribution towards high probability events [7].

I also implement **top- k sampling**. At each generation step, the candidate vocabulary is restricted to the k tokens with the highest predicted probabilities. Let V_k denote this set. The next token is sampled from the renormalized distribution [7]

$$p'(w) = \frac{p(w)}{\sum_{u \in V_k} p(u)} \quad \text{for } w \in V_k.$$

This prevents the model from selecting extremely unlikely tokens while still allowing diversity in the generated text.

In addition, following Holtzman[7] I use **top- p (nucleus) sampling**, which dynamically selects the smallest set of tokens V_p whose cumulative probability exceeds a threshold p :

$$\sum_{w \in V_p} p(w) \geq p.$$

I then sample from the renormalized distribution over this subset. Compared to top- k , this method adapts the candidate set size depending on the model’s uncertainty.

In my implementation I combine **top- k** and **top- p sampling** sequentially. First, top- k filtering removes very low-probability tokens, and nucleus sampling is then applied to select the final candidate set from which the next token is sampled.

Finally, I apply a **repetition penalty**, which downweights tokens that have already appeared in the generated sequence, and **no-repeat n-gram blocking**, which prevents the model from generating n-grams that have already occurred earlier in the sequence. The idea is that these constraints should help reduce degenerate repetition and improve the coherence of generated sonnets.

5 Experiments

5.1 Data

Experiments were conducted using the datasets provided in the CS224N default project: Stanford Sentiment Treebank (SST)[9], CFIMDB dataset, Quora Question Pairs[8], and Shakespeare Sonnet dataset. The sentiment classification tasks (SST and CFIMDB) require predicting the sentiment polarity of movie reviews. The paraphrase detection task involves predicting whether two questions from the Quora Question Pairs dataset express the same meaning. Finally, the sonnet generation task evaluates the ability of the model to generate Shakespeare-style sonnets conditioned on a prompt.

5.2 Evaluation Method

Different evaluation metrics were used depending on the task.

- Classification tasks were evaluated using accuracy and F1-score.
- Paraphrase detection was evaluated using accuracy, macro F1, and AUC.
- Sonnet generation was evaluated using the character n-gram F-score (chrF) [10], which measures character-level overlap between generated and reference texts.

5.3 General Experimental Details

All experiments were implemented in PyTorch and fine-tuned from the pretrained GPT-2 architecture. We used the base GPT-2 configuration with hidden dimension $d = 768$, 12 transformer layers, and 12 attention heads. I initialized parameters from pretrained GPT-2 weights and subsequently fine-tuned for the downstream tasks. I conducted optimization using the AdamW optimizer. I experimented with several learning rates during preliminary runs.

5.4 Specific Implementations and Results

This section provides additional detail about each specific implementation and summarizes the empirical performance of the fine-tuned GPT-2 models across the tasks in the default project. I report development-set metrics used during model selection as well as final test performance.

5.4.1 Sentiment Classification

I first evaluate GPT-2 on the sentiment classification tasks: the Stanford Sentiment Treebank (SST) and the Counterfactual IMDB (CFIMDB) dataset. I compare two fine-tuning regimes. First, I only use the final classification layer while keeping the pretrained GPT-2 parameters fixed. Second, I fine-tune the entire GPT-2 model. Table 1 reports development-set accuracy for both regimes.

Table 1: Development-set accuracy for sentiment classification under two fine-tuning regimes

Dataset	Last-layer	Full-model
SST	0.461	0.514
CFIMDB	0.861	0.988

Fine-tuning the full GPT-2 model consistently improves performance relative to training only the classification head. The improvement is particularly large for the CFIMDB dataset, suggesting that updating the pretrained representations allows the model to better capture task-specific semantic patterns. The fully fine-tuned model also achieves results similar to (slightly better than) those reported in the baseline provided by the default project’s instructions, which is expected.

5.4.2 Paraphrase Detection

For the paraphrase detection task, I fine-tuned GPT-2 on the Quora Question Pairs dataset. During training, several evaluation metrics were monitored on the development set, including accuracy, macro F1, and AUC. I selected the model using the development-set macro F1.

I experimented with several learning rates in smaller subsamples of the training data, and $\eta = 2 \times 10^{-5}$ provided the most stable training behavior and best development performance. I also experimented with the batch size, and the final batch size was set to 16. Models were trained for up to 5 epochs.

To implement the described smoothness regularization extension, two additional hyperparameters were introduced. $\lambda = 0.1$ controls the strength of the regularization term, and $\epsilon = 10^{-3}$ defines the magnitude of the adversarial perturbation applied to the input embeddings. The perturbation was approximated using one step of gradient ascent on the KL divergence between clean and perturbed predictions.

Table 2 reports the development-set performance of the baseline GPT-2 model. The model achieves strong results across all evaluation metrics, indicating that the pretrained transformer representations adapt well to the paraphrase classification task.

Table 2: Development-set performance for paraphrase detection.

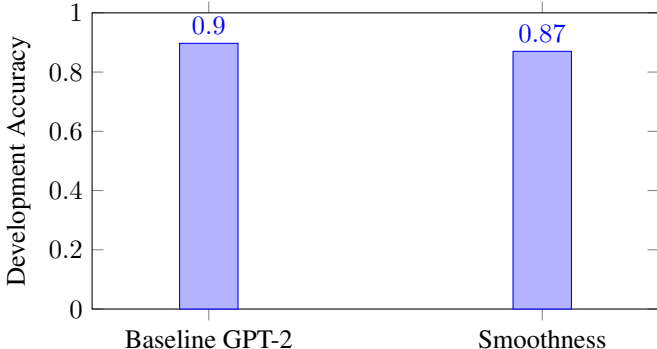
Model	Accuracy	Macro F1	AUC
Baseline GPT-2	0.897	0.891	0.961

Using the checkpoint selected on the development set, **the model achieved a final accuracy of 0.842 on the hidden test set evaluated by the autograder.**

We also evaluated a smoothness regularization extension designed to improve robustness to small perturbations in the input embeddings. While this extension introduces an additional regularization objective during training, it produced slightly lower development accuracy compared to the baseline model.

Figure 1 illustrates the comparison between the baseline GPT-2 model and the smoothness regularization extension in terms of development accuracy. The baseline model achieves an accuracy of 0.897, while the smoothness regularized model achieves 0.870. Although the difference is relatively small, the results suggest that the pretrained GPT-2 representations are already sufficiently stable for this task, limiting the gains from additional smoothness constraints.

Figure 1: Development accuracy for paraphrase detection: baseline GPT-2 vs. smoothness regularization



I expected the smoothness regularization extension to improve performance. However, the results show that the baseline GPT-2 model already performs strongly on this task. I suspect this is because the Quora dataset is relatively large and the pretrained GPT-2 representations are already well structured, reducing the need for additional regularization. In addition, the fine-tuning setup is relatively conservative, which may limit overfitting and therefore reduce the potential gains from smoothness constraints.

5.4.3 Sonnet Generation

For the sonnet generation task, I fine-tuned GPT-2 was fine-tuned on a corpus of Shakespearean sonnets. During evaluation the model receives a prompt and generates the remainder of the sonnet autoregressively. Generation quality was evaluated using the chrF metric[10].

I experimented with the extensions discussed above. In particular, I explored parameter-efficient fine-tuning using LoRA and several decoding strategies designed to improve generation quality. I observed that LoRA did not improve performance, likely because the base GPT-2 model used in this project is relatively small and full fine-tuning is already computationally inexpensive, and LoRa may even deteriorate performance in this setting. Therefore, I do not use LoRa to obtain the final performance metric.

I initially experimented with a more complex decoding strategy after fine-tuning GPT-2 for 15 epochs using a learning rate of 3×10^{-5} with GPU training. During generation, I applied temperature scaling ($\tau = 0.7$), top- k sampling ($k = 50$), and nucleus sampling ($p = 0.9$), together with a repetition penalty (1.35) and no-repeat n -gram blocking with $n = 4$. Generation was constrained to produce up to 14 lines per sonnet with a minimum of 80 and maximum of 220 tokens. However, this more constrained decoding setup did not improve performance. I therefore reverted to a simpler generation procedure. In the final run, GPT-2 was fine-tuned for 8 epochs with a learning rate of 1×10^{-5} and decoded using temperature sampling ($\tau = 0.8$) with nucleus sampling ($p = 0.85$), without additional repetition penalties or n -gram constraints. This configuration approximately recovered the course baseline, achieving a test-set chrF score of 41.093 according to the autograder.

Table 3 reports the test-set performance produced by the autograder:

Table 3: Sonnet generation performance measured using chrF

Model	chrF (test)
Fine-tuned GPT-2	41.093

These results indicate that the fine-tuned model captures some stylistic features of Shakespearean sonnets. Many generated samples begin with plausible Shakespearean phrasing and partially reproduce the poetic structure of the training corpus. However, generation quality remains limited by some repetition that often emerge in later lines of the poems.

I expected that my sampling-based decoding strategy combining temperature scaling with top- k and nucleus (top- p) sampling, in addition to a penalty for repetition and constraints, would reduce repetition and improve generation quality. In practice, however, these modifications did not substantially improve the chrF score. I then concluded that a simpler decoding strategy produced a higher performance.

6 Analysis

I conducted both error analysis and ablation analysis of the models. The ablation analysis revealed that different components of the model had different impacts on performance across tasks. For the sonnet generation task, introducing LoRA did not improve performance. This is likely because the base GPT-2 model is already relatively small and can be effectively fine-tuned directly, so parameter-efficient adaptation does not provide additional benefits in this setting.

For the Quora paraphrase detection task, the learning rate had the largest effect on performance. The batch size also influenced results to some extent, although less strongly. In contrast, the smoothness regularization that I introduced did not improve performance, suggesting that the model was sufficiently stable during fine-tuning and that additional regularization did not provide meaningful gains.

To better understand the model’s behavior, I conducted a qualitative analysis of prediction errors on the Quora development set. I inspected examples of both false positives and false negatives. False positives often occur in cases where the model confuses closely related but distinct semantic concepts. For example, questions such as “What is vaccination?” and “What is in a vaccine?” are treated as paraphrases by the model even though one refers to a general concept and the other refers to a specific composition. Similar errors appear when the model confuses personal statements with

general knowledge claims, such as interpreting “I know about quantum physics” and “We know about quantum physics” as equivalent statements. These examples suggest that the model sometimes struggles to capture subtle semantic distinctions.

False negatives tend to occur when the model fails to recognize that two different expressions refer to the same underlying concept. For example, the model may not recognize that “eating less” and “dieting” describe the same behavior in a given context. Other errors occur when paraphrases involve negation or alternative phrasing of the same question. For instance, questions such as “Why can’t planets be different shapes?” and “Why are planets spherical?” express essentially the same idea but are phrased in opposite forms, which makes them harder for the model to identify as paraphrases. These cases highlight the difficulty of capturing deeper semantic equivalence.

For the sonnet generation task, qualitative inspection suggests that degeneration during decoding was a major limitation in the more complex generation setup. While the model often began generating coherent text, it frequently collapsed into repetitive patterns after several lines, repeating the same words or phrases. I initially attempted to mitigate this problem by introducing sampling strategies and repetition control mechanisms during decoding. However, these constraints did not consistently improve results. In practice, reverting to a simpler decoding procedure reduced these degeneration effects and produced more stable generations and higher quality.

7 Conclusion

This project demonstrates that fine-tuning a pretrained GPT-2 model can achieve strong performance across multiple downstream tasks even with relatively limited task-specific data. In particular, the models performed well on the classification tasks, including sentiment analysis and Quora paraphrase detection, showing that pretrained representations transfer effectively to discriminative settings. An important achievement of this project is showing that full-model fine-tuning substantially improves performance compared to training only the classification layer, highlighting the value of adapting pretrained representations to the target task. I also explored several extensions, including smoothness regularization, LoRA-based adaptation, and sampling methods for generation. While smoothness regularization did not improve performance, it also did not harm the model, suggesting that the base fine-tuning procedure was already stable, and I achieved high accuracy in both development and test sets for the paraphrase task.

At the same time, the project highlights important limitations. The generation task proved more difficult than the classification tasks. Although the model was able to generate text resembling the structure of Shakespearean sonnets, my more complex decoding strategies incorporating sampling and repetition control mechanisms did not consistently improve performance. In practice, a simpler decoding procedure produced more stable outputs, though repetition still appeared in longer generations. These results suggest that generation quality may be constrained by the relatively small GPT-2 model used in the project and the limited training corpus. Future work could explore larger pretrained models or alternative generation strategies to improve long-range coherence.

Team Contributions

This is an individual project. Davi Ferreira Veronese conducted all steps.

References

- [1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
- [3] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Technical Report*, 2018.
- [4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.
- [5] 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. *arXiv preprint arXiv:1911.03437*, 2019.
- [6] 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. Computer Science > Computation and Language.
- [7] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*, 2020.
- [8] Quora. Quora question pairs dataset. <https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs>, 2017.
- [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] 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, Lisbon, Portugal, 2015. Association for Computational Linguistics.