GPT-2

Stanford CS224N Default Project

Aaron Jin, Brandon Bui, Eli Wandless
Department of Computer Science
Stanford University
aaronjin@stanford.edu, bbui@stanford.edu, eliwand@stanford.edu

Abstract

We present a GPT-2-based system that tackles three diverse NLP tasks: sentiment classification, paraphrase detection, and sonnet generation. By fine-tuning GPT-2 on domain-specific datasets, we exploit its pre-trained contextual representations to see strong performance in both classification and generative settings.

For sentiment analysis, we attach a simple linear head to the final token's hidden state and explore two strategies: updating all GPT-2 parameters, or "freezing" the Transformer backbone. For paraphrase detection, we frame the problem as a cloze-style question as we prompt the model to generate "yes" or "no" in response to whether two sentences are paraphrases. This lets GPT-2 use its generative prior while producing concise binary predictions. finally, we utilize top-p nucleus sampling with temperature modulation to generate Shakespearean sonnets. Our experiments demonstrate a working, strong foundation for all three of the listed tasks. Through optimizing hyperparameters, we show that GPT-2's masked multi-head self-attention mechanism, combined with carefully chosen sampling methods and optimization schemes, yields competitive results on this range of tasks that require nuanced language understanding and controlled text generation.

1 Key Information

TA mentor	Josh Singh
External collaborators	No
External mentor	No
Sharing project	No

2 Introduction

Large language models (LLMs) such as GPT-2 have transformed how we approach natural language processing (NLP) tasks such as classification, semantic matching, and text generation. Through pretraining on massive text corpora, these models learn rich contextual representations that can be adapted to a multitude of downstream settings. Despite these advantages, transfering an LLM's general language knowledge to specific tasks often brings up both practical and methodological challenges like selecting an effective fine-tuning strategy and balancing creativity with accuracy in generative contexts.

In this paper, we build GPT-2 for three distinct tasks: sentiment classification (on Stanford Sentiment Treebank and CFIMDB), paraphrase detection (via a cloze-style setup on Quora pairs), and sonnet generation (trained on Shakespearean texts). In each case, we aim to show how careful architectural choices—such as building a classification head on top of GPT-2 or employing top-p sampling with dynamic temperature—allow the model to excel in both discriminative and generative applications. Furthermore, by combining existing research on Transformer architectures and language modeling with targeted fine-tuning, we show that GPT-2 can produce strong results in tasks that demand nuanced language understanding and controlled text generation.

3 Related Work

Our work takes direct influence and builds upon foundational research described in the paper *Language Models are Unsupervised Multitask Learners* by Alec Radford et al., published by OpenAI in 2019 [1]. Here, this paper discusses limitations of traditional supervised NLP systems, proposing that large, unsupervised language models can perform tasks like translation and summarization via zero-shot learning, without explicit fine-tuning.

In their paper, Radford et al. introduced GPT-2, a Transformer-based model with 1.5 billion parameters, which extended beyond GPT-1 with improvements in the model's architecture such as repositioned layer normalization, scaled residual weights, and an expanded context window of 1024 tokens. Their model was trained on WebText, a dataset of 8 million web pages designed to minimize both redundancy and noise. To highlight, GPT-2 demonstrated notable zero-shot performance across multiple NLP tasks, including achieving competitive results on the CoQA reading comprehension task (55 F1 score) as well as English-to-French translation (11.5 BLEU). Furthermore, they hinted at a near future where their next models would be even "bigger," as performance scaling showed log-linear improvement with increased model capacity. The final interesting detail to point out was that the authors advanced byte-level Byte Pair Encoding (BPE) tokenization in order to enhance GPT-2's compatibility with rare and unique characters. However, GPT-2 was far from perfect with some limitations. For instance, its zero-shot performance was below supervised state-of-the-art on certain tasks like summarization (ROUGE-L 26.58 versus 38.34 supervised). Moreover, its evaluation was limited to English tasks, which left multilingual generalization completely unexplored. Furthermore, OpenAI initially withheld their full model parameters because ethical concerns regarding misuse rose; this sparked discussions on transparency and responsibility in AI research. Lastly, the authors described how the model has a hard time distinguishing memorization from generalization. Overall, the foundational work found in this original paper is directly relevant to our implementation and fine-tuning of GPT-2, as it influences the way we understand and utilize our model architecture, tokenization, pretraining strategies, and hyperparameter tuning.

4 Approach

We build on the on GPT-2 architecture to address three distinct NLP tasks: sentiment classification, paraphrase detection, and sonnet generation. In each case, the model's transformer structure (featuring masked multi-head self-attention) gives us contextual representations that can be adapted to downstream objectives. By selectively modifying the the final output layer (for classification) or using auto-regressive decoding strategies (for text generation), we are able to utilize GPT-2's generalizable language understanding for both discriminative and generative tasks.

4.1 Model Architecture

Our model begins with the standard GPT-2 transformer, consisting of learning token and positional embeddings that then feed into a stack of decoder-only layers. Each layer applies masked multi-head self-attention via the CausalSelfAttention mechanism in order to ensure that the embedding of each token is informed only by its preceding positions, which is essential for auto-regressive generation. Furthermore, a residual connection and layer normalization follow both the attention and feed-forward sub-layers. We also see that task-specific heads are introduced only at the final stage, where we either project the last-token representation onto discrete labels (for classification) or transform the hidden states of all tokens into a probability distribution over the next token in the vocabulary (for generation).

4.2 Sentiment Classification

To perform sentiment classification, we train GPT-2 on sentence-level data from SST-2 and CFIMDB. We attach a simple linear output layer to the final token's hidden state, which produces a logit for each sentiment category. We then train via cross-entropy loss, where the model minimizes the difference between predicted and ground-truth sentiment labels. Next, we experiment with two fine-tuning schemas: (1) Full-Model fine-tuning, which updates all GPT-2 parameters alongside the classification head, and (2) Last-Linear-Layer fine-tuning, which "freezes" the core GPT-2 layers and only updates

the final classifier. In both cases, we use AdamW with dropout and learning rate scheduling so that we can manage overfitting.

4.3 Paraphrase Detection

We reframed paraphrase detection as a cloze-style generation problem. For a given pair of sentences, we prepend the prompt: *"Is 'Sentence 1' a paraphrase of 'Sentence 2'? Answer:"*. The model then generates either the token "yes" or "no," which is mapped to a binary label. Next, we encode the input using GPT-2's byte-pair encoder (BPE) and train on a subset of the Quora Paraphrase Detection corpus. Additionally, to encourage seeing stable gradients, we adopt AdamW with gradient clipping and incorporate label smoothing in binary classification to reduce overconfidence on rare, unambiguous pairs. This way, we can use this cloze-style paradigm to allow GPT-2 to use its generative capacity while still outputting succinct binary predictions.

4.4 Sonnet Generation

For sonnet generation, the model is fine-tuned on a dataset of Shakespearean sonnets. We prompt the model with the first three lines of a sonnet, then tasking it with generating the remaining eleven lines. During this sonnet generation, we use top-p nucleus sampling along with a gradually decreasing temperature in later quatrains and the final couplet, which we found helps balance creative output from the model with correct syntax/structure. Next, we observe character-level chrF as our principal performance metric, since it captures the small, fine-grained differences in wording and syntax when compared to the original Shakespearean text. In an additional experiment, we combine standard cross-entropy training with direct preference optimization (DPO). More specifically, when we run with the –dpo_mode flag, we begin with cross-entropy as a foundation for the model's fluency and style, then gradually introduce DPO. This objective compares the model's outputs against artificially perturbed "negative" samples (e.g. line-swapped versions), which helps in refining the generation process and in promoting high-quality, human-aligned text.

4.5 Optimization

Across all these tasks listed above, we conduct extensive hyperparameter tuning involving everything from learning rates, batch sizes, dropout probabilities, to various sampling strategies. For classification, we do this to ensure stable adaptation to varied training sets (SST-2, CFIMDB, and Quora Paraphrase). For generative tasks, we fine-tune top-p and temperature values to find the best trade-off between diversity and interpretability in its final output. Thus, by evaluating these hyperparameters, we are able to achieve stable training dynamics on both discriminative and generative benchmarks.

5 Experiments

5.1 Data

For paraphrase detection, we use a subset of the Quora dataset, which consists of approximately 400,000 question pairs labeled to indicate whether one question is a paraphrase of the other. Each example in our portion of the dataset includes a unique 25-digit hexadecimal identifier, two questions, and a binary indicator (where "1" denotes paraphrase status, and "0" denotes non-paraphrase). From this dataset, we select a training set of 141,506 instances, a development set of 20,215 instances, and a test set of 40,431 instances.

For sonnet generation, we use a corpus of 154 Shakespearean sonnets. Each sonnet is a 14-line poem following the rhyme scheme: ABAB CDCD EFEF GG. Our corpus is split into a training set of 143 sonnets and a test set of 12 sonnets.

5.2 Evaluation Method

We evaluate the paraphrase detection component by computing accuracy, which is reflective of the binary nature of the labeling (i.e. paraphrase or non-paraphrase).

For sonnet generation, we are given the initial three lines of each test sonnet and tasked with generating the remaining 11 lines. We measure generation quality using the chrF metric, which is a

character-level n-gram evaluation measure similar in spirit to BLUE but based on character fragments. This allows for a more fine-grained comparison between generated lines and Shakespeare's original text.

5.3 Experimental Details

We conducted paraphrase detection experiments by running:

```
python paraphrase_detection.py --use_gpu
```

Under these settings, training on Quora took approximately 40 minutes and 45 seconds per epoch, followed by around 1 minute and 20 seconds for evaluation. Unless otherwise specified, we adopted default hyperparameter choices: a fixed random seed of 11711, a batch size of 8, a learning rate of $1e-5$, and the standard GPT-2 model size. The number of epochs was set to 8, which is slightly fewer than the default of 10, because we found that 8 epochs outputted the best results for paraphrase detection.

Next, for sonnet generation experiments, we ran:

```
python sonnet_generation.py --use_gpu
--held_out_sonnet_path data/sonnets_held_out_dev.txt
--sonnet_out predictions/generated_sonnets_dev.txt
```

for the development split, and we ran:

```
python sonnet_generation.py --use_gpu
```

for the test split. Training each epoch required about 2 seconds, while producing all generated sonnets afterward took roughly 20 seconds.

Initially, our model achieved a chrF value of around 37. To increase our performance, we structured a hyperparameter sweep, focusing first on tuning sampling parameters for text generation. In particular, we found that reducing temperature slightly in later quatrains and the final couplet (e.g. around 0.9 and 0.95, respectively) produced more coherent text. We also determined that a top-p setting of 0.95 provided us with the best balance between diversity and fluency. Although we tested alternative optimizers, learning rates, and even introduced a repetition penalty, our initial default settings generally performed best. Interestingly, the model's outputs appeared to degrade in quality after the 8^{th} epoch, which further motivated us to stop at epoch 8 for best results.

5.4 Results

5.4.1 Sentiment Analysis Results

Epochs	Learning Rate	Batch Size	Dropout	SST Acc.	CFIMDB Acc.
3	$5e-5$	8	0.1	0.288*	0.527*
6	$3e-5$	16	0.2	0.292*	0.498*
10	$1e-5$	32	0.1	0.283*	0.547*
10	$1e-5$	64	0.1	0.509	0.971
10	$1e-5$	64	0.1	0.509	0.980

Table 1: Full-Model fine-tuning on SST and CFIMDB. Rows marked with an asterisk (*) indicate our older experiments (pre-bug-fix). We highlight our highest dev-set accuracies in red.

Table 1 and Table 2 compare dev-set performances for both Full-Model and Last-Linear-Layer fine-tuning strategies. Our older runs (marked with *) incorrectly used the first token embedding rather than the GPT-2 output's final token embedding, which resulted in substantially degraded accuracy. After identifying our issue and correcting it, both approaches achieve (and, in some cases, exceed) the previously reported baselines. For SST, the new Full-Model runs reach up to 0.509 dev-set accuracy (improving by over 20 absolute points compared to our previously flawed experiments), while for CFIMDB, they surpass 0.980. In agreement with our expectations, Full-Model fine-tuning generally outperforms Last-Linear-Layer, especially on the more nuanced multi-class SST task.

Epochs	Learning Rate	Batch Size	Dropout	SST Acc.	CFIMDB Acc.
10	$1e-5$	64	0.1	0.283*	0.580*
10	$1e-5$	64	0.1	0.282	0.576
10	$1e-5$	128	0.1	0.289	0.576
10	$1e-3$	64	0.1	0.455	0.873

Table 2: Last-Linear-Layer fine-tuning on SST and CFIMDB. Rows marked with an asterisk (*) indicate our older experiments (pre-bug-fix). We highlight our highest dev-set accuracies in red.

5.4.2 Sonnet Generation Results

Epochs	Learning Rate	Temp (middle/final)	Top-p	Dev-Set chrF	Test-Set chrF
10	1e-5	0.95/0.9	0.95	41.477	41.869
9	1e-5	0.95/0.9	0.95	41.705	N/A
8	1e-5	0.95/0.9	0.95	42.153	40.893
7	1e-5	0.95/0.9	0.95	40.244	N/A

Table 3: Sonnet Generation performance. Red values denote our best dev-set or test-set chrF.

Our best sonnet generation model achieved a test-set chrF of 41.869 and a dev-set chrF of 42.153. Interestingly, we wanted to note that the model version with the highest dev-set chrF did not produce our top test-set score. Because we had limited test submissions (on GradeScope), we primarily tuned on the dev set. Our dev-set chrF steadily improved up to 42.153; however, we believe that the corresponding model was slightly overfit, as evidenced by its lower test-set chrF of 40.893. Our final submission reached 41.869 on the test leaderboard, which placed us around 57th place overall.

5.4.3 Sonnet Generation Extension Results

We additionally experimented with direct preference optimization (DPO)[2] to refine the quality of our generated outputs. Initially, our negative examples were overly coarse, which gave us a dev-set chrF of 40.128. However, after adjusting the strength of DPO loss and mixing it with cross-entropy training, we reached a dev-set chrF of 40.530, which was an improvement but still below our best purely cross-entropy-based scores (42.153). Qualitatively, we hypothesize that weak or unrealistic "negative" samples limited the benefits of implementing DPO. With more time, we would explore more nuanced, clever technqiues for generating the suboptimal sonnets in our pairwise data.

5.4.4 Paraphrase Detection Results

On the Quora paraphrase detection task, our best model achieved:

- 0.884 accuracy on the dev set,
- 0.838 accuracy on the test set (as seen on the leaderboard).

As a whole, these results aligned reasonable well with our expectations because, understandably, the cloze-style approach would lead to better transfer from GPT-2's generative priors. We believe that, with more time, further refinements and tweaks to hyperparameters and sampling approaches could give us even more additional gains for paraphrase detection.

6 Analysis

In sentiment classification, the model generally captures the correct sentiment. However, misclassifications do occur, particularly in cases where the sentiment is ambiguous. A qualitative review of these edge cases suggests that improvements might be achieved by incorporating richer contextual representations or multi-token features rather than relying solely on the last-token embedding.

Qualitatively, initial paraphrase detection trials yielded an unreasonable accuracy of 1.0. Upon deeper investigation, we uncovered an incorrect label mapping that artificially inflated our performance. Once we discovered that our dataset loader already provided binary labels, we removed the mapping, revealing a more realistic performance. A closer inspection of misclassified examples showed that the model often struggles with subtle semantic differences. For instance, pairs with minor syntactic variations sometimes result in false negatives. This behavior suggests that the model is still biased toward the majority class and may require further fine-tuning to better capture nuanced paraphrasing.

Initial qualitative evaluation on sonnet generation was poor. Early outputs were problematic: generated texts often deviated from the ABAB CDCD EFEF GG rhyme scheme, contained nonsensical phrases, or even included non-English characters. After refining our decoding strategy—primarily through top-p nucleus sampling and adjusting the temperature in later quatrains—we observed substantial improvements. The revised outputs now adhere more closely to the expected 14-line structure and stylistic conventions. However, some issues persist; while most generated sonnets maintain coherent syntax and appropriate rhyme patterns, a few lines feature forced rhymes or awkward phrasing, indicating that the model sometimes struggles to maintain thematic consistency and poetic fluency.

7 Conclusion

In this work, we adapted GPT-2 to distinct tasks like sentiment classification, paraphrase detection, and sonnet generation. More specifically, in the second portion of this project, we were able to achieve a paraphrase detection accuracy of 0.838 and a sonnet generation chrF score of 41.869 on test sets, both of which surpassed the initial baselines. Nevertheless, we observed that sonnet generation remained quite vulnerable to overfitting, and augmenting the data offered only marginal gains; this highlights the complexity of optimizing LLMs for specialized text. Introducing DPO further showed that novel objective functions can shift the trade-off between fluency and alignment with desired textual qualities, like in the case for sonnet generation. If given more time, we would explore more sophisticated augmentation methods, examining regularization strategies to control overfitting, and experimenting with other Transformer configurations that could give us both stronger performance and deeper understanding across an even wider array of NLP tasks.

Team Contributions

We all equally contributed in writing this final report.

Aaron implemented the GPT-2 model given by the GPT-2 scaffolding, the GPT-2 building block layer, and the self-attention layer. For Part 2, Aaron built the paraphrase detection feature and helped debug sonnet generation's various issues.

Eli implemented the attention mechanism for the initial model, helped develop the initial sonnet generation architecture, led the experiments for optimizing sonnet generation, and experimented with DPO as an extension to sonnet generation.

Brandon implemented the classifier pipeline for running sentiment analysis, the GPT-2 building block layer, led the experiments and debugging for paraphrase detection, and helped develop the initial sonnet generation architecture.

References

[1] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019.

[2] Eric Mitchell Stefano Ermon Christopher D. Manning Chelsea Finn Rafael Rafailov, Archit Sharma. Direct preference optimization: Your language model is secretly a reward model. *arXiv*, 2024.