

Improving Performance and Efficiency of GPT-2 on Sonnet Generation and Paraphrase Detection Tasks

Stanford CS224N {Default} Project

Divya Bhojraj

Department of Computer Science
Stanford University
db23@stanford.edu

Abstract

In this paper, I implement a basic version of GPT-2, and extend it towards sonnet generation and paraphrase detection. Additionally, I aim to improve on the quality of Sonnet Generation by augmenting with more data from other poets, adding negative samples and using Direct Preference Optimization to increase the creativity and improve language use while maintaining similarity to Shakespeare's sonnets. I also aim to improve the memory efficiency of the paraphrase detection task, using quantization. For baseline, my paraphrase detection model has around an 87 percent accuracy, and my sentiment analysis model has approximately 46.5 on SFT and 86 on CFIRDB. DPO resulted in minimal changes, while quantization significantly improved memory usage, while only very slightly decreasing accuracy.

1 Key Information to include

- Mentor: Chenglei Si
- External Collaborators (if you have any): N/A. Used Claude and ChatGPT for help with coding tasks
- Sharing project: N/A
- Late Days: 1 day late over allocation

2 Introduction

Two major challenges in the field of LLMs are aligning outputs with human preferences, as well as improving efficiency in terms of memory and speed of training and evaluation. Direct Preference Optimization (DPO) was introduced as a way to simplify reinforcement learning from human feedback and still maintain good results. Quantization is a method to reduce the precision of models to simplify the memory while often trading off accuracy.

In this paper, I implement a basic version of GPT-2 with sonnet generation and paraphrase detection capabilities. Additionally, I aim to improve on the quality of Sonnet Generation by augmenting with more data from other poets, and using DPO to increase the creativity and improve language use while maintaining similarity to Shakespeare's sonnets. I additionally improve the efficiency of the Paraphrase detection task using quantization. I also explore maintaining or improving speed and accuracy even with the memory reduction by applying quantization during training.

3 Related Work

Related work to my project falls into two major categories, one per approach.

3.1 DPO

Direct Preference Optimization was proposed in a 2019 paper [1] as a way to simplify learning from human feedback. It is a single stage method that learns a policy and reduces computational complexity of RLHF. The initial experiments in this paper showed promising results on English language tasks such as summarizing or answering queries. There are several papers on how DPO can be used for LLM applications, particularly alignment. One paper finds that data selection is critical in creating good performance, and good selection can increase efficiency more than 4 times [2]. I aim to incorporate several different types of sonnets into the training to ensure representation. Another paper proposes a approach that involves bootstrapping LLMs with DPO Implicit Rewards, and shows notable improvements [3]. DPO assumes that preferences are fairly uniform, but one paper suggests an approach for optimizing towards multiple objectives [4]. For my sonnet generation task, I aim to optimize towards one objective (similarity to Shakespeare), but it could be interesting to extend towards other styles to encourage creativity and account for different aesthetic preferences.

3.2 Quantization

There are two major types of quantization discussed in literature for LLM use: Post-Training Quantization, and Quantization-Aware Training. Post-training methods tend to have worse accuracy in general, though one paper suggests a method for mitigating this disadvantage by quantizing layer parameters jointly [5]. Quantization-aware training is more complicated, and so another paper looks into training sensitivity for different models to the two methods to determine whether it is worth applying QAT [6]. QuadAdapter is a QAT method specifically designed for GPT use. The authors find that it prevents overfitting and improves quantization performance [7]. I primarily use a QuadAdapter style approach in my experiments, and compare to post-training.

4 Approach

There are three main tasks: sentiment analysis, sonnet generation, and paraphrase detection. I will use two primary approaches for improving upon the paraphrase detection tasks and sonnet generation tasks.

- For the Sonnet Generation task, I use Direct Preference Optimization to compare different sonnets in terms in quality [8]. The DPO objective is as follows:

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

Where:

- σ is the sigmoid function
- π_{θ} represents the model being optimized
- π_{ref} is a fixed reference model (typically the pretrained or fine-tuned base model)
- β is a temperature parameter controlling the strength of the preference signal
- (x, y_w, y_l) represents a prompt and its winning/losing output pair

I generate and use synthetic data of various quality sonnets (human or GPT labeled) to compare against the "good" sonnets defined as the 154 written by Shakespeare. Each Shakespearean sonnet will be preferred to non-Shakespearean sonnets. More information is provided in the Data section under Experiments.

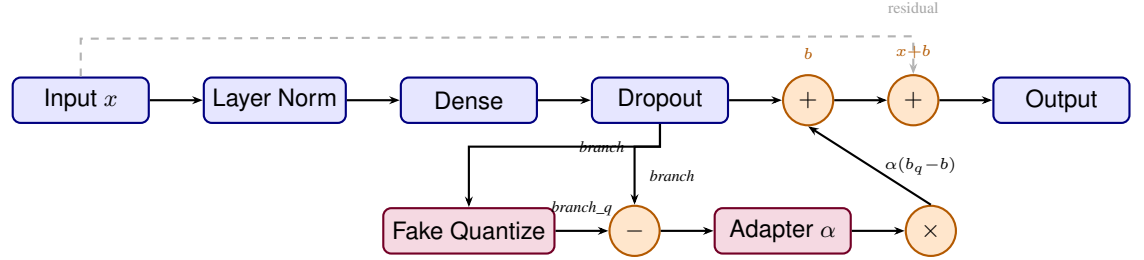
For paraphrase detection, I utilize quantization to improve my model's memory and speed [9]. Quantization is the process of reducing numerical precision. I aim to maintain accuracy in this regard, so I perform quantization-aware training rather than applying post training, though I also do the latter for comparison. I will use PyTorch's built in quantization library for PTQ, as well as a QuadAdapter style conversion pipeline. Quantization-aware training has the process of applying "fake" quantization operations pre-training to allow for adaptation of the model. Applying fake quantization to the input is as follows:

$$\hat{x} = \text{dequantize}(\text{quantize}(x, b, s), b, s)$$

$$x_q = \text{quantize}(x, b, s) = \text{clip}\left(\text{round}(s \cdot x), -2^{b-1} + 1, 2^{b-1} - 1\right)$$

$$\hat{x} = \text{dequantize}(x_q, b, s) = \frac{x_q}{s}$$

I apply fake quantization in a QuadAdapter type format, creating adapter layers in my attention and gpt forward blocks to prepare the model for applying quantization. I also try quantizing after the sub-layers on the activations instead of before on the input as in the paper for the attention and mlp layers. I still use weight quantization for attention. The architecture is modified from the baseline gpt2 layer as follows.



For a simple initial quantization test, I use dynamic quantization from Pytorch’s quantization library. This is the simplest form of quantization, which quantizes weights but not activations. It is a form of post-training quantization, applied after the regular training pipeline.

- For my GPT baseline, I used the provided GPT-2 framework, and implemented the forward and attention mechanisms [10]. The baseline for sentiment analysis is described in a 2013 paper [11]. The baseline for paraphrase detection is described in a 2008 paper [12]. I implemented the forward methods for these baselines.

5 Experiments

I ran several experiments for each task.

5.1 Data

The datasets include the Stanford Sentiment Treebank, a paraphrase dataset from Quora, and 154 Shakespearean Sonnets. For DPO, all included, there were 5 types of sonnets using in training: Shakespearean, other famous authors, GPT generated, baseline generated.

- I include 10 sonnets by poets other than Shakespeare such as Wordsworth, Keats, and Milton.
- GPT-5 generated 50 sonnets of decreasing quality (repetitive, generic or thematically bland). Some examples are shown in Figure 1.
- I had my baseline model generate the remainders of the first three lines of each Shakespearean sonnet and ranked it below the original, using the prompt as the first three lines.
- After evaluating, I added another 50 GPT-5 generated sonnets that were of bad structure rather than quality (too short, too long, bad rhyme scheme).
- My DPO file included approximately 12000 preference pairs with the following requirements: Shakespeare > Famous > GPT, and Shakespeare continuation > baseline continuation. Preference pairs were randomly sampled with these constraints. My generation scripts combined gave a preference pair with Shakespeare above other poems with probability greater than 70 percent, so as to encourage the model to learn Shakespearean sonnet structure. Overfitting was a concern, but results do not demonstrate its presence.

5.2 Evaluation method

For evaluation of sonnet generation, I used the CHRF score on the sonnets. [13]. I also did analysis on the various sonnets and similarity to the other sonnets across each model using embeddings and

Sonnet 1

The sky opens like a quiet door.
 Clouds step through slowly.
 Nothing rhymes with the sound of rain
 except the long memory of thunder.

Sonnet 2

A fragile dream of distance stirs bloom
 A fragile dream of distance stirs air
 In quiet thoughts of distance I still gloom
 A wandering mind on paths of distance fair
 Through veils of doubt appears distance part
 A fragile dream of distance stirs day
 A wandering mind on paths of distance start
 Through veils of doubt appears distance may
 My restless heart recalls distance glow
 The gentle pull of distance moves me glow
 The patient hours speak low of distance grow
 A fragile dream of distance stirs grow
 A wandering mind on paths of distance day
 In quiet thoughts of distance I still may

Figure 1: Example (bad) sonnets used in training

PCA as shown in Figures 3 and 4. Qualitatively, I looked at the differences in generated sonnets between the baseline, and my augmented model with DPO. For quantization, I focus on memory usage and speed for this task. In terms of memory and speedup, I compared the raw and percentage speedup in terms of seconds after a couple thousand batches, and decrease in model memory usage against my baseline model.

5.3 Experimental details

- I used 1 epoch to train for paraphrase detection baseline with a standard $1e-5$ learning rate. For quantization, I only quantized Linear Layers.
- I used 10 epochs for sonnet generation with a $1e-5$ learning rate. For DPO, there were around 1500 batches the first run through and 4000 batches the second run through.
- I used 10 epochs for sentiment analysis, with a $1e-5$ learning rate for the last linear layer and a $5e-5$ learning rate for the full model. I used a dropout weight of .1. For my extensions I used the same hyperparameters as above.

5.4 Results

Results for DPO showed limited success. Quantization results were more promising.

5.5 DPO

DPO slightly improved the performance of my model over the baseline. As demonstrated in Figure 2, DPO tended to actually be worse in terms of structure, despite attempts to add preference pairs that aligned with good structure. It is unclear why this is the case, as all sonnets other than the purposefully bad ones had good structure and were ranked higher. Figure 2 shows one of the worse sonnets generated by DPO, but many of the better ones were very similar to the original model. Many sonnets generated by DPO had the quality of being too short.

Figures 3 and 4 show the similarity between embeddings for different types of sonnets, where the average for each sonnets is shown by the x symbols. Figure 3 shows my original results where DPO did not have the structurally bad sonnets in training data. As visible, DPO and the original generated sonnets (in red) had basically the same mean, and the algorithm seemingly did not change the outcome. Ideally, it would have push the poems away from gen and famous on the x-axis and towards the Shakespearean poems, but it did not. Figure 2 shows the results after adding new data (around 25 percent more than the previous) to DPO that was not structurally correct. This also did not have much of an impact on DPO, as visible in Figure 4, the modified dataset did not push the mean towards the Shakespearean mean as intended. It moved it very slightly away in fact.

Generated Original

O, call not me to justify the wrong
That thy unkindness lays upon my heart;
Wound me not with thine eye but with thy
tongue;
But let me not say well these endings,
So bitter pity unto thy insolence-
And there may not breast and hear sputter;
Thy quiet, supple rebuke when they state by
delight
Touch and read hearts excited and starved,
Which slow, soft stillness might add to thine
good name,
Great enough for self, again and good most
sowers give.
But this thou wonder, I doubt if I shall even
thank,
Or if my lack makes happy birth praise.

Generated DPO

O, call not me to justify the wrong
That thy unkindness lays upon my heart;
Wound me not with thine eye but with thy
tongue;
With then my back, call thou thy woman the
trick, and profit by the network.

Figure 2: Example output sonnets used in training

5.6 Quantization

I ran a simple post-training dynamic quantization method on my model and trained on CPU using 1024 examples. This significantly improved my performance on memory and speedup while mildly reducing accuracy.

Model Type	Accuracy (%)	Memory (GB)	Speed (ms)
Original	.869	477	10.76
Dynamically Quantized	.813 (-.056)	150	3.69
Quantized - calibration (8-bit)	.761 (-.03)	232.7	11.5
Quantized - full model (8-bit)	.852 (-.009)	232.7	11.4
Quantized - full model (16 bit)	.867 (-0.02)	314.3	11.4
Quantized - calib/8-bit/no-para	.764 (-0)	232.7	11.3
Quantized - full/8-bit/no-para	.859 (-0.02)	232.7	11.5
Quantized - full/16-bit/no-para	.867 (-0.02)	314.3	11.4

Table 1: Comparison of quantization model performance metrics.

As also shown in table, I did dynamic quantization, with the original baseline as the base. While training was sped up by more than 2 times, accuracy on the full dataset decreased by around 5.6 percent. Memory decreased by around 70 percent.

After running dynamic quantization, I implemented quantization-aware training. I started by running on a calibration dataset. The calibration data was just the first 16384 datapoints, which is less than 8 percent of the original dataset and had an original accuracy of 79.2 percent. I implemented in a QuadAdapter style as detailed in the approach section. As shown in Table 1, I ran with 8 bit and 16 bit quantization, as well as with and without the paraphrase detection head being quantized.

6 Analysis

6.1 DPO

DPO had subpar performance, and barely changed the output, as demonstrated in Figure 2 and Figure 3. There are a couple reasons why this could be.

- Data: The data used spanned many different types of "bad" sonnets and ranked accordingly. However, many of the GPT-5 generated sonnets were very repetitive despite numerous

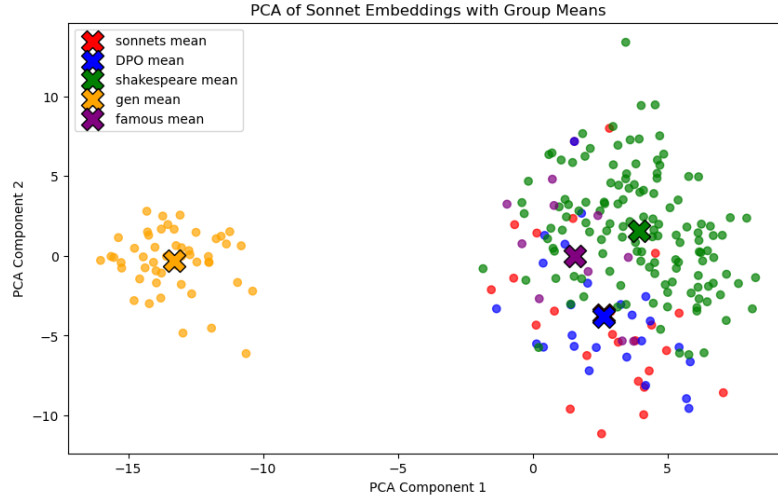


Figure 3: Sonnet Embeddings for 5 original types of sonnets

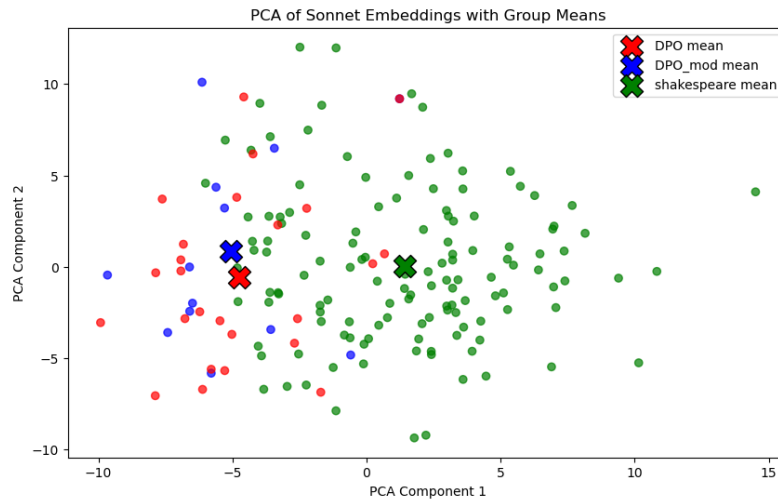


Figure 4: Sonnet Embeddings for DPO vs. Modified DPO

attempts to create variety. Perhaps with the ability to access OpenAI's API, the quality of these poems may improve, allowing the model to better learn a variety of "bad" sonnets. Additionally a greater variety of famous poems perhaps not just in sonnet form might improve performance.

- Preference Pairs: It is possible that the preference pairs did not make the best use of data. Due to time and compute limitations, the pairs with sonnet prompts only numbered around 154, while the others numbered in the thousands. It might also help generate to a DPO dataset comprised entirely of preference pairs with prompts as the first three lines, rather than just the 154 generated by the current script, and thousands of completely different sonnets, given more compute resources. This may result in overfitting to Shakespeare and lacking creativity, but currently that is not a concern in the results found.
- I used top-p sampling. Top-k sampling with $k=40$ ended up with worse results. It is possible that varying the parameters more or doing beam search based methods could improve the quality.

6.2 Quantization

While the quantization results were promising, the method can only be run on CPU, which is not particularly helpful for the time intensive task when there is a lot of data. GPU still far outperforms the speed results for the dynamically quantized model, completing the task in less than a second.

Results for QAT otherwise positive. I first ran using a small calibration model with 16384 datapoints or around 2000 batches, which resulted in my accuracy being around 3 percent less than the original calibration model, and memory and runtime being cut in half. As expected, 16-bit had higher accuracy than 8-bit and higher memory usage. With respect to speed, inference time increased for the QAT quantized models, by around 10 percent, but training time for these models went down, as they completed around 14.5 iterations per second rather than 11. Quantizing only linear layers was still able to get good results. Accuracy across the board barely decreased, and with the lack of paraphrase detection head, the model's memory was maintained while accuracy increased to almost the level of the original model. As expected, the decrease in accuracy from QAT was much smaller than from PTQ. However PTQ was far faster and used much less memory.

7 Conclusion

Overall, DPO results were minimally successful showing performance that barely differed from the original model. Quantization successfully sped up training, increased memory gains, and only reduced accuracy by a slight amount.

7.1 Limitations

There are many limitations to this work. For DPO, the most important was likely the quality and amount of data. As mentioned, data quality significantly affects the outputs of the model, so it is likely that the quality or quantity of the preference pairs was not good enough to result in improvement. Some factors that can be improved would be the variety of "bad" datapoints, and also improve the quality of said data, so that instead of re-learning algorithms that might be obvious from training, it can learn small differences that improve quality. For quantization, the results were significantly better, but though training can be done on GPU, quantizing using torch's int8 and int16 types is difficult and evaluation must be done on CPU. This slows down evaluation significantly.

7.2 Future Work

Future Work could look into how to improve performance on DPO, most likely by creating or sourcing better data and designing preference pairs that are more effective. For Quantization, it could be interesting to vary quantization parameters such as the number of bits during fake quantization, or which layers should be left out of the quantized model. As shown, just leaving out the last layer has strong positive accuracy impact. Overall, improving data quality or type in DPO might be a next step, and doing more experiments varying quantization types and parameters would also be interesting to look at.

8 Ethical Implications.

Some ethical challenges for the paraphrase task include the fact that Quora is a dataset of real questions and responses. This dataset might have similar questions asked in different ways which imply different things. For example, there might be questions that are asked in a combative or offensive way vs. genuine. Despite these being similar in obvious meaning they might not be overall. If used for detecting similarities or especially if extended to generating similar texts, it might not pick up these nuances and result in meanings being altered, which could have negative consequences. Thus, an effort would have to be made to train the model to account for these differences, which might need human labeling of sentiment or intention behind a subset of these examples. For the Sonnet Generation or any generation there is possibility of presenting something offensive or presenting a falsehood as a truth. In this particular task, sonnets cannot lie technically but there might be something offensive generated which can be prevented by training against these cases by introducing negative examples that are similar to positive ones in that they could be generated from them but have

slight variations which make them offensive. Overall for building GPT in general, ethical challenges include the possibility of untruths, or misrepresentations being spread which could be harmful, so appropriate steps should be taken to mitigate these, such as training to avoid negative examples.

References

- [1] Eric Mitchell Stefano Ermon Christopher D. Manning Chelsea Finn Rafael Rafailov, Archit Sharma. Direct preference optimization: Your language model is secretly a reward model. In *NIPS '23: Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.
- [2] Xun Deng, Han Zhong, Rui Ai, Fuli Feng, Zheng Wang, and Xiangnan He. Less is more: Improving llm alignment via preference data selection, 2026.
- [3] Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. Bootstrapping language models with dpo implicit rewards, 2025.
- [4] Zhanhui Zhou, Jie Liu, Jing Shao, Xiangyu Yue, Chao Yang, Wanli Ouyang, and Yu Qiao. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10586–10613, Bangkok, Thailand, August 2024. Association for Computational Linguistics.
- [5] Yoni Nahshan, Bartek Chmiel, Chen Baskin, and et al. Loss aware post-training quantization. *Machine Learning*, 110:3245–3262, 2021.
- [6] Xiaotian Zhao, Ruge Xu, and Xinfei Guo. Post-training quantization or quantization-aware training? that is the question. In *2023 China Semiconductor Technology International Conference (CSTIC)*, pages 1–3, 2023.
- [7] Minseop Park, Jaeseong You, Markus Nagel, and Simyung Chang. Quadapter: Adapter for GPT-2 quantization. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2510–2517, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics.
- [8] Eric Mitchell Stefano Ermon Christopher D. Manning Chelsea Finn Rafael Rafailov, Archit Sharma. Direct preference optimization: Your language model is secretly a reward model. In *NIPS '23: Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023.
- [9] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.
- [10] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019. Introduces GPT-2.
- [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] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th Annual Research Colloquium of the UK Special Interest Group for Computational Linguistics*, pages 45–52, 2008.
- [13] Maja Popović. chrF: character n-gram f-score for automatic mt evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics.