ASU AI Acceleration Lab Volunteer Internship Report

Summer 2025 - Karthick Arun

Under the mentorship of Francis Xavier Inyangat (Xavier)

Fine-Tuning GPT-2 XL for Synonym/Antonym Recommendations

My Journey Learning Small Language Model Fine-Tuning

Executive Summary

This report covers my experience learning fine-tuning concepts while building a synonym and antonym recommendation system. I fine-tuned GPT-2 XL (1.5B parameters) through multiple iterations on Google Colab and AWS SageMaker, turning theoretical knowledge into working code. This project shows how Small Language Model (SLM) fine-tuning can achieve solid results with a compact 1.5B parameter model from 2019. The below report has been formatted into the STAR (Situation Task Action Result) format to effectively display the process in which I learned fine tuning during this volunteer internship.

Situation - How I Got Started

The Challenge

I needed to learn fine-tuning for my program, but I wanted to work on something real rather than just following tutorials. I decided to create my own use case that would actually require fine-tuning to solve properly.

Why This Project?

I picked a synonym/antonym recommendation system because it hit the sweet spot:

- **Complex enough** that I couldn't just solve it with prompt engineering
- **Simple enough** that I could actually finish it as a beginner
- **Useful enough** to show real value

Where I Started

Honestly, I knew almost nothing about fine-tuning when I began. I understood basic ML concepts but had never fine-tuned a language model or even worked with the specialized datasets you need for this stuff.

Key Concepts I Had to Learn

The hardest concept for me was definitely QLora. I initially had no idea how it worked or why we even used it. After watching several YouTube videos and doing some research, I learned that QLora stands for Quantized Low-Rank Adaptation - it's a technique that lets you efficiently fine-tune very large models by significantly reducing the memory and computation required.

I also had to understand different chat formats that datasets come in and which formats work best for different use cases. I started by studying how the Unsloth Llama 3 8B parameter model worked with QLora and bnb quantization using an alpaca dataset on Google Colab.

Task - What I Set Out to Accomplish

Project Goals

My goal was straightforward: build a system that could recommend about 5 synonyms and antonyms that actually made sense for any input word, depending on the task specified (synonyms or antonyms). I didn't set any other complex requirements - I wanted to keep it focused.

Success Criteria

I defined success pretty realistically given I was working with GPT-2 XL, a 1.5B parameter model from 2019 that's microscopic by today's standards. I considered the project successful if at least a few of the generated words made sense as synonyms or antonyms and if it could consistently output 5

words in a sensible format. This was essentially Small Language Model (SLM) fine-tuning, so I kept my expectations grounded.

Action - The Implementation Journey

Platform Exploration and Early Challenges

Initially, I wanted to use AWS SageMaker with a ml.p3.12xlarge instance on Jupyter Notebook. I used ChatGPT to help figure out what cells I needed, but ran into major issues because the Hugging Face dataset was huge and I didn't want to download the entire dataset. Instead, I was trying to call it directly in the notebook from Hugging Face, but SageMaker's architecture is built around AWS's ecosystem where training data typically lives in S3 buckets. The training framework expects to pull data from S3 paths, not directly from external APIs like Hugging Face, which created compatibility problems. This would have required me to either download the huge dataset locally then upload to S3, or set up a complex data pipeline to stream from Hugging Face to S3 to SageMaker, or modify the training code to work with the Hugging Face datasets API.

After struggling with these architectural compatibility problems, I made the decision to abandon SageMaker entirely and switched back to Google Collab with NVIDIA T4 GPU. This turned out to be the right choice - I completed the entire project using CasualML to train the GPT-2 XL 1.5B parameter model from the OpenAI community on Hugging Face, using the kh4dien/synonym-antonym dataset, all within the Colab environment where I could directly use `datasets.load_dataset()` from Hugging Face without the S3 intermediary layer.

The Parameter Optimization Journey

My parameter tuning process was quite a learning experience. I started by asking ChatGPT 4o for "optimized" parameters and let the training run overnight (about 12 hours), but the kernel disconnected and timed out due to Colab's free session limits. This taught me an important lesson about working within platform constraints.

After that failure, I learned to work within Colab's constraints. I tuned down all parameters including reducing epochs from 3 to 1, and decreased batch size to 1 or 2 to train faster since I needed quicker iterations.

Trial and Error Process

Trial 1 Configuration:

```
training_args = TrainingArguments(
    output_dir="./gpt2-xl-synant",
    per_device_train_batch_size=1,
    num_train_epochs=1,
    save_steps=1000,
    save_total_limit=1,
    logging_steps=20,
    logging_dir="./logs",
    fp16=True
)
```

Trial 1 Output: Complete garbled text - unintelligible output that bore no resemblance to actual words or language patterns.

Trial 2: I realized my mistake and increased epochs to 5, reduced logging steps to 100, set batch size to 2, and made adjustments to temperature settings. This trial finally started producing actual words:

Trial 2 Results:

- Synonyms for 'bright': "dazzling, lightshowy, eyed, visageless, eye"
- Antonyms for 'cold': "warmness, mealylebrity, mealylebr, mealye, mealyebr"

While showing improvement with recognizable words, there were still issues with word quality and repetitive patterns.

Trial 3 Configuration:

```
training_args = TrainingArguments(
    output_dir="./gpt2-xl-synant",
```

```
    per_device_train_batch_size=2,
    per_device_eval_batch_size=2,
    gradient_accumulation_steps=8,
    logging_steps=100,
    save_strategy="epoch",
    num_train_epochs=5,
    learning_rate=5e-5,
    warmup_steps=100,
    weight_decay=0.01,
    logging_dir="./logs",
    save_total_limit=2,
    fp16=True,
    report_to="none"
)
```

Trial 3 Results: Significant improvement in coherence:

- Synonyms for 'lucky': "colorful, vibrant, colors, vibrant, colors"
- Synonyms for 'cold': "hot, or, hot, blooded, frozen"
- Synonyms for 'boring': "boring, looking, boring, looking, outlandish"
- Synonyms for 'dog': "puppy, dog, trainer, meadowguard, meadowguarddog"

The results became much more coherent with proper synonym/antonym relationships, though still showing some repetition.

Training Parameter Comparison

Parameter	Trial 1	Trial 2	Trial 3
Epochs	1	5	5
Batch Size	1-2	2	2
Learning Rate	Conservative	Adjusted	5e-5
Logging Steps	1000	100	100

Training Time	~20 min	~45 min	~45 min
Output Quality	Garbled text	Recognizable words with issues	Coherent with some repetition
Success Rate	0/5 words	~2/5 words sensible	~3-4/5 words sensible

Key Observation: The jump from Trial 1 to Trial 2 showed the most dramatic improvement, demonstrating that insufficient epochs (1 vs 5) was the primary bottleneck in initial results.

Results - What I Achieved

Final Model Performance

The output isn't perfect, but it produces sensible results in a clean 5-word format without repetitive hallucinations. For example, when asked for synonyms of "bright," it generated words like "dazzling, lightshowy, eyed, visageless, eye" and for "cold" it produced "hot, or, hot, blooded, frozen." While not all words are perfect matches, most make logical sense as synonyms or antonyms.

Realistic Expectations Met

These results align perfectly with what you'd expect from GPT-2 XL - a 1.5 billion parameter model that's quite small by today's standards. Historically, GPT-2 XL was never considered a "good" quality model compared to what came after, which is why OpenAI never closed-sourced it or used it as the foundation for ChatGPT, their flagship product. Given these limitations, achieving sensible word relationships and consistent formatting represents a solid success for SLM fine-tuning.

What Exceeded and Fell Short of Expectations

What impressed me was the model's ability to generate sensible outputs given both the age and relatively small size of GPT-2 XL. The fact that a 1.5B

parameter model from 2019 could learn to produce coherent synonym/antonym relationships at all was encouraging.

However, the quality of the actual word choices fell short of what I hoped for. I'm confident that with more training epochs, a larger dataset, and far smaller logging steps over a longer time period, the results would have been significantly better with the same model. The time constraints I faced limited the quality I could achieve.

Future Improvements

If I were to attempt this project again, I would use Unsloth with QLora and BNB quantization to fine-tune a more capable model like Mistral 7B or Llama 3 7B. I'd also find a dataset already formatted in the Alpaca format designed specifically for synonym/antonym tasks, and train with more aggressive parameters given a larger time budget for training.

Key Lessons Learned

Technical Insights

The most important technical lesson I learned is that certain parameters like epochs and batch size can drastically alter both the quality of the output and the overall model performance. Small adjustments in these parameters made the difference between garbled nonsense and coherent word relationships.

I also discovered that if you want genuinely good quality results, you should start with at least 7B parameters - 1.5B is simply too small for most practical applications. The exception is when you're specifically working on Small Language Model (SLM) fine-tuning rather than Large Language Model (LLM) fine-tuning, or when you need the model to run on-device where size constraints matter more than absolute quality.

Platform and Resource Management

I learned that SageMaker is really only beneficial if you have your own data that needs to be cleaned, prepared, and trained off a JumpStart model. For most learning projects, the additional complexity isn't worth it.

For Google Colab, I discovered that training bigger models without Colab Pro requires Unsloth quantization using QLora/BNB due to memory size constraints. The free tier simply can't handle larger models without these optimization techniques, unless you purchase Colab Pro compute instances with more provisioned resources.

Parameter Tuning Reality

I learned how incredibly time-consuming it is to find the right parameters and then train and retrain the model multiple times. The iterative process of adjusting parameters, waiting for training to complete, evaluating results, and then starting over again takes much longer than I initially expected. Each cycle can take hours, and you often need multiple cycles to get decent results.

Unexpected Challenges

What surprised me most about the fine-tuning process was the sheer amount of parameter changes needed and the extensive training time required. I initially thought I could get good results with minor tweaks, but it turned out that finding the right combination of parameters required much more experimentation than anticipated. The training time for each iteration was also significantly longer than I expected, making the whole process much more time-intensive than I had planned for.

Learning Process Insights

I discovered that using ChatGPT to clarify concepts was incredibly valuable for my learning process. When I encountered difficult topics like different chat formats or complex fine-tuning concepts, ChatGPT could explain them in ways that were easier for me to understand than traditional documentation or tutorials. This conversational approach to learning helped me grasp technical concepts much faster than trying to decode academic papers or dense technical docs on my own.

Advice for Future Fine-Tuning Projects

If I had to give advice to someone starting their first fine-tuning project, I'd tell them to start off with using Unsloth QLora/BNB to quantize a larger LLM like Mistral 7B or Llama 3 8B. Don't make the same mistake I did by starting

with an undersized model like GPT-2 XL. The larger models will give you much better results and a more rewarding learning experience, even if they require quantization techniques to fit within free compute constraints.

Conclusion

This project successfully demonstrated Small Language Model fine-tuning using GPT-2 XL while teaching me fundamental concepts about model optimization, parameter tuning, and resource management. Although the 1.5B parameter model had limitations, the experience provided valuable hands-on learning about the entire fine-tuning pipeline - from initial concept selection through multiple training iterations to final evaluation.

The iterative process of parameter adjustment, the challenges of working within computational constraints, and the importance of choosing appropriately sized models were all crucial lessons that will inform future projects. Most importantly, this project proved that even with limited resources and a relatively small model, meaningful results are achievable when you understand the underlying concepts and are willing to iterate through the optimization process.

Moving forward, I'm confident in applying these fine-tuning skills to larger, more capable models and more complex use cases, armed with a solid understanding of both the technical requirements and practical constraints involved in model fine-tuning.