Finetuning Meta's Llama-3-8B Model

National Park Comment Extraction

Austin Lackey

April 26, 2024

Abstract

Large language models (LLMs) have become increasingly popular in the last few years due to their ability to grapple with large amounts of text data. However, these models are often only accessible to those who have the computational resources to run them, or the money to pay for API access. In this paper, I discuss how multiple efficiency techniques can be used to finetune a model for specific use-cases, all while training and running the model locally.

Figure 1: Image generated by DALL·E 3 [6]

1 Introduction

Every year, the VUStats program at The National Park Service (NPS) combs through 400+ parks worth of data to ensure that data collection is accurate and consistent. Primarily, data collectors from each park will submit monthly numbers accompanied by a comment that describes the visitation for that month. Often times, these data collectors will submit non-zero numbers for locations but mention that the location was closed. This is a problem as the comments and the numbers contradict each other. Whether it be due to human error, fat-fingering, or a lack of attention to detail, it is the VUStats program's job to catch these errors and correct them. However, this requires a lot of reading and skimming as comments can range from a few words to sometimes hundreds. Often, most of the information is not relevant to the data collection process and can be cumbersome to read through. This is

where the use of modern language models can be of great use.

1.1 Motivation

Although large language models (LLMs) are great for dealing with subjective text, any model that is trained on a large enough dataset is often only accessible for those who have access to the necessary computational resources or pay via API. Both of these avenues are not feasible for the VUStats program as they are a government entity and lack the resources. With the power of open source language models, QLoRA finetuning, and Apples MLX technology, we can finetune a model that is runnable on most laptops.

2 Methodology

The idea here is simple, by focusing the model on a specific use-case and using a few tricks to cut computational corners, we can create a model that is feasible for local use.

1. **Data Collection:** Create an I/O domain specific dataset that is relevant to the use-case.

2. **Model Selection:** Choose a model that is small enough to run but large enough to be useful.

3. **Quantization:** Further reduce the model size by reducing the precision and memory requirements.

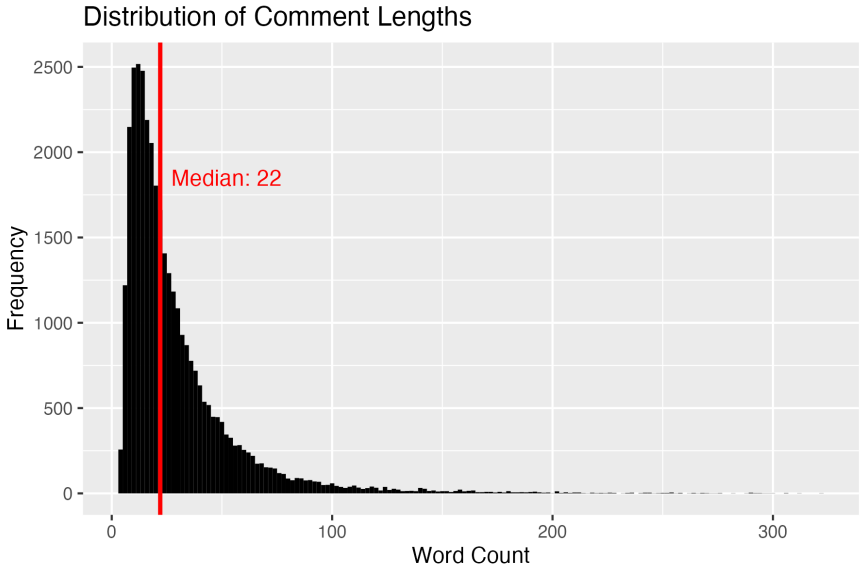
4. **MLX:** Use Apple's MLX technology to take advantage of Metal's GPU acceleration and speed up inference.

5. **QLoRA Finetuning:** Train a subset of the model on the domain-specific dataset.

Figure 2: Comment Length Distribution

2.1 Data Collection

The data collection and preparation process is arguably the most important step in creating a model that is useful. Garbage in, garbage out. Comments within the dataset are collected from the VUStats database. Here we have access to all 34,862 comments that range from 1993-2024. While most of the comments are short, and concise, some are long and verbose so its important to build a dataset that is representative of the entire population.



Its important to note that these comments often follow a pattern within each park since data collectors generally format their comments the same for each month. That means one park may post a comments that are short and concise while another park may post comments that are long and verbose. To ensure the training dataset was well represented, a stratified split was used to select 100 random parks, with each strata randomly selecting 3 years worth of data. After sampling, we have the following dataset:

We now have one last step, and thats to generate the output, or in this case, the format in which we want the model to convert these comments to. Since we only care about the

	Split	Size	Token Count
Train	75%	1,192	45,649
Test	25%	398	17,552
Total	**100%**	**1,590**	**63,201**

Table 1: Dataset Split and Size

information that is relevant to closures and visitation dips, we want a simple format that is only a few sentences long which is easy to read and understand. Rewriting the comments in this format at this scale is not feasible to do by hand with the time and resources constraints, so I used OpenAI's GPT-3.5 Turbo API to generate a good output for each comment. This took only a few minutes to run and the output was more than satisfactory. What's interesting about this technique is we are 'distilling' the capabilities of a larger Transformer into a smaller one, and it only costed me 35 cents to do so. To be clear, the use of GPT-3.5 Turbo was only used to expedite the dataset building process and is not used in the final model. Below is an example of the input and output for a comment, the closure related information is highlighted in blue for clarity.

Input:	`Comment for January: Averages for the following counters were submitted: Belle Haven (vehicles and MVT) Daingerfield (vehicles and MVT) Fort Hunt Park Turkey Run Gravelly Point Note: The sign crew supervisor will check the above counters. Averages for the follwing vehicle and traffic counters were used: U.S. Marines Corps War Memorial, Great Falls Park, Lane 1(National Airport) and Lanes 2, 3,4,5 (National Airport) Note: Data Collector was notified by email of the vehicle and traffic counters.` **Claude Moore Colonial Farm - annual January closure**
Output:	**January: Claude Moore Colonial Farm was closed for the entire month**

Table 2: Input and Output Example

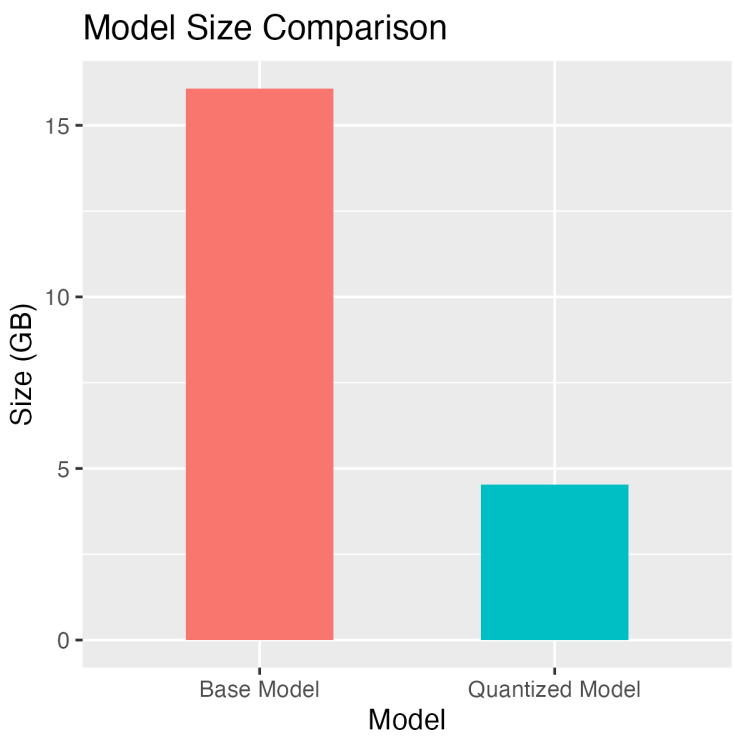
2.2 Model Selection

There are many open-source language models available to choose from, but as of completion of this project, Meta launched their Llama-3 models in both 8B and 70B parameter sizes. In fact, the Instruct tuned version of Llama-3-8B outperforms Llama-2-70B on many benchmarks, which is quite impressive considering the size difference [5]. Llama-3 was trained on over 15 trillion tokens, of publically available text with a cutoff date of March, 2023 [5]. This is great since we get both a large variety of knowledge that is up to date. Its important to note here that there are multiple versions of Llama-3-8B, which are finetuned for tasks like chatbots, instruction following, or basic autocompletion. In this project, I used the base transformer version of Llama-3-8B, which means it has not been finetuned for any specific task. This is important as we do not want to fight against the model's preconceived notions of how to generate text. Finetuning a model that is already finetuned can dillute the model's capabilities.

2.3 Quantization

Quantization is technique that reduces the precision of the model's weights and activations. This is extremely important as it allows more people to run models on their local machines. However, this comes at a tradeoff, we are increasing the efficiency at the cost of accuracy [2]. By converting models from 32-bit floating point to a lower precision, like 4-bit integers, we can reduce the model size by 4x. Llama-3-8B has about 8 billion parameters, which works out to be $8,000,000,000 \times 32/8 = 32,000,000,000$ bytes or 32 GB. With my M1-Pro MacBook Pro (16GB of RAM), I would not be able to run this model without quantization. After quantization, the model size is reduced to 8 GB, which is more than feasible to run on my machine.

Figure 3: Quantization



2.4 MLX

MLX is a NumPy-like library that is optimized for machine learning on Apple Silicon. It is built on top of Metal, which is Apple's GPU acceleration framework [3]. This is important as it allows me to take advantage of the M1 Pro's GPU to speed up computation. This is just another layer of optimization that allows me to run the model on my local machine, allowing me to train more advanced models. This is important because training LLMs on my MacBook would not have been feasible without MLX.

2.5 LoRA & QLoRA Finetuning

The process of finetuning a large language model is typically done by training the model on a domain-specific dataset to improve the model's performance on that specific task. However, this requires training all 100% of the model's parameters, which is computationally expensive.

LoRA is a technique that allows us to train a subset of the model's parameters, by introducing small, low-rank matrices that work in correspondence with the original model's weights to produce adapted outputs [4, page 2]. The authors of LoRA present two main advantages of this technique: 1) one singular pre-trained model can be used to build many task-specific LoRA adapters, and 2) LoRA's efficiency reduces the hardware requirements for training [4, page 2]. Think about these adapters as nozzles to a power washer. By swapping out these finetuned adapters, we can drastically change the way the model behaves. For example, we could train many adapters for different use-cases, effectively building a "toolbox" of LLM adapters. QLoRA extends LoRA even further by introducing quantization to further increase efficiency. Authors of QLoRA claim that their new technique reliably matches performance of 16-bit floating point precision on academic benchmarks [1, page 7].

The main takeaway here, is we are utilizing all of the efficiency techniques available to us to finetune an LLM on a domain-specific dataset for a specific use-case; and most importantly, we are doing this **locally**.

This took a lot of trial an error to get right, but after many iterations, overnight runs, and a "toolbox" full of QLoRA adapters, we can compare the performances of each adapter. In training the adapters, I experimented with many prompt formats, batch sizes, and the number of layers to finetune. The largest factor in performance was how the prompt was formatted, not so much the number of layers or batch size. Although increasing the number of layers and batch size did increase performance, it was not as significant as changing the prompt format.

Figure 4: Adapter Performance

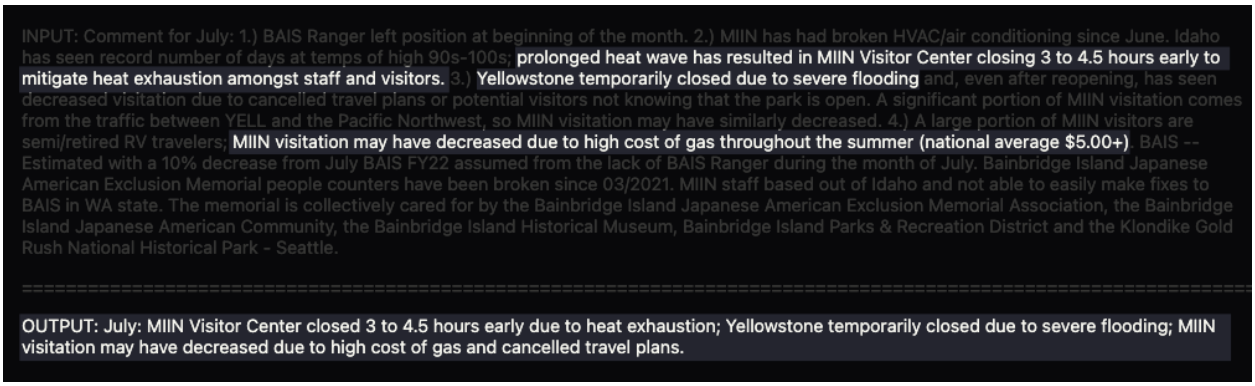
Adapter	Prompt	Batch Size	Lora Layers	Trainable Parameters
Adapter 1	Basic	1	4	0.426M
Adapter 2	Advanced	1	4	0.426M
Adapter 3	Advanced	4	32	3.408M

Table 3: Adapter Performance

In the end, Adapter 3 was the best performing adapter. Once fused with the base Llama-3-8B model, we can now run the model for inference. Below is an example of the model's output for a given input comment. For clarity, the closure related information is highlighted in the main comment. The model accurately identifies the closure information and outputs

Figure 5: Adapter 3 Output Example

it in a clear and concise format. This is a huge win for the VUStats program as it will save them time and resources in the long run.



3 Conclusion

In this paper, we show not only the feasibility of finetuning a large language model on a local machine with innovative techniques, but also the importance of domain-specific datasets. For model performance, it is clear that the prompt format is more significant than just feeding it input/output.

References

[1] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[2] Wenqi Glantz. Democratizing llms: 4-bit quantization for optimal llm inference. *Towards Data Science*, 2024.

[3] Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. MLX: Efficient and flexible machine learning on apple silicon, 2023.

[4] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[5] Meta. Llama-3-8B base model used for fine-tuning, 2024.

[6] OpenAI. DALL·E 3: Prompt: "photorealistic version of a llama with 4 legs wearing a park ranger vest and a park ranger hat wearing sunglasses", 2024. This image was generated with the assistance of AI.