

Benito Infantino

Aug 25, 2024

Advanced Applied Machine Learning

Research Paper

How Much Data Do You Really Need To Fine Tune Llama 3 Instruct 8B?

Introduction

Large language models have taken the world by storm since the release of ChatGPT. LLMs offer businesses and individuals the ability to automate many tasks that would otherwise be very difficult to automate in the past. Chatbots, summarization, content creation, brainstorming, data analysis and many other tasks have been revolutionized on the back of the mainstream adoption of LLMs.

However, businesses have been hesitant to adopt and implement LLMs since many LLM applications are closed source and therefore there is a data vulnerability. Closed source LLM models are a black box which the businesses have limited visibility into and as a result many businesses are not willing to use their private data with these LLMs therefore preventing them from leveraging the powerful new technologies.

Open source models, such as Llama, do exist however they have been lagging behind closed models in terms of performance. However, open source models are able to be fine-tuned for and adapted to any task assuming there is enough data to train the model. As a result, businesses are more likely to take an open source model and fine tune it on their task to ensure data privacy while still leveraging the power of LLMs but this comes with its own set of challenges.

The businesses must collect their own dataset and fine-tune their own model in order to harness the full power of open source LLMs. This process is not only technical but challenging. How much data is required for a successful fine-tune? Where is this data collected from? How do you acquire the data needed to fine-tune a model assuming that you don't have it readily available? What if the data you have collected can't be used since it contains personal data not fit to be used as training data? These questions are imperative and must be solved to see a widespread adoption of LLMs in business.

We aim to solve the question of how much data you need for a successful fine-tune and the dataset curation process through a multi-stage process. In this paper we leverage Llama 3 Instruct 8b to create a sustainable process for which businesses can take advantage of open source models for their unique use cases.

First, we identify a practical business use case where a business can implement a LLM to increase efficiency. Then, we benchmark Llama 3 Instruct 8b on this task to gauge the performance of the base model . Next, we create a synthetic dataset using Llama 3 Instruct 8b then we fine tune Llama 3 Instruct 8b on this new dataset and benchmark the training process to track model performance as a function of the size of the training set.

This approach leverages the power of open source models while still ensuring data privacy.

Our methods, analysis and results are explained in detail below.

Related Research

Parameter-Efficient Fine-Tuning Techniques

One of the foundational works in the area of parameter-efficient fine-tuning is the development of Low-Rank Adaptation (LoRA) by Hu et al. (2021). LoRA introduces an innovative approach to fine-tuning large-scale models by reducing the number of trainable parameters needed, thereby making the fine-tuning process more efficient in terms of both computation and data requirements. LoRA operates by freezing the pre-trained weights of the model and injecting trainable rank decomposition matrices into each layer's architecture. This technique allows for the fine-tuning of large language models like Llama 3 Instruct 8B on smaller datasets without the need for extensive computational resources. The method maintains performance levels close to those achieved by traditional fine-tuning methods which makes it effective for scenarios where computational resources are constrained. This approach is relevant in the context of fine-tuning large models on specific tasks with limited data.

Synthetic Data for Fine-Tuning

A recent contribution to the use of synthetic data in fine-tuning large language models (LLMs) is the study by researchers, titled "SyntheT2C: Generating Synthetic Data for Fine-Tuning Large Language Models on the Text2Cypher Task." This work explores the generation of synthetic data to fine-tune models for specific tasks, particularly when high-quality real-world data is scarce. The study demonstrates how combining LLM-based prompting with template-filling techniques can create diverse and high-quality synthetic datasets. These datasets significantly improve the performance of models fine-tuned on them.

Hypothesis

We propose that Llama 3 Instruct 8b Instruct can be used to generate a high quality synthetic dataset for a narrow task. Then, Llama 3 Instruct 8b can be fine tuned on this new dataset to outperform the base model on the narrow task. This gives businesses the ability to create synthetic datasets for their specific purposes while out performing the base model and maintaining data privacy. In addition, fine tuned models do not require extensive prompting since the fine tuned model understands the desired format and characteristics for the responses.

Task

The task we are focusing on is generating high quality sales social media posts. We chose this task because it is a practical business use case where a business uses an LLM to generate sales posts for their social media in their brand voice and likeness. Additionally, we can create a synthetic dataset by specifying specific criteria we would like our data to have. Then, we can test the base version of Llama 3 Instruct 8b and our fine tuned version of Llama 3 to compare the effect that fine tuning had.

We created a list of characteristics that an ideal sales social media post should have. Then we used Llama 3 Instruct 8b with specific prompts to create this synthetic dataset. Finally, we fine tuned Llama 3 Instruct 8b on this synthetic data to generate high quality sales social media posts that satisfy specific criteria without having to explicitly prompt the LLM.

We found this to be an interesting task since it allows anyone who uses this specialized model to get the maximum benefit from it without needing to know anything about effective prompting. This is useful for large scale enterprises who may want to roll out LLMs for their employees to use without having to train each and every employee how to effectively prompt an LLM.

Hardware Specifications

All of the code and experiments executed in a Google Collab notebook leveraging an A100 GPU. We purchased a Google Collab Pro subscription to ensure we had access to the A100 at all times. The total cost came out to ~\$150 USD for the compute units and subscription plan.

Compute Specs	System RAM	GPU RAM	Disk
	83.5 GB	40.0 GB	201.2 GB

Methods

Step 1: Creating Synthetic Dataset

1. Identifying Dataset Characteristics

We created our synthetic dataset using Llama 3 Instruct 8b combined with a variety of user prompts to ensure a rich and diverse fine tuning dataset. It is essential to create a diverse dataset to ensure that the fine tuned model can learn from our fine tuning dataset without ‘memorizing’ the dataset. Therefore, the first thing we did was figure out the core components of our user prompts.

We broke down our user prompts into the following categories:

- Universal Criteria
The universal criteria are universal because they apply to all of the sales posts that we want to generate. These criteria include:
 - Capture attention quickly with bold statements or questions.
 - Keep the text concise and to the point.

- Use a strong, clear call-to-action (CTA).
 - Highlight the benefits of the product.
 - Emphasize unique selling points (USPs).
 - Maintain consistent branding elements.
 - Leverage platform-specific algorithms (e.g., hashtags, keywords).
- Post Length

The post length specifies the number of sentences that the output should have. This is important because an end user may want to generate posts of various lengths to keep their social positions varied. The post lengths include:

 - Short posts with 1 - 3 sentences
 - Medium posts with 4 - 6 sentences
 - Long posts with 7+ sentences
- Post types

The post type specifies the type of post we want to generate. This ensures our fine tuned model is capable of generating a wide variety of posts which gives the end user more options. We tied the post length to the post type since the post type and the post length go hand in hand. The post types include:

 - Short Posts
 1. Promotional Posts
 2. Question Posts
 3. Feature Highlight Posts
 4. Comparison Posts
 5. Announcement Posts
 - Medium Posts
 1. Educational Posts
 2. Testimonial Posts
 3. User-Generated Content Posts
 4. Value Proposition Posts
 - Long Posts
 1. Storytelling Posts
 2. Interactive Posts
- Specific Criteria

We also specified specific criteria that we want each post type to include. We added a specific criteria since there are characteristics that may apply to some post types but not others. The specific criteria include:

 - Promotional Posts:
 1. Announce sales, discounts, or special offers.
 2. Highlight urgency with time-sensitive language.
 3. Mention exclusivity in offers.
 - Question Posts
 1. Engage the audience with questions related to the product.

2. Personalize messages to the audience.
- Feature Highlight Posts
 1. Focus on specific features or benefits of the product.
 2. Use vivid, descriptive language.
 - Comparison Posts
 1. Compare the product with competitors or previous versions.
 - Announcement Posts
 1. Inform about new product launches or updates.
 2. Align posts with seasonal or trendy themes.
 - Educational Posts
 1. Provide useful information or tips related to the product.
 2. Share educational snippets or interesting facts.
 - Testimonial Posts
 1. Share customer reviews and testimonials.
 2. Quote satisfied customers or share positive reviews.
 3. Include social proof elements like customer numbers or awards.
 - User-Generated Content Posts
 1. Share content created by customers.
 2. Highlight user experiences and benefits from the product.
 - Value Proposition Posts
 1. Clearly state what sets the product apart from others.
 2. Use a problem-solution framework.
 - Storytelling Posts
 1. Narrate a story that highlights the product's benefits.
 2. Use vivid, descriptive language.
 3. Personalize the story to the audience.
 - Interactive Posts
 1. Include polls, quizzes, or contests to engage the audience.
 2. Use interactive elements to increase engagement.
 3. Encourage audience participation and sharing.
- Company Descriptions

We created a set of 15 diverse companies to increase the size of our fine tuned dataset. The company information also provides useful information such as the type of business, target audience, products and more which result in a diverse dataset. We used GPT-4o-mini to generate fictitious companies. The company descriptions include:

 - Company name
 - Company description
 - Industry
 - Mission statement
 - Product offering
 - Company size
 - History

- Target Audience
- List of products

2. Generating Prompts To Create Fine Tune Dataset

We created several python dictionaries to hold each of the characteristics listed above. We then created a nested loop over each of the dictionaries to create the prompts that will be used to generate the fine tuning dataset. The prompt is shown below. Any variables included in {} are dynamically added from our dictionaries. Through this process we generated 17,600 unique prompts which will give us 17,600 rows of data for our fine tuning dataset.

Prompt:

```
"You are an expert sales copywriter for social media that
specializes in modern sales and the 'attention economy'."

Your task is to a social media post for sales with the
following characteristics:
1. Company name: "{company['company_name']}"
2. Company Description: "{company['company_description']}"
3. Product: "{product}"
4. Post length: "{post['length']}"
5. Post type: "{post['type']}"
6. Post characteristics: "{complete_criteria}"
7. Writing tone: "{tone}"

Your output MUST be ONLY the social media post with NO
additional information, text, commentary or anything extra."
```

Nested Loop:

```
for company in companies:
    for product in company['products']:
        for post in short_posts + medium_posts + long_posts:
            for tone in writing_tones:
                post_type = post["type"]
                length = post["length"]
                complete_criteria = f"""
                {universal_criteria}
                {post['criteria']}"""
```

3. Generating Fine Tuning Dataset

Finally, we used Llama 3 Instruct 8b through the Groq api to generate our fine tuning dataset. We chose to use the Groq api instead of a local version of the

model to speed up the time it takes to generate the fine tuning dataset. The Groq api uses novel hardware and software to dramatically increase tokens generated per second. Even with the Groq api it took about 10 hours to generate all 17,600 responses.

Step 2: Generating Baseline Responses

1. Generating Baseline Prompts

We created a set of prompts which will be used to create the baseline responses. The baseline prompts are similar to the fine tuning prompts but leave out the universal and specific criteria shown above. The goal of our fine tuned model is to generate posts that match the criteria we specified without having to explicitly mention the criteria in the prompt. We also created a simpler set of company descriptions for each company since the end user may not include detailed descriptions when prompting the fine tuned model.

Simple Company Descriptions:

- Company Name
- Company Description
- List of products

Prompt:

```
"Write me a sales post for my company's product with the
following characteristics:
1. company description: "{company['company_description']}"
2. product: "{product}"
3. post length: "{length}"
4. post type: "{post_type}"
5. post tone: "{tone}"

Your output MUST be ONLY the social media post with NO
additional information, text, commentary or anything extra."
```

Nested Loop:

```
for company in companies_simplified:
    for product in company['products']:
        for post in short_posts + medium_posts + long_posts:
            for tone in writing_tones:
                post_type = post["type"]
                length = post["length"]
```

2. Generating Baseline Responses

We decided not to baseline all 17,600 prompts as this would take unnecessarily long since the responses are generated from our local model which has a slower token generation speed. Therefore, we created a subset of 95 prompts that we will use to generate the baseline responses.

We created a subset of baseline prompts by taking random samples from our baseline prompts for each characteristic. We did this by extracting the unique values for the post tone, post type and post length. Then we sampled 5 rows from each unique value.

For example, our subset includes 5 short posts, 5 medium posts, 5 long posts, 5 of each unique tone and 5 of each unique post type. This ensures that our baseline dataset is diverse.

Step 3: Fine tuning Llama 3 Instruct 8B 4 Bit

1. Loading The Model

We decided to use the Unsloth 4 bit version of Llama 3 Instruct 8b since Unsloth is optimized for fine tuning and inference. Unsloth significantly speeds up the fine tuning and inference speed of our local model. The 4 bit version of the model is a quantized version of Llama 3 Instruct 8b that reduces the precision of the model weights which significantly reduces the GPU RAM required to load the model while still maintaining reasonable model quality. We decided this tradeoff was worth it because it allows one to fine tune Llama 3b without requiring high RAM GPUs. In addition, using the 4 bit model makes this research repeatable by others who may not have access to a high RAM GPU.

GPU RAM Usage	Model	GPU RAM Consumed
	Llama 3 Instruct 8b 4 Bit	6.1GB
	Llama 3 Instruct 8b	15.7 GB

2. Setting Up LoRA

We set up the LoRA adapters using the defaults from the LoRA paper. (Defaults used in the Llama-3.1 8b + Unsloth 2x faster finetuning.ipynb which is linked in the references)

R	Target Modules	Lora Alpha	Lora Dropout	Bias
16	q_proj, k_proj, v_proj,	16	0	None

	o_proj, gate_proj, up_proj, down_proj			
--	--	--	--	--

3. Experiment Configuration

The goal of this research is to understand how the size of a fine tuning dataset impacts the quality of the fine tuned model. However, we were also interested in the relationship between the number of fine tuning steps and the quality of the fine tuned model with respect to the size of the dataset. Therefore we split our dataset into subsets and for each subset we fine tuned the model using a variety of fine tuning steps. The number of fine tuning steps required for 1 full fine tune changed based on the size of the dataset. We decided to try 3 different lengths of training. 1 full fine tune for each dataset, 50 steps for each dataset and 150 steps for each dataset. This approach gave us a good understanding of whether the size of the dataset or length of training matters most with respect to the performance of the fine tuned model. The specific parameters we used are shown below.

Subset Configs	Subset %	Size Of Fine Tune Dataset
	100%	17,600
	50%	8,800
	10%	1,760
	1%	176
	0.5%	88

Fine Tune Step Configs	Fine Tune Steps
	50
	150
	1 Full Fine Tune Epoch

Size Of Fine Tune Dataset	Fine Tune Steps For 1 Full Training Epoch
17,600	2,200
8,800	1,100
1,760	220
176	22
88	11

All Experiments	Subset %	Size Of Fine Tune Dataset	Fine Tune Steps	Number Of Training Epochs
	100%	17,600	50	0.023
	100%	17,600	150	0.069
	100%	17,600	2,200	1.00
	50%	8,800	50	0.045
	50%	8,800	150	0.136
	50%	8,800	1,100	1.00
	10%	1,760	50	0.227
	10%	1,760	150	0.682
	10%	1,760	220	1.00
	1%	176	50	2.27
	1%	176	150	6.82
	1%	176	22	1.00
	0.5%	88	50	4.54
	0.5%	88	150	13.6
	0.5%	88	11	1.00

4. Creating Fine Tune Subsets

Creating the fine tune subsets well is crucial to ensure we can accurately assess the quality of our fine tuned model. At first, we simply sampled the subset % from our complete dataset but then we realized that the distribution of data in the subset would not match the distribution of our complete dataset. To fix this we created a temporary column in our dataset which combined the post type, post tone and post length.

For example, a row with a post tone of “professional”, post type of “promotional” and post length of “1-3_sentences” would be combined and added to this temporary column as “professional_promotional_1-3_sentences”.

We then used the train_test_split function from sci-kit learn to create a stratified subset on this new column. Therefore, our subsets had the same distribution of characteristics as the complete dataset. Finally, we dropped the temporary column and proceeded to fine tune the model.

5. Fine Tuning The Model

We used the Supervised Fine-Tuning Trainer (SFTT) to fine-tune our model. SFTT is particularly useful because it enhances our model's ability to generate coherent and contextually relevant text by refining its understanding of the data, rather than simply memorizing input-output pairs.

The parameters we used are shown below. (Defaults used in the Llama-3.1 8b + Unsloth 2x faster finetuning.ipynb which is linked in the references)

Training Arguments	per_device_train_batch_size	2
	gradient_accumulation_steps	4
	warmup_steps	5
	learning_rate	2e-4
	logging_steps	1
	optim	"adamw_8bit"
	weight_decay	0.1
	lr_scheduler_type	"linear"
	seed	3407
	output_dir	"outputs"

	num_train_epochs	1 if full fine tune else removed this argument
	max_steps	Fine tune steps (shown above) if not a full fine tune else removed this argument

SFTTrainer	dataset_num_proc	2
	packing	False
	args	Training Arguments

6. Generating Fine Tuned Responses

Once the model finished training we then used the fine tuned model to generate responses using our baseline prompts specified above. These fine tuned responses were then saved alongside the baseline responses for each experiment we ran so we could compare the two responses later on.

7. Saving The Models

Finally, we saved the LoRA adapters and the fine tuned responses to our Google Drive so we can reference them later. We chose to save the LoRA adapters instead of the whole model since we can easily load the same Unsloth model and attach the LoRA adapters which saves a significant amount of space when storing the models.

Step 4: Scoring Baseline Responses Vs Fine Tuned Responses

Once the model finished fine tuning we then passed the system prompt, user prompt, baseline responses and fine tune responses to Llama 3 70B to pick the best completion. Again, we leveraged the Groq api to access Llama 3 70B and score our responses since this significantly sped up the inference process and allowed us to leverage Llama 3 70B without a cluster of GPUs. Loading Llama 3 70b locally would require a significant amount of GPU RAM and slow down the inference process.

We tried a few different prompts to compare the baseline responses and fine tuned responses. First, we included a list of criteria that the model should consider and to rank each criteria for each response on a scale of 1 - 10. This method proved to be troublesome since the 1 - 10

score was quite arbitrary and the prompt was complex which led to variance in the response format which made parsing the response difficult.

We settled with a simpler prompt that asked the LLM to pick the best response based on a list of criteria and provide an explanation as to why it picked that response over the other. This simplified the prompt, removed the arbitrary scoring and gave the LLM an easier task of picking which response was better as opposed to scoring each of them, adding up the scores and then picking.

In order to parse the responses and save them appropriately we specified a specific format using special tokens for the LLM to follow. This made it significantly easier to parse the responses since we could locate the special tokens and extract the tokens in between to parse each part of the response.

The prompt we used to score the responses is shown below:

```
"You are tasked with evaluating the quality of two text-only social media sales posts based on the system's prompt, user's prompt, and the completions provided. Please use the following criteria to determine which post is overall better and provide a concise explanation of your choice."
```

```
### System Prompt:
"{system_prompt}"
```

```
### User Prompt:
"{user_prompt}"
```

```
### Completion 1:
"{completion1}"
```

```
### Completion 2:
"{completion2}"
```

Evaluate each completion based on these criteria:

1. **Message Clarity**: Core message clarity
2. **Brevity**: Conciseness
3. **Language Simplicity**: Ease of understanding
4. **Call to Action (CTA)**: Compelling CTA
5. **Emotional Appeal**: Emotional resonance
6. **Target Audience Fit**: Relevance to audience
7. **Timeliness**: Alignment with trends/events
8. **Tone of Voice Consistency**: Consistency with brand voice

```

9. **Benefit Highlighting**: Communication of benefits
10. **Unique Selling Point (USP)**: Effective communication of USP
11. **Creativity and Originality**: Uniqueness/creativity
12. **Social Proof**: Inclusion of testimonials/reviews
13. **Hashtag and Keyword Usage**: Effective use of hashtags/keywords
14. **Transparency**: Disclosure of partnerships/sponsorships
15. **Honesty**: Honesty in claims

### FORMAT RULES
1. Use special tokens [START] and [END] as instructed. Responses should be
   between these tokens.
2. Replace variables in **** with your outputs, excluding the ****.
3. Output only the best completion and a concise explanation. No extra text
   or labels.

### REQUIRED OUTPUT
[START_PICK]
'Baseline completion' or 'Finetune completion' (completion 1 = baseline,
completion 2 = finetune)
[END_PICK]

[START_EXPLANATION]
Explanation here (focus on key reasons for your choice).
[END_EXPLANATION]"

```

Analysis

Once all of the experiments were completed we had all of the data we needed saved and ready to analyze. We analyzed the performance of the fine tuned model, the effect of the number of fine tune steps with respect to performance and did a detailed analysis on how the fine tuned model performed with respect to post type, post length and post tone.

1. Performance Of The Fine Tuned Model

We were able to access the quality of each fine tuned model by looking at the number of times the LLM-as-a-judge picked the fine tuned response compared to the baseline response.

We had 95 total baseline responses that we compared to the 95 fine tuned responses for each experiment. If the LLM judge picked the fine tune response 70 times and the baseline response 45 then the performance ratio would be $75 / 40$ which is 1.56. Therefore, we could compare this performance ratio for each experiment to see which parameters yielded the best results. The higher the performance ratio the better the fine tuned model did.

We chose to use this performance ratio as the main measure of the performance of our fine tuned models since it directly compares the baseline responses against the fine tuned responses. This metric is also easy to implement in our LLM judgment since the LLM simply needs to pick one response over the other and give its explanation as opposed to coming up with a variety of individual scores to rank the responses on.

2. Fine Tune Steps Analysis

We stored and analyzed the training loss for each experiment to see whether the number of steps we used was sufficient. We then plotted all of the experiments with the same number of steps against each other to see if the training losses continued to drop over time or if they flattened out. This is useful since it allows us to see if the fine tuned model could have benefited from additional training or if we were overfitting the model. This analysis also gave us insight into how much training was needed with respect to the size of the dataset.

3. Number Of Training Epochs

The number of fine tune steps required for one full training epoch changed with the size of the dataset. We were interested in the relationship between the number of training epochs with respect to the performance of the fine tuned model. For example, would a smaller dataset with several training epochs perform better than a larger dataset with fewer training epochs? This analysis gives us insight into whether more data or more training is more impactful on the performance of the fine tuned model.

4. Detailed Analysis With Respect To Dataset Characteristics

The performance ratio mentioned above is useful to see how the fine tuned model performed overall however it only gave us limited insights. Therefore, we split up the results by post type, post length, post tone and a combination of all 3 to identify which characteristics our model succeeded at most and failed at most. This analysis provided significant insights into our model since we could see the results on a granular level.

This analysis was run for all experiments to get a detailed understanding of how the size of the dataset and number of fine tuning steps contributed to the performance of our fine tuned model.

Results

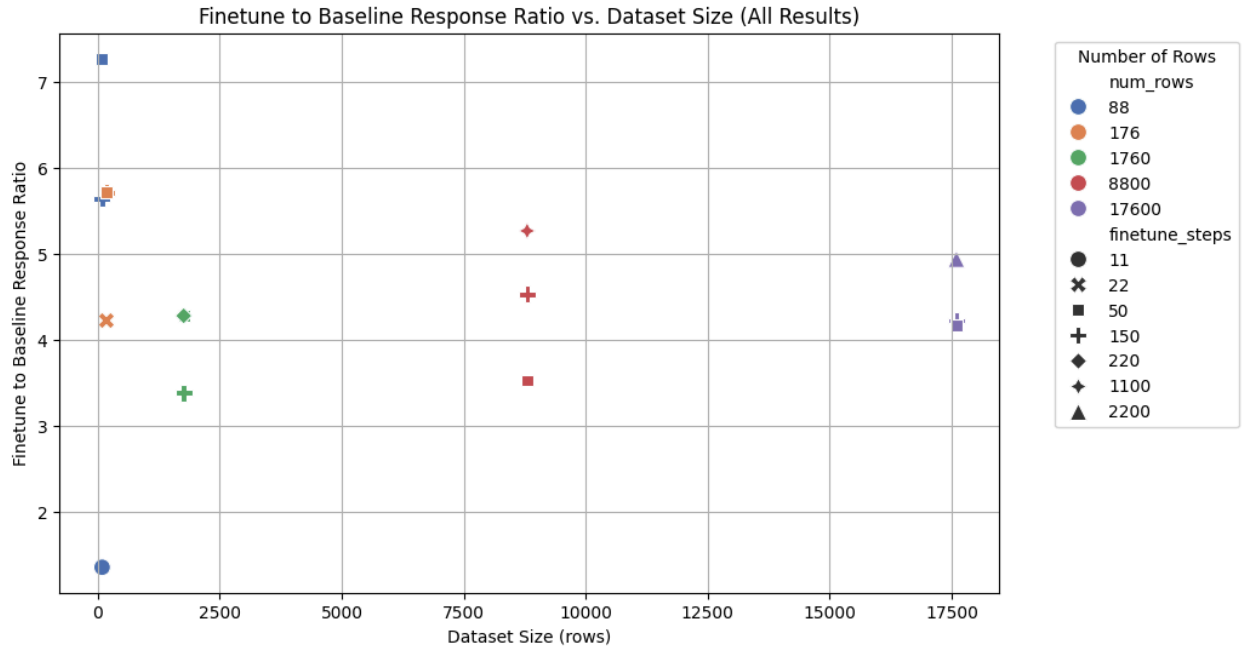
1. Performance Of The Fine Tuned Model

The chart below compares the performance of the fine tuned model with respect to the dataset size and the number of training epochs. Surprisingly, the best fine tuned model was trained on only 88 rows with a performance ratio of 7.27. However, it was trained for 50 steps which means that it had 4.54 full training epochs. For comparison, the full dataset with 17,600 rows trained on 150 steps performed significantly worse with a ratio of 4.22 but 150 steps is only 0.069 full training epochs. We expected to see the performance of the fine tuned model increase linearly with the size of the dataset however our results did not support this hypothesis. Instead, we see a much stronger

relationship between the number of training epochs with respect to the performance of the fine tuned model. We will explore this in more detail below.

Performance Ratio = Number of times the LLM judged picked the fine tuned response divided by the number of times the LLM judged picked the baseline response. The bigger the ratio the better that model did.

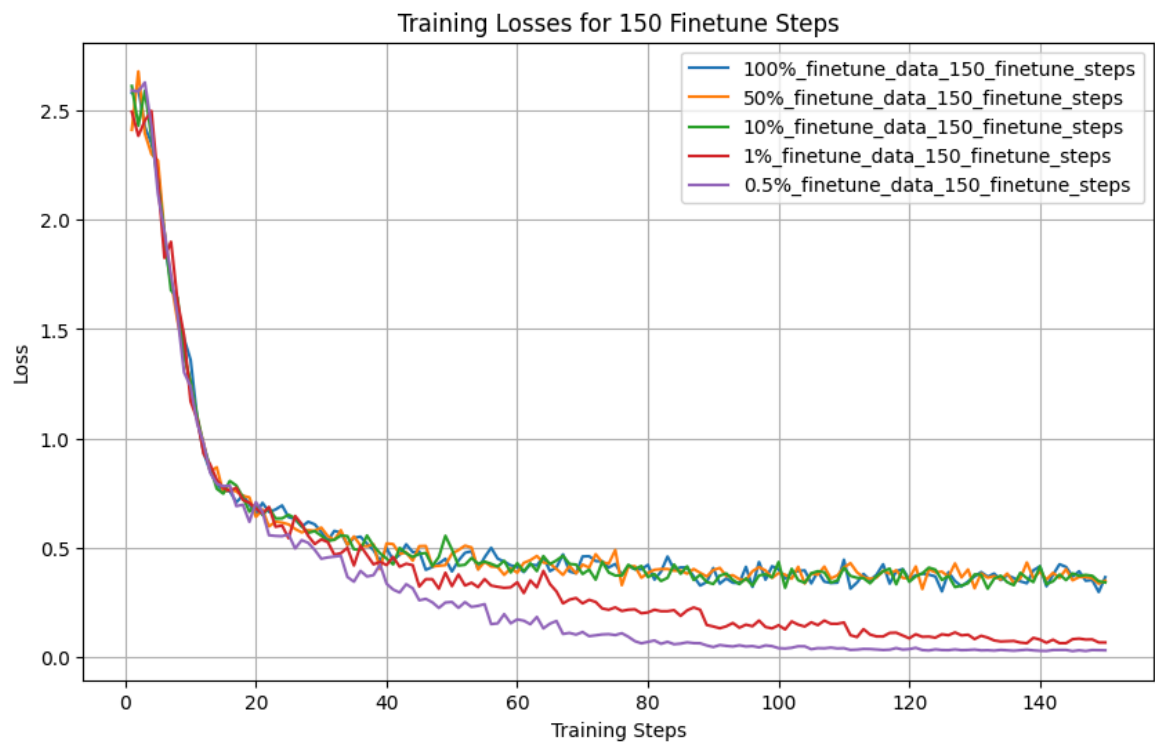
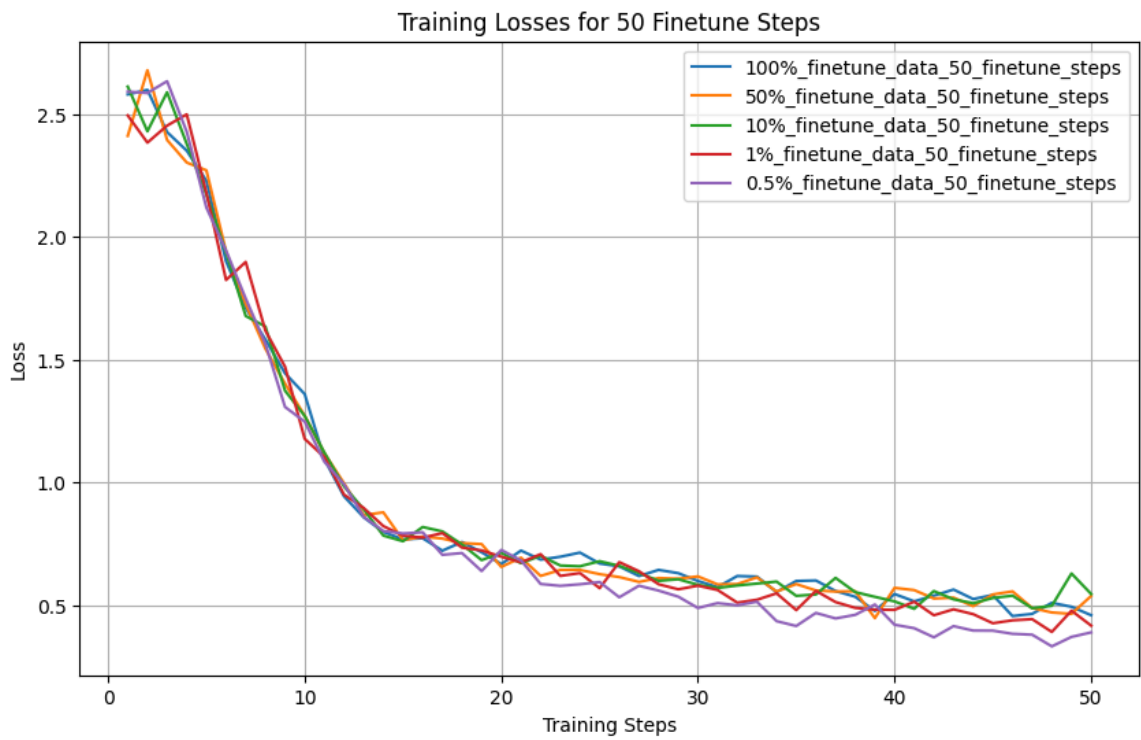
Subset %	Size Of Fine Tune Dataset	Fine Tune Steps	Number Of Training Epochs	Performance Ratio
0.5%	88	50	4.54	7.27
1%	176	50	2.27	5.71
1%	176	150	6.82	5.71
0.5%	88	150	13.6	5.64
50%	8,800	1,100	1.00	5.26
100%	17,600	2,200	1.00	4.93
50%	8,800	150	0.136	4.52
10%	1,760	50	0.227	4.27
10%	1,760	220	1.00	4.27
100%	17,600	150	0.069	4.22
1%	176	22	1.00	4.22
100%	17,600	50	0.023	4.16
50%	8,800	50	0.045	3.52
10%	1,760	150	0.682	3.38
0.5%	88	11	1.00	1.35

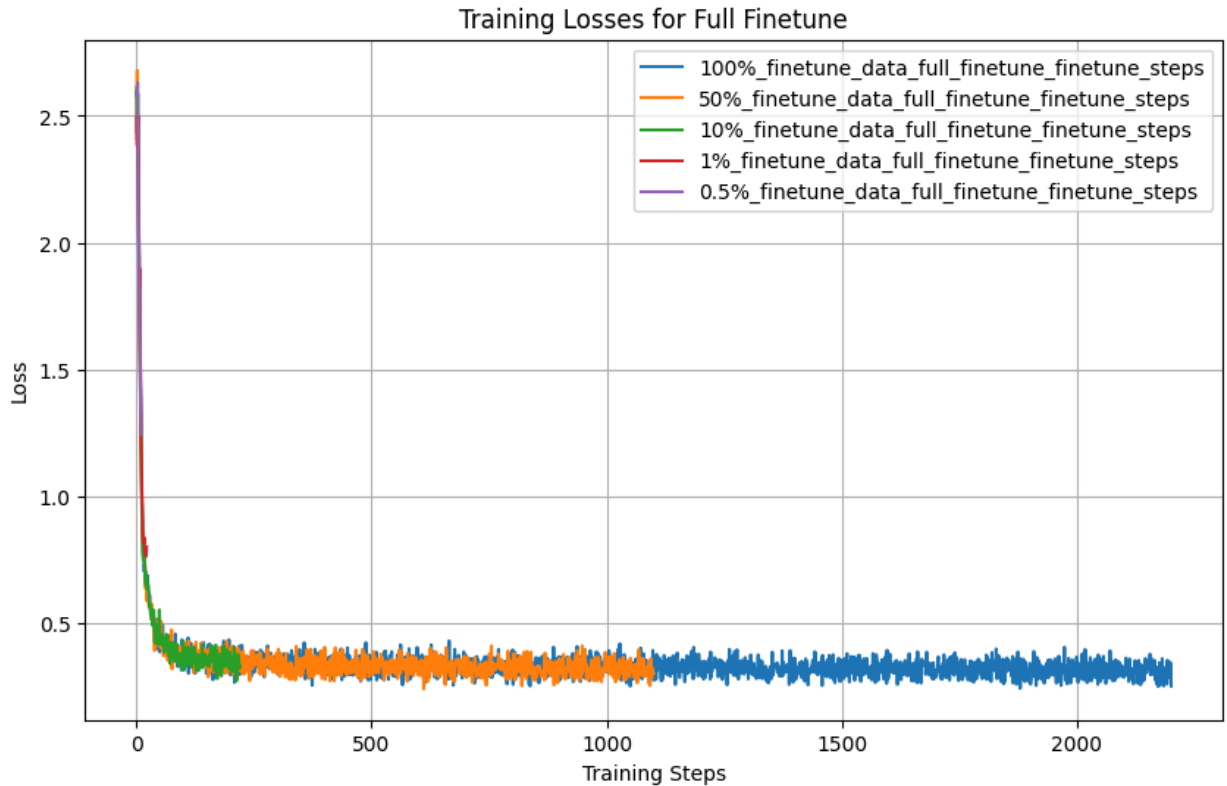


2. Fine Tune Steps Analysis

The charts below highlight the training loss during the training process which gives us insight into whether the fine tuned model is over trained or under trained. The charts show that each fine tuned model had a similar loss trajectory as training continued. The only noticeable deviation is on the 150 fine tune steps for the 1% and 0.5% subset models. This makes sense since 150 steps for these models is multiple full training epochs whereas 150 steps for the larger model is a fraction of a full training epoch. Also, it is worth noting that 1 full epoch for a small dataset may not be enough as shown by the still decreasing training loss when the training ends for smaller models. Overall, the smaller the dataset the more full training epochs that fine tuned model will need to perform compared to larger datasets which need less full training epochs for comparable performance. This does not take into account the RAM and time required to fine tune the

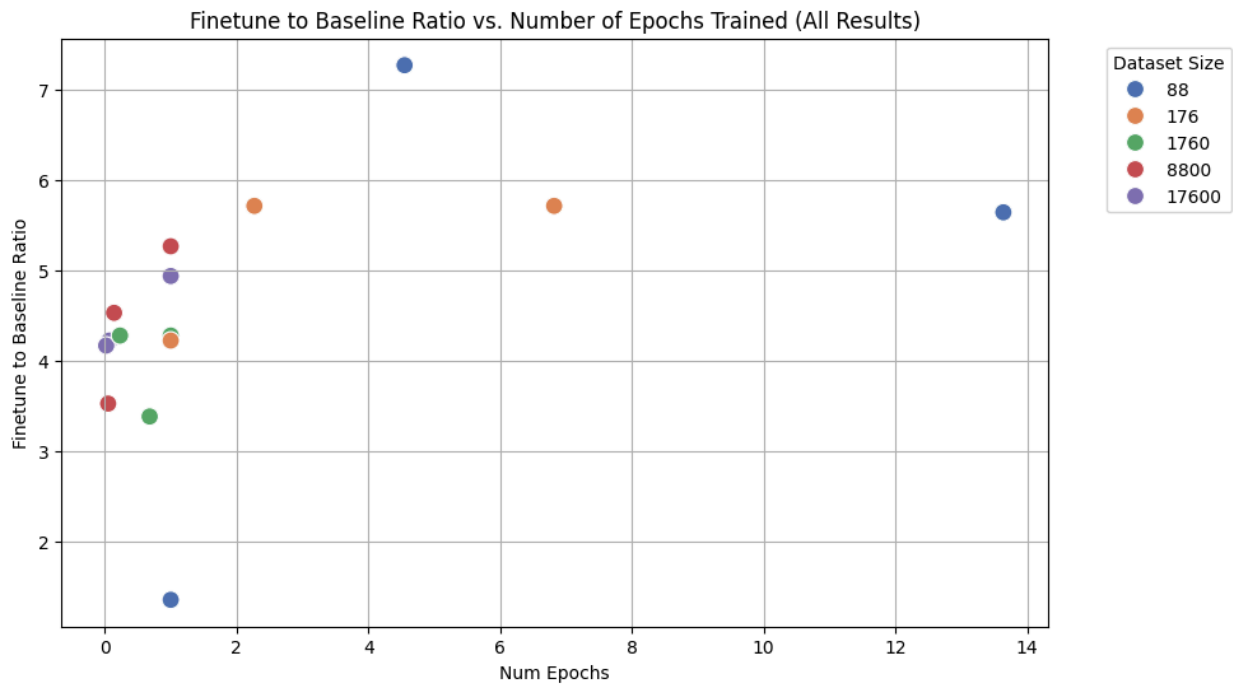
models therefore these tradeoffs are worth considering on a case by case basis.





3. Number Of Training Epochs

The chart below shows the impact on the number of training epochs with respect to the performance of the fine tuned model. There is a strong relationship between more training epochs and better performance regardless of the dataset size. However, this analysis does not take into account how the fine tuned model would perform on other tasks unrelated to writing sales social media posts. There is a risk of degrading the general performance of the base model as the fine tuning steps increase. However, for our specific task it is easy to see that the number of training epochs has a linear relationship with the performance of the fine tuned model.



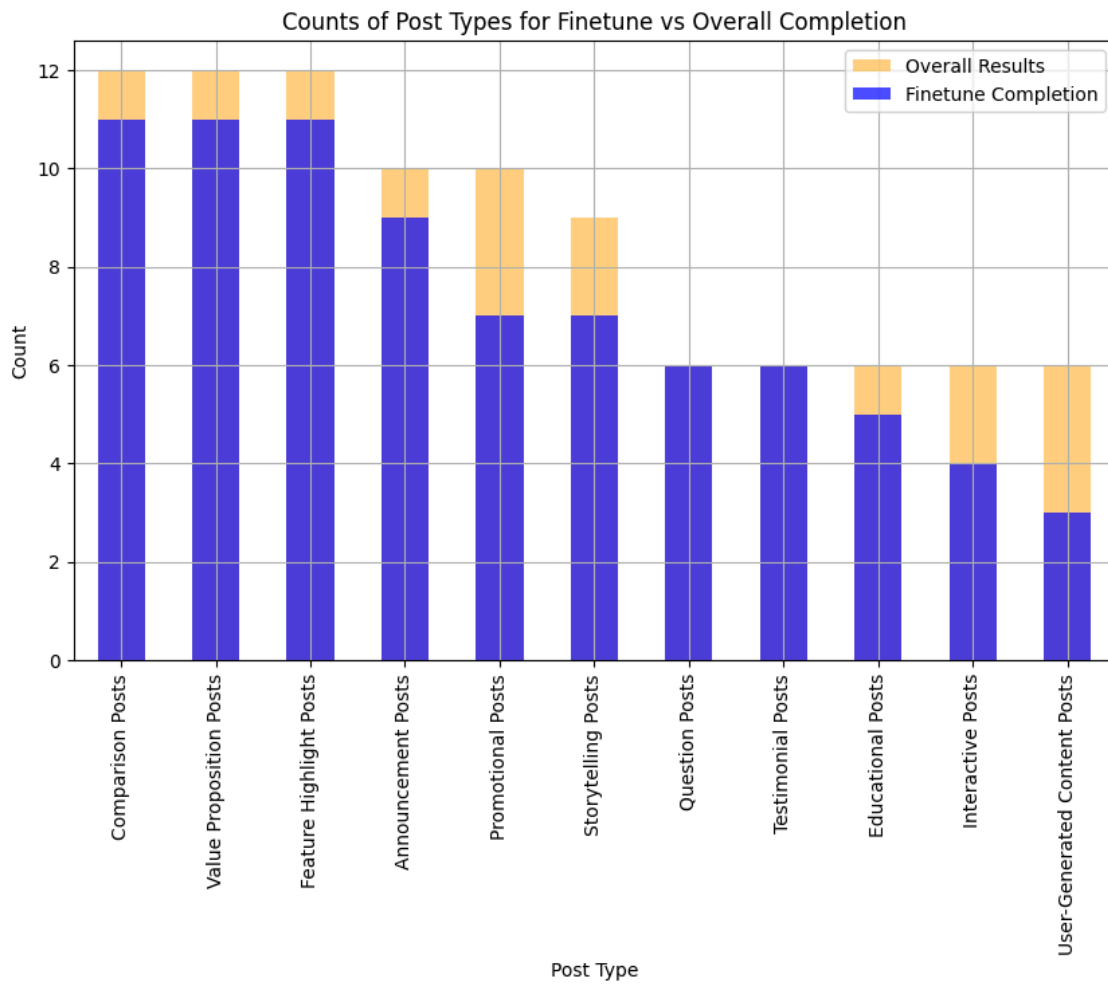
4. Detailed Analysis With Respect To Dataset Characteristics

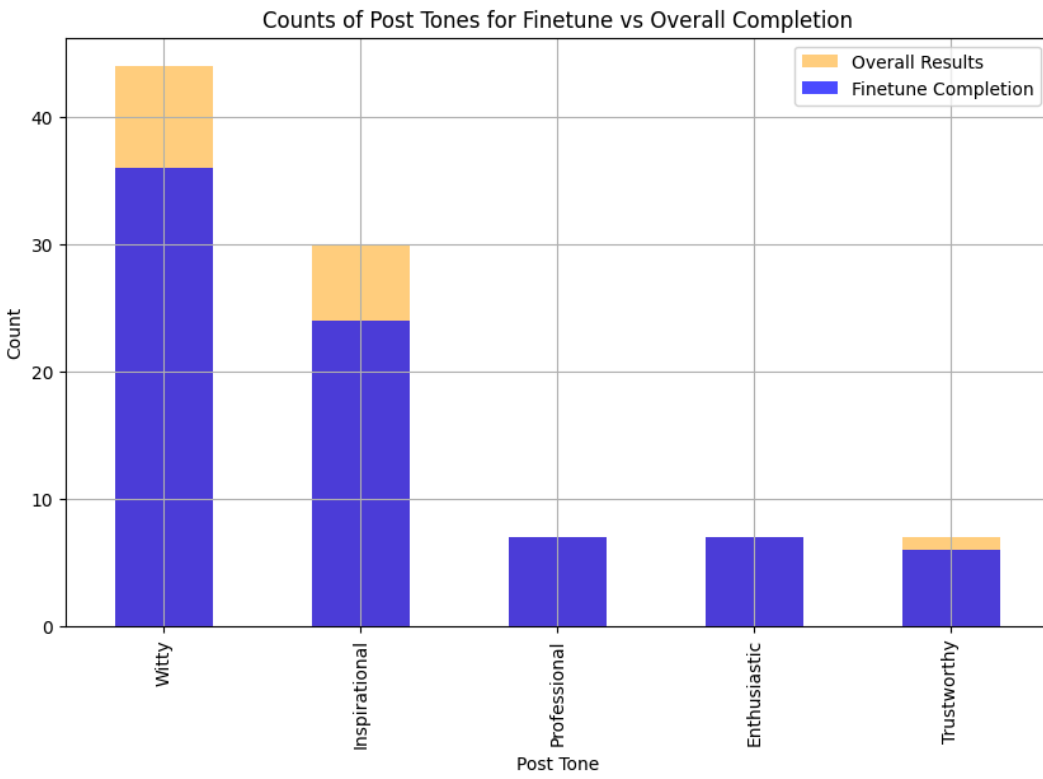
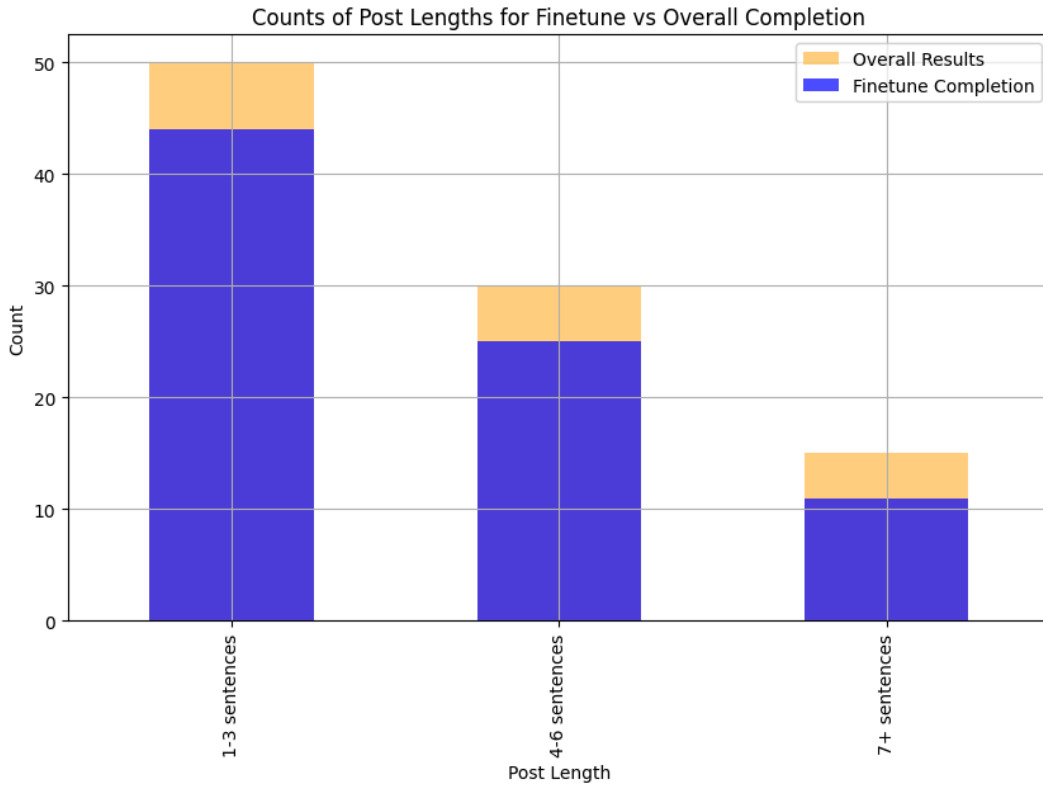
The charts below are a detailed analysis of the best performing model. The blue bars highlight the number of times the LLM judge selected the fine tuned response with respect to the dataset characteristics. The yellow bar represents the total number of data points with respect to the characteristics. Therefore, the difference between the blue bars and the yellow bars represent when the LLM judge selected the baseline response. These charts let us see for which characteristics our fine tuned model was strong or weak with respect to the baseline model.

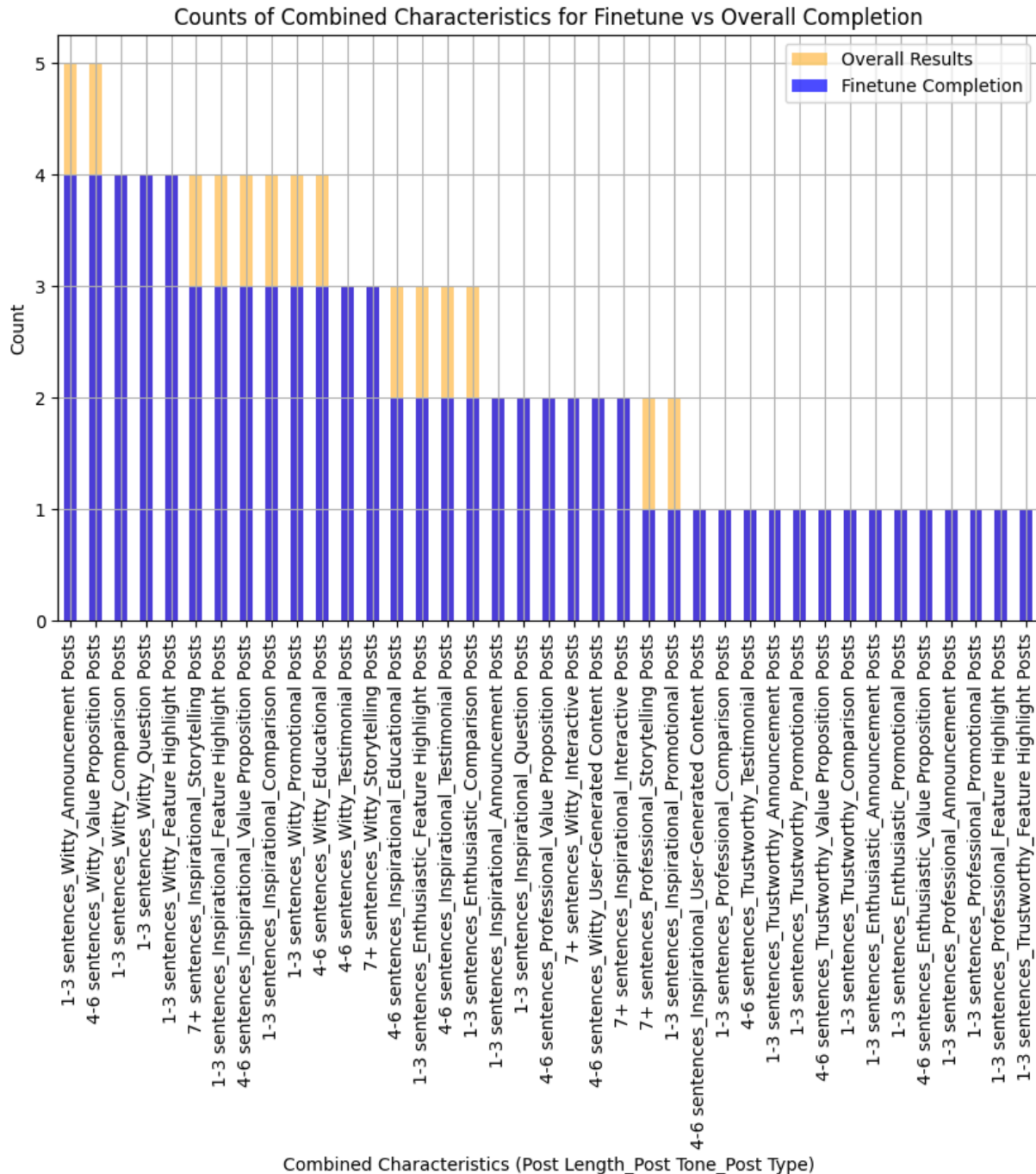
We can see that our fine tuned model performed well across the board. The fine tuned model performed worst with respect to witty tones, inspirational tones, interactive posts, user generated posts and promotional posts. The fine tuned model performed best with respect to question posts, testimonial posts, value proposition posts, comparison posts, feature highlight posts, professional tone, enthusiastic tone and all post lengths.

Overall, the fine tuned model performed well across the board without any characteristics significantly lagging behind.

Best Model Specs	Subset %	Size Of Fine Tune Dataset	Fine Tune Steps	Number Of Training Epochs	Performance Ratio
	0.5%	88	50	4.54	7.27







Conclusions


So how much data do you really need to fine tune Llama 3 Instruct 8b Instruct? Turns out, not that much. Our fine tuned model trained on 88 rows of diverse data was able to outperform another fine tuned model trained on 17,600 rows of diverse data. However, the main difference was the number of training epochs the model was fine tuned on. Although our best model was only trained on 88 rows it did 4.6 full training epochs which allowed the model to learn the

characteristics of our data. The models that trained on 17,600 rows got at most 1 full training epoch and performed worse overall than a model trained on a small dataset with more training epochs. This result surprised us since smaller datasets are not only easier to acquire / generate but also train faster. This means that businesses looking to integrate specialized fine tuned open source models into their work flows are able to do it with relative ease.

There are some limitations to our experiments that should be mentioned such as the specific task at hand and the quality of the fine tuned models on tasks outside our primary task. This research focuses on one specific narrow task and therefore we can not say conclusively that these results will hold up for various other tasks. However, the task we chose is a practical use case that a business might implement and therefore it serves as a guide for those looking to conduct similar experiments. Also, our baseline testing did not include any other tasks outside the task that we were measuring therefore we can't say for certain that we did not over train the model and cause model collapse for other tasks. However, this research was focused on implementing a fine tuned open source LLM for a narrow task since in the real world there are most likely many LLMs working together as part of any one system.

Overall, dataset size was less important than the number of full training epochs on the fine tuned model performance for our narrow task. A fine tuned model trained on a small dataset measured in the 100s with sufficient training epochs will outperform the base model for a narrow task.

References

1. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., & Chen, W. (2021). LoRA: Low-Rank Adaptation of Large Language Models. arXiv. <https://arxiv.org/abs/2106.09685>
2. Zhong, Z., Zhong, L., Sun, Z., Jin, Q., Qin, Z., & Zhang, X. (2023). SyntheT2C: Generating Synthetic Data for Fine-Tuning Large Language Models on the Text2Cypher Task. arXiv. <https://arxiv.org/abs/2406.10710>
3. Unsloth. (2024). "Llama-3.1 8b + Unsloth 2x Faster Finetuning". *Google Colaboratory*. Retrieved from  Llama-3.1 8b + Unsloth 2x faster finetuning.ipynb