

Fine-tuning Llama-2-7b on Customer Support Interactions

Adam Sam

Department of Computer Science and Networking

Wentworth Institute of Technology

550 Huntington Ave, Boston, MA 02115

sama1@wit.edu

Awele Catherine Okolie

Department of Computer Science and Networking

Wentworth Institute of Technology

550 Huntington Ave, Boston, MA 02115

okoliea@wit.edu

Abstract—Pretrained large language models (LLMs) can provide strong general-use language generation applications. However, their general purpose nature often limits their effectiveness in specialized domains where context-aware and task-specific responses are required. We propose a fine-tuned variant of MetaAI’s Llama-2-7B model using a comprehensive dataset of real-world customer support conversations to improve its ability to generate high-quality responses in a support setting. To maximize training efficiency, we employ QLoRA (Quantized Low-Rank Adaptation) for efficient training. We employ half-precision floating-point (FP16) to accelerate training, reducing computation time at a minimal cost to numerical precision. We utilize a remote lab machine equipped with dual NVIDIA RTX 3090 GPUs, each providing 24GB of VRAM, to refine our model on a domain-specific dataset. Compared to the baseline model, our fine-Tuned model demonstrated successful improvements from the base model on the task of customer support interactions using ROUGE-scoring as a metric. Overall, our work lays a foundation for such advancements in efficient LLM adaptation in the domain of generative customer-support interaction.

Index Terms—natural language processing, large language models, generative text, llama-2, customer support, chat-bot, fine-tuning

I Introduction

Pretrained large language models (LLMs) can provide strong general-use language generation applications. These models, trained on vast datasets, excel at a wide range of natural language processing (NLP) tasks, including text completion, summarization, and classification. However, their general purpose nature often limits their effectiveness in specialized domains where context-aware and task-specific responses are required, hence the need for pretrained models to be further fine-tuned on a specific task.

A key challenge arises particularly in applications such as customer support chatbots, where responses must be not only coherent, but also contextually relevant and aligned with its organization’s policies. Gen-

eral LLMs may generate unhelpful or irrelevant replies when faced with domain-specific queries, leading to poor user experiences and potential reputational risks for businesses. Additionally, without fine-tuning, these models may struggle to adopt the appropriate tone, style, and terminology expected in a professional customer service environment.

We propose a fine-tuned variant of MetaAI’s Llama-2-7B model using a comprehensive dataset of customer support conversations to improve its ability to generate high-quality responses in a support setting. The Llama-2-7B model is chosen for its relatively low computational requirements and its relatively open license for both commercial and research use [1].

To maximize training efficiency, we employ QLoRA (Quantized Low-Rank Adaptation), an advanced parameter-efficient fine-tuning (PEFT) technique that combines 4-bit quantization with Low-Rank Adapters (LoRA). This significantly reduces memory usage and computational usage while preserving model accuracy. This approach allows us to fine-tune Llama-2-7B on consumer-grade GPUs without sacrificing the model’s ability to generate high-quality responses.

```
def merge_features(row):
    row['text'] = f"[INST] {row['instruction']} [/INST] {row['response']}"
    row['text'] = tokenizer.bos_token + row['text'] + tokenizer.eos_token #add bos and eos tokens
    return row
```

Fig. 1. Preprocessing the data to align with Llama-2 formatting requirements. "Row" represents a singular sample row in the dataset.

By fine-tuning this model on a comprehensive dataset of real-world customer support conversations, we aim to improve its ability to generate accurate, context-aware, and appropriate responses, thereby improving the efficiency and effectiveness of automated customer service systems.

II Model

To adapt the base Llama-2-7B model for customer support applications, we fine-tuned it on a domain-specific dataset using QLoRA (Quantized Low-Rank Adaptation) for efficient training [2].

1) Dataset

We used the Bitext-customer-support-llm-chatbot-training-dataset from HuggingFace, which consists of over 26,000 samples of real-world customer support interactions. Each sample includes a customer query field paired with an agent response field, ensuring that the model learns domain-specific language, troubleshooting patterns, and professional communication styles. These two fields were merged together for full text processing.

Additionally, input sequences were formatted according to the model’s requirements to maintain a clear distinction between user queries and agent responses. Since Llama-2 relies on special beginning (BOS) and end-of-sequence (EOS) tokens for structured text generation, we modified the dataset to ensure proper tokenization [3]. Specifically, we appended the `</s>` token at the end of each response to help the model learn when to end conversations. Similarly, the `<s>` was appended at the beginning to denote the beginning of conversations. The tokens, `[INST]` and `[/INST]` enclose user messages within a conversation. `<<SYS>>` and `<</SYS>>` tokens represent LLM-directed instructions for the response, but since these instructions were not included in the dataset, we did not need to consider them. On a coding level, this formatting is seen in Figure 1.

For evaluation, we split the dataset using a 0.2 training and testing ratio, meaning we reserve 20% of the data for model evaluation on newly seen data.

2) Trainer Configuration

For the quantization aspect of QLora, we used 4-bit quantization from the ”BitsAndBytes” library developed by Hugging Face.

This is used in conjunction with the LORA configuration from the ”peft” library [4]. These techniques are applied to the base model through an ”SFTTrainer” (Supervised Fine-Tuning) instance from the ”TRL” library, which provides a streamlined interface for fine-tuning models using QLora [5]. We employ half-precision floating-point (FP16) to accelerate training, reducing computation time at a minimal cost to numerical precision. While FP16 can minimally impact accuracy, we found it to be an effective trade-off for faster training times.

3) Hyper-parameters

To experiment with the impacts of different model parameters, we experimented with different values on a reduced dataset of ~ 1500 samples. We found the following configuration to be suitable:

- LORA Ranks: 32
- LORA Alpha: 16
- LORA Dropout Rate: 0.05
- Optimizer: AdamW-Torch optimizer; this is a common optimizer used for fine-tuning.
- Weight Decay: 0.0001
- Epochs: 1
- Learning Rate: $2e-5$
- Per Device Batch Size: 6; we found that

| | | | | |
|------------|-------------------------|-------------|----------------------|----------------------|
| 0 | NVIDIA GeForce RTX 3090 | On | 00000000:41:00.0 Off | N/A |
| 63% | 61C P2 | 204W / 350W | 10034MiB / 24576MiB | 0% Default |
| 1 | NVIDIA GeForce RTX 3090 | On | 00000000:61:00.0 On | N/A |
| 64% | 67C P2 | 307W / 350W | 14434MiB / 24576MiB | 100% Default |
| | | | | N/A |
| Processes: | | | | |
| GPU ID | CI ID | PID | Type | Process name |
| 0 | N/A | 112751 | G | /usr/lib/xorg/Xorg |
| 0 | N/A | 717493 | C | /usr/bin/python3 |
| 1 | N/A | 112751 | G | /usr/lib/xorg/Xorg |
| 1 | N/A | 112942 | G | /usr/bin/gnome-shell |
| 1 | N/A | 717493 | C | /usr/bin/python3 |

Fig. 2. GPU utilization was monitored through nvidia-smi.

| | Metric | Baseline | Finetuned |
|---|------------|----------|-----------|
| 0 | ROUGE-1 | 0.151931 | 0.458237 |
| 1 | ROUGE-2 | 0.037789 | 0.185254 |
| 2 | ROUGE-L | 0.127279 | 0.299905 |
| 3 | ROUGE-Lsum | 0.097515 | 0.312217 |

Fig. 3. ROUGE scores for both baseline and fine-tuned inference.

this was the highest size achievable before maxing out VRAM.

-Gradient Accumulation Steps: 8

Training the on reduced dataset took ~ 20 minutes, while training on the full dataset took ~ 1 hour.

4) Training Hardware

To efficiently fine-tune our model, we leveraged a remote lab machine equipped with dual NVIDIA RTX 3090 GPUs, each providing 24GB of VRAM. This setup was accessed via SSH, allowing us to conduct training sessions remotely. VRAM usage was monitored using NVidia’s SMI interface (Figure 2), which was crucial in determining the optimal batch size.

5) Evaluation Metrics

To evaluate the performance of our fine-tuned model, we employed ROUGE (Recall-Oriented Understudy for Gisting Evaluation) scores, a standard metric for assessing the quality of text generation models. ROUGE compares the overlap of n-grams between the model-generated responses and human-written reference responses; in our case, the references are the testing split of the dataset [6]. Within the test set, representing the human-written queries and responses, we obtain the customer query as the model prompt and agent response as the reference. We feed both the baseline and fine-tuned models with the the obtained customer prompts and intuitively compare the quality of their results to the agent response reference.

We specifically tracked the following ROUGE metrics:

ROUGE-1: Measures unigram (single-word) overlap, indicating surface-level similarity.

ROUGE-2: Assesses bigram overlap, measuring deeper coherence.

ROUGE-L: Evaluates the longest common subsequence, reflecting semantic similarity and response structure.

ROUGE-scoring was chosen as a metric in particular since it aligns with customer support needs by quantifying how closely generated responses match human responses.

6) Prompt-Engineering

To determine an effective prompt for intuitively evaluating our model we constructed our prompt based on the following conventional model [7]:

- Persona: Identify as a customer support chat-bot
- Instruction: Guide the customer in answering queries
- Context: Information pertaining to the company in which the chatbot is employed
- Format: Directing the model to generate EOS tokens
- Tone: Formal and friendly tone
- User Query: The customer's query

III Results

Presented in Figure 3, our fine-tuned model showed improved ROUGE scores compared to the base Llama-2-7B model, particularly in the ROUGE-2 score, confirming its enhanced ability to generate improved customer-support responses.

To obtain an intuitive overview of the generated text, we fed both the baseline and fine-

```
#PROMPT ENGINEERING
persona = "You are a helpful and respectful customer support assistant.\n"
instruction = "Provide an appropriate response to the customer inquiry.\nAlways answer as helpfully as possible.\n"
context = f"You are employed in a shipping company.\n"
data_format = "End the response with '\n</ss>'.\n"
tone = "The tone should be formal and helpful.\n"

sys = persona + instruction + context + data_format + tone

user_query = "Is there a customer support email where I can reach an agent?"

prompt = f"<ss>[INST] <SYS></ss>{sys}</ss>\n\n{user_query} [/INST]"
prompt = clean_prompt(prompt)
#prompt = user_query
result = pipe(prompt)
print(result[0]['generated_text'])
```

Fig. 4. Engineered prompt fed to the model.

tuned models with our engineered prompt, seen in Figure 4. We initialized both models with the following parameters:

- Temperature: 0.3
- Top K: 15
- Top P: 0.6
- Max Length: 250

Because of our application for customer-support, we chose a relatively low temperature value to minimize randomness and ensure consistent, factual, and on-topic answers. This is essential for maintaining professionalism in customer-support interactions. By restricting Top K sampling to the top 15 most likely tokens at each step, and setting a slightly high Top P value (nucleus sampling), we balance diversity and relevance by allowing slight variability in phrasing.

Passing the prompt into the baseline model, it was apparent that it generated garbage results (Figure 5), as it repeatedly generated the customer's query, with no attempt at forming

```

<s>[INST] <<SYS>>You are a helpful and respectful customer support assistant.
Provide an appropriate response to the customer inquiry.
Always answer as helpfully as possible.
Please ensure that your responses are socially unbiased and positive in nature.
You are employed in a shipping company.
End the response with "</s>".
The tone should be formal and helpful.
<</SYS>>
Is there a customer support email where I can reach an agent? [/INST]
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent?
<</SYS>>
Is there a customer support email where I can reach an agent

```

Fig. 5. The output of the baseline model - notice the repetition of output.

```

<s>[INST] <<SYS>>You are a helpful and respectful customer support
assistant.
Provide an appropriate response to the customer inquiry.
Always answer as helpfully as possible.
Please ensure that your responses are socially unbiased and positive in
nature.
You are employed in a shipping company.
End the response with "</s>".
The tone should be formal and helpful.
<</SYS>>
Is there a customer support email where I can reach an agent? [/INST]
Our customer support team is available to assist you via email at
{{Customer Support Email}}. We aim to respond to your queries within 24
hours. If you need immediate assistance, you can also reach us at
{{Customer Support Phone Number}}.
We appreciate your interest in our products and services, and we're here
to help you with any questions or concerns you may have.
Thank you for contacting us! 😊"

```

Fig. 6. The fine-tuned model generated coherent responses.

an answer.

Passing the prompt into the fine-tuned model showed significantly improved results (Figure 6), as it successfully made the distinction between the customer's query, and the chatbot's response. It generated a complete response, providing a support email as well as a phone number, topped off with a smiling emoji. Noting the various flags in the generated response, such as ((Customer Support Email)) - these flags are intended by the BiText dataset that the model was trained on. The flags allow customization by the business, as the output can be further

processed by replacing the flags with the business' choice of values [8].

Overall, the fine-tuned model demonstrated successful improvements from the base-model on the task of customer-support interactions.

IV Future Work

While our fine-tuned model demonstrates promising performance in customer support applications, further optimizations may enhance its capabilities. Due to time constraints, we were unable to experiment in-depth with different configurations for our model, and we were only limited to training on one epoch.

Particularly, one deficiency of our current implementation is the lack of a method for detecting overfitting during training. While we trained for a single epoch to balance performance and computational efficiency, this approach did not include validation-based early stopping. Implementing a systematic search method (such as Random Search) over different variations of learning rates, weight decays, and LORA ranks could help better optimize our model's hyperparameters during training.

Currently, the model is only limited to

text generation. Integrating a ReAct framework would extend the model by allowing it to perform dynamic actions through the use of external tools, such as looking up order details, escalating tickets, or querying databases, while reasoning through steps, bridging the gap between conversational AI and workflow automation [9].

Additionally, the model generates responses in isolation without retaining context from previous interactions. Implementing a memory buffer would allow the model to maintain conversation history — a critical feature for customer-support chatbots.

Future efforts could prioritize these deficiencies to create a better model suitable to be deployed for businesses. Our current work lays a foundation for such advancements in efficient LLM adaptation in the domain of generative customer-support interaction.

References

- [1] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, “Llama 2: Open foundation and fine-tuned chat models,” 2023.
- [2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, “Lora: Low-rank adaptation of large language models,” 2021.

- [3] Meta, “Meta llama 2.”
- [4] HuggingFace, “Peft.”
- [5] HuggingFace, “Supervised fine-tuning trainer.”
- [6] HuggingFace, “Rouge.”
- [7] J. Alammam and M. Grootendorst, *Hands-On Large Language Models*. “O’Reilly Media, Inc.”, Sept. 2024.
- [8] Bitext, “Bitext-customer-support-llm-chatbot-training-dataset.”
- [9] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “React: Synergizing reasoning and acting in language models,” 2023.