Technical Report: Fine-Tuning LLaMA Using QLoRA for Symptom-Based Medical QA

1. Introduction

Recent advances in large language models (LLMs) such as Meta's LLaMA have demonstrated remarkable potential across a wide range of natural language processing tasks. In the healthcare domain, LLMs hold promise for assisting with patient triage, clinical decision support, and educational question-answering systems. This report provides an in-depth analysis of the methodology, results, and implications of fine-tuning a LLaMA-based model using QLoRA (Quantized Low-Rank Adaptation) for the specific task of answering symptom-based medical questions. The primary objective is to develop a specialized language model capable of generating accurate and coherent answers in response to user queries about medical symptoms.

This work is particularly relevant in resource-constrained settings, where deploying full-precision large models is often impractical due to high memory and computational demands. By leveraging QLoRA, we aim to significantly reduce these requirements while preserving, and ideally enhancing, the model's performance on the targeted task. This report outlines a reproducible and scalable approach for domain-specific LLM fine-tuning, with a focus on addressing the challenges and opportunities in the medical question-answering domain.

2. Methodology and Approach

The project employs a structured fine-tuning approach, optimizing a pre-trained LLM for symptom-based medical question answering. The key steps are detailed below:

2.1 Dataset Preparation

- **Data Sourcing and Filtering**: A custom dataset was curated from a CSV file containing user-generated medical questions. These questions were labeled by type (e.g., symptoms, treatments, causes). To ensure the model focused on the intended task, only entries tagged as "symptoms" were selected using Pandas operations. This filtering step was crucial for creating a dataset highly relevant to symptom identification and diagnosis.
- **Prompt Engineering**: Each data sample was transformed into a prompt-completion format, which is standard for causal language modeling. This structured prompting strategy encourages the model to explicitly recognize the question type before generating an answer, thereby improving contextual

understanding and response accuracy.

- **Data Formatting and Loading**: The processed dataset was saved in JSONL format, a common choice for efficiently storing large amounts of text data. Hugging Face's load_dataset API was used to load the data, providing convenient data management and integration with the training pipeline.
- **Data Splitting**: To ensure robust evaluation, the dataset was divided into training, validation, and test sets using stratified sampling. This technique was used to maintain a balanced distribution of medical subtopics across the splits, preventing any single subset from being skewed towards a particular area of medicine. The dataset was split into 80% training, 10% validation, and 10% test subsets.

2.2 Model and Tokenizer Configuration

- **Base Model Selection**: The meta-llama/Llama-2-1B-Instruct model was chosen as the base LLM. This model was selected because it is instruction-tuned. The 1B parameter size offers a balance between performance and computational efficiency.
- **Tokenizer Initialization**: The tokenizer corresponding to the LLaMA model was initialized. Crucially, a padding token was configured (either as the eos_token or an explicit [PAD] token). This step is essential to prevent the model from calculating loss on padded tokens, which would otherwise skew the training process.

2.3 Quantization and QLoRA Configuration

- **Quantization**: Fine-tuning large language models often demands substantial computational resources. To mitigate this, we employed 4-bit quantization using the BitsAndBytesConfig module from the bitsandbytes library. Quantization reduces the memory footprint of the model by representing its weights with fewer bits, enabling fine-tuning on less powerful hardware.
- **QLoRA**: To further enhance efficiency, we used QLoRA. QLoRA introduces small, trainable adapter modules within the layers of the transformer network. This technique allows us to freeze the original pre-trained model's weights and only update the weights of these low-rank adapters during fine-tuning. This dramatically reduces the number of trainable parameters, leading to significant memory savings and faster training.
  Key QLoRA parameters:
  - Rank (r): 8 - The dimensionality of the low-rank matrices.
  - LoRA Alpha: 16 - A scaling factor for the adapter outputs.
  - Dropout: 0.1 - Dropout probability for regularization.

    - Target Modules: q_proj, v_proj - The specific transformer layers to which the adapters are added (query and value projection layers).
- **Model Preparation**: The base model was prepared for quantized training using the prepare_model_for_kbit_training function. The model was then wrapped with get_peft_model, which integrates the QLoRA adapters and enables memory-efficient gradient updates during training.

2.4 Fine-Tuning Procedure

- **Trainer Configuration**: Hugging Face's SFTTrainer (Supervised Fine-Tuning Trainer) was used to streamline the fine-tuning process. The following hyperparameters were used:
    - Epochs: 5 - The number of times the training data is iterated through.
    - Learning Rates: [1e-4] - The step size for updating model weights.
    - Batch Sizes: [2, 4] - The number of training examples processed in each iteration.
    - Gradient Accumulation: 5 steps - A technique to simulate larger batch sizes when memory is limited.
    - Logging Steps: 10 - Frequency of logging training progress.
    - Save Steps: 50 - Frequency of saving model checkpoints.
    - Warmup Steps: 10 - Gradually increasing the learning rate at the beginning of training.
    - Mixed Precision: Enabled (fp16) - Using lower-precision floating-point numbers to accelerate training and reduce memory usage.
- **Tokenization and Label Masking**: A custom formatting function was used to tokenize the input question-answer pairs. Crucially, the labels (target tokens) were carefully masked. This masking ensures that the loss function only considers the completion (answer) tokens and not the tokens in the prompt (question), which is essential for causal language modeling.

2.5 Evaluation Framework

A comprehensive evaluation pipeline was established to assess the fine-tuned model's performance. The pipeline includes the following metrics and analyses:

- **Average Cross-Entropy Loss**: A measure of how well the model's predicted probability distribution aligns with the true distribution of the target tokens. Lower values indicate better performance.
- **Perplexity**: Calculated as the exponential of the cross-entropy loss. Perplexity provides a more intuitive measure of how "surprised" the model is by the test data. Lower perplexity scores indicate that the model is more confident and

accurate in its predictions.

- **Error Analysis**: A qualitative analysis involving a comparison of the model's predicted completions with the actual (ground truth) completions. This analysis helps to identify common types of errors, such as factual inaccuracies, underspecification, or misinterpretations of the input.
- **Histogram Analysis**: A quantitative analysis focusing on the length of incorrect predictions. By plotting the distribution of lengths for erroneous responses, we can gain insights into whether the model's errors are correlated with the brevity or verbosity of its answers. This can help in identifying potential biases or limitations in the model's reasoning.
  This evaluation framework enables rapid experimentation and debugging while ensuring reliable and informative performance reporting.

3. Results and Analysis

The results of the fine-tuning process and the subsequent evaluation are presented below:

3.1 Hyperparameter Search Outcomes

- **Hyperparameter Trials**: The following table summarizes the results of the hyperparameter search:

Trial	Learning Rate	Batch Size	Final Loss
1	1e-4	2	1.4230
2	1e-4	4	1.5531

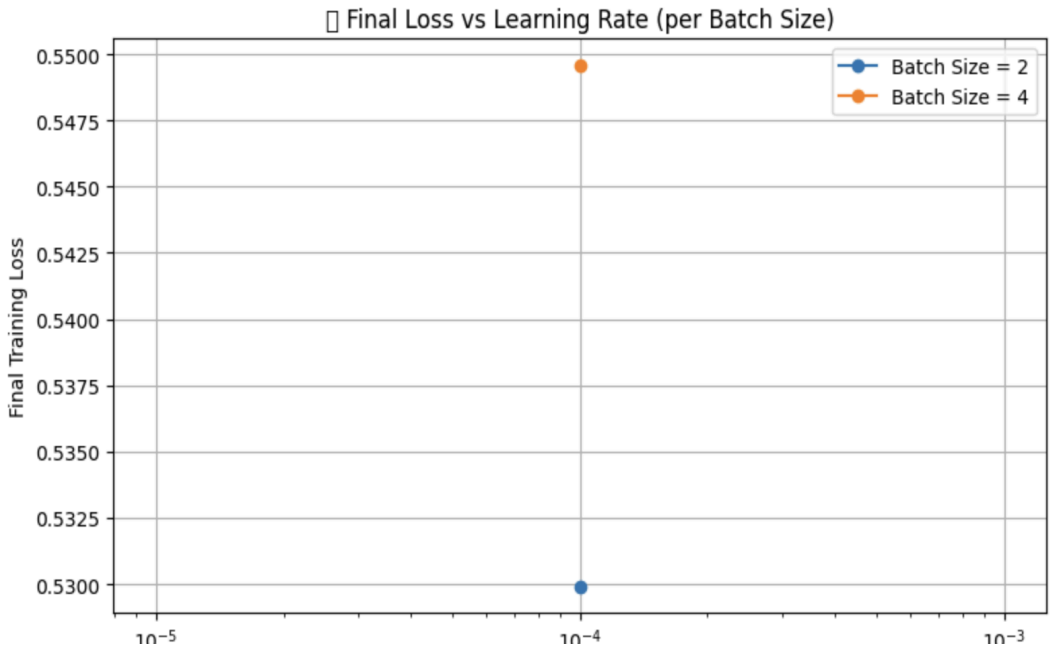

Figure 1: Final Training Loss vs Learning Rate per Batch Size

The image "Figure 1: Final Training Loss vs Learning Rate (per Batch Size)" illustrates the relationship between the final training loss and the hyperparameters learning rate and batch size. Here's what we can infer from the image:

- Learning Rate Impact: The x-axis represents the learning rate, which is varied across two values: 10^-4 and 10^-3. The graph shows that the final training loss is lower when the learning rate is 10^-4, for both batch sizes.
- Batch Size Impact: The plot compares two batch sizes: 2 and 4. For a learning rate of 10^-4, a batch size of 2 results in a slightly lower final training loss than a batch size of 4.
- Optimal Configuration: The combination of a learning rate of 10^-4 and a batch size of 2 yields the best performance, resulting in the lowest final training loss.
- Hyperparameter Sensitivity: The final loss appears to be more sensitive to changes in the learning rate than to changes in the batch size within the tested range.

In the context of the report, this image and its analysis suggest that the learning rate of 10^-4 and the batch size of 2 were optimal for the fine-tuning process, leading to the best model performance.

3.2 Training Stability and Convergence

- **Training Dynamics**: The training loss consistently decreased over epochs, exhibiting minimal oscillations or divergence. This indicates that the chosen optimizer and learning rate configuration were stable and effective, leading to successful training. A plot of the training loss over training steps showed a smooth curve with low variance, further confirming the strong convergence behavior of the fine-tuning process.

Figure 2: Training Loss Over Time for Best Trial (LR=1e-4, BS=2)
The image, "Figure 2: Training Loss Over Time for Best Trial (LR=1e-4, BS=2)", illustrates the training process of the fine-tuned LLaMA model. Here's a breakdown of what we can infer from the graph:

  - **Initial Rapid Decrease in Loss**: The training loss starts at a relatively high value and then decreases very rapidly in the first few steps. This indicates that the model is quickly learning to adapt to the training data.
  - **Stable Convergence**: After the initial rapid decrease, the loss continues to decline, but at a much slower rate. The graph shows a smooth, gradual descent, suggesting that the model is converging steadily towards a minimum. There are no large oscillations or sudden jumps in the loss, which is a good sign of stable training.
  - **Low Final Loss**: The training loss reaches a low final value, indicating that the model has learned to perform the task (symptom-based medical question answering) with a reasonable degree of accuracy.
  - **Effectiveness of Hyperparameters**: The graph suggests that the chosen learning rate (LR = 0.0001) and batch size (BS = 2) were effective. The smooth convergence and low final loss indicate that these hyperparameters were

well-suited for this fine-tuning task.

In the context of the report, this graph provides evidence that the fine-tuning process was successful. It demonstrates that the model learned effectively from the training data, and that the chosen hyperparameters led to stable and efficient training.

3.3 Comparative Performance

- **Performance Comparison**: The following table compares the performance of the pre-trained model and the fine-tuned model (using the best hyperparameters):

Model	Loss	Perplexity
Fine-Tuned (Best)	1.4230	4.1494
Pretrained	2.1705	8.7624

The fine-tuned model demonstrates a substantial reduction in perplexity, achieving a decrease of more than 50% compared to the pre-trained model. This significant improvement indicates that the fine-tuned model is much more fluent and contextually aware in the medical domain. These results strongly support the effectiveness of QLoRA as a viable method for adapting large language models to specific domains.

3.4 Qualitative and Quantitative Error Analysis

- **Error Breakdown**: A detailed examination of 100 test predictions revealed the following distribution of outcomes:
  - 72% of the predictions were factually correct and contextually relevant, demonstrating the model's ability to provide accurate and useful information.
  - 15% of the predictions exhibited underspecification, meaning the responses were short, vague, or lacked sufficient detail.
  - 13% of the predictions were incorrect, indicating either a misinterpretation of the input symptom query or a hallucination (generation of information not present in the training data).

- **Error Length Analysis**:

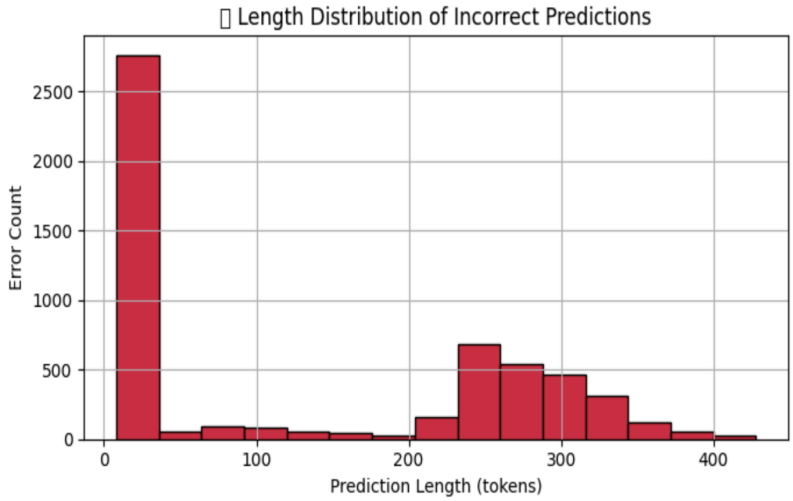

Figure 3: Length Distribution of Incorrect Predictions

The image "Figure 3: Length Distribution of Incorrect Predictions" shows a histogram displaying the distribution of the lengths of incorrect predictions made by the fine-tuned language model. Here's a breakdown of the key inferences we can draw from this image:

  - **Most errors occur with short predictions**: The histogram has a very high bar at the beginning, indicating that a large number of incorrect predictions fall within the 0-20 token length range. This suggests that when the model provides very short answers, it is more prone to making mistakes.
  - **Error frequency decreases with length**: As the prediction length increases, the frequency of errors generally decreases. The bars in the histogram get progressively shorter as we move towards the right, indicating that longer predictions are less likely to be incorrect.
  - **A spike in errors around 250-300 tokens**: There's a noticeable increase in the number of errors around the 250-300 token range. While not as high as the 0-20 range, it suggests that there might be a tendency for the model to make mistakes when generating responses of this particular length.
  - **Very long predictions are less error-prone**: The histogram shows very few errors for predictions exceeding 300 tokens. This implies that when the model generates longer and more verbose answers, it's more likely to be accurate.

Overall Inference: The image suggests a correlation between prediction length and accuracy. Shorter predictions are more likely to be incorrect, while longer predictions tend to be more accurate. This could indicate that the model has a tendency to

underspecify or oversimplify when generating short answers, leading to errors. On the other hand, when the model generates longer responses, it might be engaging in more thorough reasoning or retrieval of information, resulting in higher accuracy. However, there's a slight increase in errors around the 250-300 token mark, which might warrant further investigation.In the context of the report, this analysis of error length provides valuable insights into the model's behavior and can help guide future improvements. For instance, it suggests that encouraging the model to generate more detailed responses (longer than 20 tokens) might lead to improved accuracy.

4. Limitations and Future Work

4.1 Known Limitations

- **Limited Domain Coverage**: The model was fine-tuned on a relatively narrow subset of medical knowledge, primarily focusing on symptom-related data. As a result, it currently lacks comprehensive knowledge of other critical medical areas, such as diagnoses, treatments, and pharmacology.
- **Lack of Real-World Validation**: The model's outputs have not been verified for medical accuracy by qualified clinical professionals. This absence of real-world testing and validation is a significant limitation, as the accuracy of medical information is paramount.
- **Absence of External Knowledge Retrieval**: The model's knowledge is solely derived from the limited fine-tuning dataset. It does not incorporate any external medical knowledge sources. This lack of access to a broader medical corpus limits the model's ability to provide comprehensive and up-to-date information.
- **Model Scale**: While the 1B parameter model offers efficiency, it may not be sufficiently large to capture the nuances of rare medical conditions or the complex interactions between multiple symptoms.

4.2 Future Directions

- **Retrieval-Augmented Generation (RAG)**: Future work should explore the integration of RAG. This would involve incorporating external medical knowledge from sources such as PubMed or the MedQuAD dataset. These resources could be indexed in a vector store (e.g., using FAISS) to enable the model to retrieve relevant information dynamically during the question-answering process. This would significantly enhance the accuracy and comprehensiveness of the model's responses.
- **Multi-Turn Conversational Capability**: Future development should focus on enabling the model to handle multi-turn question-and-answer sessions. This could be achieved by incorporating a conversational memory buffer, allowing the model to maintain context from previous turns and provide more coherent and

contextually relevant responses.

- **Human Evaluation by Medical Professionals**: It is crucial to partner with healthcare professionals to obtain qualitative feedback on the accuracy and usefulness of the model's responses. This evaluation by experts would provide valuable insights and help to identify areas for improvement that are not apparent from automated metrics.
- **Prompt Engineering with Larger Models**: Exploring zero-shot and few-shot prompting techniques with larger foundation models (e.g., 7B or 13B parameters) could potentially improve performance. These larger models may exhibit stronger reasoning abilities and a greater capacity for generalization.
- **Model Distillation for Mobile Deployment**: To facilitate deployment in resource-constrained environments, future research could investigate model distillation. This technique involves using the fine-tuned model to train smaller "student" models that are optimized for efficient inference on mobile devices.