LLMs Project Report

Simon Fliegel
Computer Science (Master) / Secrets Behind LLMs
Matriculation number: 5342785
simon.fliegel@mailbox.tu-dresden.de
Codabench-Username: simonf
Source-Code: https://github.com/SimonFliegel/llms-project

Abstract

This report details the iterative development of a culturally-aware language model based on the *Meta-Llama-3-8B* architecture. The task involves addressing cultural nuances through two distinct formats: Multiple Choice Question (MCQ) and Single Answer Question (SAQ). Starting from a zero-shot baseline that exhibited significant structural failures and repetition loops, I implement a series of stepwise optimizations using Parameter-Efficient Fine-Tuning (PEFT) via Low-Rank Adaption (LoRA). I explore the transition from task-specific specialized models to a unified multi-task architecture, supported by structural contextual prompting and a higher-rank adaptation ($r = 64$). To address data scarcity, I evaluate several augmentation strategies, finding that country-choice permutation and uniform upsampling improve performance, while high-variance label augmentation introduces detrimental noise. The final system demonstrates that while model capacity is foundational, contextual grounding and data-centric refinements are the primary drivers for aligning Large Language Models (LLMs) with specialized, culturally-dependent domains.

1 Introduction

The objective of this project was to adapt the *Meta-Llama-3-8B* base model to address cultural nuances through two distinct natural language processing tasks: MCQ and SAQ. For each task, a dataset consisting of training and test sets in CSV format was provided.

The project workflow involved generating predictions for both formats and submitting them to a dedicated Codabench competition. This platform provided a standardized benchmark for evaluating model accuracy and facilitated a comparative analysis of performance improvements by providing a submission history, as well as a leaderboard for performance comparisons with other participants in the course.

The primary goal was to bridge the gap between theoretical concepts covered in the course and the practical challenges of applying an LLM to a specialized domain. In the following sections, I detail the sequential optimization steps taken and analyze the resulting gains in accuracy.

2 System Design and Optimizations

2.1 Initial Model Integration

The first phase of development involved securing access to the *Meta-Llama-3-8B* base model via the Hugging Face Hub (AI@Meta, 2026). The model was integrated using the Hugging Face Transformers API. Upon successful initialization, the model could generate text by processing a prompt alongside specific configuration arguments. My initial "Hello World" implementation followed a straightforward execution path. It accepted a user prompt, passed it through the model's transformer layers, and streamed the response to the console before terminating. This served as a critical sanity check for the environment and allowed me to continue with the next steps.

2.2 Establishing the Zero-Shot Baseline

With a functional pipeline, the next objective was to establish a performance baseline by evaluating the base model on the test datasets without any prior tuning. I analyzed the schema of the test files to extract the relevant input features. Initially, I utilized the `prompt` field for MCQs and the `en_question` field for SAQs as raw inputs.

To introduce more flexibility, I later implemented an `instruct` field. This allowed for the inclusion of contextual constraints and facilitated techniques such as in-context learning. The zero-shot results were inconsistent. While the model occasionally

handled MCQs with the correct formatting, it struggled significantly with the SAQ task. In many instances, the model exhibited "repetition loops", either echoing the question or generating irrelevant queries until the token limit was reached. Given that fine-tuning was a planned milestone, I moved forward under the hypothesis that supervised alignment would resolve these behavioral issues.

2.3 Fine-Tuning

For the training phase, I employed PEFT using LoRA (Sharma, 2025), a method that preserves the pre-trained weights of the base model while injecting trainable low-rank matrices into specified *target modules*. Mathematically, while traditional fine-tuning updates a weight matrix $W \in \mathbb{R}^{d\times d}$ directly, LoRA approximates the update ΔW through the product of two smaller matrices, $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$, where $r \ll d$.

The rank r and the scaling factor α govern the capacity and influence of the adaptation. A higher rank provides more parameters for learning complex cultural nuances but increases computational overhead (Unsloth-AI, 2026). Following common heuristics, I set $\alpha = 2r$ to balance the weight of the new adaptations against the base knowledge.

To implement this, I utilized the `unsloth` library due to its optimized kernels and memory efficiency (CHAWLA, 2025). My experimentation followed an iterative path:

- **Configuration A:** I initially set $r = 16$, targeting the attention projections: `q_proj`, `k_proj`, `v_proj`, and `o_proj`.
- **Configuration B:** I subsequently increased the rank to $r = 64$ and expanded the target modules to include the MLP layers (`gate_proj`, `up_proj`, `down_proj`) and the `lm_head`.

While the increased capacity in Configuration B yielded only marginal gains for models specialized in a single task, it proved essential for the multi-task "combined" model. As detailed in the following section, this higher-rank adaptation provided the necessary representational capacity to handle the increased complexity of switching between MCQ and SAQ formats within a single architecture.

An overview of the parameter configuration I used for fine-tuning can be found in Appendix A.

2.4 Combining Tasks

Right from the beginning I asked myself whether it was necessary to train one model for each task or if a single model can be trained for both leveraging the knowledge it learnt in the training data of the one task and applying it to questions of the other. When using a single model, the task has to be encoded somehow in the prompt so the model knows what answer format is expected. However, with the additional instruction field I added earlier, this was definitely possible.

With the initial number of samples and no added context the results were just about the same as using specialized models for each task. However the improvements in the fine tuning 2.3, contextual prompting 2.5, and the added samples 2.6, the combined performed better than the specialized models.

2.5 Contextual Prompting

To clearly distinguish between the two tasks for the combined model, I introduced a structural context to each prompt. This involved explicitly defining the task type and the target country, followed by a task-specific instruction.

This formatting ensures that the model switches to the appropriate response mode (letter-selection for MCQs vs. concise naming for SAQs). Examples of the final training prompt structures for both MCQ and SAQ tasks are provided in Appendix B.

2.6 Data Augmentation

As competition on the leaderboard intensified, I re-evaluated the training data pipeline to maximize information extraction. While I initially considered synthetic data generation via larger teacher models, I decided against this approach to avoid introducing hallucinated cultural biases and to maintain the factual integrity of the ground-truth data.

In the initial implementation for SAQ, I only extracted the answer with the highest consensus count. In an attempt to increase the dataset size, I experimented with a "one-to-many" augmentation strategy, creating separate training samples for every unique answer provided in the annotations. However, this led to Label Noise Conflict. The model was presented with identical input prompts mapped to different target strings. This ambiguity hindered convergence, probably because the loss function penalized the model for predicted answers that were

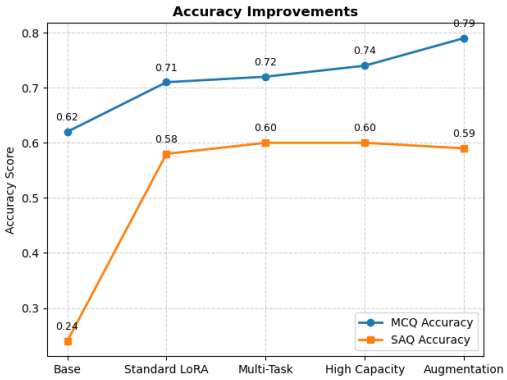

Figure 1: Development of the Accuracy for MCQ and SAQ for each iteration

technically correct but did not match the specific "noisy" label of that sample.

For the MCQ task, I implemented a more successful augmentation strategy based on Country-Choice Permutation. Since the dataset provided the corresponding country for each of the four answer options, I was able to quadruplicate the MCQ samples. By replacing the country name in the prompt with the alternative country from the choices and updating the target label accordingly, I expanded the dataset while maintaining high label fidelity.

This augmentation created a significant class imbalance for the combined model, where MCQ samples outnumbered SAQ samples. To prevent the model from developing a task-specific bias or neglecting the more complex linguistic requirements of the SAQ task, I applied Uniform Upsampling to the SAQ dataset. By duplicating the consolidated SAQ samples until a 1:1 ratio was achieved.

3 Results and Analysis

The progression of accuracy across each iterative phase is illustrated in Figure 1. While the refinements in prompt structure and data augmentation led to a consistent increase in MCQ accuracy, the SAQ task proved more resistant to these optimizations.

A manual inspection of the generated SAQ responses suggests that many predictions were contextually relevant with the question, yet many of them must have failed to match the exact ground-truth strings required for the evaluation metric. This discrepancy highlights the inherent difficulty of the SAQ task, where the narrow margin for "correctness" can mask underlying semantic improvements. The specific performance gaps observed on the leaderboard suggest that other participants may have utilized more successful strategies considering their significantly higher score in this task.

Ultimately, a systematic error analysis, comparing the model's outputs directly against the test references, would be essential to identify if these failures stemmed from formatting inconsistencies, cultural hallucinations, or specific linguistic edge cases that were not sufficiently addressed during the fine-tuning process.

4 Conclusion

This project provided a comprehensive practical application of the Large Language Model (LLM) optimization techniques covered throughout the course. The transition from a zero-shot Llama-3-8B baseline to a task-aligned, culturally-aware model highlighted the importance of the learned techniques.

The iterative nature of the project demonstrated that while hyperparameter tuning (such as increasing LoRA rank) provides a foundation, the most significant performance gains often stem from contextual grounding and data-centric refinements. The open-ended nature of the task allowed for a rigorous exploration of the "base-to-specialized" pipeline, a skill set that is directly transferable to other specialized domains requiring local LLM deployment. Future work could involve exploring higher-rank adaptations or more diverse cultural datasets to further minimize the performance gap observed in complex SAQs.

Acknowledgements and Aids

I would like to thank the course instructors for the organization of the project as well as providing access to the high-performance computing resources necessary for this experimentation. Additionally, the *Gemini* language model (Google) was utilized for coding assistance, grammar refinement, and structural proofreading during the preparation of this report.

References

AI@Meta. 2026. Llama-3-8b on hugging face.

AVI CHAWLA. 2025. Top 4 llm fine-tuning frameworks!

Sanjana Sharma. 2025. What is parameter-efficient fine-tuning (peft)?

Unsloth-AI. 2026. Unsloth ai - fine-tuning llms guide.

A Fine Tuning Configuration

Hyperparameter	Value
LoRA Configuration (PEFT)	
Rank (r)	64
Alpha (α)	128
Target Modules	All Linear + LM Head
LoRA Dropout	0.0
RS-LoRA	Enabled
Bias	none
Training Arguments (SFT)	
Epochs	5
Learning Rate	1×10^{-5}
LR Scheduler	Cosine
Batch Size (Total)	64 (16×4)
Warmup Ratio	0.1
Weight Decay	0.1
Optimizer	AdamW (8-bit)
NEFTune Alpha	5.0
Precision	BF16
Max Sequence Length	[Your value]

Table 1: Final hyperparameter configuration for the fine-tuned *Meta-Llama-3-8B* model. The total batch size is calculated as the product of per-device batch size and gradient accumulation steps.

B Prompt Templates

This appendix provides the full structure of the prompts used during the fine-tuning of the *Meta-Llama-3-8B* model.

B.1 MCQ Training Prompt

```
Context:
Task: Multiple Choice Question
Country: Iran

Instruction:
The following is a multiple-choice question
about cultural practices and preferences.
Based on the specified country, select the
single most accurate letter choice (A, B,
C, or D).

Input:
What are the required subjects in Iran's
university entrance exam?

A. chinese
B. culinary arts
C. korean language
D. mathematics.

Response:
D
```

B.2 SAQ Training Prompt

```
Context:
Task: Single Answer Question
Country: China

Instruction:
Provide a direct and concise answer to the
cultural question below. Use only a single
word or short phrase that represents the
general consensus in the specified country.

Input:
What is the main dish for Thanksgiving in
China?

Response:
Fish
```