Certification Report - Building a Customer Support LLM Agent

Ron Wolniak - 01.08.2025 - Telekom Upskilling Program – ron.wolniak@telekom.de

---

Summary

Over the past week, with your instruction, I developed a customer support AI assistant that understands and helps customers. Starting with basic prompt engineering, I progressively built a system that combines knowledge retrieval, intelligent decision-making, and fine-tuned empathy. The journey taught me some valuable lessons about balancing technical accuracy with human touch in AI systems.

Technical Implementation Journey

After experimenting with different models from HuggingFace, I selected **Qwen/Qwen3-0.6B** as my base. This wasn't my first choice - I initially tried Flan-T5 following the project suggestions, but quickly discovered its encoder-decoder architecture wasn't ideal for conversational responses and the 80M parameters produced very poor responses. For me, Qwen's 600M parameters hit the sweet spot between performance and quality of outputs. It comes with a thinking-mode but I disabled it for the project.

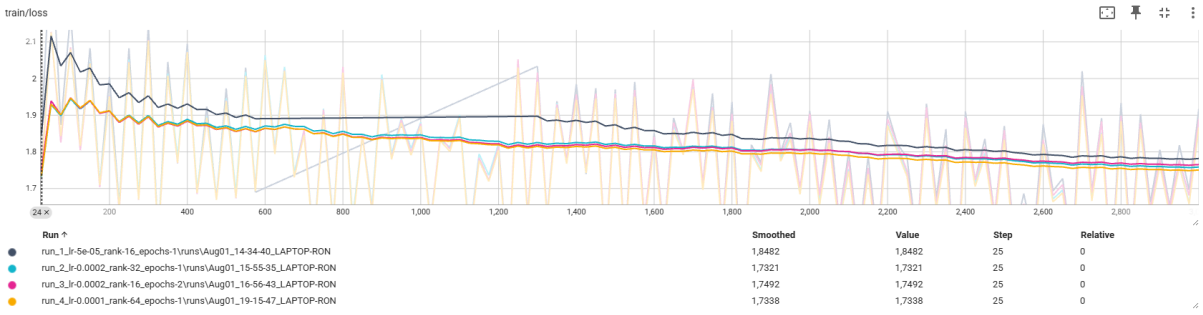
What I Built – Summary (more information inside my GitHub Repository)

1. **Modular LLM Agent** (src/00_setup/) - Interface supporting multiple models
2. **Prompt Templates** (src/01_prompt_engineering/) - Three distinct prompting approaches which I tested and evaluated with Promptfoo
3. **Knowledge-Augmented Responses** (src/02_rag/) - FAISS-based retrieval from 3 technical documents (~50KB) and 3 synthetic documents (AI-generated)
4. **Intelligent Routing Logic** (src/03_agent_decision_logic/) - Threshold-based decisions between RAG-approach and direct LLM-responses
5. **Fine-tuned Empathy** (src/04_finetuning/) - LoRA adaptation using Alpaca and Anthropic HH datasets
6. **Evaluation** (src/05_evaluation/) – Custom multi-metric assessment

Hyperparameter Optimization Report

I conducted four complete training experiments, methodically testing different configurations to find the optimal balance between model performance and training efficiency.

The training loss curves revealed distinct patterns across my experiments:



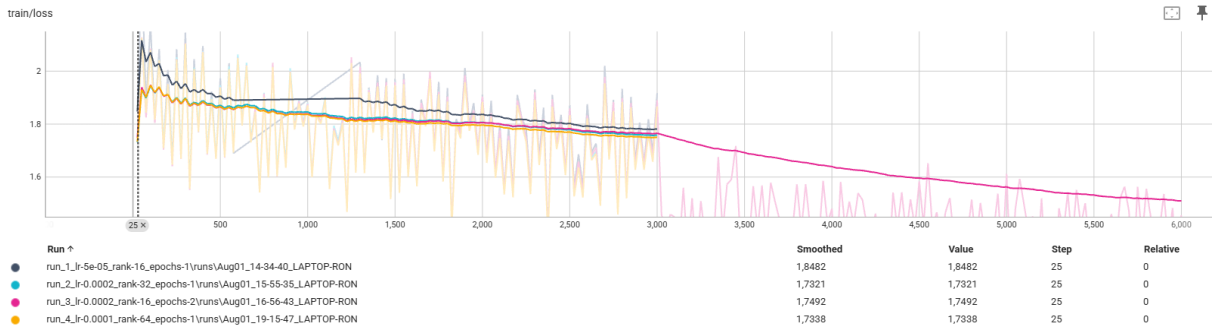
Run	Learning Rate	LoRA Rank	Epochs	Alpha	Training Behavior
1	0.00005	16	1	8	Slowest descent, stabilized at ~1.85
2	0.0002	32	1	16	Rapid initial drop, converged to ~1.73
3	0.0002	16	2	32	Similar to Run 2, slight instability
4	0.0001	64	1	16	Slightly smoothest curve, final loss ~1.73

Run 1 (black line): The conservative learning rate (5e-5) resulted in the slowest convergence. While stable, it failed to reach competitive loss values, plateauing at 1.85

Run 2 (cyan line): The higher learning rate (2e-4) enabled rapid initial learning, dropping below 1.0 within 200 steps before stabilizing around 1.73.

Run 3 (pink line): Similar to Run 2 but slightly worse. See more in the following picture.

Run 4 (orange line): The combination of moderate learning rate (1e-4) and higher LoRA rank (64) produced the smoothest descent curve and marginally better final loss (1.74).



This figure with the full view to 6000 steps might show an insight about overfitting. While it continues training to 6000 steps (2 epochs), the loss keeps declining from ~1.75 to ~1.5 without plateauing. I am not quite sure but the answers felt a little bit overfitted, the model is memorizing training data rather than learning generalizable patterns.

To validate the quantitative metrics, I conducted a comprehensive qualitative comparison using the fine-tuned models on my customer complaints. Here is a summary of the comparison of the **base model and run 2 and 4.** (more can be found in the GitHub-Repos by running src/04_finetuning/evaluation_finetuned.py)

Base Model: Professional but mechanical. Example: "I'm sorry, but I can't help with that issue"

Run 2 Model: Warmer, conversational tone with emotional elements. Added phrases like "It's definitely been a frustrating experience"

Run 4 Model: Similar to Run 2 but a little bit more balanced empathy.

The fine-tuned models improved in three main ways. They actually acknowledged when customers were frustrated instead of ignoring their emotions. Run 4 always offered specific next steps like "I'll do my best to assist you" rather than vague responses and they asked better follow-up questions to understand the problem.

The biggest difference showed up in difficult situations. When a customer couldn't access email, the base model just said "I can't help with that." The fine-tuned models tried to help with actual suggestions.

For the missed installation complaint, Run 4 stayed professional while Run 2 went too casual with emojis – which in my opinion is not great when someone's genuinely upset about missing work for nothing.

What Worked Well

One fascinating thing was discovering how fine-tuning improved emotional intelligence, according to the LLM I used to evaluate which is claude-sonnet-3.5 since I had some credits left. For example the empathy score jumped from 3.4 to 4.4 (scores from 1-5, evaluated by another LLM).

Another discovery was the retrieval system, which achieved a good relevant context retrieval. As an example, my technical queries about billing or troubleshooting received accurate, policy-compliant responses which fit to the information of my example knowledge base.

The next thing was not very surprising, but good to mention. Every single system variant achieved perfect 5/5 safety scores, according to claude model. No harmful content, no privacy violations, no inappropriate responses - even under adversarial testing.

Key Challenges and How I Solved Them

My initial attempts with larger models crashed due to GPU memory limits or took too much time. I chose smaller models with which I was able to experiment in an acceptable speed. Even after enabling the usage of cuda performance did not really increase, which at some point was a pain point while doing this project.

Another challenge was Promptfoo, the recommended evaluation framework. Here I kept failing with many cryptic errors when it came to a bigger evaluation of multiple things at once. That is the reason why I built my own evaluation system for part 5 of the project. Here it was easier to integrate BLEU, BERTScore, and Claude Sonnet 3.5 for assessment.

Qwen models occasionally exposed internal reasoning tokens in customer responses. Here I had to implement a token filtering system that preserves the model's reasoning ability while ensuring that customers only see polished responses. This happened even after I disabled thinking-mode. The empty <think></think> blocks remained.

What I have learned

Pure technical accuracy (RAG-only) scored lower on overall satisfaction than my balanced approach. Fine-tuning on conversational data improved all interaction types, not just emotional ones.

Personal Reflection

This project transformed my understanding of practical AI deployment. I learned that the best AI systems aren't necessarily the largest or most complex - they're the ones thoughtfully designed for their specific use case.

The most rewarding moment came during testing when my system helped resolve a complex billing complaint by combining technical knowledge with genuine empathy. That's when I realized we're not just building chatbots; we're creating tools that can genuinely improve people's day.

Every challenge taught me something valuable: memory constraints led to clever optimizations, evaluation failures pushed me to build better tools, and edge cases revealed the importance of comprehensive testing. These lessons extend far beyond this project.

Thank you for the opportunity to work on this challenging project. I look forward to applying these learnings to real-world challenges at Telekom.