Weekly Report – LLM Deployment Efficiency Experiments

Name: Anbhi Thakur
Date: 31/08/2025

---

1. Work Completed This Week

- Implemented **caching strategies** for frequently used prompts and embeddings, reducing repeated computation overhead.
- Ran **prompt optimization tests**, experimenting with shorter context windows, structured templates, and prompt re-use.
- Benchmarked **token usage optimization techniques**, including truncation of unnecessary history and response length control.
- Conducted preliminary **quantization experiments** on model weights to test inference speed improvements.
- Tested **speculative decoding methods** for parallelized token generation to improve responsiveness in the coding copilot use case.

---

2. Experimental Results

- **Caching:** Reduced average latency by ~18% for repeated queries, with negligible storage overhead.
- **Prompt Optimization:** Achieved up to 22% reduction in token usage, making responses more cost-efficient.
- **Token Management:** Optimized history length handling, reducing overall cost by 20% without affecting context retention significantly.
- **Quantization:** Achieved up to 35% faster inference times with 8-bit quantization, though slight accuracy trade-offs were noted.
- **Speculative Decoding:** Improved token generation speed by ~25%, particularly effective in interactive coding scenarios.

---

3. Challenges Faced

- **Accuracy vs. Efficiency Trade-off:** Quantization and aggressive prompt trimming sometimes reduced output quality.
- **Caching Invalidations:** Determining when cached results were outdated proved complex in dynamic contexts.
- **Scaling Issues:** Maintaining low latency under concurrent high load requires further infrastructure optimization.
- **Cost Modeling:** Predicting real-world cost savings is difficult without larger-scale deployment data.

---

4. Next Week's Plan

- Add **visual performance dashboards** to track latency, token usage, and cost in real-time.
- Experiment with **adaptive model routing** (smaller models for lightweight queries, larger ones for complex reasoning).
- Test **hybrid RAG setups**, combining vector search with structured caching for more reliable results.
- Automate **prompt optimization workflows**, possibly using reinforcement learning to adapt prompts over time.
- Investigate **memory-efficient inference techniques**, such as parameter-efficient fine-tuning and retrieval-augmented compression.

---

5. Notes & Recommendations

- **Short-term:** Maintain a balance between caching and freshness of responses to avoid outdated outputs.
- **Medium-term:** Adopt hybrid inference pipelines (quantized small models + full-precision fallback).
- **Long-term:** Explore serverless deployment and elastic scaling strategies for cost control in production.
- Continue running **controlled A/B tests** to measure improvements in responsiveness and cost-effectiveness.