Report on Model Optimization for Search Classification

Prepared by: Baraa (with assistance from ChatGPT)

Date: 2025-09-06

Executive Summary

This project set out to determine the optimal large language model (LLM) and parameters for classifying whether user questions require an external web search. The primary goals were: 1. To balance speed and accuracy on low-powered hardware. 2. To develop a repeatable evaluation methodology with reliable metrics. 3. To document the rationale behind testing, challenges encountered, and solutions implemented. Through multiple rounds of experiments with Ollama models, parameter sweeps, prompt refinements, and evaluation harnesses, the project identified Qwen2.5:0.5b-instruct as the best performing model overall. While Granite3.3:2b was more accurate in early tests, Qwen2.5 ultimately delivered a superior balance of speed, accuracy, and confidence when prompted under stricter conditions.

Introduction

The project’s purpose was to build a lightweight but accurate search-needed classifier that could run efficiently on CPU-bound environments, provide consistent JSON outputs for downstream processing, and handle short, independent user queries without context memory. Constraints included: - Input length capped at 512 tokens (~650 characters). - Strict schema: {"search_needed":0 or 1,"confidence":float}. - Low tolerance for formatting errors or hallucinations.

Methodology

Prompt Engineering

- Started with generic prompts requiring strict JSON outputs. - Iteratively refined to emphasize “no reasoning, JSON only,” reducing model drift. - Added domain-specific few-shot examples (medical, programming, general, mental health). - Later introduced balanced two-shot prompts (1 yes, 1 no), selected by input length, to prevent bias toward trivial or complex examples.

Model Selection

Focused on smaller Ollama-hosted models for performance reasons: - Qwen3:0.6b - Qwen2.5:0.5b-instruct - Granite3.1-MoE:1b - Granite3.3:2b - Llama3.2:1b - Phi-4 mini (3.8B) - Falcon3:1b

Options Sweeps

Two broad parameter regimes were tested: - Conservative: temperature=0.0–0.3, top_k=1–5, top_p=0.5–1.0. - Liberal: temperature=0.4–1.0, top_k=10–60, top_p=0.1–0.5. Other constants: - repeat_penalty=1.1, num_predict=64–128, format="json", raw=True.

Evaluation Harness

- Built a Python test suite to run all models across labeled questions. - Metrics collected: latency, CPU time, discrepancies, confidence, errors, and domain-level performance. - Automated CSV logging of runs for analysis. - Added visualizations (bar charts, scatter plots, domain-level breakdowns).

Challenges & Solutions

1. JSON Drift: Some models emitted or reasoning. - Solution: Forced raw=True, format=json, strict schema instructions. 2. Bias Toward YES or NO: Early prompts skewed results. - Solution: Balanced two-shot examples (1 yes, 1 no) by input length. 3. Performance Variability: Different prompt styles produced different outcomes. - Solution: Split analysis by results.csv vs results2.csv. 4. Discrepancy vs Latency Trade-off: Granite more accurate but slower. - Solution: Use Qwen2.5 as default, Granite as fallback for critical cases.

Results

Results 1 (results.csv): - Granite3.3:2b → lowest discrepancies (17), ~2.05s latency. - Qwen2.5:0.5b → fastest (~0.63s latency), slightly higher discrepancies (~26). Results 2 (results2.csv, stricter prompts): - Qwen2.5:0.5b → best overall: 23 discrepancies, ~0.63s latency, ~0.99 confidence. - Granite3.3:2b → 24 discrepancies, ~2.24s latency. Domain-Level: - Granite excelled in math. - Qwen2.5 strong in general, programming, medical. - Llama3.2 weakest (low confidence). - Falcon3 consistently underperformed.

Conclusion

Qwen2.5:0.5b-instruct is the optimal model for production: - Fastest (sub-1s latency). - High confidence (~0.99). - Accuracy nearly tied with Granite3.3:2b under stricter prompts. Granite3.3:2b remains useful as a fallback when maximum accuracy is needed. Final trade-off: sacrificing ~5 discrepancies to save ~1.5 seconds per query is justified for large-scale workloads.

Future Work

1. Hybrid Router: Implement confidence-based routing between Qwen2.5 and Granite3.3. 2. Fine-tuning: Explore lightweight fine-tunes on labeled datasets. 3. Scalability: Test cluster-level overhead for parallel classification. 4. Domain Specialization: Use domain-specific prompting if certain areas remain weak.