Relazione Homework 3 - SQE 2025/2026

Software Performance Benchmarking

LLM Inference Latency Analysis

Authors:

Giovanni Altieri (Matricola: 309006)

Matteo Patella (Matricola: 308056)

Table of Contents

Note: In this project, any occurrence of `llama3.2:latest` refers to the **Llama 3.2: 3b** model.

1. Introduction

This document describes the work carried out for Homework #3 of the Software Quality Engineering course, whose main objective is to analyze the inference performance of Large Language Models (LLMs) executed locally.

The specific objectives of the project are:

- Perform inference benchmarks on selected LLM models
- Collect LLM-specific latency metrics (TTFT, ITL, E2E)
- Analyze metrics to characterize the performance behavior of the models
- Investigate the relationships between metrics, inputs, and outputs

To achieve these objectives, we have developed a benchmarking framework in Python that interfaces with Ollama for model execution and automatically collects all relevant metrics.

2. Execution Environment

2.1 Hardware Configuration

The benchmark was run on a desktop PC with the following configuration:

Component	Specification
CPU	Intel Core i5-9600K @ 3.70GHz (6 cores)
RAM	32 GB DDR4
GPU	NVIDIA GeForce GTX 1660 SUPER (6 GB VRAM)
Storage	NVMe SSD
Power	Desktop PC connected to the mains

LLM model inference was performed entirely on the NVIDIA GPU, leveraging CUDA acceleration for optimal performance. The decision to use the GPU was crucial in ensuring acceptable response times, especially for larger models.

2.2 Software Configuration

Software	Version
Operating System	Windows 10
Ollama	v0.14.3
Python	3.11.9
CUDA	Integrated with NVIDIA drivers
IDE	Visual Studio Code

During benchmark execution, the system was dedicated exclusively to testing: only the Ollama terminal and the Python benchmarking script, run via VSCode, were active. No other significant processes were running to avoid interference with the measurements.

3. Benchmarking Methodology

3.1 Model Selection

Three models with significantly different sizes were selected to analyze how performance scales with model complexity:

Model	Parameters	Size	Reason
qwen2.5:0.5b	0.5 billion	~400 MB	Lightweight model, baseline speed
llama3.2:latest	3 billion	~2 GB	Medium model, good balance
mistral:7b	7 billion	~4 GB	Large model, industry standard

The choice of these models was guided by two main criteria: (1) variety in size to observe performance patterns as complexity varies, and (2) compatibility with available VRAM (6 GB), which excluded larger models such as llama3.1:70b or mixtral:8×7b.

3.2 Benchmark Parameters

Parameter	Value	Reason
Number of prompts	100	Dataset provided by the instructor
Iterations per prompt	5	Trade-off between statistical reliability and execution time
Max token output	256	Standard value for concise responses, sufficient for comparative analysis
Temperature	0	Ensures deterministic output and reproducibility of results
Total inferences	1500	100 prompts x 5 iterations x 3 models

3.3 Warm-up procedure

Before each benchmark session for each model, a warm-up call is performed with a simple prompt ('Hello, how are you?'). This procedure ensures that:

- The model is fully loaded into VRAM
- The CUDA caches are initialized
- The first measurements are not affected by the cold start

4. Prompt Dataset

The prompt dataset (prompts_group_6.jsonl) was provided by the instructor and contains 100 prompts of varying types and lengths.

4.1 Dataset Statistics

Statistic	Value
Total number of prompts	100
Minimum length	54 characters
Maximum length	11,690 characters
Average length	810 characters

4.2 Distribution by Category

Prompts were categorized based on their length:

Category	Range (characters)	Number of prompts	Percentage
Short	< 200	23	23
Medium	200 - 500	25	25
Long	500 - 1000	25	25
Very Long	> 1000	27	27

The balanced distribution across categories allows for meaningful analysis of the relationship between prompt length and latency metrics.

4.3 Prompt Types

The dataset includes different types of tasks:

- Algorithm development (e.g., sentiment analysis, NLP)
- Implementation of specific functions
- Process automation (e.g., conda package builds)
- Explanation of code and concepts

5. Metrics collected

The metrics collected during the benchmark were designed to capture every aspect of the LLM inference process.

5.1 Time To First Token (TTFT)

Definition:

The time between sending the request and receiving the first output token.

Formula:

TTFT = load_duration + prompt_eval_duration

Interpretation:

Measures the responsiveness of the model. A low TTFT is crucial for interactive applications where the user expects an immediate response. It is mainly influenced by the length of the input prompt.

5.2 Inter-Token Latency (ITL)

Definition:

The average time between the generation of one token and the next during the output phase.

Formula:

ITL = eval_duration / (eval_count - 1)

Interpretation:

Measures the fluidity of generation. It is particularly important for streaming applications where tokens are shown to the user as they are generated.

5.3 Time Per Output Token (TPOT)

Definition:

The average time to generate each token, calculated over the entire generation phase.

Formula:

TPOT = eval_duration / eval_count

Difference with ITL:

TPOT divides by n (total number of tokens), while ITL divides by n-1 (number of intervals). TPOT is slightly more statistically stable.

5.4 End-to-End Latency (E2E)

Definition:

The total time from sending the request to receiving the last token.

Formula:

E2E = end timestamp - start timestamp

Relationship with other metrics:

E2E ≈ TTFT + (output_tokens x TPOT)

5.5 Throughput

Definition:

The number of tokens generated per second during the generation phase.

Formula:

Throughput = output_tokens / generation_time

Interpretation:

Measures the overall efficiency of the model in generation. It is inversely proportional to TPOT.

5.6 Coefficient of Variation (CV)

Definition:

Measures the relative variability of results across multiple iterations.

Formula:

CV = standard deviation / mean

Interpretation:

- CV < 0.1: Very stable results
- CV 0.1 - 0.3: Moderate variability
- CV > 0.3: High variability

6. Implementation

6.1 System Architecture

The benchmarking framework consists of two main Python scripts:

- py: Benchmark execution and metric collection
- py: Statistical analysis and visualization generation

6.2 Metrics Collection

Metrics are collected from two sources:

1. **Ollama metrics (more accurate):**

Ollama provides precise internal metrics in the response payload: load_duration, prompt_eval_duration, eval_duration, eval_count. These metrics are calculated directly by the inference engine and do not include network overhead.

1. **Metrics measured via HTTP:**

Timestamps captured during token streaming. Useful as a reference but less accurate due to network buffering.

Our script uses Ollama metrics as the primary source to ensure reliable measurements.

6.3 Data Format

Results are saved in two formats:

- CSV: For compatibility with Excel and analysis tools
- JSONL: To preserve the complete data structure

Each record includes: identifiers (model, prompt_id, iteration), input characteristics (prompt_char_length, category), metrics (ttft, itl, tpot, e2e_latency, throughput), and raw Ollama metrics.

7. Results

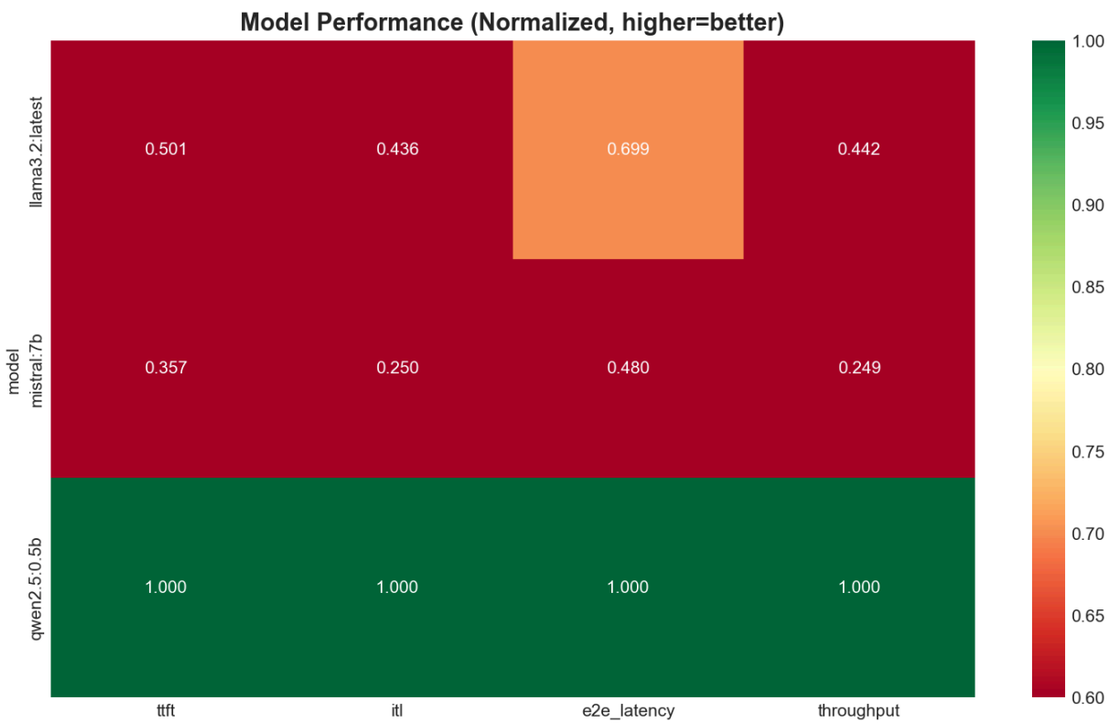
7.1 General Statistics

Parameter	Value
Total Records	1500
Valid records	1500 (100%)
Errors	0
Total benchmark duration	~2.5 hours

7.2 Metrics per Model

Average values calculated on 500 inferences per model (100 prompts x 5 iterations):

Model	TTFT (s)	ITL (ms)	TPOT (ms)	E2E (s)	Throughput (tok/s)
qwen2.5:0.5b	0.144	5.13	5.10	3.67	197.2
llama3.2:latest	0.287	11.67	11.58	5.24	87.1
mistral:7b	0.403	20.45	20.36	7.63	49.1



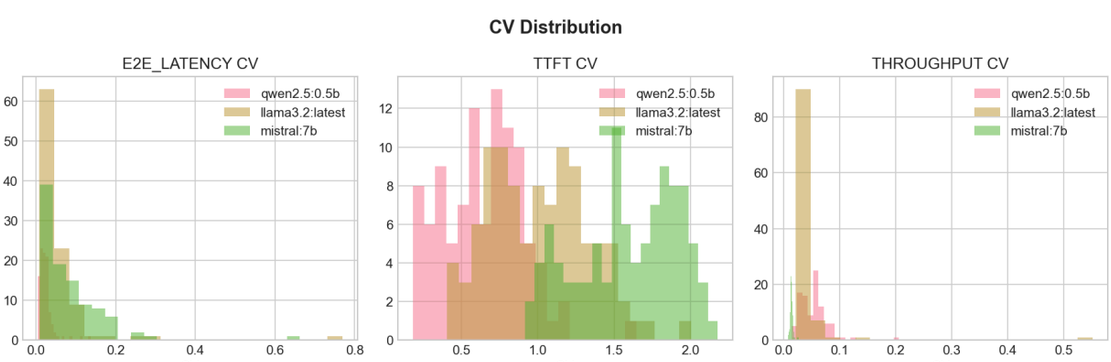
Observations:

- 5:0.5b dominates all metrics thanks to its small size
- Metrics scale proportionally with model parameters
- The throughput of mistral:7b (49 tok/s) is about 4x lower than qwen2.5:0.5b

7.3 Coefficient of Variation

Analysis of the stability of results between iterations:

Model	E2E CV	TTFT CV	Throughput CV
qwen2.5:0.5b	0.024	0.646	0.054
llama3.2:latest	0.055	1.007	0.039
mistral:7b	0.082	1.588	0.014



Observations:

- E2E Latency and Throughput show very low CV (<0.1): stable results

- TTFT shows high CV (>0.6): variability due to dynamic model loading
- mistral:7b has the most stable throughput (CV=0.014) but more variable TTFT

8. Analysis of Results

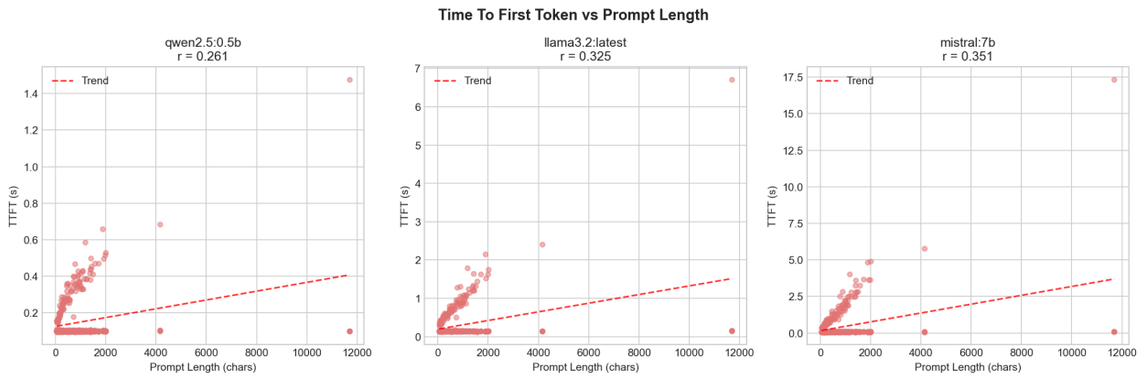
8.1 Relationship between Metrics and Prompt Length

Question: Is there a relationship between metrics and prompt input?

We analyzed the correlation (Pearson's r) between prompt_char_length and the main metrics:

Model	TTFT	E2E	Throughput
qwen2.5:0.5b	r = 0.261 *	r = 0.220 *	r = 0.120 *
llama3.2:latest	r = 0.325 **	r = -0.097	r = -0.653 ***
mistral:7b	r = 0.351 **	r = 0.406 **	r = -0.684 ***

Key: * weak, ** moderate, *** strong



Interpretation:

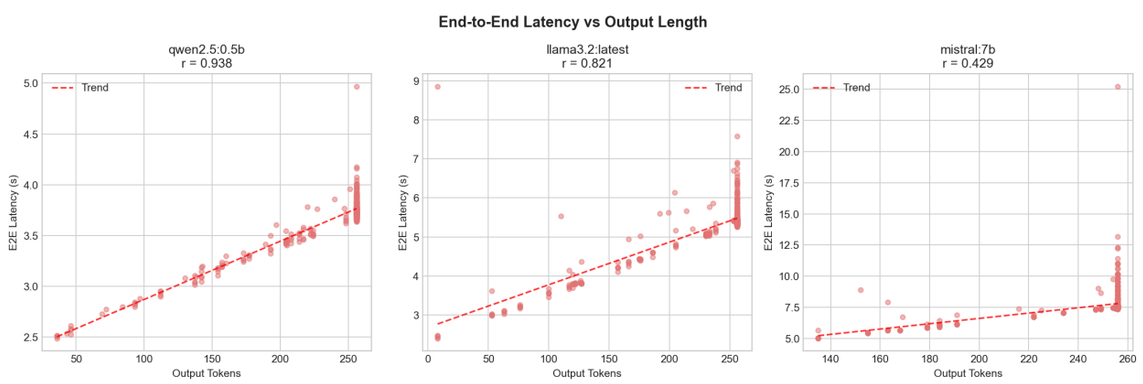
- TTFT increases with prompt length for all models (positive correlation)
- Throughput decreases sharply for larger models (strong negative correlation)
- 5:0.5b is less sensitive to prompt length

8.2 Relationship between Metrics and Output Length

Question: Is there a relationship between metrics and output length?

Correlation between output_tokens and E2E latency:

Model	Correlation between E2E and Output Tokens
qwen2.5:0.5b	r = 0.938 *** (very strong)
llama3.2:latest	r = 0.821 *** (strong)
mistral:7b	r = 0.429 ** (moderate)



Interpretation:

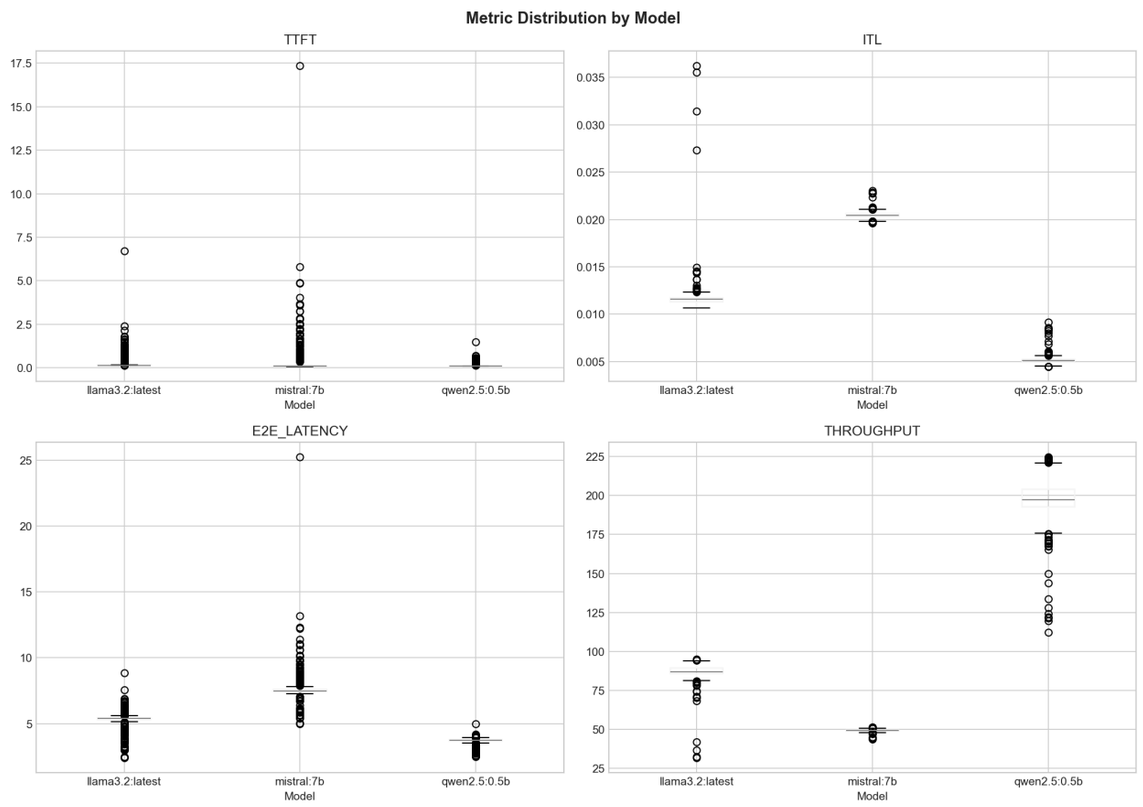
The positive correlation confirms that E2E increases linearly with the number of tokens generated. The relationship is closer for small models, while mistral:7b shows greater variance.

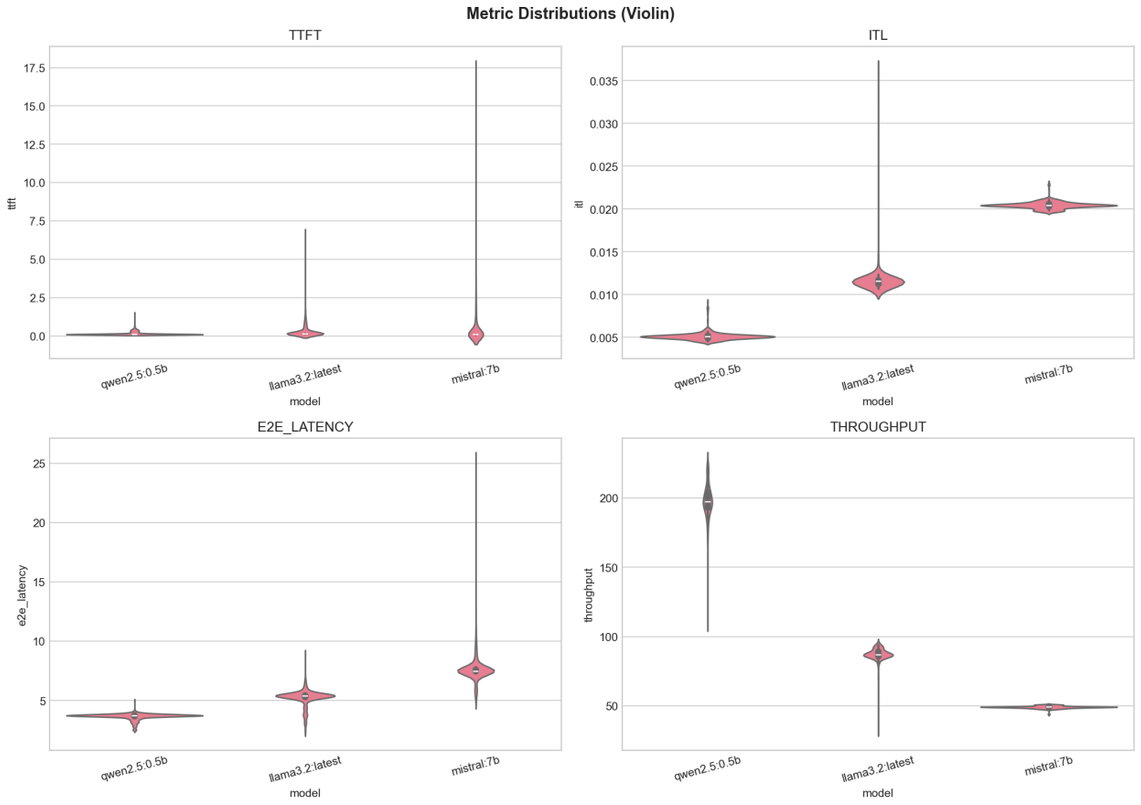
8.3 Differences between Models

Question: Do the metrics change when using different models/prompts?

The Kruskal-Wallis test was performed to check for statistically significant differences between models:

Metric	p-value	Significance
TTFT	< 0.001	*** Highly significant
ITL	< 0.001	*** Highly significant
E2E	< 0.001	*** Highly significant
Throughput	< 0.001	*** Highly significant





All metrics show statistically significant differences between models, confirming that model choice has a substantial impact on performance.

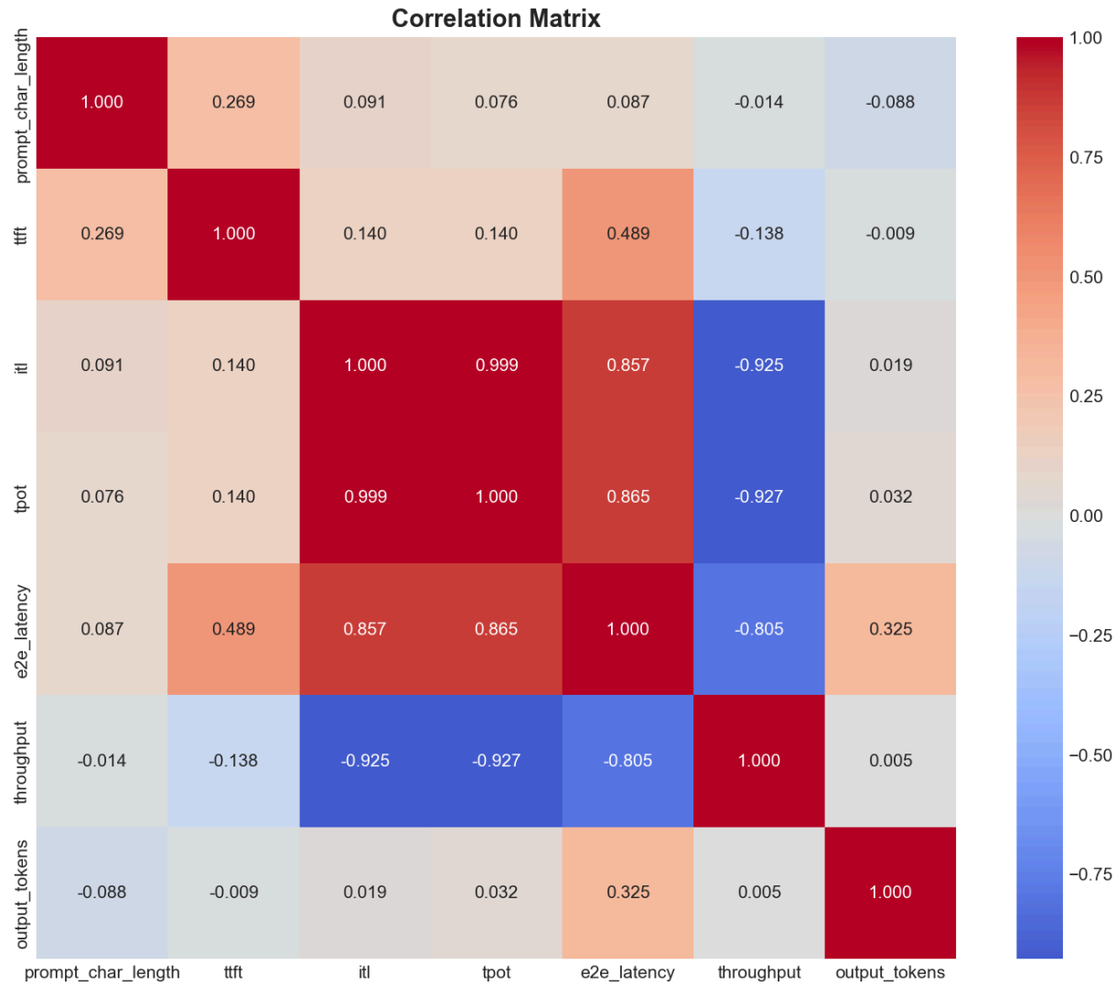
8.4 Variability between Iterations

Question: Do the results vary when running multiple times with the same workload?

Analysis of the Coefficient of Variation shows that:

- E2E Latency and Throughput are highly reproducible (CV < 0.1)
- TTFT shows significant variability, probably due to:
- Dynamic GPU memory management
- Variations in model loading time
- Minimal operating system interference

The use of temperature=0 ensured deterministic outputs, eliminating variability related to stochastic generation.



9. Problems Encountered and Solutions

9.1 Incorrect Calculation of Inter-Token Latency

Problem:

In the first version of the script, ITL was calculated by measuring the time intervals between chunks received via HTTP streaming. The resulting values were in the order of microseconds (10^-6 s), which was clearly incorrect.

Cause:

Ollama does not send tokens individually via HTTP, but groups them in buffers before transmission. As a result, HTTP timestamps do not reflect the actual token generation time.

Solution:

We modified the script to use the internal metrics provided by Ollama in the response payload:

ITL = eval_duration / (eval_count - 1)

This formula uses the total generation time (eval_duration) and the number of tokens generated (eval_count) to calculate an accurate ITL.

9.2 Compatibility Issues

Issue:

During development, compatibility errors were encountered between Python library versions on different test machines.

Machines tested:

- Laptop with integrated GPU: insufficient performance

- Workstation with different configuration: dependency errors

Solution:

A requirements.txt file was created with specific library versions optimized for Python 3.11:

- pandas >= 2.0
- numpy >= 1.24
- matplotlib >= 3.8
- scipy >= 1.11

9.3 Execution Times

Problem:

The complete benchmark took approximately 2.5 hours, making debugging and incremental validation difficult.

Solution:

A --limit-prompts option was implemented to run quick tests on a subset of prompts, allowing code validation before the full benchmark.

10. Conclusions

10.1 Summary of Results

The benchmark allowed us to characterize the performance of three LLM models of different sizes, highlighting clear patterns:

1. Performance scales inversely with model size: qwen2.5:0.5b is about 4x faster than mistral:7b
2. TTFT is affected by prompt length, with larger models being more sensitive
3. E2E latency correlates strongly with output length
4. The results are highly reproducible for E2E and throughput (CV < 0.1)

10.2 Recommended Model for Use Case

Use Case	Recommended Model	Reason
Real-time applications	qwen2.5:0.5b	Optimal TTFT and throughput
Quality/speed balance	llama3.2:latest	Good compromise
Maximum quality	mistral:7b	Most capable model

10.3 Study limitations

- Benchmark performed on a single machine: results may vary on different hardware
- 5 iterations: a higher number would increase statistical reliability
- The quality of responses was not evaluated, only performance

10.4 Future Developments

- Extend the benchmark to larger models with adequate hardware
- Include response quality metrics
- Test the impact of quantization on performance

11. Appendix: Repository Structure

The GitHub repository contains the following files:

Benchmarking Scripts:

- benchmark.py - Main script for running the benchmark
- analyze_results.py - Script for analysis and visualization

Dataset and Results:

- prompts_group_6.jsonl - Prompt dataset provided by the instructor
- benchmark_results/ - Directory with results in CSV and JSONL format
- results_analyzed/ - Directory with graphs and analysis reports

Configuration:

- requirements.txt - Python dependencies
- README.md - Project documentation

Generated graphs:

- ttft_vs_prompt_length.png - TTFT/prompt length correlation
- e2e_vs_output_length.png - E2E/output token correlation
- model_comparison_boxplot.png - Metric distribution by model
- model_comparison_violin.png - Violin plot of distributions
- model_comparison_heatmap.png - Normalized performance heatmap
- cv_distribution.png - Coefficient of Variation distribution
- correlation_matrix.png - Correlation matrix between metrics

GitHub repository link: https://github.com/G-Altieri/homework-3-sqe