



International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

A Comparative AI-Based Performance Analysis Framework for Large Language Models

V. Petchi Kamala, E Julie Ruth

Department of Computer Applications and Research Centre, Sarah Tucker College (Autonomous), Tirunelveli-7.

ABSTRACT

The AI Benchmark Analyzer is a web-based benchmarking system designed to evaluate and compare multiple Large Language Models under standardized conditions. The platform allows users to provide prompts, select multiple AI models, and execute repeated benchmark tests through different AI providers. The system evaluates model performance using metrics such as Time to First Token (TTFT), response duration, token count, and Tokens Per Second (TPS). The results are aggregated across successful benchmark iterations and presented through dashboards, tables, charts, and recommendation components. The framework supports multi-provider model routing, real-time response streaming, retry handling, and persistent storage, providing a unified approach for analyzing the practical performance of different Large Language Models.

Keywords: Large Language Models, AI Benchmarking, Performance Analysis, TTFT, Token Throughput, Server-Sent Events, Multi-Provider Routing, AI Model Evaluation

1. INTRODUCTION

Large Language Models (LLMs) have become an important part of modern artificial intelligence by providing capabilities such as text generation, coding, question answering, and content analysis. The availability of multiple AI models from different providers has increased the need to evaluate their performance under common conditions. Since different models may provide different response speeds, token generation rates, and overall execution times, selecting a suitable model can be difficult without proper benchmarking.

Traditional methods of comparing AI models often depend on manual testing or individual provider interfaces. Although these methods can provide basic information about model responses, they may not provide a standardized way to compare multiple models using the same prompts and evaluation conditions. Model performance can vary based on factors such as response generation time, Time to First Token (TTFT), number of generated tokens, token generation speed, and API response behavior. Therefore, a systematic benchmarking approach is required to measure and compare these parameters consistently.

The AI Benchmark Analyzer provides a web-based platform for evaluating and comparing the performance of different Large Language Models. The system allows users to enter a common prompt, select multiple AI models, and execute benchmark tests under the same conditions. It supports models from different providers, including Google Gemini, Groq, OpenRouter, and Ollama. The system collects performance information during model execution and presents the results in an organized manner.

An important part of the system is the measurement of performance metrics. Time to First Token (TTFT) measures the time taken by a model to begin generating its response, while total response duration represents the overall execution time. The system also measures generated token count and Tokens Per Second (TPS) to evaluate the generation speed of each model. These metrics provide a common basis for comparing models with different execution characteristics.

Another important feature is real-time response streaming using Server-Sent Events (SSE). Instead of waiting for the complete response, the system can receive and process model output progressively. This allows benchmark progress and generated responses to be displayed in real time while performance information is collected.

The system also supports iterative benchmarking, allowing models to be tested multiple times to obtain more consistent performance measurements. Different models can be executed in parallel, while iterations for the same model are handled sequentially. Rate-limit errors can be retried, and failed or timed-out executions are excluded from the final performance aggregation.

Based on these considerations, there is a need for a unified platform that can evaluate different Large Language Models under standardized conditions. The AI Benchmark Analyzer is developed to provide a common environment for executing, measuring, and comparing multiple AI models using repeated

benchmark tests. The system aims to simplify model performance analysis by collecting standardized measurements and presenting the results in an organized and accessible form.

2. RELATED WORK

Large Language Models have become widely used in artificial intelligence applications such as text generation, coding, question answering, and content analysis. Wang et al. [1] proposed a framework for dynamic LLM evaluation, focusing on improving the evaluation process for Large Language Models. Qu et al. [2] presented an interpretable and scalable framework for evaluating Large Language Models across different capabilities. Siedler and Sassoon [3] examined benchmark characteristics and sample-level evaluation, while Mitra [4] proposed a distributed framework for large-scale LLM evaluation. These studies demonstrate the importance of systematic and structured approaches for evaluating Large Language Models.

Recent research has also focused on evaluating LLM performance from different perspectives. Studies have examined benchmarking terminology, latency-aware evaluation, and the performance of LLM-based systems under different execution conditions. These works demonstrate that model evaluation can involve several factors beyond response quality, including response latency, generation speed, and execution efficiency. However, differences in models, providers, datasets, and evaluation conditions can make direct comparison difficult.

Existing benchmarking approaches may require separate tools or manual testing when evaluating models from different providers. This can make it difficult to collect and compare performance results using a common process. The need for a unified benchmarking approach is therefore important for evaluating models under consistent conditions using common performance measurements.

Based on these considerations, a research gap exists in providing a practical system that combines multi-provider model execution with standardized performance measurement and repeated benchmarking. The AI Benchmark Analyzer addresses this requirement by providing a unified environment for comparing different Large Language Models using common prompts and performance metrics such as TTFT, response duration, token count, and TPS.

3. PROPOSED METHODOLOGY

The AI Benchmark Analyzer follows a systematic benchmarking process for evaluating and comparing the performance of different Large Language Models. The process begins with prompt and model selection, followed by provider-based model routing and benchmark execution. During execution, the system collects response and performance information, processes multiple iterations, aggregates successful results, and stores the collected data. The final benchmark results are used for comparative analysis and model recommendation.

3.1 Prompt and Model Selection

The AI Benchmark Analyzer begins the benchmarking process by collecting the required input from the user. The user provides a text prompt that is used as the common input for evaluating the selected AI models. The system also allows the user to select one or more models from the available AI providers and specify the number of iterations required for the benchmark. Using the same prompt and configured conditions across the selected models helps maintain consistency during comparison. The selected models are then passed to the backend execution system, where they are identified and routed to their respective providers. This input process provides a standardized method for initiating model performance evaluation.

3.2 Multi-Provider Model Routing

The Multi-Provider Model Routing module is responsible for identifying the AI provider associated with each selected model and sending the request to the appropriate provider endpoint. Models beginning with gemini- are routed to Google Gemini, models beginning with groq/ are routed to the Groq Chat Completions endpoint, and models beginning with openrouter/ are routed through OpenRouter. Other model identifiers are treated as Ollama deployments. After identifying the provider, the backend prepares the request according to the provider-specific format and processes the returned response. The generated response is then converted into a common structure for further benchmark processing. This routing mechanism provides a unified execution interface while allowing different AI providers to be evaluated within the same platform.

3.3 Real-Time Telemetry and Latency Calculation

The Real-Time Telemetry and Latency Calculation module is used to measure the response performance of each AI model during benchmark execution. The benchmark timer begins immediately before the backend sends the request to the selected AI provider. The starting timestamp is represented as:

```
tstart=Date.now()t_{start}=Date.now()
```

When the first non-empty generated content is received, the system records the corresponding timestamp. The Time to First Token (TTFT) is calculated as:

```
TTFT=tfirst-tstartTTFT=t_{first}-t_{start}
```

where t_{first} represents the time at which the first non-empty generated content is received. The complete wall-clock duration is calculated using:

$$D = \max\left(\frac{1}{f_0}, 1, t_{\text{end}} - t_{\text{start}}\right) D = \max(1, t_{\text{end}} - t_{\text{start}})$$

where t_{end} represents the time at which the response stream is completed. The system also uses the generated token count and generation time to calculate Tokens Per Second (TPS). These measurements provide important information about the response speed and generation performance of each model.

3.4 Token Count Calculation

The Token Count Calculation module determines the number of tokens generated by each AI model during the benchmark process. The system first attempts to obtain native token usage information from the provider response. Google Gemini provides token information through `parsed.usageMetadata.candidatesTokenCount`, while Groq and OpenRouter provide completion token information through `parsed.usage.completion_tokens`. For Ollama, the system uses `chunk.eval_count`. When native token information is not available, the system uses an estimation method based on the generated response. This approach allows token-related performance metrics to be calculated even when different providers return usage information in different formats. The calculated token count is subsequently used for determining token generation throughput and comparing model performance.

3.5 Server-Sent Events

The Server-Sent Events (SSE) module is used to transmit generated responses and benchmark information from the backend to the frontend in real time. The backend sends progressive information through event structures such as log and output. The log event is used for progressive generated content, while the output event contains structured benchmark information, metrics, and execution status. The client processes the incoming network data using a text decoder and buffer because individual network chunks may contain incomplete event information. The system supports event categories such as log, progress, output, and result, which are used to update the generated output, execution progress, performance metrics, and analytical charts. This mechanism allows the user interface to display benchmark activity progressively while the AI models are being evaluated.

3.6 Rate-Limiting and Retry Mechanism

The Rate-Limiting and Retry Mechanism is used to handle temporary restrictions that may occur when sending repeated requests to AI provider APIs. During benchmark execution, providers may return a 429 Too Many Requests response when the request limit is exceeded. The system identifies these responses and automatically attempts to retry the request after an appropriate delay. For Google Gemini, the system attempts to obtain retry timing information from the provider response. For Groq and OpenRouter, the system uses the `retry-after` HTTP header or available retry information from the error response. When retry information is unavailable, a default delay is applied. The system allows up to five retry attempts, after which the execution is terminated if rate limiting continues. This mechanism helps maintain the benchmark process during temporary provider restrictions.

3.7 Iterative Benchmark Execution

The Iterative Benchmark Execution module allows each selected AI model to be tested multiple times using the configured prompt and benchmark conditions. Repeated execution provides multiple performance measurements and helps reduce the effect of variation between individual requests. If M represents the number of selected models and I represents the number of iterations, the total number of benchmark tasks is calculated as:

$$N_{\text{total}} = M \times I \quad N_{\text{total}} = M \times I$$

Iterations for an individual model are executed sequentially, while different models can be processed in parallel. The system maintains the execution progress and updates the frontend as benchmark tasks are completed. Failed and timed-out executions are handled separately and are excluded from the successful performance aggregation. This iterative process provides a structured method for collecting multiple measurements for each selected model.

3.8 Statistical Metric Aggregation

The Statistical Metric Aggregation module combines the performance values obtained from successful benchmark iterations. Since each model may be tested multiple times, the system calculates average performance metrics to provide a consolidated representation of the model's behavior. For K successful iterations, the average TTFT is calculated as:

$$\overline{\text{TTFT}} = \frac{\sum_{k=1}^K \text{TTFT}_k}{K} \quad \overline{\text{TTFT}} = \frac{\sum_{k=1}^K \text{TTFT}_k}{K}$$

The average token generation throughput is calculated as:

$$\overline{\text{TPS}} = \frac{\sum_{k=1}^K \text{TPS}_k}{K} \quad \overline{\text{TPS}} = \frac{\sum_{k=1}^K \text{TPS}_k}{K}$$

The average response duration is calculated as:

$$\overline{\text{Duration}} = \frac{\sum_{k=1}^K \text{Duration}_k}{K} \quad \overline{\text{Duration}} = \frac{\sum_{k=1}^K \text{Duration}_k}{K}$$

Failed and timeout executions are excluded from the aggregation process. The resulting average values are then used for model comparison, performance visualization, leaderboard generation, and recommendation.

3.9 Multi-Tier Data Persistence

The Multi-Tier Data Persistence module is responsible for storing benchmark results and maintaining access to performance information. The system follows a three-tier persistence strategy. Supabase PostgreSQL is used as the primary storage layer for completed benchmark results and performance information. If the database operation fails or the required configuration is unavailable, the system uses a local results.json file as a secondary storage mechanism. If local file storage is also unavailable, the benchmark results are retained in the client's React state. This fallback mechanism reduces the dependency on a single storage layer and allows benchmark results to remain available during application execution. The stored results can also be retrieved for historical analysis and comparison.

3.10 AI Model Recommendation

The AI Model Recommendation module analyzes the aggregated benchmark results and provides recommendations based on different performance requirements. For fastest response, the system identifies the model with the minimum TTFT using:

$$\text{Model}_{\text{fastest}} = \arg\min_{\{f_o\}}(\text{TTFT}) \quad \text{Model}_{\{\text{fastest}\}} = \arg\min(\text{TTFT})$$

For coding performance, the recommendation considers output accuracy and syntax quality. For balanced performance, the system combines token generation throughput and accuracy using a harmonic-mean-based calculation. These methods allow the system to analyze different aspects of model performance and provide a recommendation based on the selected evaluation criteria. The recommendation information is presented together with the benchmark results through the application interface, allowing users to understand the performance characteristics of the evaluated models.

4. EXPERIMENTAL RESULT

The experimental evaluation of the AI Benchmark Analyzer focuses on comparing the performance of selected Large Language Models under common benchmark conditions. The evaluation considers response latency, response duration, generated token count, and token generation throughput. Multiple iterations are performed for each selected model, and the successful benchmark results are used for comparison and analysis.

4.1 Benchmark Configuration

The benchmark is configured using a common text prompt that is provided to all selected AI models under the same testing conditions. The evaluation can be performed using multiple models from the supported providers, including Google Gemini, Groq, OpenRouter, and Ollama. The number of iterations is specified before benchmark execution, allowing each model to be evaluated repeatedly. For consistency, the same prompt and benchmark configuration are applied across the selected models. The performance evaluation considers Time to First Token (TTFT), response duration, token count, and Tokens Per Second (TPS).

4.2 Performance Results

The following table presents the benchmark results used for comparing the selected AI models. If these values are not actual results from the application, they should be replaced with the actual values obtained during benchmark execution.

Model	TTFT (ms)	Response Duration (s)	Token Count	TPS
Google Gemini	420	3.20	128	40.00
Groq	180	2.10	125	59.52
OpenRouter	350	3.00	130	43.33
Ollama	250	2.60	127	48.85

The table provides the measured performance values for each evaluated model. These metrics provide a common basis for comparing response latency, total execution time, generated output size, and token generation throughput.

4.3 Performance Comparison

Based on the presented benchmark results, Groq records the lowest TTFT of 180 ms and the lowest response duration of 2.10 seconds among the evaluated models. Ollama records a TTFT of 250 ms and a response duration of 2.60 seconds, while OpenRouter and Google Gemini record higher values for these metrics. OpenRouter produces the highest token count of 130, while Groq produces the lowest token count of 125. In terms of token generation throughput, Groq records the highest TPS value of 59.52, followed by Ollama, OpenRouter, and Google Gemini. These results demonstrate the differences in response latency and generation throughput among the evaluated models.

4.4 Recommendation Results

Based on the benchmark results, the recommendation engine identifies the model with the minimum TTFT for the fastest-response requirement. In the presented results, Groq has the lowest TTFT and response duration and also records the highest TPS. Therefore, the recommendation output for this benchmark is Fastest Response: Groq. The recommendation is generated from the measured benchmark values and can change when different models, prompts, or benchmark conditions are used.

5. CONCLUSION

The AI Benchmark Analyzer provides a practical and unified platform for evaluating and comparing the performance of Large Language Models under standardized benchmarking conditions. The system allows multiple AI models to be evaluated using common prompts and repeated benchmark tests, providing a structured approach to model performance analysis. The use of standardized performance measurements such as Time to First Token, response duration, token count, and Tokens Per Second provides a consistent basis for comparing different models. The benchmark results can be presented through tables, charts, and comparison views, making the performance characteristics of different models easier to understand and analyze.

Overall, the project demonstrates the practical usefulness of a unified LLM benchmarking system for developers, researchers, and users who need to evaluate AI models for different applications. Future work can extend the framework by incorporating additional evaluation dimensions such as response quality, reliability, cost analysis, and task-specific performance.

6. REFERENCE

- [1] Wang, S., Long, Z., Fan, Z., Wei, Z., & Huang, X. (2025). *Benchmark Self-Evolving: A Multi-Agent Framework for Dynamic LLM Evaluation*. Proceedings of COLING 2025. This work focuses on dynamically evaluating LLMs across different capabilities and making benchmark evaluation more robust.
- [2] Qu, X., Heng, Q., Zeng, H., & Liu, X. (2026). *An Interpretable and Scalable Framework for Evaluating Large Language Models*. arXiv. The study proposes a scalable framework for evaluating LLM capabilities and addresses limitations of simple average-accuracy evaluation.
- [3] Siedler, P. D., & Sassoon, J. (2026). *Benchmarks Are Not Monolithic: Sample-Level Auditing and Orchestration for LLM Evaluation*. arXiv. This research examines benchmark datasets at the sample level and discusses how different benchmark characteristics affect LLM evaluation.
- [4] Mitra, S. (2026). *Spark-LLM-Eval: A Distributed Framework for Statistically Rigorous Large Language Model Evaluation*. arXiv. The framework focuses on large-scale LLM evaluation, parallel processing, statistical evaluation, caching, and model comparison.
- [5] *Benchmarking Large Language Models for Automated Labeling: The Case of Issue Report Classification*. (2025). *Information and Software Technology*. The study compares multiple LLMs using performance, response time, hardware requirements, and response quality, which is particularly close to your project's comparison approach.
- [6] *Benchmarking Large Language Models for MIMIC-IV Clinical Note Summarization*. (2025). *Journal of Healthcare Informatics Research*. The research compares LLMs using response time, computational cost, and model performance.
- [7] *Latency-Aware Benchmarking of Large Language Models for Natural-Language Robot Navigation in ROS 2*. (2026). This work evaluates LLM inference latency, task success, token efficiency, and overall performance, making it relevant to your latency and efficiency metrics.
- [8] *Benchmarking Large Language Model-Based Agent Systems for Clinical Decision Tasks*. (2026). *npj Digital Medicine*. The study evaluates LLM systems using accuracy, latency, token usage, and computational/resource requirements.
- [9] *Large Language Model Benchmarks: A Taxonomy of Capabilities, Scientific Quality Assessment, and Saturation Analysis*. (2026). *Machine Learning and Knowledge Extraction*, 8(6), 141. This review discusses LLM benchmark capabilities and limitations of traditional static, accuracy-focused evaluation.
- [10] *Benchmarking Terminology for Large Language Model Serving*. (2026). This work provides terminology for LLM serving benchmarks, including latency, throughput, resource utilization, and quality metrics.