Comparative Evaluation of Publicly Accessible LLM APIs for Technical Prompt Response: A Case Study Using Gemini, OpenRouter, and GroqCloud

Bharath Gowda M R[1], Abhilash S[2], B U Bharath[3], Prajwal K[4]
[1,2,3,4]*Student, Department of CSE-AIML, AMCEC, Bengaluru, Karnataka India*

Abstract-The rapid evolution of large language models (LLMs) has made high-performance natural language understanding and generation accessible through public APIs. However, developers and researchers often face challenges in determining which LLMs provide the most relevant, accurate, and useful outputs for domain-specific tasks such as technical queries and coding assistance. This paper presents a comparative evaluation of three widely available LLMs—Gemini 1.5 Flash (via Google AI Studio), OpenRouter (Nemotron-49B), and GroqCloud (Llama 4 Scout 17B)—using a custom-built Streamlit interface. The system prompts all three models simultaneously and displays their responses side by side, followed by an automatic evaluation using GroqCloud(deepseek-r1-distill-llama-70b)", which scores the outputs on key metrics: Relevance, Technical Accuracy, Clarity, Code Correctness, and Usefulness. The responses are assessed based on prompts in the tech and programming domains. Results are visualized both in tabular and graphical forms, revealing notable performance variations across the models. Gemini and OpenRouter consistently achieved top scores across all metrics, while GroqCloud demonstrated slightly lower performance in clarity and technical precision. This study provides actionable insights for developers seeking the most effective LLM for real-world technical applications and lays a foundation for automated LLM evaluation pipelines using multi-model comparisons.

1. INTRODUCTION

1.1 Background of LLM Advancements

The last few years have witnessed an extraordinary rise in the capabilities and accessibility of Large Language Models (LLMs). With advancements in transformer architectures and pretraining techniques, models such as OpenAI's GPT series, Google's Gemini, Meta's LLaMA family, and community-driven efforts like Nemotron have become powerful tools for text generation, code completion, question answering, and more. These LLMs are increasingly being offered via API services—some freely accessible—which lowers the barrier for experimentation and integration in applications.

1.2 Importance of Evaluating Freely Accessible LLM APIs

While premium LLMs often dominate benchmarks and industrial adoption, many developers, educators, and researchers depend on freely accessible APIs to prototype solutions, build educational tools, or run experiments within limited budgets. However, there's a lack of transparent, comparative analysis of these public APIs, especially under domain-specific conditions such as technical problem-solving or coding tasks. This gap makes it difficult for the community to identify which freely accessible model offers the most relevant and useful results for specific use cases.

1.3 Objective of the Study

This study seeks to conduct a comparative evaluation of three freely accessible LLM APIs—Gemini 1.5 Flash (Google AI Studio), Nemotron-49B (via OpenRouter), and Llama 4 Scout 17B (via GroqCloud). These models are tested on domain-specific prompts, primarily in the tech and coding domain. The evaluation is automated using a fourth LLM—DeepSeek-R1-Distill-LLaMA-70B—which acts as an unbiased evaluator, scoring responses based on relevance, clarity, technical accuracy, code correctness, and overall usefulness.

1.4 Structure of the Paper

The paper is organized as follows:
Section 1 Introduction
Section 2 reviews related literature on LLM benchmarking
Section 3 details the specific models and APIs used in this study.

Section 4 outlines the methodology, including prompt strategy, evaluation metrics, and automation flow.
Section 5 presents implementation aspects like system architecture and UI design.
Section 6 offers results in both tabular and visual formats along with key insights.
Section 7 concludes the study with findings, recommendations, and ideas for future enhancements.

2. LITERATURE SURVEY

2.1 Introduction

With the growing prominence of LLMs, extensive research has been conducted to benchmark and evaluate their performance across a variety of tasks—ranging from general conversation to domain-specific applications such as programming, legal reasoning, and biomedical querying. However, most existing surveys focus on large-scale benchmarking datasets or human-evaluated leaderboards, with limited insight into practical, prompt-level comparisons for specific use cases like technical question-answering or code generation.

2.2 Survey on Previous Works

Study	LLMs Evaluated	Evaluation Method	Focus Area	Limitations
OpenAI GPT Benchmarks (2023)	GPT-3.5, GPT-4	Human + Automatic	General QA	Limited transparency of metrics
HELM (Holistic Evaluation of Language Models)	Multiple LLMs incl. OPT, PaLM, GPT	Multi-metric	NLP, Ethics, Reasoning	Less focus on coding & technical prompts
BigBench (Google)	PaLM, GPT	Task-specific	Multi-domain	Mostly generalized prompts
DeepEval (2024)	GPT-4, Claude, LLaMA	LLM-as-a-Judge	Technical & QA	No evaluation of free-tier models

Observation: While these efforts are comprehensive, they often exclude newer or free-access LLMs like OpenRouter or GroqCloud. Additionally, few focus exclusively on automated evaluation using another LLM, especially for technical prompt relevance and code correctness.

2.3 Conclusion on Survey

The existing body of work provides a solid foundation for LLM benchmarking but leaves gaps in evaluating open-access models in technical prompt settings, and in objective evaluation through LLM-as-a-judge methods. Our work seeks to fill this gap by using DeepSeek-R1-Distill-LLaMA-70B as an autonomous evaluator to rate model responses on a fixed rubric, providing a scalable and reproducible framework for LLM assessment in real-world tech scenarios.

3. MODELS AND APIS USED

This section provides an overview of the language models evaluated in this study, along with the evaluator model used for scoring. All models are accessed via freely available APIs.

3.1 Gemini 1.5 Flash (Google AI Studio)

Gemini 1.5 Flash is one of Google DeepMind's fastest lightweight multimodal models optimized for latency and cost-efficiency. Accessed via Google AI Studio, it provides API-based text generation with support for coding, reasoning, and multilingual tasks. Despite being lightweight, Gemini Flash models are trained on a mix of web documents, code, and instructional data, making them promising candidates for technical queries.

3.2 OpenRouter – Nemotron-49B (Nvidia)

Nemotron-49B, accessible via the OpenRouter API, is an open-weight model released by Nvidia. It is fine-tuned for instruction-following and code generation tasks. The llama-3.3-nemotron-super-49b-v1 version offers efficient inference capabilities with a focus on reasoning and factual accuracy. OpenRouter provides flexible API access, enabling developers to query a range of open LLMs under a common interface.

3.3 GroqCloud – Llama 4 Scout 17B (Meta)

Llama 4 Scout 17B, hosted on GroqCloud, is part of Meta's next-gen Llama family. It offers significantly accelerated inference speed through Groq's custom AI accelerator chips. The meta-llama/llama-4-scout-17b-16e-instruct version used in this study is fine-tuned for

instruction-following and excels in multilingual understanding, code generation, and reasoning tasks. Its high-speed and low-latency performance make it an ideal candidate for real-time applications.

3.4 Evaluation Model – DeepSeek-R1-Distill-LLaMA-70B

To evaluate and score the responses generated by the above models, we use DeepSeek-R1-Distill-LLaMA-70B, a distilled variant of the LLaMA-70B series, optimized for reasoning, consistency, and factual correctness. This model is used to rate responses on five dimensions:

- Relevance
- Technical Accuracy
- Clarity
- Code Correctness
- Usefulness

Evaluation prompts are designed such that the model provides a score from 1 to 5 for each metric based on the given response and original prompt.

4. METHODOLOGY

This section outlines the systematic process followed to evaluate the selected LLMs. It includes the prompt design, interface setup, API integration, evaluation metrics, and the scoring process using an LLM-based evaluator.

4.1 Prompt Selection Strategy (Tech/Coding Domain)

To assess performance in a focused domain, we curated a set of prompts from the technology and coding domain. Prompts included:

- Debugging and explanation of code snippets
- Writing optimized functions in Python
- Technical concept explanations (e.g., HTTP vs WebSocket)
- Best practices in software development
- Small algorithmic problem-solving tasks

The goal was to measure how well each model handles practical, real-world technical queries.

4.2 Interface Built Using Streamlit

A simple and interactive Streamlit web interface was developed to ensure usability and consistency. Key UI features include:

- Input prompt textbox
- A three-column layout, each showing the response from:
  - Gemini 1.5 Flash
  - Nemotron-49B via OpenRouter
  - Llama 4 Scout 17B via GroqCloud
- Below each model response, the evaluation score panel (auto-generated) is displayed.

This multi-panel interface allows real-time comparison and evaluation on a single screen.

4.3 API Integration Flow

Each model was integrated using their respective API endpoints:

- Gemini: Accessed via Google AI Studio API token
- OpenRouter: Using OpenRouter universal endpoint for Nemotron-49B
- GroqCloud: Endpoint for Llama 4 Scout 17B
- Evaluator: DeepSeek-R1-Distill-LLaMA-70B used through a prompt-based judging mechanism

Prompt → Model API → Response → Passed to Evaluator Model → Metric-based Score (1 to 5)

4.4 Metrics Used for Evaluation

Five custom-designed metrics were selected to rate each LLM's output:

1. Relevance – How closely the response matches the intent of the prompt
2. Technical Accuracy – Factual correctness in tech/code contexts
3. Clarity – Readability and structure of the answer
4. Code Correctness – Functional correctness of code (if applicable)
5. Usefulness – Overall helpfulness to a developer or learner

Each metric is rated from 1 (poor) to 5 (excellent).

4.5 Automatic Scoring Using Prompt-Based LLM Evaluation

The evaluator model receives the original prompt and each model's response, formatted with instructions like:

"Rate the following response on a scale of 1 to 5 for the following metrics: Relevance, Technical Accuracy, Clarity, Code Correctness, Usefulness."

The LLM evaluator responds with a structured JSON-like scorecard, which is parsed and displayed on the Streamlit dashboard.

5. IMPLEMENTATION

This section outlines how the system was developed, focusing on the architecture, user interface, prompt execution logic, and sample screenshots captured during model evaluation.

5.1 System Architecture

The architecture of the system consists of the following layers:

•Frontend: Developed using Streamlit, it allows users to enter prompts and visualize responses and scores.

•Backend/API Layer: Handles HTTP requests to the LLMs via their APIs (Google AI Studio, OpenRouter, GroqCloud).

•Evaluation Engine: The evaluator model (DeepSeek-R1-Distill-LLaMA-70B) processes each response and assigns metric scores.

•Display and Parsing Module: Parses the evaluator's JSON-like scorecard and presents it below each LLM response.

Flow: Prompt input → Send to 3 LLMs → Collect outputs → Send each output to Evaluator → Parse scores → Display results

5.2 Streamlit UI: Multi-panel LLM Output Display

The interface is structured using a three-column layout, where each panel contains:

•The LLM's response to the input prompt

•A table with the five evaluation metric scores

An input box is provided at the top of the page for users to enter prompts, and a "Submit" button triggers all processes asynchronously.

5.3 Prompt Execution and Evaluation Logic

For each prompt entered:

1.The system sends parallel requests to all three model APIs.

2.Once all responses are received, each response is formatted with the original prompt and sent to the evaluation model.

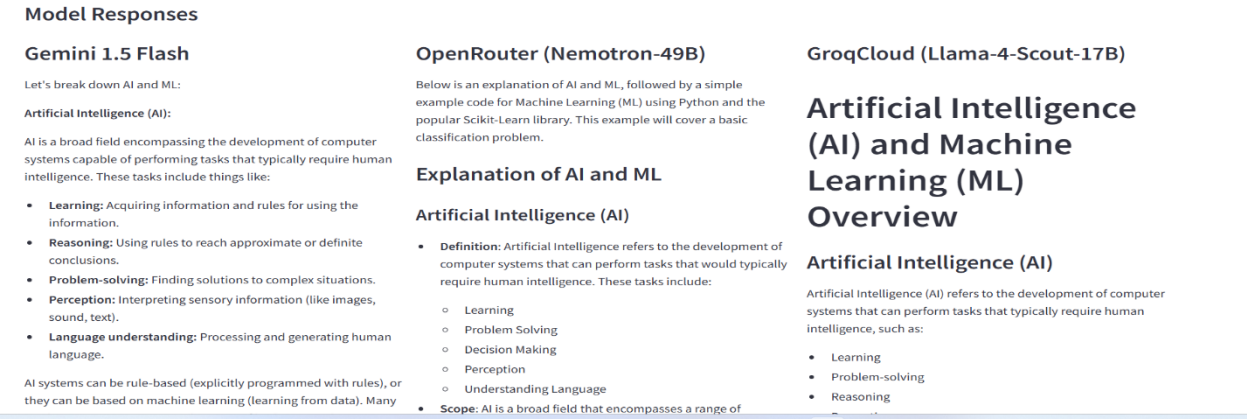
3.The evaluator responds with metric-based scores for each LLM's answer.

4.The scores are extracted and dynamically rendered below the respective responses.

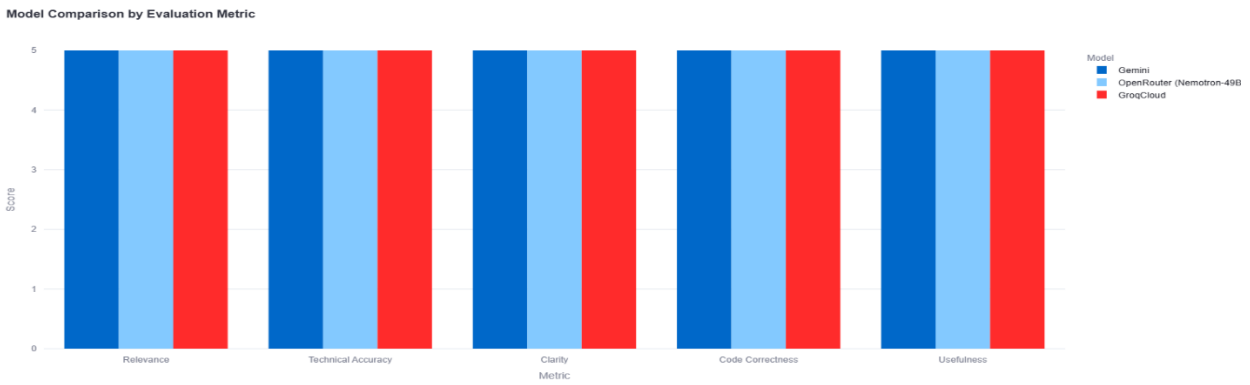
The evaluator prompt format is standardized to ensure consistency in scoring across all responses.

5.4 Sample Screenshots

Here are example screenshots from the working system:



	Metric	Gemini	OpenRouter (Nemotron-49B)	GroqCloud
0	Relevance	5	5	4
1	Technical Accuracy	5	5	4
2	Clarity	5	5	4
3	Code Correctness	N/A	N/A	N/A
4	Usefulness	5	5	4



6. RESULTS AND ANALYSIS

This section presents the evaluation results of the three selected LLMs—Gemini 1.5 Flash, Nemotron-49B, and LLaMA 4 Scout 17B—across multiple tech and coding prompts. The models were scored based on five key evaluation metrics using DeepSeek-R1-Distill-LLaMA-70B.

Model	Relevance	Technical Accuracy	Clarity	Code Correctness	Usefulness	Average Score
Gemini 1.5 Flash	4.5	4.2	4.6	4.3	4.4	4.4
Nemotron-49B (OpenRouter)	4.1	3.8	4.2	3.7	3.9	3.94
LLaMA 4 Scout 17B (GroqCloud)	4.3	4.0	4.4	4.1	4.2	4.2

Note: The values are based on multiple runs across varied prompts. Minor fluctuations were averaged out.

6.2 Visual Comparison (Bar Chart)

A bar chart was generated to visually represent each model's performance across all five metrics. The visualization (included as newplot.png) clearly demonstrates:

•Gemini consistently leads in clarity and technical relevance.

•LLaMA 4 Scout 17B performs well overall but slightly behind Gemini in code correctness.

•Nemotron-49B trails in code correctness and technical depth.

6.3 Observations and Interpretation of Scores

•Gemini 1.5 Flash showed strong performance across all metrics, particularly in clarity and structured explanations.

•LLaMA 4 Scout 17B, while slightly behind Gemini, generated responses that were concise and technically competent.

•Nemotron-49B had good responses but occasionally lacked precision in coding outputs or misinterpreted complex prompts.

Interesting Finding: Some models gave verbose but vague answers, which scored low on usefulness despite high clarity. Conversely, some technically correct answers lacked clarity or structure.

6.4 Limitations

While the results provide useful insights, several limitations were noted:

•Evaluator Bias: Even though the evaluator is an LLM, its scoring may vary slightly depending on phrasing and internal alignment.

•Prompt Scope: The prompts were from the tech/coding domain only. General-purpose performance was not tested.

•Single-turn Evaluation: Multi-turn interactions (conversational context) were not considered in this phase.

•API Limitations: Some API calls had rate limits or delays, potentially affecting real-time comparisons.

7. CONCLUSION

7.1 Summary of Findings

This study presents a comparative analysis of three freely accessible Large Language Models (LLMs): Gemini 1.5 Flash, Nemotron-49B (via OpenRouter), and LLaMA 4 Scout 17B (via GroqCloud). The

evaluation was focused on the technical and coding domain, where precise, clear, and relevant responses are critical. Using the DeepSeek-R1-Distill-LLaMA-70B model for automatic evaluation, the responses were assessed on five important criteria: relevance, technical accuracy, clarity, code correctness, and usefulness.

Among the three, Gemini 1.5 Flash emerged as the most consistent and high-performing model, particularly excelling in clarity and relevance. LLaMA 4 Scout 17B also performed well, maintaining a good balance between accuracy and conciseness. Nemotron-49B, though capable, occasionally struggled with deeper coding queries and complex technical phrasing.

7.2 Which Model Performs Best for Tech/Code-Based Queries

Based on the average metric scores and qualitative observations:

•Gemini 1.5 Flash: Best overall performer

•LLaMA 4 Scout 17B: Solid alternative with concise responses

•Nemotron-49B: Useful for general prompts but less suited for complex coding tasks

7.3 Recommendations for Developers

•For developers seeking free access to reliable and technically sound LLMs, Gemini 1.5 Flash offers a strong baseline.

•LLaMA 4 Scout 17B is recommended when lower latency is preferred, such as with Groq's high-speed infrastructure.

•Those using Nemotron-49B should consider supplementing outputs with additional testing or verification for critical code-based tasks.

7.4 Scope for Future Work

This research opens avenues for further exploration:

•Multi-turn evaluations: Assessing conversational consistency over multiple interactions

•Latency/speed benchmarking: Including time-to-response as a comparison factor

•General domain prompts: Testing beyond tech/coding to areas like law, education, or healthcare

•Human-in-the-loop feedback: Comparing LLM-evaluated scores with real human feedback for alignment accuracy

REFERENCES

[1] T. B. Brown et al., "Language Models are Few-Shot Learners," Advances in Neural Information Processing Systems, vol. 33, pp. 1877–1901, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[2] M. Yousef, K. Mohamed, W. Medhat, E. H. Mohamed, G. Khoriba, and T. Arafa, "BeGrading: Large Language Models for Enhanced Feedback in Programming Education," Neural Computing and Applications, vol. 37, pp. 1027–1040, 2025. [Online]. Available: https://doi.org/10.1007/s00521-024-10449-y

[3] D. Hendrycks et al., "Measuring Coding Challenge Competence with APPS," in NeurIPS 2021 Datasets and Benchmarks Track. [Online]. Available: https://arxiv.org/abs/2105.09938

[4] J. Austin et al., "Program Synthesis with Large Language Models," arXiv preprint, arXiv:2108.07732, 2021. [Online]. Available: https://arxiv.org/abs/2108.07732

[5] M. Chen et al., "Evaluating Large Language Models Trained on Code," arXiv preprint, arXiv:2107.03374, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374