A Comparative Analysis of Leading Language Models for System Integration

Chapter 1

Understanding the Landscape

Overview of current LLM offerings

Model Descriptions

Providers tested:

1 - OpenAI: Gpt

2 - Google: Gemini

3 - Hosted Open Source Model : meta-llama

GPT-4o-Mini

- Cost-effective, swift OpenAI model
- Successor to GPT-3.5 Turbo
- Supports text and vision inputs

Gemini 2.5 Flash

- Google's balanced model: speed, quality, cost
- Ideal for user-facing applications
- "Thinking" capabilities for reasoning

Gemini 2.0 Flash

- Combining stability and accuracy , and the most cost effective and used model in production.
- High volume , low latency , and moderate accuracy for tasks that require reasoning , excels in text based tasks.

Open Source Model:

1

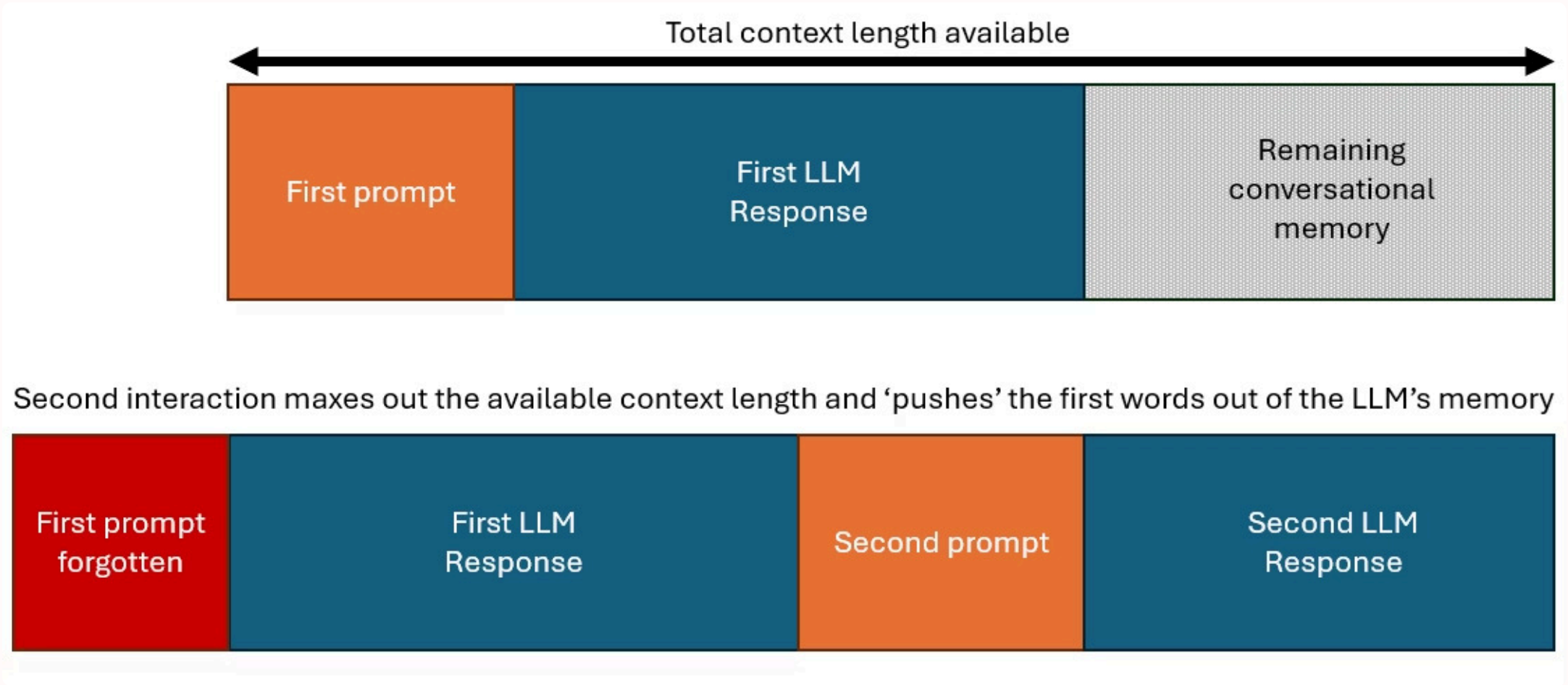
meta-llama/Llama-3.1-8B-Instruct

- Open-source instruction-tuned model from Meta (8B parameters).
- Strong reasoning and language quality for its size.
- **Pros:** Free to use, customizable, can fine-tune for domain-specific tasks.
- **Cons:** Requires significant compute to host (GPU memory ~16–24GB), infra scaling is non-trivial, monitoring/updates are on you.
- Best suited for teams that want **full control + on-prem/cloud deployment**, but comes with higher ops overhead.

Context Window Capabilities

What is Context Window?

The context window defines the amount of text a model can process, vital for long documents or conversations.



When using an AI system, how much it “remembers” depends on the situation:

- **Chatting with the AI**: The AI needs to remember past messages so it can respond in a natural way and keep track of the conversation.
- **Creating things like quizzes, flashcards, or questions**: The AI doesn’t need to keep the history. Once it generates the content, we’re done and can move on without it remembering what came before.
- **Adjusting the AI’s personality or style**: This is somewhere in between. For example, if we want the AI to act like a tutor, we set its “persona” to teacher mode and it stays that way for the whole session. If later we want it to act in a different style, like being more playful or curious, we just update the instructions and it switches modes.

This matters because how the AI remembers (or forgets) shapes how useful it is—whether it’s for having a conversation, generating content, or adapting to different situations.

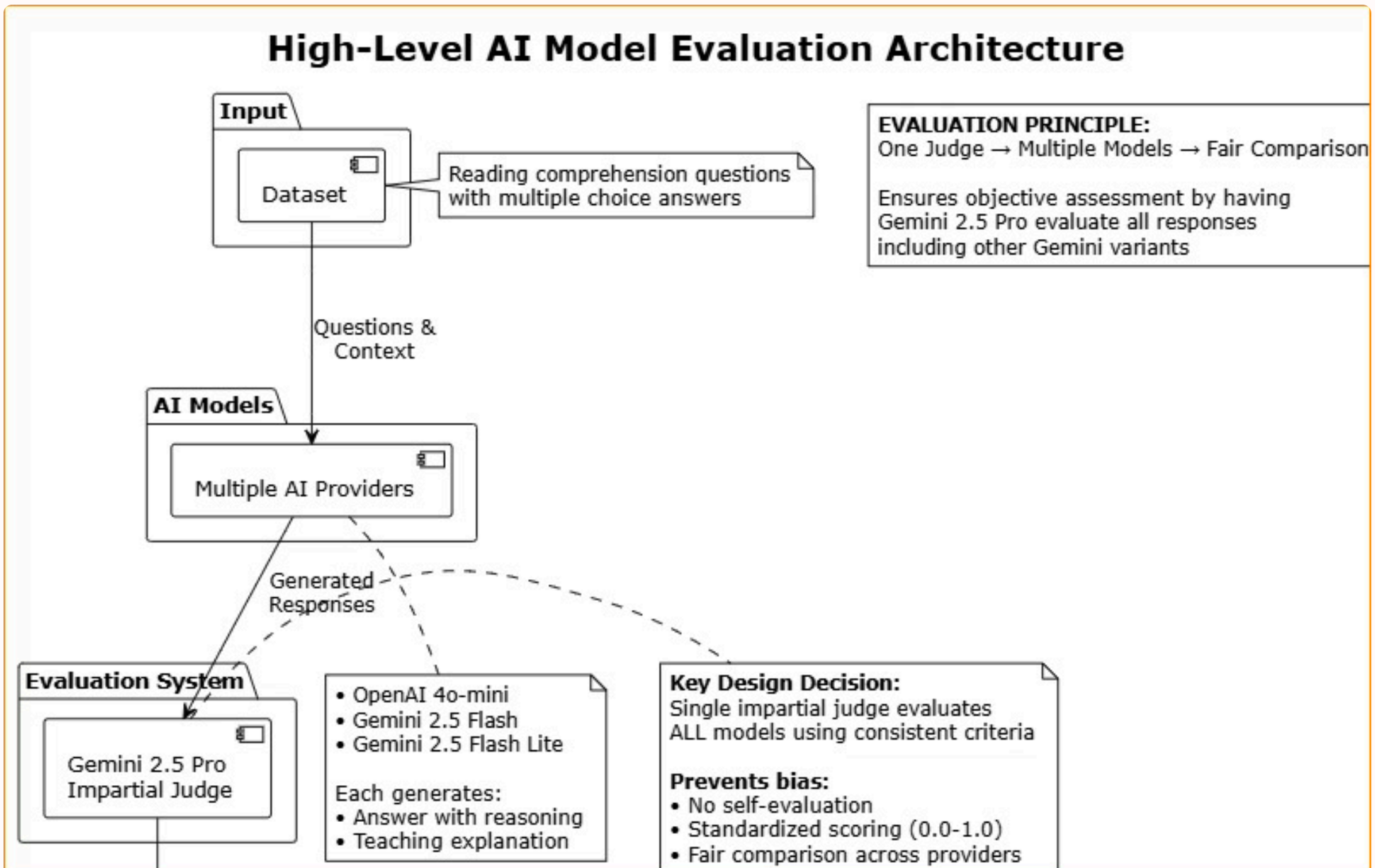
Model	Context Window Size
gpt-4o-mini	128,000 tokens
gemini 2.5 flash	1,048,576 tokens
gemini 2.0 flash	1,048,576 tokens
meta-llama/Llama-3.1-8B-Instruct	128,000 tokens

Chapter 2

Performance Metrics

Accuracy, latency , Cost.

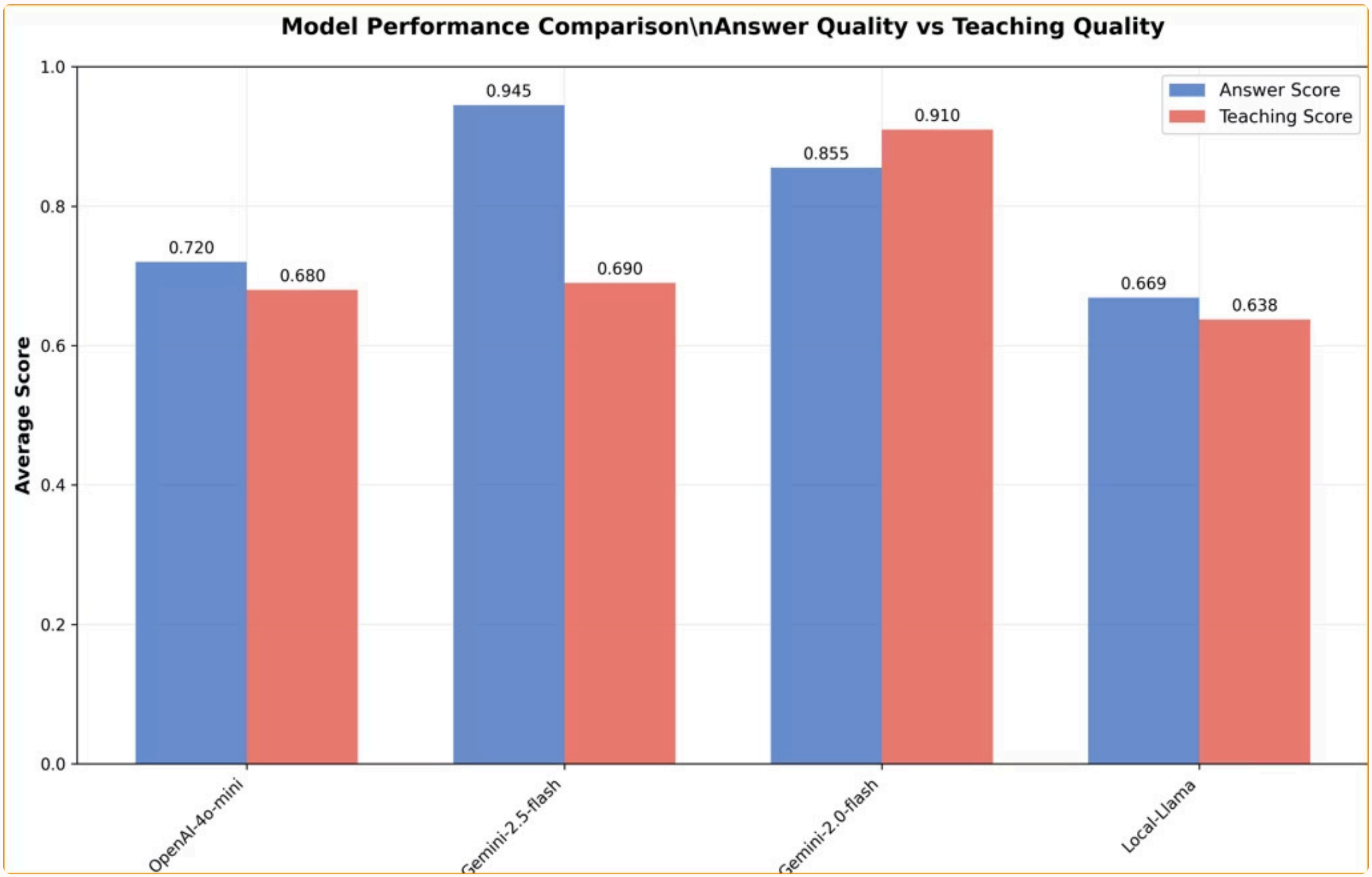
Testing method:



Accuracy Benchmarks

Evaluating model performance against industry standards.

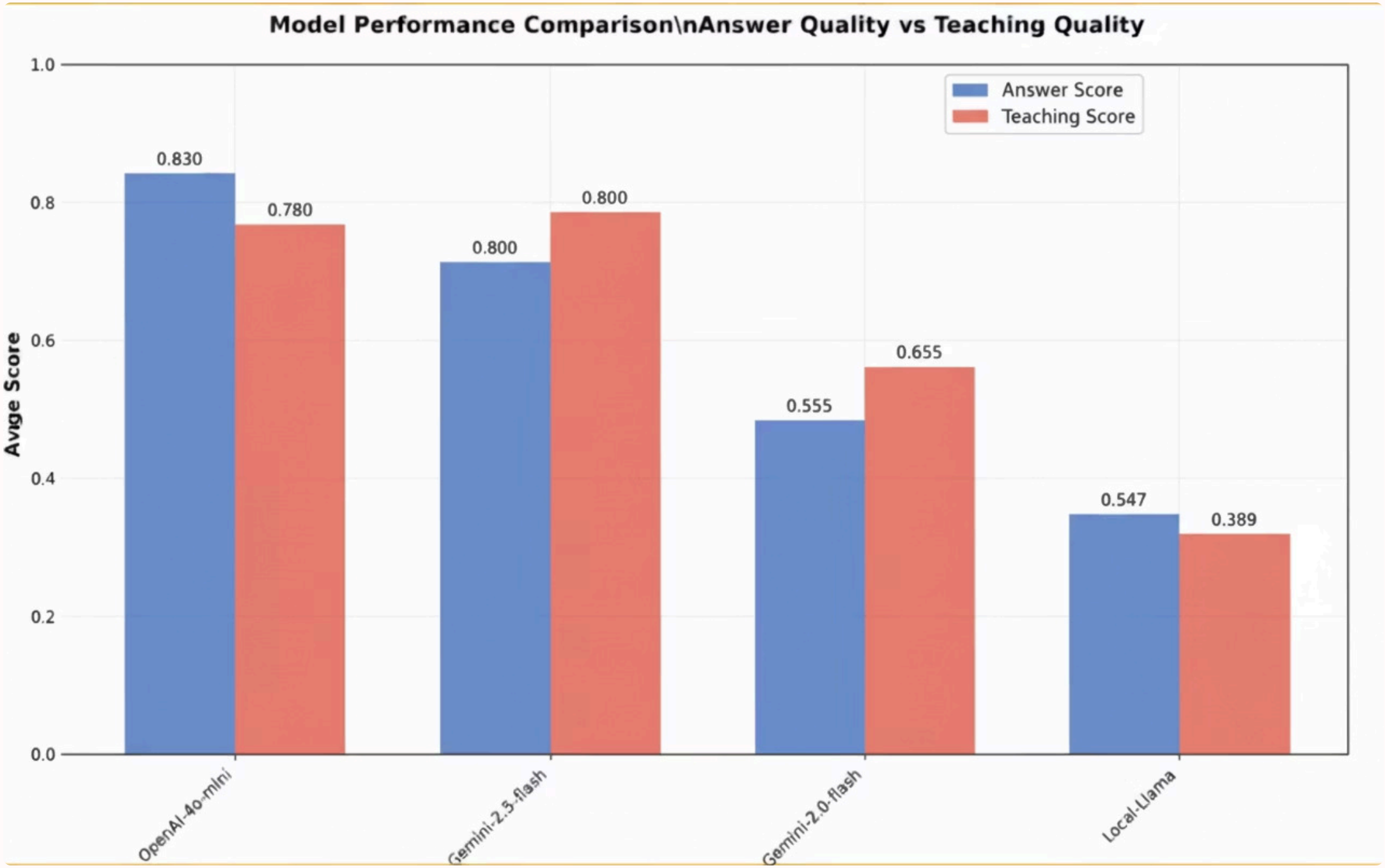
1 - English Set :



- Gemini 2.5 flash has the highest accuracy in general
- Gemini 2.0 flash is quite close to it , even though costing way less
- on average OpenAI models are performing worse than the gemini family, and cost a bit more.
- The local Llama model showed the lowest results , considering it for this project is not viable.

This benchmark was done for a set of English questions , that test comprehension , and ability to explain, and tutor.

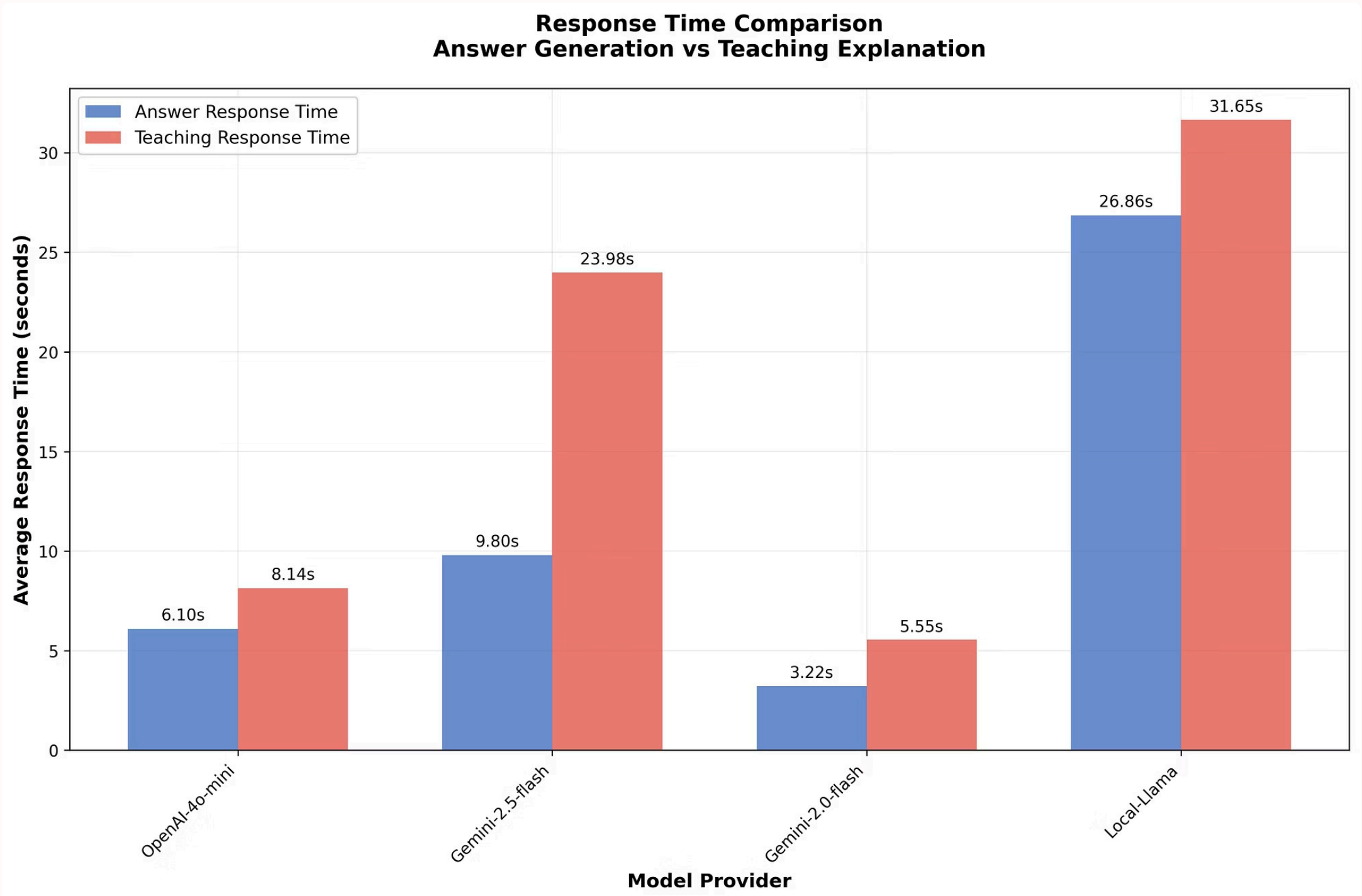
2- History Set (Malay language):



- GPT-4o mini gives better answers in Malay compared to Gemini 2.5 flash, but it struggles to clearly explain those answers.
- Gemini 2.5 flash stays strong overall, with more balanced performance between answering and teaching.
- Gemini 2.0 flash performs poorly in both answering and teaching, while the local LLaMA model is not usable for Malay at all.

This benchmark was done for a set of Malay questions, testing both direct answering and the ability to explain like a tutor.

Latency:



- In testing, Gemini 2.5 Flash showed higher delays during teaching tasks.
- Gemini 2.0 Flash was noticeably faster, with much lower delays in both teaching and answering, while still delivering nearly the same quality. This makes it feel more efficient and refined.
- OpenAI’s models were generally the fastest to respond, but earlier benchmarks showed they were less accurate for this teaching task.
- The local LLaMA model had very high delays, making it unusable in any interactive setting, whether chat or content generation.

Choosing Between Cloud Models and Open-Source Models

Fine-Tuning

Cloud Models (e.g., GPT-4o mini, Gemini 2.0/2.5 Flash):

Fine-tuning options are limited or not available at all (mostly prompt-based control).

Open Source (e.g., LLaMA):

Can be fine-tuned extensively, adapted to niche tasks or languages.

Setup Overhead

Cloud Models:

No setup required — access instantly through an API.

Open Source Models:

Requires infrastructure (GPUs/servers), installation, and optimization.

Flexibility

Cloud Models:

Easier to scale and integrate; flexibility comes from APIs and built-in features.

Open Source Models:

Full control over weights, training data, and deployment environment.

Maintenance

Cloud Models:

Always updated by the provider; you automatically use the latest version.

Open Source Models:

Require frequent updates, retraining, and patching; updates can be slow.

Which Models Are Most Suitable?

1

Gemini 2.5 Flash

- Highest overall accuracy in English and Malay.
- Balanced between answering and teaching.
- Slower response times during teaching tasks.

2

Gemini 2.0 Flash

- Nearly as strong as 2.5 in English, with lower cost.
- Faster and more efficient, but weaker in Malay performance.

3

OpenAI GPT-4o Mini

- Strong in Malay answering.
- Weaker in explanations and teaching tasks.
- Very fast response times.

4

Local LLaMA

- Lowest accuracy in both English and Malay.
- Very high latency → unusable for interactive use.
- Not viable for this project.

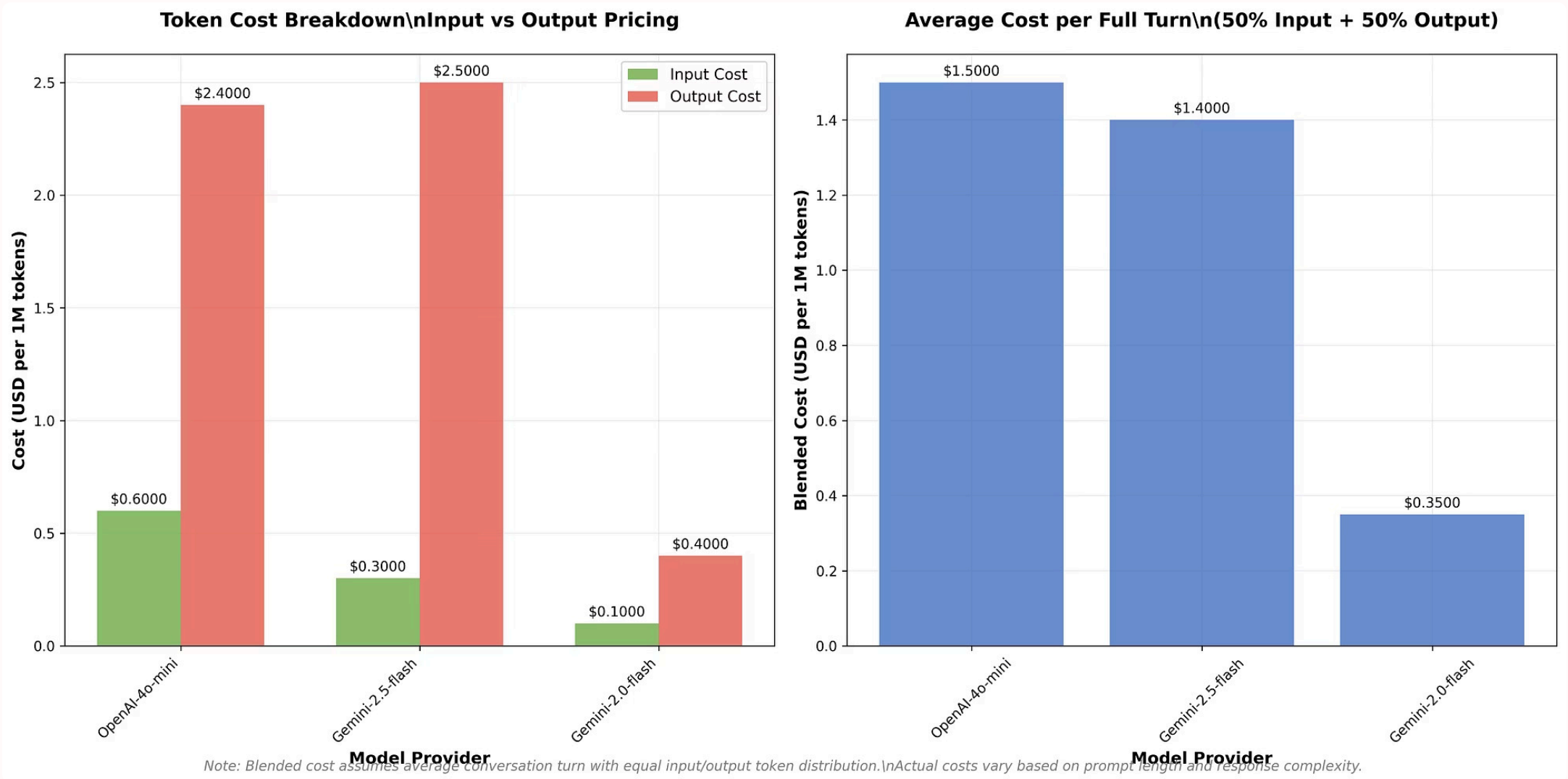
Conclusion

Cloud models (Gemini and OpenAI) are the most suitable choice due to their balance of accuracy, speed, and ease of deployment, while local open-source models are not practical.

Moving on to pricing.

Cost Breakdown

Pricing per 1 million tokens (Input/Output).



Why Output Costs More Than Input

When using AI language models, it usually costs more to get answers (output) than to give instructions or text (input). This isn't random—it comes down to how the AI works.

- **Input (what you type in):** The AI can read and process all your text in one go. Think of it like scanning a page—it happens fast and efficiently.
- **Output (what the AI writes back):** The AI doesn't write everything at once. Instead, it creates one word (or piece of a word) at a time. Each new word depends on the previous ones, so the AI has to "think" repeatedly until the full answer is finished. That makes output more expensive.

What a "Turn" Means in a Conversation

A **turn** is one back-and-forth between you and the AI. It has two parts:

1. **Your input** – this includes your message, plus any background information the system adds.
2. **The AI's output** – the response it gives back.

A conversation is just a series of these turns, where each new turn builds on the memory of the last ones, so the AI can stay on topic and make sense.

How does input, output prices affect our choice?

When making an AI system , Each component or model receives different amounts of input , and geneartes different amounts of output , optimizing this is crucial for cost effectiveness.

1- RAG:

This diagram explains **how RAG (Retrieval-Augmented Generation) works**.

1. You ask a question.
2. The system looks up relevant information in a special database (like a knowledge library).
3. That information is added to your question.
4. The AI then uses both your question and the extra information to give a more accurate and informed answer.

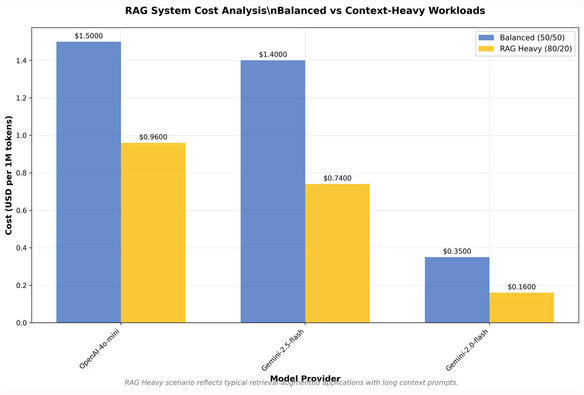
So instead of the AI relying only on its memory, it first retrieves supporting facts and then generates a response.

Analogy

Think of it like asking a friend for help with homework:

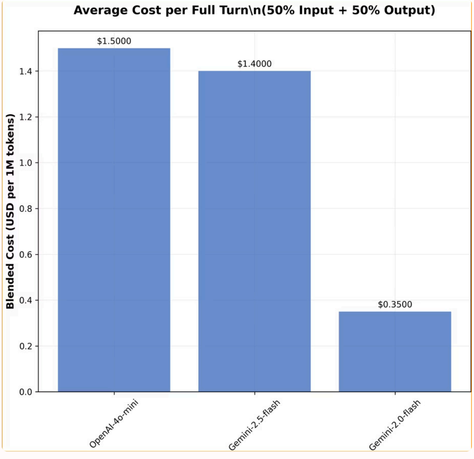
- **You (the student)**: Ask the question.
- **The friend**: Doesn't just guess—they quickly check a textbook or notes (the database) for useful information.
- **They combine** your question with what they found and explain the answer to you.

This way, the response is both smarter and more trustworthy.



- in a rag system since the context gets provided in the prompt it's an input heavy system.
- 80/20 sometimes.
- In this case it gives us more freedom and space to use a premium model.
- balancing the accuracy and cost effectiveness.

2- reacting to enviroment:

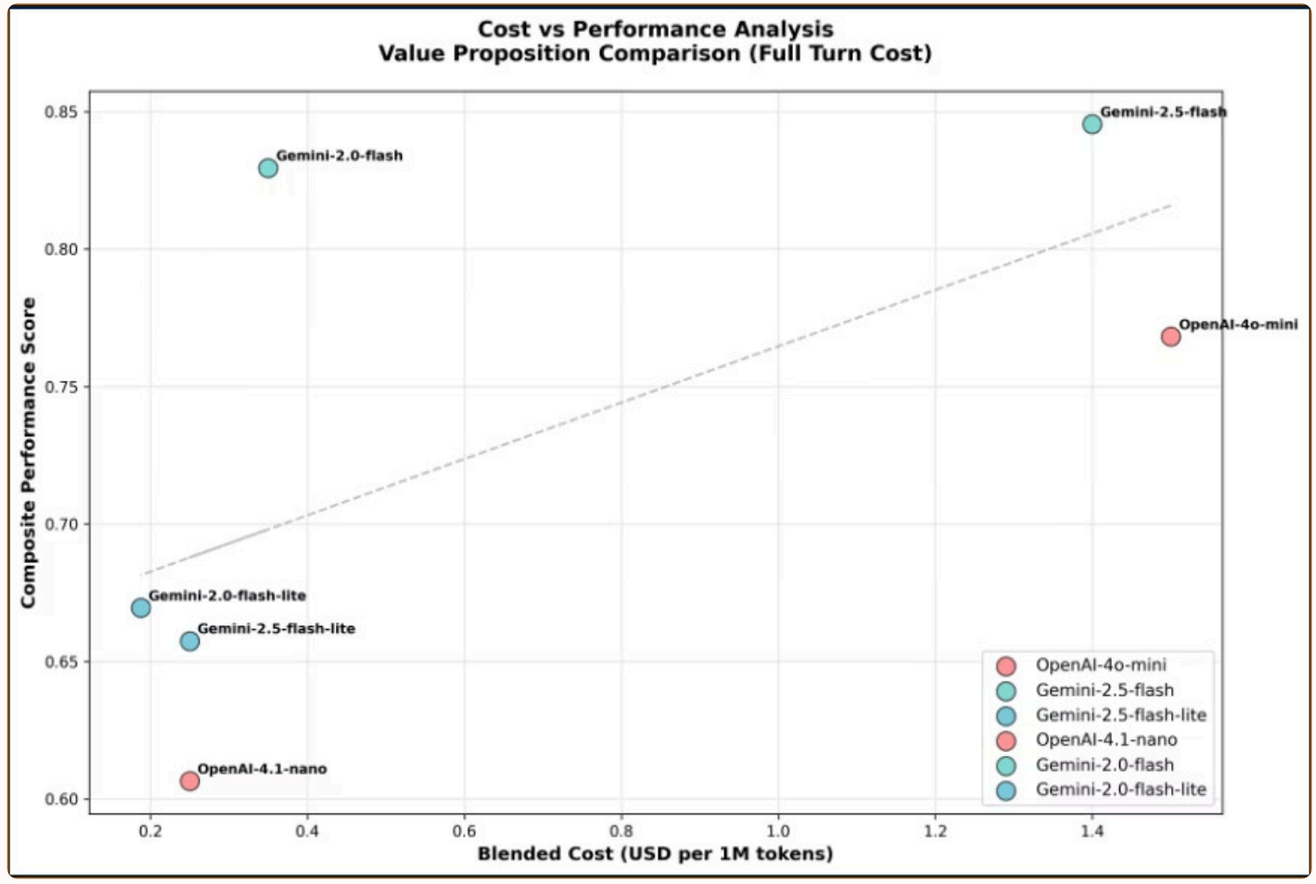


- Sometimes the AI uses information from its surroundings, like your preferences or details about the situation. This is a **moderate level of usage**—it's about half relying on stored knowledge and half on outside context.
- If the outside information is **simple**, a regular model is usually good enough. But if the information is **complex or detailed**, it's worth using a more advanced (and more expensive) model to make sure the answers are accurate and reliable.

3- Generative:

- When the goal is to **generate content like questions, quizzes, or flashcards**, the balance is different. It's mostly **about producing output (80%)** and only a little about understanding the input (20%).
- Because generating lots of output is the **most expensive part of using AI**, cost becomes a key factor to think about in this type of task.

Correlation, cost and performance:



- This scatterplot shows how performance relates to cost.
- The **Gemini models** are performing better than the **OpenAI models**, while also being more affordable.
- The standout is **Gemini 2.0 Flash**—its performance is close to Gemini 2.5 Flash but comes at a much lower cost, making it the best value overall.

Cost Impact on Model Choice

Input-Heavy Models (e.g., RAG)

Lower input token price is key. **Gemini 2.5 flash** Would reason over context better.

Balanced Models (e.g., Conversational AI)

Seek low combined input/output cost. **Gemini 2.0 flash** offers balance, it would be costeffective and have direct instructions.

Generation-Heavy Models (e.g., Content Creation)

Output token price dominates. **Gemini 2.0 Flash** is budget-friendly for extensive text generation, with the right architecture i can make it way better.