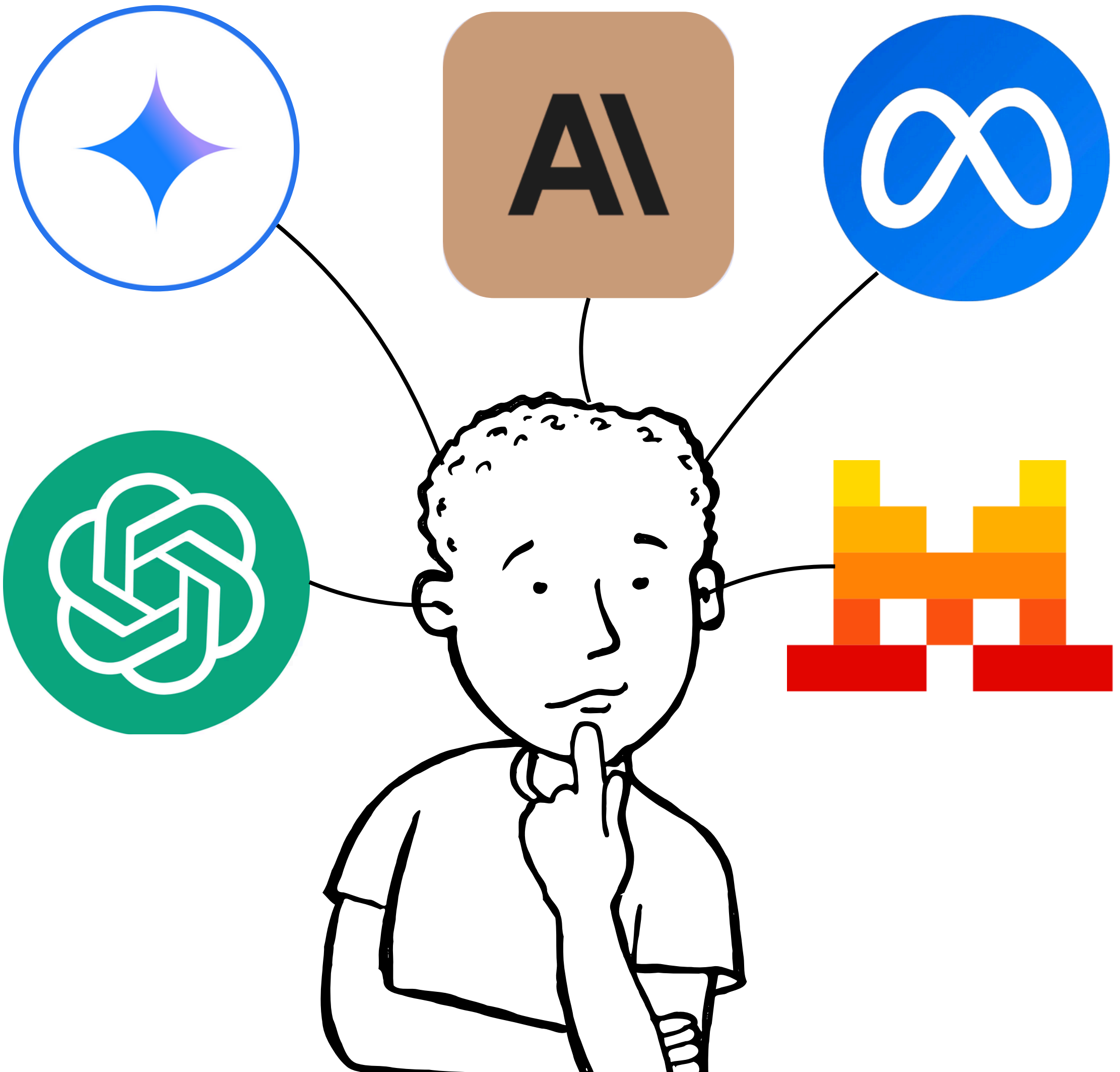
AI Accelerator Program

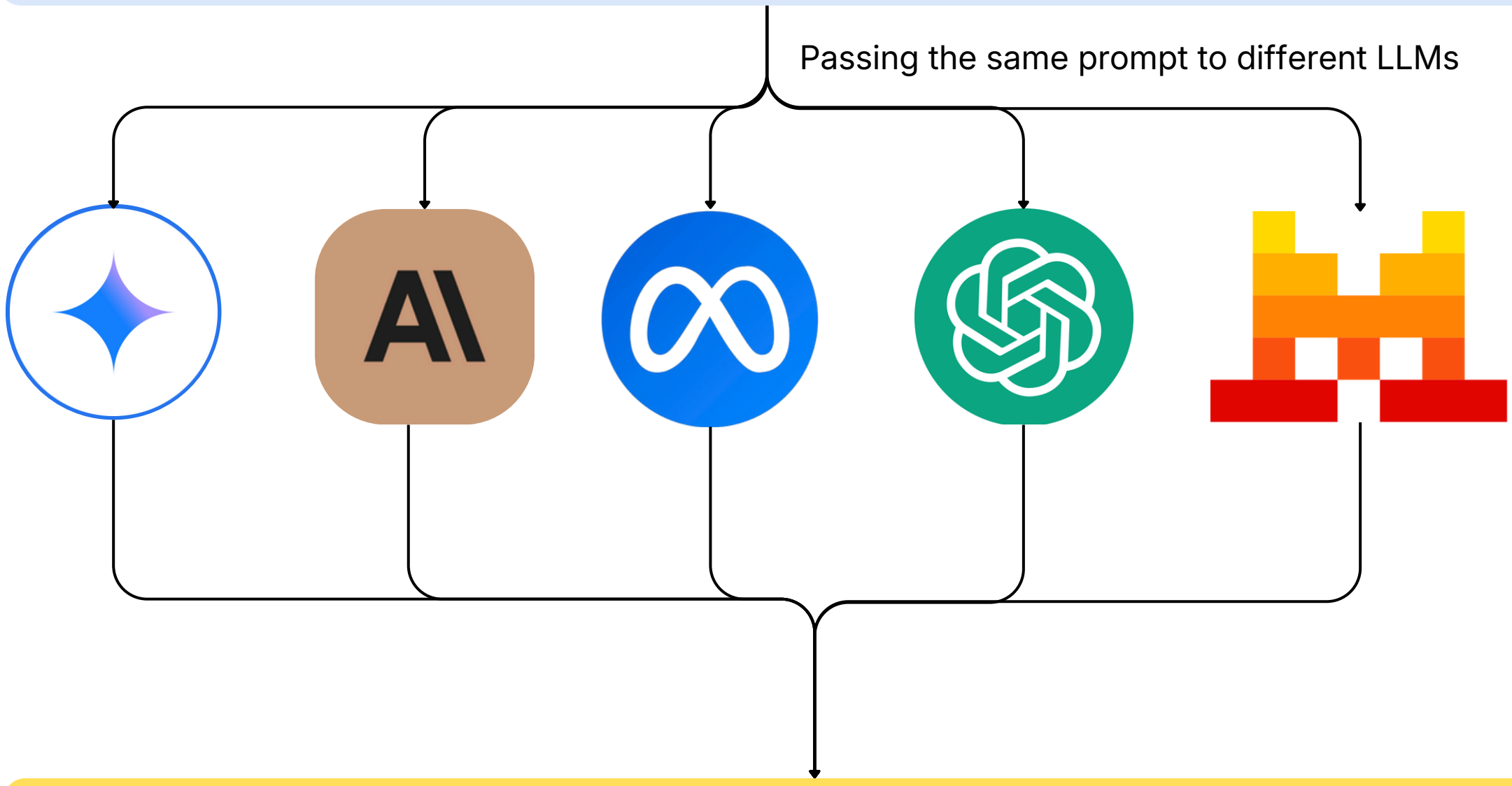
Analytics Vidhya

How to choose the right LLMs for your GenAI Application



Prompt

Write an email to my HR requesting sick leave



LLMs respond differently to the same prompt

Different models are good at different things. Some are better at **coding**, while others are stronger at **reasoning**, **summarizing**, or having **conversations**.

For example, we use **ChatGPT** for everyday questions, formatting text, and quick research, but we choose **Claude** when we need more advanced coding help.

The main point is that **there isn’t one model that’s best at everything**.

Why do LLMs perform differently?

Before choosing or evaluating a model, it helps to understand why LLMs don’t all perform the same.

A few key factors explain the differences.

1. Training data and domain

A model’s performance depends heavily on the data it was trained on. Models trained on large amounts of code tend to be stronger at programming, while those trained on academic or general web content may be better at reasoning and summarization.

2. Fine-tuning and RAG

Most real applications are domain-specific. For example, an employee system must follow company-specific rules and policies. Two common ways to adapt models to such needs are fine-tuning and Retrieval-Augmented Generation (RAG).
RAG adds relevant external information to guide the model’s answers without changing the model itself. Fine-tuning goes further by retraining the model on domain-specific data.

3. Architecture differences

Even though many LLMs use transformer architectures, they can still perform quite differently. Variations in model size, training data, and design choices lead to noticeable performance gaps.
Understanding these factors makes it easier to see why model evaluation is important and when it becomes necessary.

When Do You Need to Evaluate an LLM?

Model evaluation becomes essential in the following scenarios.

1. Before You Start Building

If you're building a production-grade GenAI application, early model selection is critical.

At this stage, you should clearly define the problem: the application's scope, your expected number of users, any latency expectations, and privacy requirements. You should also identify non-negotiable requirements (SLOs). For example, perhaps you need accuracy to be above 90% and latency below 2 seconds. You'll need to consider cost implications as well, such as funding constraints at early stages, expected user growth, and request volume and scaling.

Common evaluation factors include:

- Speed and latency
- Accuracy and reliability
- Data privacy and compliance

2. When Upgrading an Existing Application to a New Model

Another common use case is upgrading a model when the application is already in production.

In this scenario:

- Core metrics usually remain the same
- The features will be already implemented and also benchmarked on existing model.
- There is already a baseline performance threshold that must be preserved

Credits: FreeCodeCamp

Upgrading a model is not always straightforward. System prompts that worked well previously may behave very differently with a new model.
From personal experience, after upgrading an LLM, responses that were previously well formatted suddenly became inconsistent and poorly structured.

When an application is live, evaluation focuses on regression testing and measurable improvement:

- Existing features and prompts must be revalidated
- Metrics should be evaluated feature by feature
- Improvements should be data-driven, not anecdotal

Key Factors to Evaluate

These are the most important factors to evaluate when you're choosing a model for your task:

1. Accuracy and Consistency

Accuracy and consistency are in most cases the most important factors when building LLM-based applications.

Accuracy refers to whether the responses generated by the model are correct or not, while consistency measures the model's tendency to produce the same response when given the same input multiple times. Ideally, a model should demonstrate both accurate and consistent behavior.

For example, consider a RAG application where a user asks a question. If the model generates the correct answer on the first attempt, an incorrect answer on the second attempt, and then the correct answer again on the third attempt, this indicates that the responses are not consistent even if accuracy is occasionally achieved.

Credits: FreeCodeCamp

When selecting an LLM, ask yourself the following questions:

- Does the model hallucinate on simple or complex queries?
- Are responses consistent across multiple runs?
- Does accuracy degrade for edge cases?

2. Latency

Alongside accuracy, it is important to consider the performance of your application. From a user's perspective, a system with high latency or slow performance can lead to negative feedback or decreased usage, even if the responses are accurate.

For example, consider a streaming-response RAG application that delivers answers chunk by chunk. If the first chunk arrives after 15 seconds and the complete response after 60 seconds, this indicates poor performance from a user experience standpoint.

When evaluating LLMs, ask yourself the following questions:

- How quickly does the model respond?
- Is latency predictable under load?

3**. Cost**

LLMs are not free, and each token comes with a price. So it's important to consider cost when selecting a model. You should perform proper calculations and assessments to estimate the expected load. Consider how many requests you'll make per minute and the size of each request, as this will directly impact your overall expenses.

When evaluating LLMs, ask yourself the following questions:

- What is the cost per request or per token?
- Is the model viable for your expected traffic, especially in early-stage or proof-of-concept phases?
-

4. Ethical and Responsible AI Considerations

With generative AI, it has become even more critical to enforce ethical constraints and implement responsible AI. Without these guidelines and restrictions, models can produce content that is harmful to society, which should never be tolerated.

For example, your application should **not provide assistance for harmful requests**, such as “How to make a bomb.”

When evaluating LLMs, ask yourself the following questions:

- Does the model adhere to safety and community guidelines?
- Are harmful, biased, or disallowed requests properly rejected?

Responsible AI is not optional. It’s a shared responsibility across developers, product owners, and managers. Ignoring ethical considerations can harm both the product and society.

5. Context Window

If your application processes large documents or relies on long conversations, the context window becomes a critical factor.
The context window includes both **input and output tokens**, not just the response.

Examples:

- GPT-3: 4K tokens
- GPT-3.5 Turbo: 8.1K tokens

Credits: FreeCodeCamp