WORKING WITH LARGE LANGUAGE MODELS

LLM *evaluation* , from first prompt to production.

How teams measure whether a model — and the product built around it — is accurate, safe, and aligned with the job it was hired to do.

AUDIENCE

Engineers, PMs, anyone shipping LLM features

IN EIGHT SLIDES

Two kinds of evals · Benchmark example · Pricing · Why hard · The lifecycle

SOURCE

evidentlyai.com / llm-guide

THE CORE DISTINCTION

Benchmarks are *school exams*. Product evals are *job performance reviews*.

A · MODEL EVALUATION

How capable is the LLM in general?

Tests the raw model on standardized benchmarks — coding, translation, reasoning, safety — to compare it against other LLMs.

SCOPE	The model itself, via direct prompts
DATA	General benchmark datasets (MMLU, etc.)
RUN WHEN	A new model is released
USEFUL FOR	Picking which LLM to build with

B · PRODUCT EVALUATION

How well does *your* system do *its* job?

Tests the whole application — prompts, retrieval, guardrails — against the specific scenarios your real users send in.

SCOPE	The full system, end-to-end
DATA	Custom, scenario-based test cases
RUN WHEN	Continuously, from dev to production
USEFUL FOR	Shipping a reliable product

EXAMPLE · FRONTIER MODEL BENCHMARKS

How capable is the LLM? Benchmarks turn that question into *numbers you can compare.*

BENCHMARK		GEMINI 3.5 FLASH	GEMINI 3.1 PRO	CLAUDE SONNET 4.6	CLAUDE OPUS 4.7	GPT-5.5
CODING	Terminal-bench 2.1 Agentic terminal coding	76.2%	70.3%	—	66.1%	**78.2%**
	SWE-Bench Pro Diverse agentic coding tasks	55.1%	54.2%	—	**64.3%**	58.6%
AGENTIC	MCP Atlas Multi-step workflows using MCP	**83.6%**	78.2%	69.5%	79.1%	75.3%
	Toolathlon Real-world general tool use	**56.5%**	—	—	—	55.6%
EXPERT	Finance Agent v2 Financial analysis & decisions	**57.9%**	43.0%	51.0%	51.5%	51.8%
MULTIMODAL	CharXiv Reasoning Synthesis from complex charts	**84.2%**	83.3%	72.4%	82.1%	84.1%
	MMMU-Pro Multimodal understanding	**83.6%**	80.5%	74.5%	75.2%	81.2%
LONG CTX	MRCR v2 (8-needle) 128k average · long context	77.3%	84.9%	84.9%	59.3%	**94.8%**
REASONING	Humanity's Last Exam Academic reasoning, text + MM	40.2%	44.4%	33.2%	**46.9%**	41.4%
	ARC-AGI-2 Abstract reasoning puzzles	72.1%	77.1%	58.3%	75.8%	**84.6%**

Bold & warm = best in row

Source · deepmind.google / evals-methodology / gemini-3-5-flash

PRICING SNAPSHOT · JANUARY 2026

How expensive is the LLM? Output tokens cost *4–8× more* than input tokens.

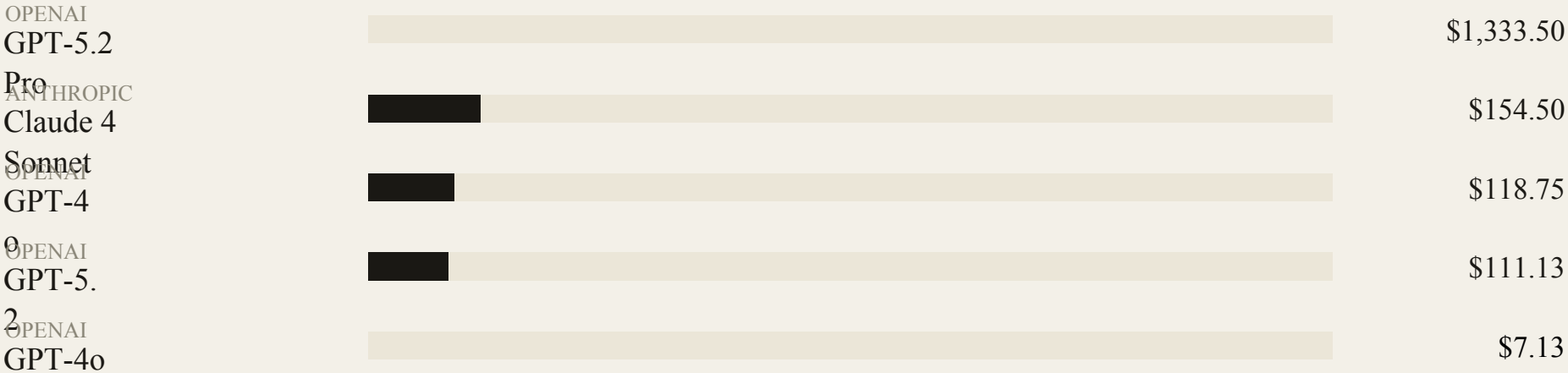
PROVIDER	MODEL	CONTEXT	INPUT $ / 1M TOKENS	OUTPUT $ / 1M TOKENS	OUTPUT : INPUT
OPENAI	GPT 5.2	400K	$1.75	$14.00	8.0×
OPENAI	GPT 5.2 Pro	400K	$21.00	**$168.00**	8.0×
OPENAI	GPT 4o	128K	$2.50	$10.00	4.0×
OPENAI	GPT 4o Mini	128K	$0.15	$0.60	4.0×
ANTHROPIC	Claude Sonnet 4	200K	$3.00	$15.00	5.0×
ANTHROPIC	Claude Opus 4	200K	$15.00	$75.00	5.0×
DEEPSEEK	DeepSeek V3.2	164K	$0.27	$0.42	1.6×
META	Llama 4 Maverick	1.05M	$0.27	$0.85	3.1×

Why output costs more — input tokens are processed in parallel, but output tokens are generated one at a time. Long, verbose responses balloon costs even when prompts are small.

WORKED EXAMPLE · SUPPORT TICKET AUTOMATION

A single call is pennies. *Ten thousand tickets* is a different conversation.

Workload · 3,150 input tokens (system prompt + retrieved docs + user message) · 400 output tokens · per ticket



PER TICKET — MINI

$0.0007

Less than a tenth of a cent.

PER TICKET — 5.2 PRO

$0.1334

Roughly 190× more than Mini.

DAILY SPREAD · 10K TICKETS

$7 vs $1,333

Same workload, same task, two orders of magnitude apart.

STRATEGY · MODEL MIXING AT THE ORCHESTRATION LAYER

Send *most traffic* to the cheap model. Escalate only when you must.

Baseline · 1,000,000 requests · 2,000 input + 400 output tokens each

ALL-IN	BALANCED	AGGRESSIVE	EDGE-CASE
100% Claude Sonnet 4	60% GPT-4o Mini 40% Sonnet 4	90% GPT-4o Mini 10% Sonnet 4	99% GPT-4o Mini 1% GPT-5.2 Pro
COST PER 1M REQUESTS	COST PER 1M REQUESTS	COST PER 1M REQUESTS	COST PER 1M REQUESTS
$12,000	$5,124	$1,686	$1,627
Baseline	−57% savings	−86% savings	−98.5% savings

Layered with batching & caching — async/batch endpoints run at roughly half price, and caching static system prompts removes repeat tokenization. Combined, these levers cut total inference cost 5–10×.

IT ISN'T TRADITIONAL SOFTWARE TESTING

Probabilistic systems don't have *one right answer* — and the test set can never cover every input.

01

Non-deterministic behavior

Identical inputs can produce different outputs. You can't assert equality — you assert that a range of responses fits expectations.

"Write me a welcome email." → ten valid replies, all different.

02

No single correct answer

Open-ended tasks have many valid responses. Exact-match scoring fails; you need semantic similarity, rubrics, or LLM judges instead.

summary_a ≠ summary_b but both convey the same meaning.

03

Unbounded input space

Real users push the system beyond your test set — with typos, edge cases, new topics, and adversarial prompts you didn't plan for.

Plus new risk classes: hallucinations, jailbreaks, data leaks.

EVALS AT EVERY STAGE

A single product needs *five evaluation workflows* — each building on the one before.

STAGE 01

Comparative experiments

Pick the model, the prompt, the chunking strategy. Measure each change against a curated test set.

STAGE 02

Stress-testing & red-teaming

Expand coverage to edge cases and adversarial inputs before launch. Probe for failure modes.

STAGE 03

Guardrails

Real-time checks on inputs and outputs in production. Block or repair problematic responses on the fly.

STAGE 04

Observability

Trace live interactions and score them continuously. Spot drift, surface new topics, learn from real users.

STAGE 05

Regression testing

When you ship a fix, re-run the suite. Confirm the change improves quality without breaking what worked.

BEFORE LAUNCH · OFFLINE EVALUATION

AFTER LAUNCH · ONLINE EVALUATION

Bottom line — evals are not a single metric. They are a system: a test dataset that grows with the product, automated scoring that scales human judgment, and the discipline to run both at every stage of the lifecycle.