LLM Evaluation — Cheat Sheet

Why, how, and with what metrics LLM apps & foundation models are evaluated before, during, and after production.

1 · CORE IDEA

LLM Evaluation = the systematic, repeatable measurement of an LLM/app's quality, correctness, reliability, safety, and production-readiness — using datasets, rubrics, and metrics instead of guesswork.

Build → Evaluate → Fix → Deploy → Monitor (continuous loop)

Vibe Testing

Judging quality from a handful of manual prompts + gut feel. No metrics, not reproducible, misses edge cases. OK for toy projects; unsafe at production scale.

2 · WHY IT MATTERS: CASE STUDIES

Air Canada chatbot
Hallucinated bereavement-fare policy → customer sued → court ruled the company liable for its own chatbot's words.

Chevrolet chatbot
Jailbroken via prompt tricks into "agreeing" to sell a car for $1 — no guardrails against manipulation.

Lawyer + ChatGPT
Cited hallucinated, fake case law in a real court filing → sanctioned for not verifying AI output.

Lesson: hallucination + jailbreak + zero evaluation = legal, financial, and reputational risk.

3 · COMMON FAILURE TYPES

- **Hallucination** — confidently wrong/fabricated info
- **Jailbreak** — bypassing safety rules via crafted prompts
- **Prompt Injection** — hidden instructions in input/content hijack behavior
- **Prompt Leakage** — system prompt/internal instructions exposed
- **Toxic Output** — offensive, biased, unsafe content
- **Privacy Leakage** — PII or training data exposed

4 · DETERMINISTIC VS PROBABILISTIC

Traditional Software	LLM Applications
Same input → same output	Same input → variable output (sampling/temperature)
Exact pass/fail assertions	No single "correct" string — needs judgment-based scoring
Fixed test suites	Statistical, distribution-aware evaluation

5 · MODEL EVAL VS APPLICATION EVAL

Model Evaluation

Tests the raw foundation model via public benchmarks (MMLU, GSM8K, HumanEval…). Done by frontier labs, third parties, and teams comparing models.

Application Evaluation

Tests the system built *on top* (chatbot, RAG, agent) for a specific use case: task success, safety, latency, cost.

6 · EVALUATION DIMENSIONS (15)

Correctness · Factuality · Groundedness · Faithfulness · Relevance · Completeness · Conciseness · Fluency · Tone · Safety · Bias · Robustness · Latency · Cost · User Satisfaction

Groundedness = supported by retrieved context. **Faithfulness** = doesn't contradict the source. **Factuality** = objectively true regardless of source.

7 · THREE EVALUATION METHODS

Programmatic (deterministic)

Exact-match, regex, Recall@K / Precision@K. Fast & cheap; only works for verifiable outputs.

Human Evaluation

Direct rating, red-teaming, A/B testing, golden-dataset creation, human-in-the-loop. Gold standard, but slow/expensive — use a rubric + multiple raters.

LLM-as-a-Judge

A strong LLM scores outputs vs a rubric. Scalable for open-ended text; the judge itself must be validated (MAE vs human scores) — cheaper than humans, can drift or be biased.

Reference-based needs a gold answer
Reference-free judges quality without one

8 · APPLICATION-SPECIFIC EVALUATION

- **RAG** — Retriever: Recall@K, Precision@K, MRR. Generator: faithfulness, groundedness, answer relevance.
- **Agents** — task completion, tool-call correctness, step efficiency, planning quality
- **Safety** — jailbreak resistance, red-teaming, toxicity, PII leakage
- **Operations** — latency, cost, uptime, logging/tracing, drift detection

9 · 12-STEP APPLICATION EVAL WORKFLOW

Define task & target → Define success criteria → Build golden dataset → Choose eval method → Run model → Evaluate results → Analyze results → Improve system → Repeat evaluation → Deploy → Monitor → Update golden dataset ↺

10 · OFFLINE VS ONLINE EVALUATION

Offline — pre-deployment, golden dataset. Used for release gating, version comparison, regression testing. Misses: unanticipated inputs, emergent failures, drift.

Online — post-deployment on real traffic. Captured signals (/ , retries) + computed signals (latency, refusal rate). Goal: detect *abnormality*, not just incorrectness.

11 · WHY MULTIPLE EVAL PIPELINES PER APP

Reason 1 — Multiple failure points: retriever ≠ generator ≠ workflow ≠ whole app; each layer needs its own eval.

Reason 2 — Multiple risk categories: Quality, Safety, and Operations each need a dedicated pipeline.

12 · MODEL EVAL: 8 CORE CAPABILITIES

Knowledge & Reasoning · Coding & Software Engineering · Mathematics · Long Context · Vision & Multimodal · Agentic & Tool Use · Safety & Alignment · Instruction Following

13 · ANATOMY OF A BENCHMARK

- **Dataset** = input + gold answer
- **Run Config** = zero/few-shot, CoT vs direct, temperature, pass@k, tools on/off
- **Scoring** = extract answer → compare (exact-match or judge/rubric)
- **Aggregation** = mean, macro vs micro average, weighted by category

14 · WHO PERFORMS MODEL EVALUATION

- **Frontier Labs** — self-report at model release (read cautiously — marketing incentive)
- **Third-Party Evaluators** — independent, standardized, comparable across labs
- **Companies / Eng Teams** — custom evals for their exact use case (build vs buy)

15 · KNOWLEDGE & REASONING BENCHMARKS

Benchmark	Year	Focus	Status
MMLU	2020	57-subject general knowledge, MCQ	Saturated
TruthfulQA	2021	Truthfulness vs misconceptions	Saturated
AGIEval	2023	Real exams (SAT/LSAT/Gaokao)	Saturated
GPQA	2023	Grad-level "Google-proof" science	Near-saturated
MMLU-Pro	2024	Harder MMLU, 10-option, CoT-heavy	Active
SimpleQA	2024	Closed-book recall + calibration	Active
HLE	2025	100+ fields, expert-written, ~2.5k Qs	Active

16 · COMMON BENCHMARK PROBLEMS

- **Contamination** — test data leaks into training data
- **Saturation** — models max out, benchmark stops discriminating
- **Configuration Gaming** — cherry-picked settings inflate scores
- **Aggregation Bias** — one average hides per-category weaknesses

17 · KEY TAKEAWAY

Building an LLM application is only half the job. Evaluating, securing, monitoring, and continuously improving it is the other half — that's what separates a prototype from a production AI system.