Tech Jargon

Part 6

for

Product Managers

Building with LLMs

Prompts | Tokens | RAG | Agents | Evals | Cost

From an Engineer-Turned-PM

The LLM Stack Mental Model

A way of understanding how **large language models work in production** — so you can make smart decisions about prompts, architecture, cost, and quality.

It helps PMs move from "can we use AI?" to "how do we build this well?" Get this wrong, and you ship AI that's slow, expensive, or impossible to improve. Get it right, and you build AI products that are cost-effective and systematically getting better.

When you understand this mental model, you can:

- Choose between RAG and fine-tuning based on data, timeline, and quality — not just engineering preference
- Reason about token costs, context windows, and latency to make real product tradeoffs
- Define evals and quality metrics so you ship with evidence, not vibes
- Scope agentic features realistically — knowing what needs orchestration, error handling, and human checkpoints

This guide covers the 10 essential terms that bring this mental model to life.

LLM / Foundation model

The pre-trained models you build on top of

A large language model (LLM) is an AI system trained on massive amounts of text that can understand and generate human language. A foundation model is the broader category — any large, pre-trained model designed to be adapted for specific tasks. You don't build these from scratch — you build on top of them.

Think of it like building an app on top of iOS rather than writing your own operating system.

We're evaluating Claude vs. GPT-4 as our foundation model — the cost-per-token and context window differences matter for our use case.

For PMs, this means our first decision isn't "build AI" — it's "which model, and why?" Different models have different strengths, costs, and limitations — and those tradeoffs shape what's even possible.

Prompt engineering

Shaping AI behavior through instructions

Prompt engineering is the practice of crafting the instructions, examples, and context you send to an LLM to get the output you want. Small changes in wording or structure can dramatically shift output quality.

Think of it like briefing a new contractor: the more specific your brief — examples, constraints, context — the better the output. A vague brief gets vague work.

The output quality jumped once we added few-shot examples to the prompt. We went from 60% accuracy to 85% without touching the model

For PMs, this means prompts are a product lever you can influence. When output quality isn't right, the fix is often in the prompt — not a model swap or a retraining cycle.

Tokens / Context window

The capacity constraint that affects cost and capability

Tokens are the units LLMs process — roughly ¾ of a word in English. Every input and output is measured in tokens, and you pay per token. The context window defines how many tokens the model can take as input, with a separate, smaller limit on output.

Think of it as a desk that can only hold so many pages at once. Your prompt, conversation history, and documents all compete for desk space — once it's full, something has to go.

We're hitting the context window limit. The conversation history plus the system prompt doesn't leave enough room for a useful response.

For PMs, this means context windows constrain what your product can do in a single request. You need to know your model's limit and design the UX around it — what gets included, what gets summarized, and what gets cut.

RAG

Retrieval-Augmented Generation

RAG is a pattern where you retrieve relevant information from your own data — documents, databases, knowledge bases — and inject it into the prompt before the model generates a response. Instead of retraining the model to "know" your data, you feed it the right context at query time.

Think of it like handing someone the relevant page of a manual before asking them a question, rather than making them memorize the entire book.

We're using RAG against the help center docs — the model just needs the right articles in context at query time.

For PMs, this means RAG is often the fastest, cheapest way to make an LLM "know" your product's data. But quality depends on retrieval — if the system pulls the wrong documents, the model gives wrong answers.

Fine-tuning

Customizing the model itself for your domain

Fine-tuning is training an existing foundation model on your own data — adjusting its internal weights so it performs better on your specific task, tone, or domain.

RAG is like giving someone a cheat sheet before an exam. Fine-tuning is like coaching them over weeks — their instincts and approach shift permanently.

Prompt engineering got us to 85%, but we need to fine-tune to hit the accuracy target. We'll need a few hundred high-quality labeled examples.

For PMs, this means fine-tuning is a bigger investment — labeled data, training runs, and ongoing evaluation. Always ask: "Can we get there with better prompts or RAG first?" before committing to fine-tuning

Agent / Agentic AI

AI that takes actions, not just generates text

An agent is an AI system that doesn't just generate a response — it decides what steps to take, calls tools or APIs, and chains actions to complete a goal.

A chatbot gives you instructions: "here's how to cancel your subscription." An agent actually cancels it for you — reading the account, clicking the buttons, confirming the change.

The agent reads the ticket, pulls customer history, drafts a response, and routes to the right team — if any step fails, it retries or escalates.

For PMs, this means agents are powerful but harder to control. Every step can fail or take an unexpected path. Define what the agent can do autonomously, what requires human approval, and scope small.

Evals

How you measure if the AI is actually good

Evals (evaluations) are the tests and benchmarks you use to measure an LLM's output quality — accuracy, relevance, tone, safety. Without evals, you're shipping on vibes.

Think of it as grading essays instead of checking math — you need rubrics, not just answer keys.

We need to build an eval set before we swap models. Otherwise we won't know if the new one is actually better or just different.

For PMs, this means you define what "good" looks like — criteria, test cases, edge cases — and engineers build the system to measure it. Insist on evals before any model or prompt change ships.

AI cost / Token economics

Why every AI call costs money

Every time your product calls an LLM, you're billed by the token — input tokens (what you send) and output tokens (what the model returns). Unlike traditional software, AI costs scale linearly with usage.

Think of it like paying postage per letter — every message costs money, and heavier letters cost more. Traditional software is more like a flat-rate mailbox.

Each query costs anywhere from $0.001 to $0.10 depending on the model — at 100K queries a day, that adds up to thousands daily.

For PMs, this means AI features have a per-use cost that traditional features don't. Understand cost per query, how it scales, and where to optimize. Token economics should be in your business case, not an afterthought.

Grounding

Connecting outputs to verified data

Grounding is tying an LLM's output to verified, traceable sources — so the response is based on real data, not just patterns learned during training. A grounded answer cites a source. An ungrounded answer might be a confident hallucination.

Think of it like a journalist's rule: don't publish a claim without a source. An ungrounded model writes from memory; a grounded model writes with footnotes.

The summary looks good, but it's not grounded — we need the model to reference specific sources so users can verify.

For PMs, this means Grounding is how you build trust. If your product makes claims or answers questions, design for citations, source links, and clear signals when the
model is generating beyond the data.

Multimodal

Models that handle text, images, audio, and video

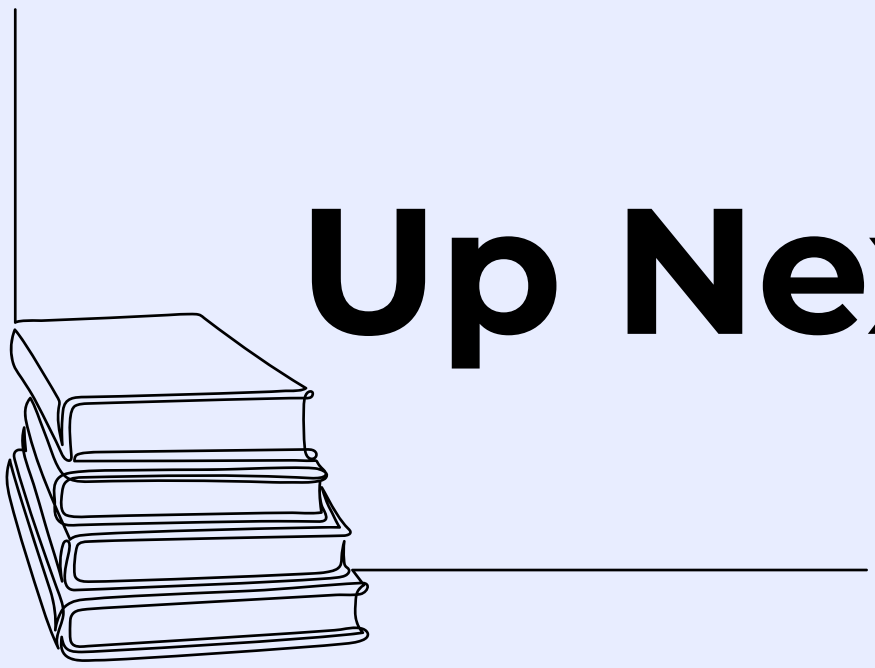
Multimodal means the model can process more than just text — images, audio, video, or a combination. A multimodal model can "see" an image you upload, transcribe a voice note, or generate a diagram from a description.

Think of it like hiring someone fluent in multiple languages versus just one — each extra language adds capability, but also cost and complexity.

The model supports vision now, so we could let users upload a photo of the error screen instead of making them describe it.

For PMs, this means multimodal opens new interaction patterns — but each modality adds cost and latency. Ask whether the multimodal path is genuinely better for the user or just a cool demo.

Stay Tuned



Up Next

A plain-English guide to metrics, funnels, and experiments — the terms that turn product instinct into product evidence. Because launching without measurement is just hoping.