BAADEN Scientific

AI: What You Need to Know A Map for the Overwhelmed

A Map for the Overwhelmed

Marc Baaden · Research Director, CNRS · Founder, Baaden Scientific · 2026

Contents

1 Introduction 3
2 Part 1 — The Map 3
2.1 What Are We Actually Talking About? 3
2.2 Five Things Everyone Must Know 3
2.3 Privacy: Does Your Data Leave Your Computer? 4
2.4 Beyond Text: Image, Audio, and Video 4
2.5 Avatars, AI Assistants, and AI Workspaces 4
2.6 The Ladder: Five Levels of AI Use 5
3 Part 2 — Going Deeper 8
3.1 Prompt Engineering and Building a Prompt Library 8
3.2 RAG, Vector Databases, and Local LLMs 8
3.3 The Invisible Glue: APIs, Fine-Tuning, and Model Size 9
3.4 Building Real Automations: The WAT Framework in Practice 9
3.5 Trust, Safety, and Where This Is Going 9

1 Introduction

Everyone around you seems to be talking about AI. New tools launch weekly, the jargon shifts monthly, and every headline sounds like it was written to make you feel behind. If you're overwhelmed, that's not a personal failing — it's rational. The field is moving faster than any single person can track, and most introductions assume you already know half the vocabulary.

This document is not a manual. It's a map. Think of it as the kind of overview a well-informed friend would give you over coffee: enough to orient yourself, spot what matters to *you*, and feel confident enough to explore further. We'll walk through five levels of AI use — from simply chatting with a bot to orchestrating teams of autonomous agents — and along the way we'll cover the terms, tools, and trade-offs that actually matter.

Whether you're a curious beginner, a knowledge worker wondering how this changes your job, or a semi-technical person trying to make sense of the landscape, this is for you.

———

2 Part 1 — The Map

2.1 What Are We Actually Talking About?

When people say "AI" today, they almost always mean **generative AI**: systems that create new text, images, audio, or video rather than just classifying or sorting existing data. At the heart of most of these systems sits a **large language model** (LLM) — a program trained on enormous amounts of text that has learned to predict, and therefore produce, language.

The main players right now are OpenAI (GPT-4o, o3), Anthropic (Claude), Google (Gemini), Meta (Llama), and Mistral. Some of these are *closed* models — you use them through a company's website or app, your data goes to their servers, and you don't see the internals. Others are *open* (or open-weight) models — you can download them, inspect them, and run them on your own hardware.

This closed-vs-open distinction matters enormously for privacy and cost, and we'll come back to it. For now, just know that the landscape is not one company or one tool; it's an ecosystem with genuinely different philosophies.

One more thing: AI is no longer just about text. **Multimodality** means these systems can now work with images, audio, video, and code — sometimes all at once. You can show Claude a photo of a handwritten note and ask it to transcribe and translate it. You can ask Gemini to describe a video. The text-only era is already behind us.

2.2 Five Things Everyone Must Know

Before we go any further, five concepts form the absolute foundation. Skip these and nothing else will make sense.

Hallucinations. AI models sometimes produce statements that sound confident but are entirely false. This is not a bug that will be patched next quarter — it's structural, a consequence of how these models work. They predict plausible text, not verified truth. Always verify anything that matters.

Tokens. LLMs don't read words; they read *tokens* — chunks of text that are roughly three-quarters of a word in English. Tokens are the unit of everything: they determine how much a query costs, how long your input can be, and how models are compared. When someone says a model has a "200K context window," they mean 200,000 tokens.

Context window. This is the AI's working memory — everything it can "hold in mind" during a single conversation. Once the context window fills up, older parts of the conversation effectively vanish. Bigger context windows are better, but they also cost more.

Knowledge cutoff. Every model was trained on data up to a certain date. It genuinely does not know what happened after that. Some tools compensate by searching the web in real time, but the base model itself has a hard boundary.

Cost. Chatting casually is cheap or free. But as you move up the levels we'll discuss — giving AI tools, letting it write code, granting it computer access — costs can climb fast. A complex agentic coding session can easily run several dollars. Knowing where the money goes helps you make good decisions.

2.3 Privacy: Does Your Data Leave Your Computer?

This might be the single most important question to ask about any AI tool: *where does my data go?*

When you type something into ChatGPT, Claude.ai, or Gemini, your text travels to the company's servers, gets processed, and the answer comes back. That means a copy of your input exists, at least temporarily, on someone else's infrastructure. For casual questions this is fine. For confidential work documents, medical information, or proprietary code, it might not be.

The good news: you have options. An increasing number of tools let you run AI *locally*, meaning everything stays on your machine. Apple Intelligence processes many tasks directly on your iPhone or MacBook without sending data anywhere. Ollama lets you download and run open models on your own computer — completely offline if you want. LM Studio provides a graphical interface for the same purpose. We'll discuss these in more depth in Part 2.

Here's a simple decision framework you can use right now. Before using any AI tool, ask yourself three questions: What data am I sharing? Who can see it (the company, their partners, anyone)? And do I have the option to keep it local? If the answer to the third question is yes and the data is sensitive, use the local option.

2.4 Beyond Text: Image, Audio, and Video

AI's creative abilities now extend well beyond text, and this is one of the areas moving fastest.

Image generation has matured remarkably. Google's Nano Banana Pro (built on Gemini 3 Pro) is currently the standout — it produces high-fidelity images with accurate text rendering in multiple languages, supports up to 4K resolution, and can reason about what it's creating. Other strong players include Midjourney (known for artistic quality), Flux by Black Forest Labs (open-source and gaining traction fast), Ideogram (excellent at text in images), and Adobe Firefly (integrated into Creative Cloud). The quality gap between AI-generated and professional photography is narrowing fast.

Audio has two sides. On the input side, transcription tools like OpenAI's Whisper can convert speech to text with remarkable accuracy across dozens of languages — this is genuinely production-ready and already embedded in countless apps. On the output side, voice cloning and text-to-speech have reached the point where a few seconds of sample audio can produce a convincing replica of someone's voice. This is powerful (think accessibility, content creation, translation) and raises obvious ethical questions.

Video generation is the current frontier. Tools like Sora (OpenAI), Runway, and Kling can generate short video clips from text prompts, but the results are still inconsistent, the computing costs are very high, and generation times are slow. This is impressive technology that is not yet practical for most people — but it's worth watching closely because progress is rapid.

2.5 Avatars, AI Assistants, and AI Workspaces

You may have seen videos of eerily realistic AI-generated presenters — a person on screen who looks and sounds human but is entirely synthetic. These are **AI avatars**, and tools like HeyGen are making them accessible to anyone. You upload a script, choose (or clone) a face and voice, and get a polished video in minutes.

Here's the thing: avatars aren't really a new category of AI. They're a combination of the technologies we've already discussed — language models for the script, voice cloning for the audio, image generation for the face, and video synthesis to animate it all. Recognizing this helps you see through the marketing: an "AI assistant" with an avatar is fundamentally the same technology as a chatbot, just with a visual layer on top.

AI workspaces are a different story and deserve attention. Google's AI Studio gives you direct access to Gemini models with fine-grained controls — it's where developers and power users experiment. NotebookLM takes a different approach: you feed it your own documents and it becomes an expert on *your* material, capable of generating podcast-style audio summaries, answering questions, and finding connections you might have missed. These workspace tools are powerful because they combine the model's intelligence with *your* specific knowledge, and they're becoming increasingly accessible to non-technical users.

2.6 The Ladder: Five Levels of AI Use

LEVEL 0: JUST TALK TO IT

The simplest way to use AI: open a chat interface and type. Claude.ai, ChatGPT, Gemini, Mistral's Le Chat — they all work this way. No setup, no configuration, no technical knowledge required.

Think of it as talking to a very well-read friend who has an extraordinary memory for facts but occasionally makes things up with complete confidence. You can ask questions, brainstorm, draft emails, summarize long texts, translate, explain complex topics in simple terms, or just think out loud with a patient interlocutor.

The limits are real, though. The AI has no memory between conversations (unless the tool explicitly offers it). It can't access the internet (unless web search is enabled). And it can only work with what you give it in the conversation window.

> **i TIP**
>
> Add to the end of your prompt: "*Ask me any clarifying questions and iterate until you are 95% sure.*" This one sentence transforms the interaction. Instead of guessing, the AI will interview you, surface ambiguities, and produce something far closer to what you actually need.

> **i TIP**
>
> When the output isn't quite right, don't start over — *correct it within the conversation*. Say "That's too formal, rewrite it like you're texting a friend." The conversation *is* the context.

LEVEL 1: MAKE IT YOURS — SYSTEM PROMPTS AND KNOWLEDGE BASES

Level 0 is a generic assistant. Level 1 is *your* assistant.

A **system prompt** is a set of instructions you give the AI before the conversation starts — a persona, a set of rules, a tone. "You are a senior patent attorney specializing in biotech. Always cite relevant case law. Write in plain English." The AI will follow these instructions throughout the conversation, producing dramatically more relevant results than a generic chat.

RAG (retrieval-augmented generation) takes this further. Instead of relying solely on the AI's training data, you give it access to your own documents — company policies, research papers, product specs, meeting notes. When you ask a question, the system retrieves the most relevant passages from your documents and feeds them to the AI as context. The AI can now answer questions *about your specific material* rather than about the world in general. Tools like Claude Projects and custom GPTs make this accessible without any coding.

> **i TIP**
>
> Give the AI a role before you give it a task. “You are a senior tax advisor” before “Help me with my return” produces dramatically better results than the task alone.

> **i TIP**
>
> Copy-paste a sample of your own writing and say “Match my style.” This instantly solves the “it doesn’t sound like me” problem.

LEVEL 1.5: GIVING AI HANDS — TOOLS AND EXTENSIONS

Up to this point, AI is purely reactive — it reads your input and produces text. At Level 1.5, it starts *doing things*.

MCP servers (Model Context Protocol) are connectors that let AI reach out to external systems — your calendar, your files, a database, a web API. Think of them as USB ports for the AI: each one adds a new capability. Claude can use MCP to read your Google Drive, query a database, or interact with a development tool. This is one of the key developments making AI genuinely useful in day-to-day work.

Skill files (like SKILL.md) are reusable instruction sets — a recipe the AI follows for a specific type of task. You write it once, and the AI can execute it consistently every time.

Prompt libraries are your personal collection of tested, refined prompts. If you find yourself doing the same type of task repeatedly (summarizing meeting notes, writing product descriptions, reviewing contracts), saving your best prompts is one of the most underrated productivity tools available.

This is also the level where **integration becomes a superpower**. AI’s ability to connect to existing tools through APIs and plugins means it can pull data from your CRM, post to your project board, check your inbox, and file documents — often in a single workflow. Tools like n8n, Make, and Zapier make these integrations accessible to non-programmers by providing visual, drag-and-drop workflow builders with AI steps built in.

And then there are the **agentic browsers** — a new category that deserves attention. Fellou (with over a million users) calls itself the first agentic AI browser: it can browse, research, fill forms, and execute multi-step web tasks autonomously while you do other things. Others in this space include Dia (by The Browser Company), Comet, and Atlas. These sit at the boundary between “tool” and “agent” — the AI isn’t just answering questions; it’s navigating the web on your behalf.

LEVEL 2: AI THAT WRITES AND RUNS CODE

This is where the paradigm shifts. At Level 2, AI doesn’t just suggest code — it writes it, executes it, sees the errors, fixes them, and iterates until the task is done.

The flagship tools are Claude Code (Anthropic’s command-line coding agent), Cursor (an AI-native code editor), and GitHub Copilot (Microsoft’s inline coding assistant). But the concept is broader than any single tool: it’s the idea that natural language becomes a programming language.

Andrej Karpathy coined the term **vibe coding** — the practice of describing what you want in plain English and letting the AI figure out the implementation. This is the entry point: just talking about what you want built. But there’s a spectrum beyond that. At the next stage, you’re running iterative loops where the AI writes code, tests it, reads the errors, and tries again automatically. Further still, you get *self-healing* systems — code that detects its own failures and repairs them without human intervention.

You don’t need to be a programmer to use these tools. Many people with zero coding background are building functional applications, automations, and websites by describing what they want in natural language. The AI handles the syntax; you handle the intent.

> **i TIP**

> Break big tasks into steps yourself. "First, outline the pages. Then design the homepage. Then add login." The AI performs dramatically better on well-scoped tasks than on sprawling ones.

Influential voices in this space include Andrej Karpathy (who coined the vocabulary and demonstrated the practice), Nate Herk (who teaches Claude Code workflows and the WAT framework to non-technical audiences), and Sabrina Ramonov (who focuses on practical AI automation for businesses).

LEVEL 3: THE WIDE-OPEN ZONE

At this level, AI gets the keys to your computer. Not just code execution, but full access: it can see your screen, move your mouse, open applications, browse the web, read and write files, and take actions on your behalf.

This is powerful and genuinely useful — but it's also the level where risk, privacy, and trust become critical questions.

OpenClaw (by Peter Steinberger) is an open-source personal AI assistant that went viral. It runs on your own machine, is accessible through WhatsApp, Telegram, or iMessage, has persistent memory, can control your browser, access your file system, and even build its own new skills. If self-hosted, your data stays with you. **Claude Computer Use** lets Claude see your screen and operate your computer like a human would. **Claude Cowork** provides a desktop environment where Claude can work with your files, run code, and use tools autonomously while you supervise. **Perplexity** offers AI-powered web research with autonomous browsing.

TOOL	PRIVACY RISK	AUTONOMY GRANTED	EASE OF SETUP
Claude Computer Use	Medium	High	Easy
Claude Cowork	Medium	Medium	Easy
OpenClaw (self-hosted)	Low	Very High	Medium
OpenClaw (cloud)	Medium	Very High	Medium
Perplexity	Medium	Low	Very Easy

This is also where structured frameworks become essential. The **WAT framework** — Workflows, Agents, and Tools — provides a three-layer architecture for organizing complex AI projects, particularly in Claude Code. *Workflows* are step-by-step process descriptions. *Agents* are the AI reasoning engines. *Tools* are the specific capabilities agents invoke. Separating these three layers makes complex automations dramatically easier to build, debug, and extend.

LEVEL 4: TEAMS OF AGENTS

The final level — and still the most experimental. Instead of one AI agent working alone, multiple agents collaborate: one plans, another researches, a third writes code, a fourth reviews the output. They delegate tasks to each other, share context, and work in parallel.

Frameworks like Microsoft's AutoGen and multi-agent Claude pipelines make this possible today, but we're still in early territory. The orchestration is complex, error propagation between agents is a real problem, and costs can escalate quickly when multiple agents are each consuming tokens.

Why does it matter? Because the most ambitious AI applications — running a complete software development cycle, managing a research pipeline, coordinating a business workflow end-to-end — require more than one agent's context window and skill set. Teams of agents are how AI scales beyond the capabilities of any individual model.

For now, this is mostly relevant to developers and advanced power users. But it's the direction things are heading, and understanding that it exists helps you see where the "just chatting" paradigm eventually leads.

———

3 Part 2 — Going Deeper

3.1 Prompt Engineering and Building a Prompt Library

Prompt engineering sounds technical but it's really a craft — the art of communicating clearly with an AI so it produces what you actually want. The core techniques are simpler than most guides make them seem.

Role and context. Tell the AI who it is and what situation it's in. "You are an experienced grant reviewer for the European Research Council. I'm going to show you a proposal summary and I want you to identify the three weakest points."

Format and constraints. Be explicit about what the output should look like. "Give me a bulleted list with exactly five items, each under 20 words." The more specific you are about format, the less editing you'll do afterward.

Examples. Show, don't just tell. If you want the AI to write product descriptions in a specific style, paste one example and say "Follow this pattern." One concrete example is worth fifty words of abstract instruction.

> **i TIP**
>
> Say what you want, not what you don't want. "Write in short, direct sentences" produces better results than "Don't write long, complex paragraphs."

Chain of thought. For complex reasoning tasks, ask the AI to "think step by step" or "show your reasoning before giving the answer." This dramatically improves accuracy on math, logic, and analysis tasks.

A **prompt library** is simply a collection of your best prompts, saved somewhere accessible. Every time you craft a prompt that produces great results, save it. Over time, this library becomes one of your most valuable productivity assets. Organize it by task type (writing, analysis, coding, brainstorming) and refine entries as you learn what works.

3.2 RAG, Vector Databases, and Local LLMs

In Part 1, we introduced RAG — retrieval-augmented generation — as the idea of giving AI access to your own documents. Let's go a level deeper.

When you upload documents to a RAG system, the text gets broken into chunks and converted into mathematical representations called *embeddings* — lists of numbers that capture the meaning of each passage. These embeddings are stored in a **vector database** (Pinecone, Chroma, Weaviate, and others). When you ask a question, your question is also converted into an embedding, and the system finds the document chunks whose embeddings are closest in meaning to your question. Those chunks are fed to the AI as context.

You don't need to understand the math. What matters is the principle: RAG lets the AI work with *your* knowledge, not just its training data. This is how companies build internal AI assistants that know about their specific products, policies, and processes.

Local LLMs are the other major theme. Ollama lets you download and run open-source models (Llama, Mistral, Phi, Qwen, and others) entirely on your own machine. No internet connection required, no data leaving your computer, no per-token costs. LM Studio provides a user-friendly graphical interface for the same purpose. The trade-off is that local models are generally less capable than the largest cloud models (you're running a 7B or 13B parameter model on a laptop versus a massive model on a datacenter), but for many tasks — summarization, translation, first drafts, data extraction — they're more than good enough. And the privacy guarantee is absolute.

3.3 The Invisible Glue: APIs, Fine-Tuning, and Model Size

Three concepts you'll encounter constantly as you go deeper.

APIs (Application Programming Interfaces) are how software talks to software. When you use Claude through a chat interface, there's an API behind the scenes — your browser sends your message to Anthropic's servers, the model processes it, and the answer comes back. APIs are the reason AI can be embedded in virtually any tool: your email client, your spreadsheet, your CRM, your website. Every integration you've seen — AI in Slack, AI in Notion, AI in your phone — runs through an API.

Fine-tuning is often confused with RAG, but it's fundamentally different. RAG gives the model temporary access to your documents at query time. Fine-tuning permanently modifies the model itself by training it further on your specific data. Think of RAG as giving the AI a reference book to consult; fine-tuning is like sending it back to school. Fine-tuning is more powerful for specialized tasks but much more expensive, technically demanding, and hard to update. For most users, RAG is the right starting point.

Model size and parameters. When someone says a model has "70 billion parameters" (70B), they mean the model contains 70 billion adjustable numerical values that were learned during training. More parameters generally means more capability — but also more computational cost and slower responses. The industry is getting better at making smaller models perform well (Google's Gemini Flash, Anthropic's Haiku, Microsoft's Phi), which is why you can now run useful models on a laptop. Bigger is not always better; it depends on the task.

3.4 Building Real Automations: The WAT Framework in Practice

In the Ladder section, we introduced the WAT framework — Workflows, Agents, and Tools — as a way to structure complex AI projects. Let's see what this looks like in practice.

Imagine you want to build an automation that monitors your inbox for customer complaints, classifies them by severity, drafts a response, and logs the issue in your project tracker. Without structure, you'd write one giant prompt and hope for the best. With WAT, you separate concerns.

The **Workflow** is a markdown file that describes the process: "Step 1: Check inbox for new messages tagged 'complaint.' Step 2: Classify severity (low/medium/high). Step 3: Draft response using the customer service template. Step 4: Log to project tracker with severity and summary. Step 5: If high severity, also notify the team lead."

The **Agent** is Claude Code (or another AI reasoning engine) that reads this workflow, makes decisions at each step (Is this actually a complaint? What severity?), and calls the appropriate tools.

The **Tools** are the specific capabilities: an email reader (MCP server), a classification prompt, a response template, a project tracker API, a notification service.

This separation matters because when something breaks — and it will — you know exactly where to look. Is the workflow logic wrong? Is the agent misclassifying? Is a tool failing? Each layer can be tested and fixed independently.

Nate Herk teaches this approach to non-technical audiences, showing how Claude Code can be used to build sophisticated automations without traditional programming knowledge. Sabrina Ramonov takes a complementary approach, focusing on practical business automations using n8n and similar platforms, making the case that AI automation is not just for developers.

The key insight: complex AI automations are not magic. They're just well-organized systems with clear responsibilities at each layer.

3.5 Trust, Safety, and Where This Is Going

We've covered the map. Now let's talk about the territory's dangers and its horizon.

Hallucination mitigation in practice means building habits, not waiting for a technical fix. Cross-reference important claims. Ask the AI to cite its sources (and then check them). Use RAG to ground answers in

your own verified documents. When accuracy is critical, treat AI output as a first draft that needs human review, not as a finished product.

Prompt injection becomes relevant the moment AI has tools. If an AI agent can read your email and execute actions, a malicious email could theoretically contain hidden instructions that manipulate the agent. This is not science fiction; it's an active area of security research. The practical implication: the more autonomy you grant an AI, the more carefully you should evaluate what inputs it processes.

When to trust AI output. A useful rule of thumb: trust is inversely proportional to the stakes. For brainstorming, first drafts, and low-stakes tasks, trust freely and iterate. For anything with real consequences — financial decisions, medical information, legal documents, public statements — verify independently. AI is an extraordinary tool for thinking faster; it is not (yet) a substitute for judgment.

Where is this going? Rather than making grand predictions, two trends seem clear. First, AI is getting embedded in everything — every app, every service, every device — which means the skills you're building by reading this document will be relevant far beyond any single tool. Second, the balance between capability and control will remain the central tension. More capable AI is more useful and more risky. The five levels we've discussed are really five points on a single axis: how much power are you willing to delegate? The answer is personal, contextual, and evolving. This map is meant to help you decide for yourself.

— Marc Baaden, 2026

Marc Baaden
Research Director, CNRS · Founder, Baaden Scientific