POSTNIKOV.AI — SYSTEMS

CONTEXT ENGINEERING KIT

PLAYBOOK #1

How to talk to LLMs

From zero to a working system in one evening

MAX POSTNIKOV

postnikov.ai

Contents

Context Engineering Kit — Playbook 1

> How to Talk to LLMs

From zero to a working system in one evening

Author: Max Postnikov *Website:* postnikov.ai

Disclaimer

Prompt engineering is not enough.

I've been working with AI every day for a year — writing content, building products, running a community of 100+ professionals. Here's what I learned: *the prompt is 10% of the equation. The other 90% is context.* Knowing how to write prompts matters, but without context, even a perfect prompt gives average results. This playbook covers both.

This isn't about "write the magic word and AI produces a masterpiece." There are thousands of those guides. They don't work. You've probably already tried.

This playbook is about a system. Read it — implement it — it works. No magic. A clear, step-by-step plan.

This is the first playbook in a series. It covers the fundamentals: context, patterns, debugging. More coming on postnikov.ai.

Who This Is For

You already use AI. ChatGPT, Claude, Gemini — doesn't matter. But the results frustrate you.

AI responds:

> Too generic — like Wikipedia, only worse
> Off-topic — you're asking about business, it's talking about "let's consider all perspectives"

> Long and empty — three paragraphs of fluff instead of one answer

Sound familiar? You're in the right place.

There are hundreds of free prompting guides online. OpenAI and Anthropic publish their own. The difference: they teach you to use *their* tool. This playbook teaches you to use *any AI* — and builds a system that works regardless of which model is currently best.

What's Inside

1. Why AI doesn't understand you (spoiler: it's not the model or the prompt)
2. 4 levels of context — from "stranger" to "best friend"
3. How to create your context file in 10 minutes
4. 10 patterns that work — ready-to-use formulas
5. What to do when AI talks nonsense — a debugging checklist
6. Context drift — why AI gets dumber in long chats (and how to deal with it)
7. Projects — how to build an AI assistant that doesn't forget
8. 2 exercises — level up in one evening

Included — ready-to-use tools:

> 5 context file templates — open, fill in 5 fields, AI understands you
> AI Readiness Scorecard — find your level in 2 minutes
> 30-Minute Setup Checklist — set up your system in half an hour
> Pattern Cheat Sheet — one page, all formulas
> Meta-prompt — AI writes your context file for you
> Debug prompt — one command that fixes 80% of bad responses

How to Read

30-40 minutes cover to cover. But better — one chapter at a time, with practice. Each chapter ends with an action.

Most chapters include a "What's Happening Right Now" section: fresh trends worth knowing.

Let's go.

Chapter 1. Why AI Doesn't Understand You

I asked AI to help me write blog posts. I got a perfectly smooth, completely dead text. As if it was written by a robot from 2019 that had read every motivational LinkedIn post and decided *that's how humans talk*.

Then I made AI read my blog. Understand my style. Remember how I build sentences, which words I use, where I place periods.

Same AI. Same model. Night and day difference.

The difference? *Context.*

> The Problem

Imagine: you walk into an unfamiliar consultant's office. Sit down. And say:

> "Write me a strategy."

He starts talking. Confidently, smoothly, at length. SWOT analysis, segmentation, competitive advantages. Sounds right. But it's *not about you*. It's a generic strategy he could give to anyone.

This is exactly what you do with AI every day.

> The Formula

```
Response quality = f(Context quality)
```

Not "prompt quality." *Context.* The prompt is part of the context, but not the whole context.

> What Is Context

Context is everything AI knows about you and your problem when it starts responding:

- *Who you are* — role, experience, level
- *What you're doing* — project, task, deadline
- *Why* — goal, success criteria
- *How you want the answer* — format, length, style
- *What you've already tried* — so it doesn't repeat

Without this, AI is forced to *guess*. And it guesses toward the average. Average level, average request, average audience. That's why the answer sounds like Wikipedia.

> Example

Without context:

> "Write a LinkedIn post."

AI will write a generic motivational piece with emojis. Because that's what most people ask for.

With context:

> "I'm a tech entrepreneur, writing for an audience of 30-45 year old IT professionals. Tone: direct, no BS, with personal experience. Topic: why I stopped hiring juniors and replaced them with AI agents. Format: 150-200 words, hook in the first line."

Same AI. Same model. But the answer will be 10x more accurate, because you gave context.

> Anti-Advice

Here's what everyone recommends: *"Make your prompt as detailed as possible."*

Here's why it doesn't work: a detailed prompt without context is a long letter to a stranger. You can write three paragraphs of instructions, but if AI doesn't know *who you are* — it'll still produce an average result. Just a longer one.

Don't search for the right words. *Give context.*

→ *Action:* Open your last AI conversation. Look at your first prompt. How many of the five context elements (who, what, why, how, what you've tried) did you provide? Probably zero or one.

What's Happening Right Now

Prompt Engineering → Context Engineering → Harness Engineering. In March 2026, the industry started talking about a new term: Harness Engineering. The idea: the model is the engine, but without the harness — retries, validation, guard rails, and context management — the engine is useless. Prompt engineering was about words. Context engineering is about information. Harness engineering is about *the system around AI*.

To start understanding Harness Engineering, you need to master the two previous levels first. That's why you're here. That's why let's keep reading.

Chapter 2. 4 Levels of Context

Not all context is equally useful. There's a hierarchy — from "stranger" to "best friend." The higher the level, the better AI understands you.

> Level 0: Stranger

```
You: "Help me write a blog post."
AI: *5 paragraphs of motivational fluff with emojis
    and the phrase "In today's world..."*
```

Zero context. AI doesn't know who you are, what you write about, for whom, in what style. It guesses the average. And the average on the internet is a LinkedIn motivational post.

Result: 90% of people operate at this level and complain that "AI is useless." No, bro. AI isn't useless. Zero context is useless.

> Level 1: Acquaintance

```
You: "I'm a marketer at a B2B SaaS company. Write a cold email
to a potential client — CTO of a 50-person startup.
Product: analytics platform. Tone: professional,
non-aggressive. 3-4 sentences."
```

You provided: role, audience, tone, format. AI understands the *task*.

Result: The answer is already 70% closer to what you need. But every time you need to explain who you are. Every. Single. Time.

> Level 2: Colleague

At this level, AI knows you *before the conversation starts*. You don't explain who you are and what you do — it's already recorded.

How? Through *system context*: a context file, system prompt, or project instructions.

```
# System Context
I'm Alex, a marketer at a B2B SaaS startup (20 people).
I write in English. Tone: direct, no fluff.
Audience: CTOs and VP Engineering, 30-45 years old.
My product: analytics platform for engineering teams.
```

Now:

```
You: "Write a cold email for a CTO."
AI: *personalized email, right tone, right product*
```

Result: AI works like a colleague who knows the context. No need to repeat who you are.

> Level 3: Best Friend

This is where the magic begins.

I asked Claude a simple question: "Tell me about this service." I expected a standard overview. But Claude didn't just describe the service — it added which of *my specific projects* this service could be useful for. With examples. Without being asked.

This is level 3. AI knows not only who you are and what you do, but also:

> Your style — how you write, which words you use
> Your preferences — what you like and what annoys you
> Your history — what worked, what didn't
> Your projects — and how they connect to each other

```
# Extended Context
## Who I Am
Alex, marketer. B2B SaaS, analytics for engineers.

## My Style
- Short sentences. No fluff.
- Specific numbers instead of "significant growth"
- No emojis in work correspondence
- Humor is ok, but dry

## What Works
- Cold emails with a question in the subject line (open rate 34%)
- Case studies with specific metrics

## What Doesn't Work
- "Hope you're doing well" — deleted unread
- Emails longer than 5 sentences
- Generic value propositions
```

Result: AI writes *like you*. Not “for you” — exactly the way you would write, just faster.

> Summary Table

Level	Name	What AI Knows	Your Effort	Quality
0	Stranger	Nothing	Minimum	10%
1	Acquaintance	The task	Every prompt	60%
2	Colleague	You + task	Set up once	85%
3	Best Friend	You + style + history	Set up once + update	95%

> Where Are You Now?

Most people are stuck at level 0-1. You can jump to level 2 in 10 minutes — that’s what the next chapter is about.

→ *Action:* Identify your current level. If you're at 0-1, the next chapter will show you how to reach level 2 in 10 minutes.

What's Happening Right Now

Context Rot. AI response quality degrades long before the context window fills up completely. Even if a model supports 1 million tokens, that doesn't mean it works equally well throughout. Context rot is a real problem, and in chapter 7 we'll break down how to fight it.

Chapter 3. Your Context File

> What It Is

A context file is text that AI reads *before it starts talking to you*. Like a note for a new colleague: “Here’s who I am, here’s what we do, here’s how things work around here.”

In Claude Code, this file is called CLAUDE.md. In ChatGPT — Custom Instructions. In Gemini — System Instructions. Different names, same idea: *write once — AI understands you every time.*

> My Experience

My own context file has gone through 10+ iterations. I update it roughly once a month.

Why? Because people change. Their projects change. Their priorities change. *Your context file should change with you.*

The first version was three lines: who I am, what I do, what language to respond in. The current version is a full document with my style, my projects, my preferences and anti-preferences.

And you know what? The difference between the first and current version is like the difference between level 1 and level 3 from the previous chapter. Same AI, but a *different conversation*.

> Structure

A good context file has 4-5 sections:

1. Who I Am (2-3 lines)

```
## Who I Am
Maria, UX designer. Working at a fintech startup (30 people).
5 years experience, specializing in mobile interfaces.
```

Why: AI calibrates the level. Won’t explain what a wireframe is.

2. What I’m Doing (2-3 lines)

```
## Current Project
Redesigning a mobile app for stock trading.
Target audience: 25-35, investment beginners.
Deadline: June 2026.
```

Why: AI understands the task context. Won’t suggest enterprise solutions for a consumer product.

3. How to Respond (3-5 rules)

```
## How to Help Me
- Be specific: wireframe > description
- If unsure — ask, don't assume
- Critique my decisions if you see a problem
- Format: short bullet points, not essays
- Language: English, technical terms as-is
```

Why: AI adapts to your working style.

4. What NOT to Do (more important than it seems)

```
## What NOT to Do
- Don't suggest Figma plugins (I use Sketch)
- Don't give generic advice like "run an A/B test"
- Don't start responses with "Great question!"
```

Why: AI stops suggesting things that annoy you. This section is the most underrated.

5. Examples of My Work (level 3)

```
## My Style
Example of a good screen description:
"Onboarding screen: 3 steps. Step 1 — name and email.
Step 2 — strategy selection (conservative/balanced/aggressive).
Step 3 — first deposit. Progress bar at top."
```

Why: AI copies your style of phrasing, rather than generating its own.

> Where to Use It

AI Tool	Where to Put Context
Claude (claude.ai)	Project Knowledge or start of chat
ChatGPT	Custom Instructions → “What would you like ChatGPT to know about you?”
Gemini	System Instructions in Google AI Studio
Claude Code	`CLAUDE.md` file in project root
Any chat	Copy text to the beginning of a new chat

> How Long Does It Take

First time: *10 minutes*. Fill in 4 sections (who, what, how, don’t). That’s enough for level 2.

Level up to 3: *another 10 minutes*. Add work examples and preferences.

Update: *2 minutes once a month* (or when your project changes).

> Life-Hack: Let AI Write It For You

Don’t want to write the context file yourself? Let AI ask you questions and assemble it from your answers.

Here’s the prompt — copy and paste into any chat:

```
You are an expert in configuring AI assistants.
Your task: help me create my personal context file
(system prompt / CLAUDE.md / Custom Instructions).

Ask me questions ONE AT A TIME. Wait for my answer before the next
question.
Start with the most important:
1. Who I am and what I do
2. What I'm currently working on
3. How I prefer to receive answers
4. What annoys me about AI responses
5. Examples of my style (if applicable)

Don't limit yourself to just these questions. Decide what else to ask
based on my answers to other questions.
Our goal is to build the ideal system prompt for our collaboration.

After all questions — assemble a ready-to-use context file
in markdown format. Show it to me and ask what to change.
```

Yes, it's meta: AI writes instructions for itself. But it works. And it's 3x faster than writing from scratch.

→ *Action:* Open any AI chat. Paste the prompt above. Answer 5 questions. Get your context file in 5 minutes. Or, if you want control — fill in a template from the appendix manually.

What's Happening Right Now

Context files grow — and that's a problem. People start with 10 lines, and a month later they have 3000 words. AI drowns in instructions and starts ignoring half of them.

3 ways to keep the file short:

1. *Move details to Projects.* Context file = who you are and how to work. Details of specific tasks go in Project Knowledge (chapter 8). The file stays at 300-500 words.
2. *The "one screen" rule.* If the file doesn't fit on one screen — it's too long. Remove anything AI can understand from the task context.
3. *Monthly review.* Delete rules that AI already follows on its own. If you haven't reminded it "don't use emojis" in a month — it's learned that. Delete it.

Chapter 4. 10 Patterns That Work

A pattern is a proven formula. Not a prompt to copy, but a *structure* you fill with your own context.

Each pattern: name → when to use → formula → example → common mistake.

Anti-advice: Don't try to memorize all 10. Patterns 1-3 are the foundation — they cover 80% of tasks. Master those. Patterns 4-10 are a reference. Come back to them when a specific need arises.

> Part 1: Foundation (start with these three)

> Pattern 1: Role + Context + Task

When: Works for 80% of tasks. Your "default" mode.

Formula:

```
You are [role].
Context: [situation].
Task: [what to do].
Format: [how to deliver the result].
```

Example:

```
You are an experienced product marketer.
Context: I'm launching a SaaS for task management.
Target audience: team leads at IT companies of 50-200 people.
Task: Write 3 landing page headline options.
Format: Each option — one line + explanation in parentheses.
```

Common mistake: “You are the world’s best expert at everything.” The more specific the role, the better the response. “Product marketer in B2B SaaS” > “marketer” > “expert.”

> Pattern 2: Chain of Thought

When: You need reasoning, not a direct answer. Analysis, strategy, problem-solving.

Formula:

```
[Task].
Think step by step:
1. First, identify [X].
2. Then analyze [Y].
3. Based on that, propose [Z].
Show your reasoning.
Ask clarifying questions.
```

Example:

```
My competitor dropped prices by 30%.
Think step by step:
1. Why might they have done this? (3 hypotheses)
2. How will this affect our market?
3. What are my options for responding?
Show reasoning, then recommendation.
Ask clarifying questions if needed.
```

Common mistake: Using Chain of Thought for simple tasks. “Think step by step: write a tweet” — that’s overkill.

> Pattern 3: Few-Shot Examples

When: AI doesn’t understand the format you need. Easier to show than to explain.

Formula:

```
Task: [description].

Examples:

Input: [example 1 input]
Output: [example 1 output]

Input: [example 2 input]
Output: [example 2 output]

Now:
Input: [your request]
Output:
```

Example:

```
Rewrite headlines in our blog's style.

Examples:

Input: "How to use machine learning in marketing"
Output: "ML in marketing: 3 cases that actually work"

Input: "Review of best data analytics tools"
Output: "5 analytics tools we can't live without"

Now:
Input: "Benefits of software test automation"
Output:
```

Common mistake: One example isn't enough. AI may latch onto a random detail. Two or three examples establish a pattern.

> Part 2: Reference (come back as needed)

> Pattern 4: Persona Calibration

When: You need a specific style or tone. Especially useful for content.

Formula:

```
Write in the style of [persona description].
Style characteristics:
- [trait 1]
- [trait 2]
- [trait 3]
Do NOT: [anti-patterns].
```

Example:

```
Write like an experienced tech blogger who:
- Explains complex things in simple words
- Uses analogies from everyday life
- Isn't afraid to say "I don't know" or "this is hype"

Do NOT:
- Don't use emojis
- Don't start with "In today's world..."
- Don't be condescending
```

Common mistake: “Write like Steve Jobs” — too abstract. Describe style *traits*, not a name. AI doesn’t know how you imagine “Steve Jobs.”

> Pattern 5: Structured Output

When: You need JSON, a table, a list, or any structured format.

Formula:

```
[Task].
Output the result in this format:
[example structure]
```

Example:

```
Analyze 3 competitors.
Output as a table:

| Competitor | Price | Strengths | Weaknesses | Our Advantage |
```

Common mistake: Not showing the example structure. “Output as a table” — AI guesses the columns. Show which columns you need.

> Pattern 6: Iterative Refinement

When: The result needs step-by-step improvement. Text, design, strategy.

Formula:

```
Step 1: Create [draft/first version].
Step 2: Critically evaluate the result by criteria: [criteria].
Step 3: Improve by fixing the problems found.
Show the final version and what exactly changed.
```

Common mistake: Asking to “write it perfectly on the first try.” Two passes with critique beat one “perfect” attempt.

> Pattern 7: Constraint Setting

When: AI is too verbose, goes off-topic, or adds unnecessary content.

Formula:

```
[Task].
Constraints:
- Maximum [N] words/sentences/points
- Only [topic/area]
- Without [what to exclude]
- Level: [audience]
```

Example:

```
Explain what Kubernetes is.
Constraints:
- Maximum 5 sentences
- For a manager, not a developer
- No technical details about pods and nodes
- One analogy from real life
```

Common mistake: Only constraining length. “Write it short” — AI doesn’t know what “short” means to you. Give a number.

> Pattern 8: Devil’s Advocate

When: You need criticism, not agreement. Testing an idea, plan, or decision.

Formula:

```
I'm planning [solution/idea].
Act as a harsh critic:
1. Find 3-5 weak points
2. For each: why it's a problem + specific failure scenario
3. Suggest how to close each weak point
Don't be polite — be useful.
```

Example:

```
I'm planning to leave my job and go freelance.
Current situation: salary €5000/month, 6 months of savings,
2 potential clients.

Act as a harsh critic:
1. Find 3-5 weak points in my plan
2. Specific failure scenarios
3. How to close each risk
Don't be polite.
Swear if you need to.
```

Common mistake: Asking for “feedback.” AI is polite and supportive by default. Explicitly ask for criticism. The harsher the framing — the more honest the response.

> Pattern 9: Knowledge Boundary

When: It’s important that AI honestly says “I don’t know” rather than making things up.

Formula:

```
[Task].
Important:
- If unsure — say so directly
- Don't invent facts or numbers
- Distinguish verified data from assumptions
- Mark confidence level: ✅ certain / ⚠️ probable / ❓ unsure
- Include links to sources
```

Example:

```
What's the market size for AI tools for HR in Europe?
Important:
- Only verifiable data
- If exact numbers don't exist — say "I don't know" and suggest where to find them
- Mark: ✅ certain / ⚠️ estimate / ❓ assumption
- Include a list of sources with links at the end
```

Common mistake: Trusting AI unconditionally. Even with this pattern — *verify key facts*. AI hallucinates. It's not a bug — it's a property of the technology.

> Pattern 10: Multi-Step Pipeline

When: The task is complex and needs to be broken into steps. Each step is a separate input.

Formula:

```
We'll work in 3 steps. Execute one at a time.
Step 1: [task]. Show the result and wait for confirmation.
Step 2: [task based on step 1].
Step 3: [finalization].
Start with step 1.
```

Example:

```
We'll write an article in 3 steps:
Step 1: Create an outline — 5-7 points. Show it and wait for my OK
or my comments.
Step 2: Write the first draft following the approved outline.
Step 3: Edit: remove fluff, add examples, cut to 800 words.
Start with step 1.
```

Common mistake: Giving all steps at once and expecting the final result. Better: step → review → next step.

> Cheat Sheet: which pattern when

Task	Pattern
Any standard task	1. Role + Context + Task
Analysis, strategy, “think about it”	2. Chain of Thought
AI doesn’t give the right format	3. Few-Shot Examples
Need a specific tone/style	4. Persona Calibration
Table, JSON, structure	5. Structured Output
Text needs improvement	6. Iterative Refinement
Too long/off-topic	7. Constraint Setting
Test an idea	8. Devil’s Advocate
Accuracy matters	9. Knowledge Boundary
Complex task, many steps	10. Multi-Step Pipeline

→ *Action:* Pick one task you do regularly. Apply the right pattern. Compare the result with what you used to get.

What’s Happening Right Now

Popular prompting advice often doesn’t work. No surprise: tips get copied from blog to blog without verification, accumulate myths, and go stale within a month. Conclusion: don’t copy other people’s

prompts. Understand the patterns — and write your own. That’s exactly why this playbook teaches *structures*, not ready-to-paste text.

Chapter 5. Prompt Debugging — When AI Talks Nonsense

AI gave a bad answer. What do you do?

Most people: delete the chat, open a new one, try a different phrasing. Repeat 5 times. Give up.

There's a better way. Prompt debugging is a *systematic approach* to understanding why AI responded poorly, and fixing it in one step.

> Checklist: 7 Reasons for Bad Responses

When AI "doesn't understand" — go through the list top to bottom. The problem is usually in the first three.

1. No Context

Symptoms: Response is generic, "textbook-like," doesn't account for your situation.

Fix: Add who you are, what you're doing, and why. Use the "Role + Context + Task" pattern.

Before: "How to improve sales?" *After:* "I'm a B2B SaaS founder for HR (ARR $200K, 30 clients). How to increase trial-to-paid conversion? Currently 8%, want 15%. Main reason for churn — users don't complete onboarding."

2. Too Many Tasks in One Prompt

Symptoms: AI responds superficially to everything. Or only answers part of the question.

Fix: One task — one prompt. If there are multiple tasks — use Multi-Step Pipeline.

Before: “Analyze competitors, write a strategy, and create a quarterly content plan.” *After:* “Analyze 3 main competitors in a table format: price, target audience, strengths, weaknesses.”

3. No Examples

Symptoms: AI gives the right content but wrong format.

Fix: Show 1-2 examples of what you want. Use Few-Shot Examples.

4. Wrong Model

Symptoms: Response is too simple for a complex task. Or too slow for a simple one.

Fix:

> Simple tasks (translation, formatting, routine) → fast model (Haiku, GPT-4o mini)
> Complex tasks (analysis, strategy, code) → powerful model (Opus, o1)
> Creative tasks (writing, ideas) → mid-tier model (Sonnet, GPT-4o)

> If your subscription allows it, I always recommend using the most powerful model with Extended Thinking mode. But remember — this approach can quickly burn through your limits.

5. Prompt Too Long

Symptoms: AI “gets lost” — responds to part, ignores the rest.

Fix: Cut it down. Remove everything you can live without. Focus > volume. If you have a lot of context — use a context file (chapter 3) so you don’t repeat the basics every time.

6. AI “Stuck” in Context

Symptoms: In long chats, AI starts repeating itself or stubbornly holds onto a mistake you’ve already corrected.

Fix: Start a new chat. Copy the final version of the task (not the conversation history) and ask the question fresh. Clean context = clean answer.

> For large tasks, use models with large context windows. Both Claude and ChatGPT offer context sizes up to 1M tokens. That’s enough for 99% of tasks.

But even 1M tokens isn’t infinity. More on context drift in chapter 7.

7. Task Beyond AI's Capabilities

Symptoms: AI confidently lies. Makes up numbers, links, facts.

Fix: Accept the limitation. AI works poorly with:

> Exact numbers (statistics, financials) — verify
> Recent events — doesn't know
> Highly specialized domains — stays superficial
> Math (sometimes) — calculate yourself

Use the Knowledge Boundary pattern so AI honestly admits when it doesn't know.

> Quick Diagnosis

```
AI gave a bad answer
     │
     ├─ Generic/off-topic? → Add context (#1)
     ├─ Superficial? → Split into tasks (#2)
     ├─ Wrong format? → Give an example (#3)
     ├─ Dumb response? → Change model (#4)
     ├─ Ignores parts? → Shorten prompt (#5)
     ├─ Repeating itself? → New chat (#6)
     └─ Making things up? → AI limitation (#7)
```

> Anti-Advice About Email

Speaking of emails. Today people often use AI to write long, perfectly crafted emails. And what does the recipient do? Right — runs it through *their* AI to get a summary.

Two AIs talking to each other through you as a proxy. Brave new world. Office slop.

If you want to write to a human — *write short*. 3-4 sentences. In your own words. You don't need AI for that.

→ *Action:* Think of the last time AI disappointed you. Go through the checklist — which point was the cause?

One Prompt That Fixes 80% of Problems

When AI misses the mark — don't rewrite your prompt. Paste this:

```
Stop. Your last answer missed the target.
Before responding again — ask me 3 clarifying questions.
```

The best debug is making AI debug itself. It'll ask exactly what was missing from the context. You answer. The next response will be dramatically better.

When Debug Won't Help (and When It Will, But You Didn't Know)

Common myth: "AI doesn't know recent data."

This was true in 2024. In 2026 — no. Most AI tools can search the internet. But they don't do it by default — *you need to ask explicitly*.

```
Find data from the last 2 weeks on the topic [X].
Use internet search. Include links to sources.
```

I personally use this constantly — for trend research, competitor analysis, fact-checking. AI handles it brilliantly when it knows it's *allowed* to go beyond its knowledge.

What genuinely can't be fixed:

> *Exact numbers without sources* — AI can generate convincing statistics out of thin air. Always ask for links. Always verify.
> *Legal advice* — see a lawyer.
> *Medical questions* — see a doctor.
> *Decisions you're accountable for* — AI helps you think, but you decide.

AI is a powerful tool, but not an oracle. Knowing its limits is a skill. Knowing how to push those limits — an even more valuable one.

Chapter 6. Practice — Level Up in One Evening

Theory without practice is zero. Here are two exercises that will turn knowledge into skill.

You already created your context file in chapter 3 (you did, right?). Now — patterns and debugging.

> Exercise 1: Pattern Refactoring

Time: 15 minutes *Result:* Understanding how patterns change response quality

Steps:

1. Pick a task you do regularly. Examples:
   > Analyze data
   > Come up with content ideas
   > Review your plan
2. Write a “naive” prompt — the way most beginners write: > “Come up with ideas for LinkedIn posts.”
3. Rewrite it using the right pattern from chapter 4: > “You are a content strategist for B2B SaaS. > My audience: CTOs and VP Engineering, 30-45 years old, skeptics. > My style: direct, with personal experience, no emojis. > Task: 5 LinkedIn post ideas for this week. > Format: Headline + hook (first line) + brief description.”
4. Send both prompts to the same AI. *But in separate chats.* Compare.

How to know it worked: The difference between responses is obvious. The second is more specific, more relevant, closer to what you’d actually publish.

> Exercise 2: Prompt Debugging in Action

Time: 20 minutes *Result:* Ability to quickly diagnose and fix issues

Steps:

1. Open your AI chat history
2. Find 3 conversations where you were unhappy with the result
3. For each:
   > Identify the cause using the checklist from chapter 5
   > Rewrite the prompt with the fix
   > Send the fixed prompt and evaluate the difference
4. Fill in the table:

Original Prompt	Cause of Failure	Fix	Got Better?
…	No context (1)	Added role + situation	Yes/No
…	Too many tasks (2)	Split into 2 prompts	Yes/No
…	No examples (3)	Added 2 examples	Yes/No

How to know it worked: You start seeing patterns in your own mistakes. For most people, 80% of problems come from the same cause (usually 1 or 2).

Two more bonus chapters ahead. They're about what happens *after* you've set everything up. About context drift and Projects.

Bonus 1. Context Drift — Why AI Gets Dumber in Long Chats

You set up your context file. Learned the patterns. AI responds great. You're happy.

40 minutes into the conversation. And AI starts… getting dumb. Repeating itself. Forgetting what you agreed on. Stubbornly doing what you already told it not to.

Welcome to *context drift*.

> What's Happening

Every AI has a *context window* — the amount of text it can "hold in mind" at once. Think of it as RAM. When the window fills up, AI starts "forgetting" the beginning of the conversation.

But there's a nuance that few people talk about.

Heuristic: quality drops sooner than you think. In my experience and based on community observations, AI starts degrading long before the context window fills up completely. Roughly at the midpoint — sometimes earlier.

This means: even if a model supports 1 million tokens, it doesn't work equally well throughout. Trust your instincts: if AI starts getting dumb — the context is probably overloaded.

> Context Rot

Context rot is the gradual degradation of AI quality as the conversation grows:

1. *Lost instructions.* AI "forgets" rules from the start of the chat. You said "write short" — 20 messages later it's writing essays again.

2. *Style drift.* AI starts "averaging" the style. Was direct and specific — becomes polite and watery.
3. *Repetition loops.* AI stubbornly returns to the same solutions, even if you've already rejected them.
4. *Hallucinations.* The longer the context, the higher the probability that AI will start making things up.

> Compaction — Built-in Protection

Some AI tools (e.g., Claude Code) use *compaction* — automatic context compression. When the context window fills up, the system "squeezes" key facts from the conversation and continues with them, discarding details.

It's like lecture notes instead of a full recording. Works, but with losses.

The compaction problem: AI decides what's important and what's not. Sometimes it throws away exactly what you need.

> 4 Rules for Working with Long Chats

1. New Chat = Clean Context

If the task is done — start a new chat. Don't continue in the old one "so AI remembers." Your context file already "remembers" — the details of the previous task only get in the way.

2. Context Overloaded? Summary + New Chat

If you feel AI is getting dumb — the context is probably overloaded.

> Ask AI to create a detailed summary:

```
Create a detailed summary of our conversation:
1. What we decided (key decisions)
2. What remains unresolved (open questions)
3. Context that needs to be preserved
```

> Check that the summary includes everything important
> Start a new chat and paste the summary from the previous chat at the beginning
> Keep working in the new chat until you notice context drift again

3. One Task — One Chat

Don't mix different tasks in one chat. "Help me prepare for a client meeting" and "analyze competitors" — those are two separate chats. Even if you're too lazy to open a new window.

4. Compact Instructions

If you use tools with compaction (Claude Code, Cursor) — mark critical instructions so they survive compression. Keep them at the beginning of your context file, not in the middle of the conversation.

> When to Start a New Chat

Situation	Action
Task completed	New chat
AI starts repeating	Summary + New chat
AI forgets instructions	Summary + New chat
Changed topic	New chat
Chat running >30 minutes	Check quality, maybe new chat
Everything works fine	Continue

→ *Action:* Next time AI "gets dumb" in a long chat — don't try to fix it with a prompt. Start a new chat with your context file. Compare the difference.

What's Happening Right Now

AI is starting to remember. Both Claude and ChatGPT can now save facts about you between sessions. You say "I work in fintech" — and a week later, in a new chat, AI remembers this.

This isn't a replacement for a context file — memory is still uncontrollable and unpredictable. But the direction is clear: soon context drift will stop being a problem, because AI will remember not just the current chat, but your entire work history.

Until that happens — use the rules from this chapter. But watch for updates: AI memory is evolving fast.

Bonus 2. Projects — AI That Doesn't Forget

You created a context file. That's level 2-3. But what do you do when you have *multiple tasks*?

One context file for the blog. Another for work. A third for a side project. Copy the right one every time? Switch Custom Instructions?

There's a better solution: *Projects*.

> What Are Projects

Both Claude and ChatGPT support Projects — isolated workspaces within a single account. Each project is:

- > Its own context file (Project Instructions / System Prompt)
- > Its own uploaded files (Knowledge Base)
- > Its own chat history
- > Its own AI "personality"

Essentially, you have *multiple different AI assistants*, each configured for its own task.

> Why You Need This

Without Projects: One AI knows everything about everything. Or rather, nothing about anything — because context gets diluted.

With Projects: You have a separate AI for each role:

- > "Blog assistant" — knows your style, audience, tone of voice
- > "Work analyst" — knows your product, metrics, competitors
- > "Career coach" — knows your resume, goals, market
- > "Automation helper" — knows your tasks, tools, level

Each remembers context between sessions. No need to explain again.

> How It Works

Claude Projects (claude.ai)

1. Go to claude.ai → sidebar → “Projects”
2. Create a new project (e.g., “My Blog”)
3. Add *Project Instructions* — this is your context file for this project
4. Add files to *Knowledge* — text samples, guidelines, data
5. All chats within the project automatically see these instructions and files

What to put in Knowledge:

> Examples of your best writing (for blog)
> Product documentation (for work)
> Your resume and cover letter (for career)
> Learning materials (for education)

ChatGPT Projects

1. Go to ChatGPT → sidebar → “Projects”
2. Create a project
3. Add *Instructions* and *Files* — works the same as Claude
4. All chats in the project use these instructions

> 5 Projects to Start With

Project	Instructions (essence)	Knowledge (files)
My Blog	Tone of voice, audience, style, anti-patterns	5-10 best posts
Work	Role, product, metrics, team	Product docs, OKRs
Career	Current role, goals, skills	Resume, LinkedIn
Learning	Topic, level, learning style	Learning materials
Side Project	Description, stack, stage	Code, wireframes, notes

> Anti-Advice

Don't create 20 projects. Start with 2-3 of your most frequent tasks. Add more as needed. 3 well-configured projects are better than 15 empty ones.

> Projects vs Context File

Context File	Projects
What it is	Text with instructions
Workspace	*Where it lives*
One for everything	One per task
What it remembers	Who you are and how to work
1. uploaded files + history	*When to use*
One main task	Multiple different tasks

Projects are context files on steroids. The system prompt says "who you are." Projects say "who you are *in this role*, with this data, in this project."

→ *Action:* Create one project in Claude or ChatGPT. Start with your most frequent task. Add instructions from your context file + 3-5 relevant files. Work for a week — evaluate the difference.

Key Nuance: Files in a Project ≠ Files in AI's Head

You uploaded 15 files to a project. It seems like AI now "knows" them all. Not quite.

AI doesn't re-read all files with every request. It searches for relevant fragments — and sometimes misses. The more files, the higher the chance the right one gets lost.

Pro tip: At the start of a new chat, hint which files to focus on.

```
In this chat we're working with files:
- tone-of-voice.md — my writing style
- audience.md — audience description
The rest of the project files aren't needed right now.
```

It’s like telling a colleague: “Open these two documents, the rest doesn’t matter right now.” Simple action — but response quality noticeably improves.

What’s Next

You’ve read the guide and done the exercises. Now you:

> Understand the difference between a prompt and context
> Know your level and how to level up
> Have a context file that makes AI personal
> Know 10 patterns for different tasks
> Can diagnose and fix bad responses

Next step: Use it. Every day. In a week you won’t remember how you worked without context.

This is Playbook #1 — the foundation. Next playbooks are released at postnikov.ai.

Want a system, not one-time advice → hellonew.pro