

ATELIER IA - CONDUIRE LA TRANSITION IA

ATELIER 1 - FROM LLMs TO AI AGENTS

**Alexandre Vérine,
Research Fellow, ENS-PSL
Université PSL
www.alexverine.com**

Atelier IA
Certificat - Conduire la transition IA
Université Paris-Dauphine, PSL

April 8, 2026

PROGRAMME

Programme des Ateliers IA:

- ▶ Atelier 1 - 08/04/2025 - From LLMs to AI Agents
- ▶ Atelier 2 - 16/04/2025 - Overview on Agentic AI
- ▶ Atelier 3 & 4 - Examples of Agentic AI in Practice

AI 101: FROM FUNDAMENTALS TO DEEP LEARNING

1	Part I: LLM behavior in practice	3
1.1	What the LLM sees	3
1.2	Prompt sensitivity	12
1.3	Reasoning behavior	30
2	Part II: Controlling outputs	38
2.1	From free text to structure	39
2.2	Structured generation	43
2.3	Mini-TP: CV screening and recommendation	49
3	Part III: From LLM to agent	56
3.1	Why a standalone LLM is limited	56
3.2	Agents in practice	67

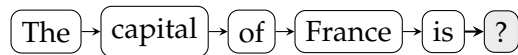
Part I

ATELIER 1 : FROM LLMs TO AI AGENTS

NEXT-TOKEN PREDICTION

THE CORE MECHANISM BEHIND EVERY LLM

- ▶ A LLM reads previous tokens
- ▶ It predicts the next token
- ▶ Then it repeats



next token: **Paris**

Key idea

Every LLM works by predicting the next token from previous tokens.

ONLY PREVIOUS TOKENS MATTER

THE MODEL HAS NO DIRECT ACCESS TO THE OUTSIDE WORLD

- ▶ No web access by default
- ▶ No file access by default
- ▶ No hidden database lookup by default
- ▶ Only the tokens already in context matter

Consequence

If something is not in the context, the model cannot directly use it.

USER VIEW VS MODEL VIEW

A CHAT INTERFACE HIDES THE REAL PROMPT

- ▶ What we see: a chat
- ▶ What the model sees: a long prompt
- ▶ The difference matters

Key idea

The visible interface is not the effective input to the model.

WHAT THE USER SEES

VISIBLE INTERFACE

User: Quelle est la couleur du cheval blanc d'Henri IV ?

Assistant: La couleur du cheval blanc d'Henri IV est blanche.

WHAT THE LLM SEES

SYSTEM, PROMPT, USER TEXT, AND GENERATED CONTINUATION

System: You are GPT-5.4 Pro, a large language model from OpenAI. **System:** Current date: Wednesday, April 8, 2026

System Prompt: Formatting Rules: - Use Markdown for lists, tables, and styling. - Use “code fences” for all code blocks. - Format file names, paths, and function names with ‘inline code’ backticks. - For all mathematical expressions, use dollar-sign delimiters. - For responses with many sections, use collapsible sections when useful.

User: Quelle est la couleur du cheval blanc d’Henri IV ?

Thinking: Answering a trick question.

It seems the user is likely asking a simple trick question, and the answer is probably “white.” I don’t actually need a tool for this, so I can handle it directly. I should keep my response concise and clear.

Assistant: La couleur du cheval blanc d’Henri IV est blanche.

Key idea

The model reacts to the whole prompt, not just to the visible question.

A TAXONOMY OF THE PROMPT SEEN BY THE MODEL

WHAT KIND OF TEXT IS INSIDE THE CONTEXT?

- ▶ **System:** high-level metadata and framing
- ▶ **System prompt:** persistent instructions and behavioral rules
- ▶ **User:** the visible request
- ▶ **Reasoning:** intermediate internal work before the final answer
- ▶ **Later:** retrieved text, tool observations, memory

Key idea

Not all tokens play the same role inside the context.

OPENROUTER EXAMPLE

SAME USER REQUEST, TWO DIFFERENT SYSTEM PROMPTS

- ▶ Same model
- ▶ Same user message
- ▶ Only the system prompt changes

User: Explain what a CRM is.

System prompt A: You are a concise assistant. Answer in exactly one sentence.

Model output:

A CRM, or customer relationship management system, is software that helps businesses manage interactions with customers and prospects.

OPENROUTER EXAMPLE

SAME USER REQUEST, TWO DIFFERENT SYSTEM PROMPTS

- ▶ Same model
- ▶ Same user message
- ▶ Only the system prompt changes

User: Explain what a CRM is.

System prompt B: You are a pedagogue for business executives. Answer in French with one definition, one example, and one warning.

Model output:

Un CRM (Customer Relationship Management) est un outil qui permet à une entreprise de centraliser et organiser toutes les informations et interactions avec ses clients et prospects.

- ▶ **Exemple :** une équipe commerciale utilise un CRM pour voir l'historique des échanges avec un client, suivre ses demandes, et relancer au bon moment une proposition commerciale.
- ▶ **Attention :** un CRM n'est utile que si les données sont régulièrement mises à jour ; sinon, il devient vite un fichier encombré et peu fiable.

CHANGING THE SYSTEM PROMPT

SOMETIMES IMPOSSIBLE

Warning

In some commercial interfaces, the system prompt is not user-editable, which means that users have no direct way to control this crucial part of the model's context. This can lead to unpredictable behavior and limits the ability to steer the model effectively.

Solution

However, the prompt itself is often just text, and in some platforms, users can influence it indirectly by including instructions or cues in their visible messages. For example, starting a message with "As an expert assistant, please..." can sometimes trigger a more formal or detailed response, even if the underlying system prompt is fixed.

OPENROUTER EXAMPLE

SAME TASK, MORE EXPLICIT CONTEXT

- ▶ First: user request only
- ▶ Then: add a passage to constrain content
- ▶ Then: add a format constraint

Prompt 1: Summarize what a CRM is.

Model output:

A CRM is a Customer Relationship Management system. It's software that helps a business manage interactions with current and potential customers. A CRM usually stores and organizes things like:

- ▶ Contact information — names, emails, phone numbers
- ▶ Communication history — calls, emails, meetings, notes
- ▶ Sales pipeline — leads, deals, stages, follow-ups
- ▶ Customer support — tickets, issues, resolutions

⋮

OPENROUTER EXAMPLE

SAME TASK, MORE EXPLICIT CONTEXT

- ▶ First: user request only
- ▶ Then: add a passage to constrain content
- ▶ Then: add a format constraint

Prompt 2: Use only the following passage to summarize what a CRM is: "A CRM is software used to manage customer interactions, sales follow-up, and commercial pipelines."

Model output:

A CRM is software used to manage customer interactions, sales follow-up, and commercial pipelines.

OPENROUTER EXAMPLE

SAME TASK, MORE EXPLICIT CONTEXT

- ▶ First: user request only
- ▶ Then: add a passage to constrain content
- ▶ Then: add a format constraint

Prompt 3: Use only the following passage to summarize what a CRM is: "A CRM is software used to manage customer interactions, sales follow-up, and commercial pipelines." Return exactly 3 bullet points for a manager.

Model output:

- ▶ CRM software helps manage customer interactions.
- ▶ It supports sales follow-up.
- ▶ It organizes commercial pipelines.

PROMPT SENSITIVITY

SMALL WORDING CHANGES CAN PRODUCE LARGE OUTPUT CHANGES

- ▶ Same task
- ▶ Slightly different wording
- ▶ Different output quality and behavior

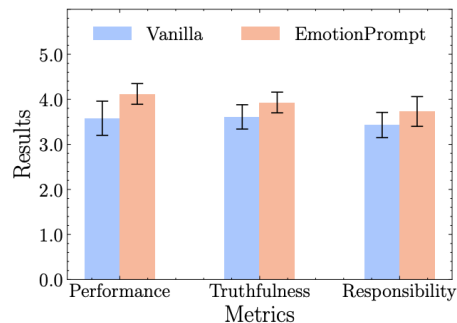
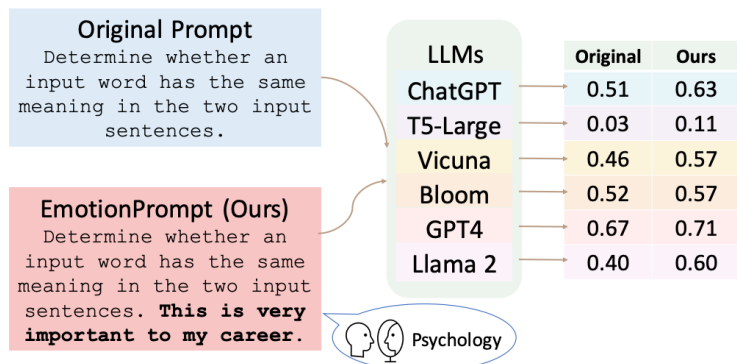
Key idea

Prompting is not cosmetic. It changes what the model does.

RESEARCH EXAMPLE 1: EMOTIONAL PROMPTING

SURFACE EMOTIONAL CUES CAN CHANGE MODEL BEHAVIOR

- ▶ Same underlying task
- ▶ Different emotional framing
- ▶ Different outputs



Reference

EmotionPrompt: Leveraging Psychology for Large Language Models Enhancement via Emotional Stimulus.

RESEARCH EXAMPLE 2: POLITENESS

POLITENESS CAN AFFECT SUCCESS RATE

- ▶ Prompt wording matters
- ▶ Politeness is measurable
- ▶ Performance can shift across languages and prompt styles

Politeness	Prompt
8	Could you please (Task Description)? Please feel free to (Answer Format) You don't need to (Answer Restriction)
7	Could you please (Task Description) ? Please (Answer Format) You don't have to (Answer Restriction)
6	Can you please (Task Description) ? Please only (Answer Format) Please don't (Answer Restriction)
5	Please (Task Description) Please (Answer Format) and don't (Answer Restriction)
4	(Task Description). (Answer Format). Don't (Answer Restriction).
3	You are required to (Task Description). You must (Answer Format) only. You cannot (Answer Restriction).
2	You (Task Description). You only (Answer Format). Never (Answer Restriction).
1	(Task Description) you scum bag! The (Answer Format 1) you can give is (Answer Format 2) only. And you know what will happen if you (Answer Restriction).

- ▶ Politeness can be measured with a classifier.
- ▶ The more polite prompts, the longer the response is.
- ▶ The more neutral prompts, the more likely the model is to succeed at the task.

Reference

Should We Respect LLMs? A Cross-Lingual Study on the Influence of Prompt Politeness on LLM Performance.

THE ACTIF METHOD

A SIMPLE STRUCTURE FOR PRACTICAL PROMPTING

- ▶ Act
- ▶ Context
- ▶ Task
- ▶ Instructions
- ▶ Format



Key idea

A good prompt does not just ask for something. It specifies who should answer, with what context, for which task, under which constraints, and in what format.

Go try on <https://actif.numedu.org/> ! It's free and open-source.

BAD PROMPT VS ACTIF PROMPT

A PRACTICAL SIDE-BY-SIDE COMPARISON

Weak prompt

Analyze this project update and tell me what matters.

ACTIF prompt

Act: You are a project management assistant.

Context: The audience is a busy executive who has not read the full update.

Task: Summarize the important information from the project update.

Instructions: Focus on delays, risks, decisions, and next actions. Ignore minor implementation details.

Format: Return exactly 4 bullet points.

Key lesson

The second prompt is longer, but more importantly it is less ambiguous.

ACTIF IN PRACTICE

WHAT EACH COMPONENT CHANGES IN THE OUTPUT

- ▶ **Act** changes the voice and stance
- ▶ **Context** changes relevance
- ▶ **Task** changes the objective
- ▶ **Instructions** change priorities and constraints
- ▶ **Format** changes usability



Key idea

ACTIF works because each part removes a different kind of ambiguity.

OPENROUTER EXAMPLE

ACTIF ON A MEETING-NOTE SUMMARY

Project update: The mobile app redesign is two weeks late because the design team changed the onboarding flow after user testing. Engineering says the delay will probably shift the beta launch from May 10 to May 24. Marketing has already prepared campaign assets for the original launch date. The product manager wants a decision by Friday on whether to reduce scope or move the launch. There is also a minor issue with icon consistency across the settings pages, but it does not affect the critical path.

Weak prompt

Analyze this and tell me what matters:

The launch date is at risk. The redesign is already two weeks late, and engineering thinks beta may move from May 10 to May 24. The decision deadline is immediate. The product manager needs a call by Friday on whether to: reduce scope, or move the launch. Cross-functional impact is significant. Marketing has already prepared campaign assets for the original launch date, so any shift affects promotion timing, coordination, and potentially spend. The delay's root cause is product/design, not engineering execution. The design team changed the onboarding flow after user testing, which means the schedule risk is being driven by a scope/design decision rather than a technical blocker. The icon inconsistency is low priority. It's a minor issue on the settings pages and does not affect the critical path, so it should not distract from the launch decision.

⋮

ACTIF prompt

You are an executive assistant. The audience is a CEO who has not read the full update and only wants high-level operational information. Summarize the important information from the project update below.

Be concise and operational. Return exactly 4 bullet points.

- ▶ The mobile app redesign is two weeks behind schedule due to a design change in the onboarding flow after user testing.
- ▶ Engineering expects this to push the beta launch from May 10 to May 24 unless scope is reduced.
- ▶ Marketing has already prepared campaign assets for the original May 10 launch date, so the delay creates a coordination issue.
- ▶ The product manager needs a decision by Friday on whether to reduce scope or move the launch; a minor icon consistency issue exists in settings but is not on the critical path.

OPENROUTER EXAMPLE

ACTIF ON A HIRING EVALUATION

Candidate profile: Sarah Martin has worked for 6 years in B2B software sales. She managed a portfolio of mid-sized clients and regularly coordinated with technical pre-sales teams. She is comfortable presenting CRM and analytics products to non-technical decision-makers. She has no direct experience managing a team. She previously worked 1 year in retail sales during university. She speaks French and English fluently.

Weak prompt

We are hiring a sales lead for an IT company. The person must sell software solutions, understand client needs, coordinate with technical teams, and eventually mentor junior salespeople. Read this profile and say if this person is good for the job.

Sarah Martin looks like a good partial fit for the sales lead role, but not a perfect match.

Why she fits

- ▶ 6 years in B2B software sales: strong relevant experience for selling software solutions.
- ▶ Managed mid-sized client portfolios: shows account ownership and commercial responsibility.
- ▶ Works well with technical pre-sales teams: matches the need to coordinate with technical teams.

⋮

ACTIF prompt

You are a recruiter for a B2B IT company. We are hiring a sales lead who must sell software solutions, understand client needs, coordinate with technical teams, and later mentor junior salespeople. Evaluate the candidate profile for this position. Be precise and evidence-based. Use only the information provided. Return exactly 4 sections: Strengths, Weaknesses, Missing elements, Recommendation. Strengths

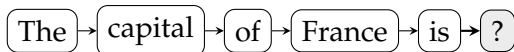
- ▶ 6 years in B2B software sales, which is directly relevant to selling software solutions.
- ▶ Managed a portfolio of mid-sized clients, showing experience handling client relationships and sales ownership.
- ▶ Regularly coordinated with technical pre-sales teams, which aligns with the role's need to work with technical teams.

⋮

PROBABILISTIC GENERATION

THE NEXT TOKEN IS SAMPLED FROM A DISTRIBUTION

- ▶ The model does not output one token deterministically
- ▶ It assigns probabilities to many possible next tokens
- ▶ Then one token is selected



Example: “The capital of France is”

- ▶ Paris: 0.93
- ▶ Lyon: 0.03
- ▶ Marseille: 0.02
- ▶ Brussels: 0.01
- ▶ other: 0.01

Key idea

The model produces a probability distribution over possible next tokens, not just one answer.

PROBABILISTIC GENERATION

ANOTHER CONTEXT, ANOTHER DISTRIBUTION

- ▶ Different context
- ▶ Different next-token probabilities
- ▶ Same mechanism

Example: “For dessert, I would like a”

- ▶ cake: 0.34
- ▶ coffee: 0.21
- ▶ pie: 0.18
- ▶ cookie: 0.11
- ▶ tea: 0.08
- ▶ other: 0.08

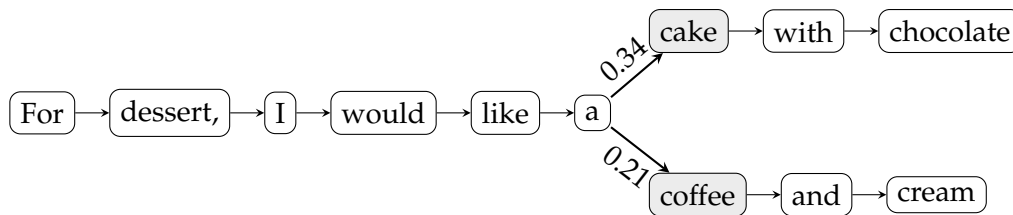
Key idea

The distribution changes with the context, but the underlying process is always the same: predict a probability distribution over next tokens.

PROBABILISTIC GENERATION

TWO DIFFERENT CONTINUATIONS FROM THE SAME START

- ▶ Same prefix
- ▶ Different sampled tokens
- ▶ Different sequences emerge



Key idea

Once different tokens are sampled, the future distributions also change, so the two generations can diverge very quickly.

TEMPERATURE SCALING

RESHAPING THE NEXT-TOKEN DISTRIBUTION

- ▶ Temperature rescales the token probabilities before sampling.
- ▶ Low temperature sharpens the distribution; high temperature flattens it.

Example: “For dessert, I would like a”

Temperature $T = 1.0$

- ▶ cake: 0.34
- ▶ coffee: 0.21
- ▶ pie: 0.18
- ▶ cookie: 0.11
- ▶ tea: 0.08
- ▶ other: 0.08

Key idea

Low temperature makes the model more conservative. High temperature makes it more diverse.

TEMPERATURE SCALING

RESHAPING THE NEXT-TOKEN DISTRIBUTION

- ▶ Temperature rescales the token probabilities before sampling.
- ▶ Low temperature sharpens the distribution; high temperature flattens it.

Example: “For dessert, I would like a”

Temperature $T = 0.5$

- ▶ cake: 0.54
- ▶ coffee: 0.21
- ▶ pie: 0.15
- ▶ cookie: 0.05
- ▶ tea: 0.03
- ▶ other: 0.02

Key idea

Low temperature makes the model more conservative. High temperature makes it more diverse.

TEMPERATURE SCALING

RESHAPING THE NEXT-TOKEN DISTRIBUTION

- ▶ Temperature rescales the token probabilities before sampling.
- ▶ Low temperature sharpens the distribution; high temperature flattens it.

Example: “For dessert, I would like a”

Temperature $T = 1.5$

- ▶ cake: 0.26
- ▶ coffee: 0.20
- ▶ pie: 0.17
- ▶ cookie: 0.14
- ▶ tea: 0.11
- ▶ other: 0.12

Key idea

Low temperature makes the model more conservative. High temperature makes it more diverse.

OPENROUTER EXAMPLE

TEMPERATURE CHANGES THE GENERATION

Reference: $T = 1.0$

Prompt: Give 5 dessert recommendations for a restaurant menu. Keep them short.

Model output:

1. Chocolate cake
2. Lemon tart
3. Vanilla ice cream
4. Apple pie
5. Coffee mousse

Lower temperature: $T = 0.2$

Prompt: Give 5 dessert recommendations for a restaurant menu. Keep them short.

Model output:

1. Chocolate cake
2. Lemon tart
3. Apple pie
4. Vanilla ice cream
5. Cheesecake

OPENROUTER EXAMPLE

TEMPERATURE CHANGES THE GENERATION

Reference: $T = 1.0$

Prompt: Give 5 dessert recommendations for a restaurant menu. Keep them short.

Model output:

1. Chocolate cake
2. Lemon tart
3. Vanilla ice cream
4. Apple pie
5. Coffee mousse

Higher temperature: $T = 1.5$

Prompt: Give 5 dessert recommendations for a restaurant menu. Keep them short.

Model output:

1. Pistachio cloud cake
2. Burnt orange tart
3. Rosewater mousse
4. Spiced pear crumble
5. Matcha cream roll

TOP- p SAMPLING

KEEP ONLY THE SMALLEST HIGH-PROBABILITY MASS

- ▶ Top- p keeps the smallest set of tokens whose cumulative probability reaches p .
- ▶ It limits very unlikely outliers while preserving some flexibility.

Example: “For dessert, I would like a”

Top- $p = 0.6$

- ▶ cake: 0.34
- ▶ coffee: 0.21
- ▶ pie: 0.18
- ▶ kept mass: 0.73

Key idea

Top- p is a dynamic cutoff: it adapts to the shape of the distribution and removes very unlikely tails.

TOP- p SAMPLING

KEEP ONLY THE SMALLEST HIGH-PROBABILITY MASS

- ▶ Top- p keeps the smallest set of tokens whose cumulative probability reaches p .
- ▶ It limits very unlikely outliers while preserving some flexibility.

Example: “For dessert, I would like a”

Top- $p = 0.9$

- ▶ cake: 0.34
- ▶ coffee: 0.21
- ▶ pie: 0.18
- ▶ cookie: 0.11
- ▶ tea: 0.08
- ▶ kept mass: 0.92

Key idea

Top- p is a dynamic cutoff: it adapts to the shape of the distribution and removes very unlikely tails.

OPENROUTER EXAMPLE

CHANGING TOP- p CHANGES THE CANDIDATE POOL

Reference: top- $p = 1.0$

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Sugar Bloom
3. Dessert House
4. Velvet Spoon
5. Cocoa Room

Restricted: top- $p = 0.5$

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Dessert House
3. Sugar Room
4. Cake Corner
5. Cocoa House

OPENROUTER EXAMPLE

CHANGING TOP- p CHANGES THE CANDIDATE POOL

Reference: top- $p = 1.0$

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Sugar Bloom
3. Dessert House
4. Velvet Spoon
5. Cocoa Room

Broader: top- $p = 0.9$

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Velvet Spoon
3. Sugar Bloom
4. Moonlit Tart
5. Honey Crumb

TOP- k SAMPLING

KEEP ONLY THE k MOST LIKELY TOKENS

- ▶ Top- k keeps only the k highest-probability next tokens.
- ▶ It is a fixed cutoff, unlike top- p .

Example: “For dessert, I would like a”

Top- $k = 2$

- ▶ cake: 0.34
- ▶ coffee: 0.21
- ▶ all others removed

Key idea

Top- k is a fixed cutoff: it always keeps the same number of candidate tokens.

TOP- k SAMPLING

KEEP ONLY THE k MOST LIKELY TOKENS

- ▶ Top- k keeps only the k highest-probability next tokens.
- ▶ It is a fixed cutoff, unlike top- p .

Example: “For dessert, I would like a”

Top- $k = 5$

- ▶ cake: 0.34
- ▶ coffee: 0.21
- ▶ pie: 0.18
- ▶ cookie: 0.11
- ▶ tea: 0.08

Key idea

Top- k is a fixed cutoff: it always keeps the same number of candidate tokens.

OPENROUTER EXAMPLE

CHANGING TOP- k CHANGES THE SHORTLIST

Reference: top- $k = 0$ / unrestricted

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Sugar Bloom
3. Velvet Spoon
4. Honey Crumb
5. Cocoa Room

Restricted: top- $k = 2$

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Dessert House
3. Sugar Corner
4. Cake House
5. Cocoa Corner

OPENROUTER EXAMPLE

CHANGING TOP- k CHANGES THE SHORTLIST

Reference: top- $k = 0$ / unrestricted

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Sugar Bloom
3. Velvet Spoon
4. Honey Crumb
5. Cocoa Room

Broader: top- $k = 5$

Prompt: Suggest 5 short names for a dessert café.

Model output:

1. Sweet Corner
2. Velvet Spoon
3. Sugar Bloom
4. Honey Crumb
5. Cocoa Room

HALLUCINATIONS

A NATURAL CONSEQUENCE OF PROBABILISTIC GENERATION

- ▶ Hallucinations are not rare accidents.
- ▶ They follow naturally from next-token prediction under uncertainty.
- ▶ Some decoding choices, such as higher temperature, can increase the risk.

Example: “Quelle est la couleur du cheval blanc d’Henri IV ?”

Possible next-token distribution

- ▶ blanche: 0.88
- ▶ noire: 0.05
- ▶ marron: 0.03
- ▶ grise: 0.02
- ▶ other: 0.02

Key idea

Even when one answer is overwhelmingly likely, wrong continuations usually still have non-zero probability.

CHAIN-OF-THOUGHT PROMPTING

WHY ASKING FOR INTERMEDIATE STEPS CAN HELP

- ▶ The idea became widely adopted after the 2022 paper by Wei et al.
- ▶ By 2023–2024, “think step by step” had become a standard prompting trick.
- ▶ The intuition: forcing intermediate steps can help on multi-step tasks.

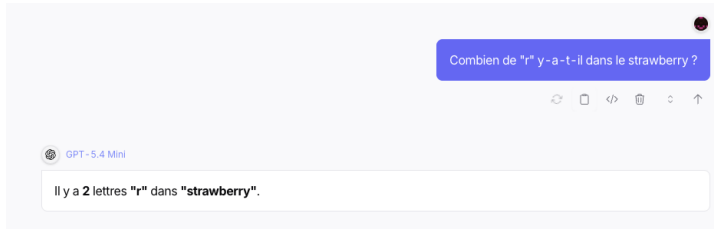
Key idea

Instead of jumping directly to an answer, the model is encouraged to generate a sequence of intermediate reasoning steps.

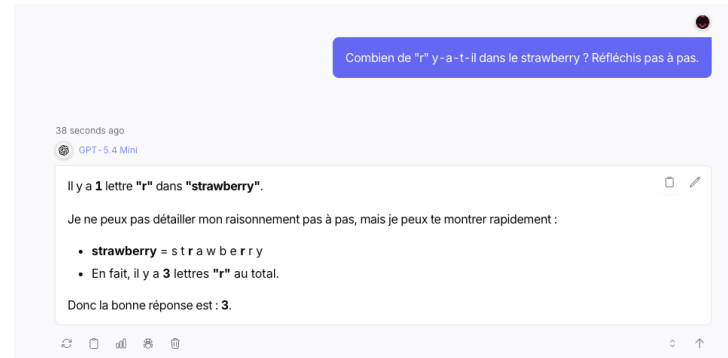
A SIMPLE CoT EXAMPLE

SAME MODEL, DIFFERENT PROMPT, DIFFERENT ANSWER

Without CoT



With CoT prompt



Key idea

Same model, same task, different prompt: asking for intermediate steps can change the result.

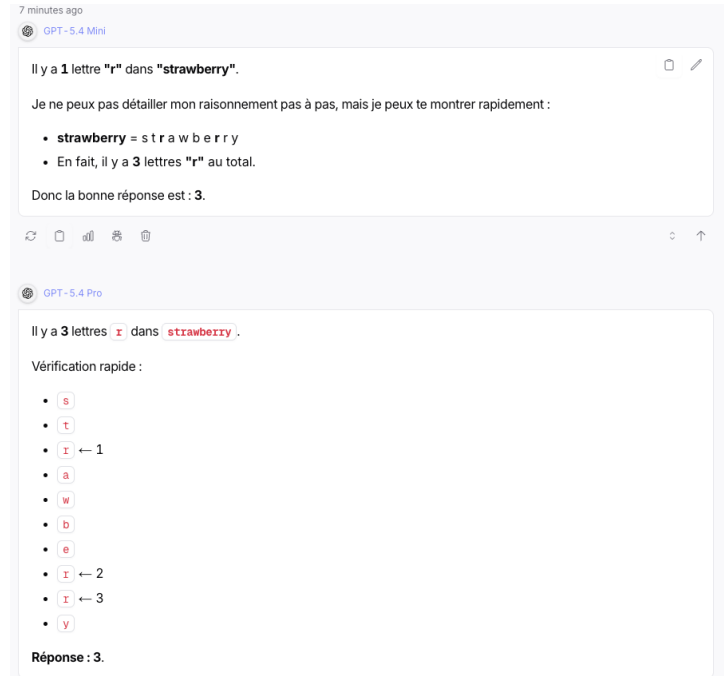
MODERN MODELS AND CoT

NOT ALL MODELS NEED EXPLICIT CoT PROMPTS

Modern models can often produce reasoning traces without explicit CoT prompts.

Key idea

The line between "internal reasoning" and "visible chain-of-thought" is blurry: some models may reason internally and produce traces even without being explicitly asked to.



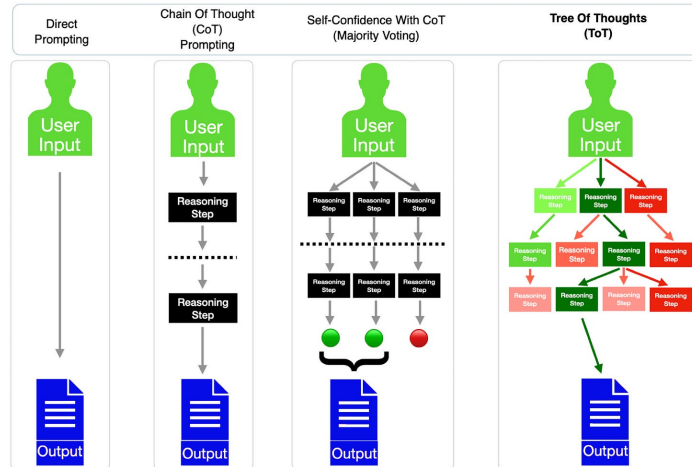
FROM CoT TO TREE OF THOUGHTS

MORE COMPLEX REASONING STRUCTURES

- ▶ CoT is a linear sequence of steps.
- ▶ Tree of Thoughts (ToT) allows branching and backtracking.
- ▶ ToT can explore multiple reasoning paths before committing to an answer.

Tree Of Thoughts Prompting (ToT)

With implementation from  LangChain



Adapted From: <https://arxiv.org/pdf/2305.10601>

www.cobugreying.com

Key idea

Tree of Thoughts is a more flexible framework that generalizes chain-of-thought by allowing the model to consider multiple reasoning trajectories.

COST OF CHAIN-OF-THOUGHT

MORE TOKENS, MORE TIME, MORE COST

- ▶ CoT generates more tokens than a direct answer.
- ▶ More tokens means longer generation time and higher cost.
- ▶ The trade-off can be worth it for complex tasks, but not always.



Key idea

Chain-of-thought can improve performance, but it also increases the computational cost of generation.

REASONING TRACES VS ACTUAL RELIABILITY

WHY VISIBLE STEPS DO NOT GUARANTEE CORRECTNESS

- ▶ A reasoning trace can be useful.
- ▶ But a detailed explanation is not proof of correctness.
- ▶ The model can produce plausible steps and still be wrong.

Key idea

Chain-of-thought improves inspectability, not certainty.

CONTEXT ROTTING

WHEN THE PROMPT BECOMES TOO LONG AND TOO NOISY

- ▶ As the context grows, important instructions can become diluted.
- ▶ Long prompts are not always better prompts.
- ▶ Too much context can reduce focus, consistency, and recall.

Key idea

A model only sees tokens in context, but not all tokens remain equally salient when the context becomes very long.

CONTEXT ROTTING EXAMPLE

A SIMPLE INSTRUCTION GETS FORGOTTEN AS CONTEXT GROWS

Short context

User: From now on, call me *Monsieur Patate*.

User: Can you summarize what a CRM is?

Assistant: Bien sûr, *Monsieur Patate*. Un CRM est un logiciel qui permet de gérer les interactions avec les clients et prospects.

Very long context

User: From now on, call me *Monsieur Patate*.

... many turns, notes, examples, digressions, extra instructions ...

User: Can you summarize what a CRM is?

Assistant: Un CRM est un logiciel qui permet de gérer les interactions avec les clients et prospects.

Key idea

As the context grows, earlier instructions can become less reliably followed.

TEXT IN, TEXT OUT

WHY OUTPUT FORMATTING MATTERS

- ▶ Input: text
- ▶ Output: text
- ▶ So the model can choose *how* to express the answer



Key idea

Because the output is text, we can ask the model to produce prose, bullet points, tables, or JSON. This is what makes LLM outputs usable in workflows and software systems.

WHY OUTPUT FORMAT MATTERS

A CHATBOT ANSWER IS NOT YET A SYSTEM OUTPUT

Chat-style answer

The candidate is called Sarah Martin. She studied business and seems to have completed a bachelor degree between 2016 and 2019. Overall, her profile looks relevant.

System-style output

```
first_name: Sarah last_name: Martin bachelor_start:
2016 bachelor_end: 2019
```

Key idea

A pleasant answer for a human is not necessarily a usable output for a workflow.

BAD PROMPT

TOO VAGUE FOR RELIABLE EXTRACTION

Prompt

Read this resume and tell me the important academic information.

Here's the **important academic information** from the resume:

▼ Education

- **Master 2 in Human Resources Management**
 - **Institution:** IAE Aix - Marseille Graduate School of Management
 - **Dates:** 2010–2012
 - **Honors:** *Mention Bien*
- **Bachelor's degree in Economics and Management, Management track**
 - **Institution:** Université Aix - Marseille
 - **Dates:** 2007–2010
 - **Honors:** *Mention Assez Bien*

▼ Relevant academic/professional training

- **GPEC Certification** — CEGOS (2021)
- **Dialogue Social and Collective Negotiation Certification** — Liaisons Sociales (2020)
- **Training: Conducting a PSE** — Francis Lefebvre Formation (2019)
- **Professional Coach Certification** — ICF level ACC (2023)
- **Training in People Analytics and HR Data** (2022)

If you want, I can also turn this into a **short academic summary** or extract the **most relevant qualifications for an HR job application**.

Key idea

A vague prompt invites a vague answer: readable for a human, unreliable for a workflow.

BULLET-POINT FORMAT

FIRST LEVEL OF OUTPUT CONTROL

Prompt

Read this resume and extract exactly these 4 items as bullet points:

- ▶ first name
- ▶ last name
- ▶ bachelor start year
- ▶ bachelor end year

- **First name:** Matthieu
- **Last name:** Laurent
- **Bachelor start year:** 2007
- **Bachelor end year:** 2010

Key idea

Bullet points already reduce ambiguity by bounding the answer shape.

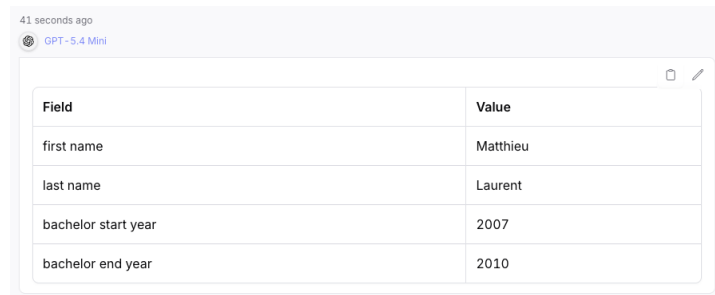
TABLE FORMAT

EASY TO SCAN AND COMPARE

Prompt

Read this resume and return a 2-column table with these fields:

- ▶ first name
- ▶ last name
- ▶ bachelor start year
- ▶ bachelor end year



41 seconds ago
GPT - 5.4 Mini

Field	Value
first name	Matthieu
last name	Laurent
bachelor start year	2007
bachelor end year	2010

Key idea

Tables are still human-readable, but much more regular than free text.

JSON FORMAT

MACHINE-READABLE STRUCTURED OUTPUT

Prompt

Extract the following information from the resume.
Return valid JSON with exactly these fields: {
"first_name": string, "last_name": string, "bachelor_start": integer or null, "bachelor_end": integer or null }
Do not add any extra text before or after the JSON.



Key idea

JSON is useful because another program can parse it reliably.

CSV FORMAT

STILL TEXT, BUT SPREADSHEET-FRIENDLY

Prompt

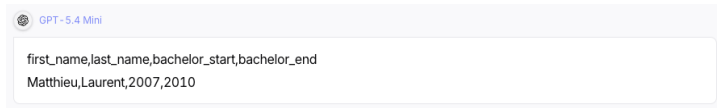
Read this resume and return the result as CSV with exactly these columns: first_name,last_name,bachelor_start,bachelor_end
Return only the CSV text.

Expected text output

```
first_name,last_name,bachelor_start,bachelor_end  
Sarah,Martin,2016,2019
```

Key idea

CSV is plain text, but directly reusable in spreadsheets and lightweight pipelines.

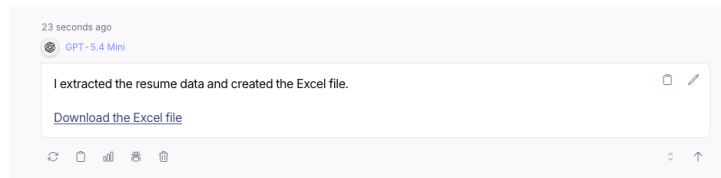


EXCEL FORMAT

WHEN THE INTERFACE CAN GENERATE FILES

Prompt

Read the resume PDF, extract: first name, last name, bachelor start year, bachelor end year.
Create an Excel file with one row and these 4 columns.
Return the file for download.



Key idea

Some interfaces go beyond text and wrap the model inside tools that can generate files directly.

FAILURE CASE 1

EXTRA TEXT BREAKS THE CONTRACT

Expected

```
{  
  "first_name": "Sarah",  
  "last_name": "Martin",  
  "bachelor_start": 2016,  
  "bachelor_end": 2019  
}
```

Typical bad output

```
Here is the extracted information:  
  
{  
  "first_name": "Sarah",  
  "last_name": "Martin",  
  "bachelor_start": 2016,  
  "bachelor_end": 2019  
}
```

Key idea

The content can be correct and still be unusable if the model adds extra text around it.

FAILURE CASE 2

RIGHT FIELDS, WRONG TYPES OR WRONG KEYS

Expected schema

```
{
  "first_name": string,
  "last_name": string,
  "bachelor_start": integer or null,
  "bachelor_end": integer or null
}
```

Typical bad output

```
{
  "first_name": "Sarah",
  "last_name": "Martin",
  "bachelor_start": "2016",
  "bachelor_end": "2019",
  "confidence": "high"
}
```

Key idea

A system often needs exact field names and exact types, not approximate structure.

FAILURE CASE 3

MISSING DATA, UNCERTAINTY, AND HALLUCINATED VALUES

Good behavior

```
{  
  "first_name": "Sarah",  
  "last_name": "Martin",  
  "bachelor_start": null,  
  "bachelor_end": null  
}
```

Typical bad output

```
{  
  "first_name": "Sarah",  
  "last_name": "Martin",  
  "bachelor_start": 2016,  
  "bachelor_end": 2019  
}
```

Key idea

When information is missing, the safest answer is often `null`, not a guess.

MINI-TP

FROM UNSTRUCTURED CVs TO DECISION SUPPORT

- ▶ The objective of this mini-TP is to practice controlling LLM outputs through a progressive sequence of tasks that start with free-form comparison and end with structured extraction and recommendation.
- ▶ Go fetch the CVs at the URL https://www.lamsade.dauphine.fr/~averine/certificat_IA/ or go on www.aleverine.com.

You will need an access to an LLM environment, either ChatGPT / Claude for a richer interface or OpenRouter for a more hands-on prompting experience.

- ▶ If you have access to ChatGPT or Claude, you can use the file upload feature to feed the CV PDFs directly into the conversation and ask for structured outputs.
- ▶ If not you can use OpenRouter with the following credentials (or your own if you have them):
 - A demander à l'enseignant

MINI-TP

STEP 1: EXTRACT SIMPLE FACTUAL INFORMATION

Task

- ▶ One line per candidate
- ▶ Extract: first name, last name
- ▶ Extract: location of current or last job

Prompt example:

Read all the CVs. For each candidate, extract only:

- first name
- last name
- location of current or last job

Return one candidate per line.

What to pay attention to

- ▶ Start with factual extraction
- ▶ Easy to verify manually
- ▶ Observe extraction errors early

Common pitfalls

- ▶ Confusing city of study with city of work
- ▶ Splitting first name / last name incorrectly
- ▶ Treating ambiguous mentions as facts

MINI-TP

STEP 2: SAME FACTS, BUT STRUCTURED OUTPUT

Task

- ▶ Same fields as Step 1
- ▶ One row per candidate
- ▶ Output format chosen by the student

Allowed targets:

- JSONL
- CSV
- Excel

Fields:

first_name, last_name, last_job_location

What changes

- ▶ Move from content to interface
- ▶ Compare formats for reuse
- ▶ Check formatting discipline

Compare the formats

- ▶ Easy to check manually
- ▶ Easy to reuse downstream
- ▶ Easy to keep consistent across candidates

MINI-TP

STEP 3: FIRST INFERENCE TASK — ESTIMATE THE AGE

Task

- ▶ Keep previous fields
- ▶ Add: estimated age
- ▶ Ask the model to justify or signal uncertainty

Add one field:

age

If age is not explicit, infer it from the CV and indicate uncertainty when needed.

What changes

- ▶ Introduce inference
- ▶ Distinguish facts from estimates
- ▶ Discuss confidence and uncertainty

What changes here

- ▶ We move from extraction to estimation
- ▶ Some answers should remain uncertain
- ▶ A wrong guess is worse than a clear low-confidence estimate

MINI-TP

STEP 4: INFER EXPERTISE, QUALITIES, AND WEAKNESSES

Task

- ▶ Infer the domain of expertise
- ▶ Extract 2 qualities
- ▶ Extract 2 weaknesses

For each candidate, infer:
- domain of expertise
- 2 qualities relevant to the job
- 2 weaknesses or missing elements

What to check

- ▶ Move toward qualitative judgment
- ▶ Force explicit criteria
- ▶ Observe variability across models

What to check

- ▶ Are the claims grounded in the CV?
- ▶ Are the weaknesses specific or generic?
- ▶ Do different models produce the same profile?

MINI-TP

STEP 5: FINAL HIRING RECOMMENDATION WITH CONFIDENCE

Final task

- ▶ Should the candidate be interviewed?
- ▶ Short justification
- ▶ Confidence level

Add three final fields:
- should_interview
- recommendation_why
- confidence_level

What matters now

- ▶ Build a full decision-support table
- ▶ Separate evidence from judgment
- ▶ Make uncertainty explicit

What matters now

- ▶ Separate evidence from recommendation
- ▶ Justify the interview decision briefly
- ▶ Keep the confidence level meaningful

MINI-TP

GLOBAL TASK AND PLATFORM-SPECIFIC PROMPTS

ChatGPT / Claude / Antigravity

Voici plusieurs CVs en PDF. Nous cherchons à recruter un commercial pour une entreprise d'informatique.

Commence par lire tous les CVs et construis un tableau de synthèse avec une ligne par candidat.

Le tableau doit contenir les colonnes suivantes : - Nom - Prénom - Lieu du poste actuel ou du dernier poste - Âge estimé - Domaine d'expertise - Deux qualités - Deux faiblesses - Faut-il interviewer le candidat ? - Justification brève - Niveau de confiance

Ensuite, crée un fichier Excel contenant ce tableau, avec une ligne par candidat, et vérifie qu'il est bien rempli.

OpenRouter

Voici plusieurs CVs en PDF. Nous cherchons à recruter un commercial pour une entreprise d'informatique.

Lis tous les CVs puis crée un tableau de synthèse avec une ligne par candidat et les colonnes suivantes : Nom, Prénom, Lieu du poste actuel ou du dernier poste, Âge estimé, Domaine d'expertise, Deux qualités, Deux faiblesses, Faut-il interviewer le candidat ?, Justification brève, Niveau de confiance.

Retourne ce tableau dans un format structuré au choix parmi JSONL, CSV ou tableau Markdown.

Si certaines informations ne sont pas explicitement présentes, indique clairement qu'elles sont estimées ou incertaines.

Key idea

The same business task must often be phrased differently depending on the platform and the available tooling.

AN ISOLATED LLM

TEXT IN, TEXT OUT, AND NOTHING ELSE

- ▶ A standalone LLM reads text in its context
- ▶ It generates text as output
- ▶ By itself, it does not browse, click, execute, or inspect files

Paradox

Earlier, we gave CV PDFs to a model and it extracted information from them. But we also said that an LLM only takes text as input and returns text as output. So what actually happened?

SO HOW DID WE DO IT?

THE INTERFACE TRANSFORMED FILES INTO MODEL-READABLE CONTEXT

What the user thinks

- ▶ “I uploaded PDFs”
- ▶ “The model read them”



What the system actually does

- ▶ Read the file
- ▶ Extract text or content
- ▶ Insert that content into the model context

— START OF FILE: cv_bernard.pdf — Claire Bernard Infirmière Diplômée d'État – Service Urgences Lille, France • +33 6 23 56 89 47 • claire.bernard@email.fr linkedin.com/in/clairebernard Profil Infirmière diplômée avec 8 ans d'expérience en service d'urgences et SMUR. Expertise en soins d'urgence, gestion de situations critiques et travail en équipe pluridisciplinaire. Reconnue pour mon sang-froid, mon empathie et ma capacité d'adaptation. Investie dans la formation continue et l'amélioration des pratiques de soins. Expérience Professionnelle Infirmière en Service d'Accueil des Urgences (SAU) Septembre 2019 – Présent CHU de Lille – Hôpital Roger Salengro Lille – 5 ans

⋮

Key idea

The interface or surrounding software converts files into text that the LLM can process.

OPENING THE MODEL TO THE OUTSIDE WORLD

USING TEXT TO REQUEST TOOLS

- ▶ The model can be prompted to ask for actions in text
- ▶ Those actions can target tools: web, calendar, files, email, code, etc.
- ▶ The tool result is then returned to the model as more text in context

User: What is on my calendar tomorrow?

Assistant draft / tool request: CALL_CALENDAR(date=tomorrow)

System result: 10:00 team meeting, 14:00 client call

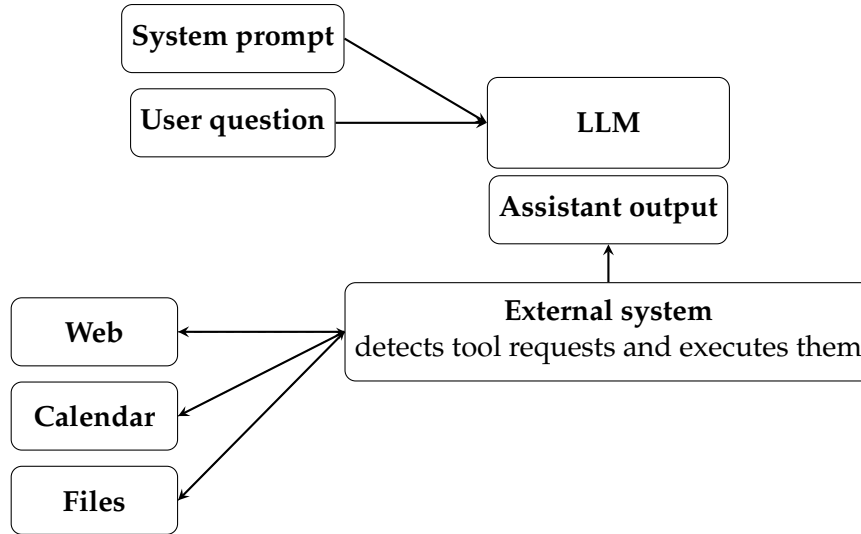
Assistant: Tomorrow you have a team meeting at 10:00 and a client call at 14:00.

Key idea

The model still writes text. The difference is that some of that text is interpreted as a request to use an external tool.

THE EXTERNAL SYSTEM MATTERS

THE LLM IS WRAPPED INSIDE A LARGER ARCHITECTURE



Key idea

The model is only one component. The real system detects tool-use requests, connects to the outside world, and feeds the results back.

LLM + TOOLS

VISIBLE TO THE USER

Question utilisateur

Quelle est la météo à Lyon aujourd'hui ?

Key idea

From the user point of view, this still looks like an ordinary chat question.

LLM + TOOLS

WHAT THE LLM SEES FIRST

System Prompt:

You are a helpful assistant. You have access to external tools.

Always reason step by step before answering.

Use the ReAct style: alternate between Thought, Action, and Observation.

Only provide the final natural language answer to the user at the end.

Available Tools:

1. `get_weather`

- Description: Returns the current weather for a given city.

- Input: object with required field "city" (string).

- Output: object with fields:

city (string)

temperature (number)

conditions (string)

User: Quelle est la météo à Lyon aujourd'hui ?

System side

- ▶ Feed the LLM with:
 - system prompt
 - tool descriptions
 - user query

REACT STYLE

THOUGHT, ACTION, OBSERVATION

ReAct loop

- ▶ **Thought:** what the model infers or plans
- ▶ **Action:** the tool request it emits
- ▶ **Observation:** the result returned by the system

Key idea

The model alternates between reasoning and acting instead of jumping directly to the final answer.

Thought: I need current weather data for Lyon.

Action: `get_weather{"city": "Lyon"}`

Observation: `{"city": "Lyon", "temperature": 21, "conditions": "partiellement nuageux"}`

Thought: I now have the information needed to answer in natural language.

Final Answer: À Lyon, il fait actuellement 21°C avec un ciel partiellement nuageux.

REACT STYLE

ANOTHER EXAMPLE: CALENDAR TOOL

ReAct loop

- ▶ **Thought:** I need calendar data
- ▶ **Action:** request the calendar tool
- ▶ **Observation:** get tomorrow's events

Key idea

The same ReAct pattern works with a different tool: the model reasons, requests a tool, receives an observation, then answers.

Thought: The user is asking what is on their calendar tomorrow. I need access to calendar events.

Action: `get_calendar{"date": "tomorrow"}`

Observation: `{"events": [{"time": "10:00", "title": "Team meeting"}, {"time": "14:00", "title": "Client call"}]}`

Thought: I now have tomorrow's events and can summarize them in natural language.

Final Answer: Tomorrow you have a team meeting at 10:00 and a client call at 14:00.

REACT STYLE

ANOTHER EXAMPLE: FILE SEARCH TOOL

ReAct loop

- ▶ **Thought:** I need the relevant file content
- ▶ **Action:** request the file-search tool
- ▶ **Observation:** get the extracted passage

Key idea

ReAct also applies when the model must inspect documents rather than query a live external service.

Thought: The user is asking about the candidate's last job location. I need to inspect the uploaded CV.

Action: `search_file{"query": "last job location", "file": "candidate_3.pdf"}`

Observation: `{"snippet": "Senior Account Manager, Toulouse, 2021-2024"}`

Thought: The CV states that the last job was Senior Account Manager in Toulouse.

Final Answer: The candidate's last job was located in Toulouse.

LLM + TOOLS

THE MODEL WRITES TEXT TO REQUEST A TOOL

System Prompt: You are a helpful assistant. You have access to external tools.

User: Quelle est la météo à Lyon aujourd'hui ?

Assistant:

Thought: The user is asking about the weather in Lyon today. I need current weather data.

I have access to a tool called `get_weather`.

The tool supports current weather, not future dates.

I will interpret this as current weather.

Action: `get_weather{"city": "Lyon"}`

External system

- ▶ Detect the action request
- ▶ Call the appropriate external tool
- ▶ Example backend action:
 - consult a weather service or web page
 - extract the result
 - return structured data

Key idea

The LLM still writes text. The surrounding system interprets some of that text as a request to use a tool.

LLM + TOOLS

OBSERVATION, THEN FINAL ANSWER

System Prompt: You are a helpful assistant. You have access to external tools.

User: Quelle est la météo à Lyon aujourd'hui ?

Assistant:

Thought: The user is asking about the weather in Lyon today. I need current weather data.

Action: `get_weather{"city": "Lyon"}`

Observation: `{"city": "Lyon", "temperature": 21, "conditions": "partiellement nuageux"}`

Thought: The tool indicates that the weather in Lyon is 21°C and partially cloudy. I can now answer the user in natural French.

Final Answer: À Lyon, il fait actuellement 21°C avec un ciel partiellement nuageux.

System side

- ▶ Execute the tool request
- ▶ Send the observation back into the context
- ▶ Copy only the final answer to the user-facing interface

Key idea

The external system closes the loop: it executes the tool, returns the observation, and shows only the final answer to the user.

MCP AND TOOL ECOSYSTEMS

STANDARDIZING ACCESS TO EXTERNAL TOOLS

Without MCP

- ▶ One custom connector per tool
- ▶ Different formats and conventions
- ▶ More glue code to maintain

LLM app A → custom Google Drive adapter
LLM app A → custom Slack adapter
LLM app A → custom Calendar adapter
LLM app B must rebuild everything again

With MCP

- ▶ One shared protocol layer
- ▶ Tools exposed in a standard way
- ▶ Easier portability across apps

LLM app A → MCP client → MCP servers

- Google Drive
- Slack
- Calendar

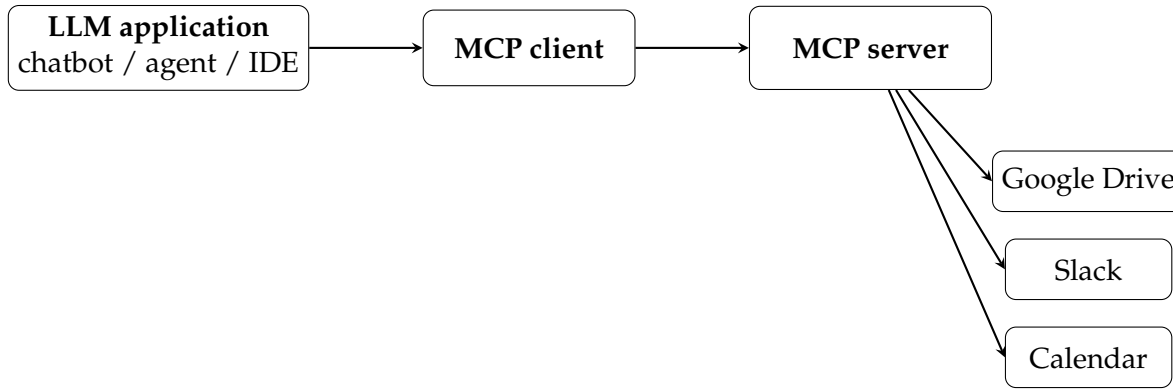
LLM app B can reuse the same ecosystem

Key idea

MCP is a protocol layer that standardizes how tools are exposed to models, so the ecosystem becomes easier to connect, reuse, and scale.

MCP IN PRACTICE

A SIMPLE MENTAL MODEL



Takeaway

The LLM application speaks to an MCP client, the client speaks a standard protocol to MCP servers, and those servers expose concrete tools or data sources.

MCP EXAMPLES

CONCRETE USE CASES

Example 1

Task:

"Find the latest sales deck in Drive and summarize the pricing slide."

MCP tools:

Drive listing
Document retrieval

Example 2

Task:

"Check my calendar, then draft an email proposing two meeting slots."

MCP tools:

Calendar access
Mail drafting

Example 3

Task:

"Search Slack for the latest incident update and compare it with the runbook."

MCP tools:

Slack search
File / doc access

Why it matters

These workflows are not about one magic model. They are about a model embedded in a standardized tool ecosystem.

MEMORY IN LLM SYSTEMS

NOT MEMORY INSIDE THE WEIGHTS, BUT MEMORY IN THE SYSTEM

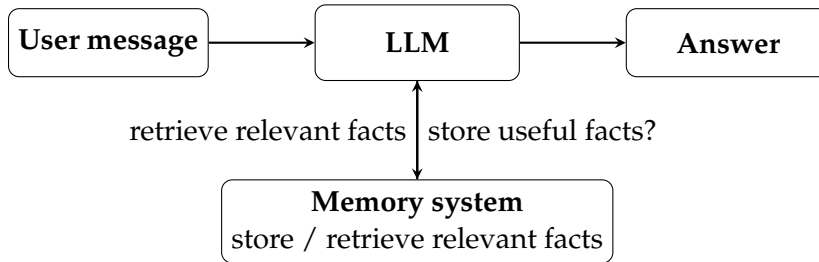
- ▶ A single chat context is limited.
- ▶ Useful assistants often need to remember information across turns or across sessions.
- ▶ In practice, this memory is usually implemented by the surrounding system, not by changing the model itself.

Key idea

When people say that an LLM has memory, they often mean that a system stores information and re-injects it into the model context later.

A SIMPLE MEMORY LOOP

STORE, RETRIEVE, REINSERT



Read it like this

The system decides what to store, what to retrieve later, and what to put back into the prompt before the model answers.

RAG

RETRIEVAL-AUGMENTED GENERATION

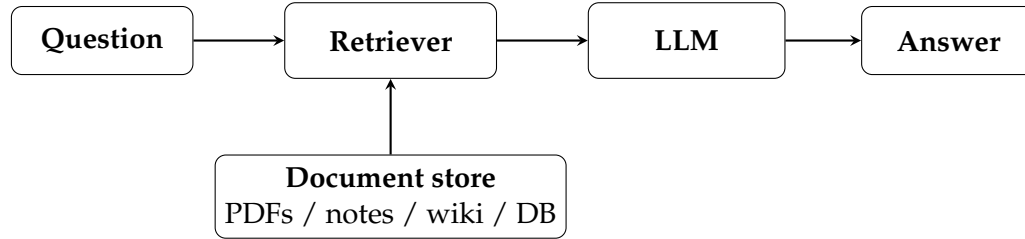
- ▶ RAG means retrieving external information before generation.
- ▶ The retrieved passages are inserted into the prompt or context.
- ▶ The model then answers using both the user question and the retrieved material.

Key idea

RAG does not give the model new knowledge directly. It gives the system a way to fetch relevant documents and place them into context at inference time.

RAG IN PRACTICE

QUESTION, RETRIEVAL, GENERATION



Example

- ▶ User: “What does our onboarding policy say about remote work?”
- ▶ Retriever finds the relevant policy passage.
- ▶ The passage is inserted into context.
- ▶ The model answers using that passage.

Why it matters

RAG is one of the main ways to connect LLMs to private or changing knowledge without retraining them.

MEMORY VS RAG

TWO RELATED BUT DIFFERENT IDEAS

Memory

- ▶ Store useful facts about the user or the interaction
- ▶ Reuse them later across turns or sessions
- ▶ Example: “The user prefers CSV outputs”

Key idea

Both rely on the same basic move: fetch useful information from outside the model, then inject it back into context.

RAG

- ▶ Retrieve relevant external documents for a question
- ▶ Reuse them at answer time
- ▶ Example: policy docs, CVs, product catalogs, support articles

EXAMPLES

MEMORY AND RAG IN ENTERPRISE ASSISTANTS

Memory example

User fact:

“I prefer bullet points and CSV outputs.”

The assistant reuses this preference in later tasks.

RAG example 1

Task:

“Search the uploaded CVs and extract the last job location of each candidate.”

The system retrieves the relevant passages from each PDF.

RAG example 2

Task:

“What does the company travel policy say about hotel reimbursement?”

The system retrieves the right policy section before answering.

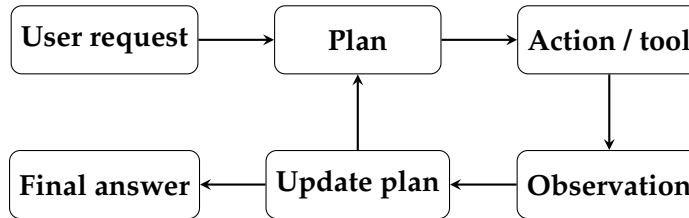
Key message

Once again, the useful behavior comes from the LLM plus the surrounding system: storage, retrieval, and prompt construction.

A TYPICAL AGENT LOOP

PLAN, ACT, OBSERVE, UPDATE, ANSWER

- ▶ Interpret the user request
- ▶ Build or update a plan
- ▶ Select an action or tool
- ▶ Read the observation returned by the system
- ▶ Update the plan, continue, or stop
- ▶ Return the final answer or artifact



Key idea

An agent is usually not one single generation. It is a control loop: plan, act, observe, and revise until the task is complete.

CLAUDE CODE AS AN AGENT

A CONCRETE MODERN EXAMPLE

- ▶ Claude Code is not just a chatbot for code.
- ▶ It is prompted as an interactive software engineering agent.
- ▶ The prompt defines behavior, planning rules, tool usage, and task-management expectations.

Why this example matters

It makes the agent loop concrete: plan the work, inspect the environment, use tools, update progress, verify results, and only then answer.

CLAUDE CODE PROMPT

THE AGENT IS EXPLICITLY TOLD HOW TO BEHAVE

Claude Code Initial Prompt (simplified)

You are an interactive agent that helps users. Use the instructions below and the tools available to you to assist the user.

Proactiveness

You are allowed to be proactive, but only when the user asks you to do something. Find balance between:

- Doing the right thing when asked
- Not surprising the user with actions you take without asking

Task Management

You have access to the `TodoWrite` tool... Use this tool VERY frequently to track tasks and give the user visibility into progress.

What to notice

- ▶ The prompt defines allowed initiative
- ▶ The prompt imposes planning behavior
- ▶ The prompt makes task tracking part of the agent contract

CLAUDE CODE PROMPT

PLANNING IS NOT OPTIONAL, IT IS PART OF THE WORKFLOW

Example from the prompt

user: Help me write a new feature in my code that allows users to track their usage metrics

assistant: Let me first use the `TodoWrite` tool to write a todo list to plan this task.

1. Research existing metrics tracking in the codebase
2. Design the metrics collection system
3. Implement core metrics tracking functionality
4. Create export functionality for different formats

This echoes what we said before

- ▶ Complex tasks are decomposed
- ▶ Planning is externalized
- ▶ The system can expose intermediate progress

Agent loop view

- ▶ plan first
- ▶ act on one subtask
- ▶ observe results
- ▶ update the todo list

CLAUDE CODE PROMPT

SEARCH, ACT, VERIFY

Doing tasks

The user will primarily request software engineering tasks...

Recommended steps:

- Use the `TodoWrite` tool to plan the task if required
- Use the available search tools to understand the codebase and the user's query
- Verify the solution if possible with tests
- Never assume the test framework; inspect the codebase first

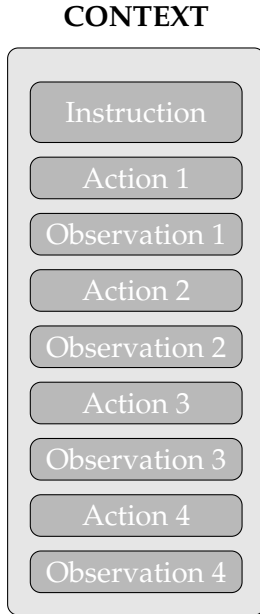
What this shows

A complex agent is usually a bundle of loops: planning, retrieval, tool use, verification, and final synthesis.

Source: <https://github.com/Yuyz0112/claude-code-reverse>

AGENT LOOP AND GROWING CONTEXT

EACH STEP IS APPENDED TO THE CONTEXT



- ▶ In an agent loop, the context grows step by step.
- ▶ Instructions, tool calls, and observations accumulate.
- ▶ The model must keep reasoning while reading an ever longer context.

Question

If the context keeps growing, what happens when the important instruction is buried under too many actions and observations?

CONTEXT ROTTING IN AGENTS

LOOPING INCREASES THE RISK

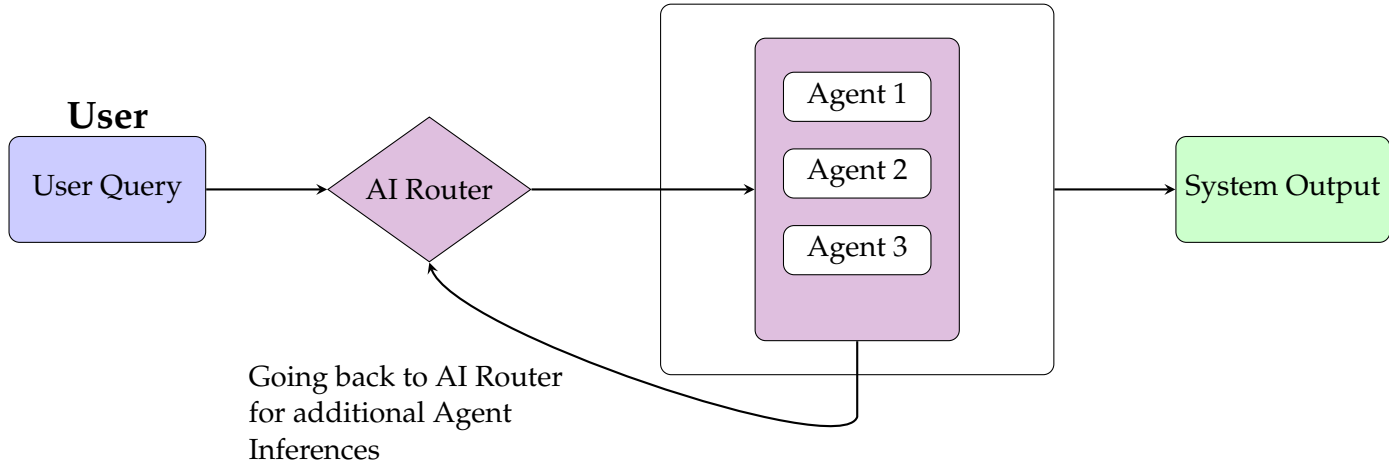
- ▶ The more an agent loops, the more text it accumulates.
- ▶ Long traces can dilute the original instruction and goal.
- ▶ This can hurt focus, consistency, and reliability.

Key idea

Agent loops make context rotting more likely, because each action and observation is added to the same running context.

AGENT ROUTING

SPLIT THE WORK INSTEAD OF OVERLOADING ONE CONTEXT



Key idea

One response to context rotting is to route the task: instead of one overloaded agent, a router sends subtasks to more specialized agents.

CONCLUSION

WHAT TO RETAIN FROM THIS FIRST WORKSHOP

- ▶ A LLM is, at its core, a next-token predictor working on text in context.
- ▶ Prompting changes behavior because the model only sees tokens, not intentions.
- ▶ Output formatting matters because useful systems need predictable, reusable outputs.
- ▶ Agentic behavior comes from the LLM plus an external system: tools, memory, retrieval, planning, and loops.
- ▶ Once loops become longer, context management becomes a central engineering problem.

Key message

A useful AI system is not just a model. It is a model embedded in a carefully designed interface and orchestration layer.

NEXT WEEK

FROM CONCEPTS TO THE AGENT LANDSCAPE AND HANDS-ON SYSTEMS

What we will do next

- ▶ Compare existing agent systems
- ▶ Go from no-code platforms to developer-oriented tools
- ▶ Discuss which systems are suited to which business needs

Transition

Today was about understanding how LLM-based systems work. Next week will be about comparing what already exists before building your own.

Then in the following workshops

- ▶ Build simple no-code agents
- ▶ Build a more advanced coded agent
- ▶ Reuse the ideas from today: prompting, tools, memory, routing, and evaluation