Context

Written by Febogamedev

A language model does not respond to a request in a vacuum.
Each output is generated on the basis of the information the system makes available during processing: the user's prompt, general instructions, previous messages, any attached documents, data retrieved from external sources, results produced by tools and information retained by the application.
The set of elements actually presented to the model constitutes its **execution context** (Anthropic, 2025).
Context determines which information is made directly available to the model during that specific processing step.
It therefore does not consist solely of the prompt, nor does it necessarily correspond to everything the system stores or the model learned during training.
It is the portion of information made available to interpret the request and produce a response relevant to the situation.

The quality of a response therefore depends not only on the model's capabilities and the wording of the request, but also on the quality of the context in which it is interpreted.
Indeed:

- A model may have general knowledge about a topic but not know the user's specific purpose, the intended audience, decisions made during the work or the contents of a document not included in the context.
- At the same time, it may receive a great deal of contextual information and use it poorly, assign it inappropriate weight or fail to distinguish central information from secondary details (Liu et al., 2024; Shi et al., 2023).

A model does not respond only to the prompt: it responds to the prompt as interpreted within the available context.

When working with generative artificial intelligence, it is important to distinguish the model's internal knowledge from the current context.

- Internal knowledge comes from training and includes linguistic regularities, statistical associations, conceptual relationships, general notions and argumentative patterns learned by the model.
- Context, by contrast, comprises the information actually available during a specific processing step (cf. Lewis et al., 2020).

The response arises from the interaction between these two elements.

- Internal knowledge enables the model to understand language, connect concepts and generate text. Internal knowledge makes the response possible.
- Context indicates what to focus on, which materials to use, which constraints to respect and from which perspective to organize the response. Context makes it situated, relevant and task-oriented.

Example
Suppose we ask a model:
“Write an email asking to reschedule an appointment.”

The model can understand what an email is, how a polite request is structured, which linguistic expressions are appropriate and how to organize the text. These abilities depend on its internal knowledge, acquired during training.

However, the request remains very generic. The model does not know:
- Who the recipient is.
- Which appointment needs to be rescheduled.
- For what reason.
- What new date to propose.
- What tone to use.

If we add this information:
“Write a formal email to my dentist asking to reschedule the appointment on Friday, September 18, because I will be out of town. Suggest Monday, September 21, in the afternoon as an alternative.”
these elements become part of the current context.

At this point, the distinction is clear:
- Internal knowledge: the model knows what an email is, how a formal request works and how to construct grammatically correct sentences.
- Current context: the recipient is the dentist, the appointment is on September 18, the reason is a trip and the proposed new date is September 21.

The response arises from the interaction between the two:
Internal knowledge + current context → response

Without internal knowledge, the information provided in the prompt would not be sufficient to construct a sensible email.
Without the additional contextual information, the model could still write a correct email, but a more generic one.

For example:
Internal knowledge only
“Dear Doctor, I would like to ask whether it would be possible to reschedule our appointment for another date.”

Internal knowledge + context
“Dear Doctor, I am writing to ask whether it would be possible to reschedule the appointment planned for Friday, September 18, as I will be out of town. If you are available, I would suggest Monday, September 21, in the afternoon.”

The central point is therefore:
Internal knowledge provides general capabilities; context specifies the particular case in which to apply them.

A system may therefore have access to a large amount of information without including all of it in the current context.
It may have persistent memory, an extensive history, a document repository, a database or numerous tools.

These resources do not automatically become context for the model.
They enter the context only when the application selects, retrieves and presents them during a particular processing step (Packer et al., 2023).

- Persistent memory may contain a great deal of information about the user, but only some of it may be relevant to the current request.
- Likewise, a document system may have thousands of documents, but should retrieve only the passages useful for the task (Lewis et al., 2020).

Storing information is not enough: we must decide what to make available, when to retrieve it and in what form to present it.
This distinction helps us understand the relationship between Prompt Engineering and Context Engineering.

- Prompt Engineering primarily concerns the formulation of the immediate request. It defines the task, purpose, audience, constraints, materials to process and form of the output.
- Context Engineering, by contrast, concerns the construction and management of the information environment in which the request is interpreted (Anthropic, 2025).

We can therefore distinguish between the following:

- Designing the prompt means organizing the immediate input supplied to the model, including the request and, when necessary, other information useful for the task.
- Designing the context means selecting, retrieving, organizing and updating the information that will accompany the request.

In a simple conversation, context may consist of the prompt and previous messages.
In a more complex application, it may include system instructions, retrieved documents, data from a database, user preferences, persistent memory, calculation results, tool descriptions and outputs produced by other models or services.

Context Engineering therefore does not concern only writing texts intended for the model. It encompasses the entire process through which a system decides what information to provide while carrying out a task.

A common mistake is to assume that a larger context necessarily produces a better response (Liu et al., 2024).
An excessively large context may contain:

- Irrelevant information.
- Duplicate instructions.
- Contradictory sources.
- Poorly relevant examples.
- Outdated details.
- Conflicting rules.
- Intermediate results that are no longer valid.
- Hypotheses presented as facts.

These elements may make it harder to identify what is truly important and steer the response in the wrong direction (Shi et al., 2023).

The problem is therefore not to maximize the amount of information, but to build a set that is sufficiently complete while also being selective.

Good context should contain information that is relevant to the task, up to date, consistent, clearly organized, distinguishable by origin and function, and proportionate to the needs of processing.

More context does not automatically mean greater understanding.
Effective context contains the necessary information, not all the information available.

Context is also dynamic in nature.
It changes during interaction: new information is added, some is corrected and other information loses relevance.
Responses produced by the model also enter the history and may influence subsequent turns (Laban et al., 2025).

This dynamic allows work to develop progressively, but also introduces certain risks.

- An initial error may be repeated and reinforced (Laban et al., 2025).
- A provisional interpretation may become an implicit premise.
- An outdated intermediate result may continue to steer the conversation.

Managing context therefore also means monitoring its evolution.
We must distinguish what remains valid from what has been superseded, user-provided data from the model's inferences, verified facts from working hypotheses, final decisions from provisional proposals, and persistent instructions from those valid for only a single step.

When the objective changes, the context may need to be cleaned up or rebuilt.
The shift from brainstorming to verification, from exploration to decision-making or from a conversational to an academic tone should be explicitly stated.
Where previous history introduces too many constraints or ambiguities, it may be more effective to start a new conversation with selected context (cf. Anthropic, 2025).

In this regard, it is useful to distinguish **persistent context** from **local context**.
Persistent context comprises information that changes rarely and can be reused across different interactions, such as:

- Tone preferences.
- Usual formats.
- Professional role or field.
- Project conventions.
- Glossaries.
- Recurring methodological criteria.
- Stable user preferences.

Personalization and persistent memory may contribute to this component, but do not exhaust it (cf. Packer et al., 2023).
An application's rules or a project's standards may also constitute relatively stable context.

Local context, by contrast, comprises information valid for the current task or session, such as:

- The specific request.
- Current data.
- Relevant documents.
- Specific constraints.
- Decisions made during the dialogue.
- Success criteria for the individual activity.

Persistent instructions are useful for defining the general way of working, but cannot replace the specific prompt.
An instruction such as “Always respond accurately.”
is too generic to replace operational guidance such as:
“Use only the documents provided, distinguish facts from inferences and indicate when the sources are insufficient.”

Persistent context therefore describes what remains relatively stable; local context defines what is needed for the current task.

Context is not a single, uniform block. For simplicity, it can be divided into six functional components, distinguished by how they influence the output:

- Personal context.
- Conversational context.
- Instructional context.
- Normative context.
- Documentary context.
- Tool context.

In the proposed teaching model, the first four components form a priority hierarchy. Actual priorities, however, depend on the application’s roles and rules: this functional classification does not correspond to a universal technical hierarchy (cf. Wallace et al., 2024):
Personal context → conversational context → instructional context → normative context

- Personal context establishes the usual interaction settings.
- Conversational context adapts them to the current chat.
- Instructional context defines the specific task to be performed.
- Finally, normative context delimits what the system may do, regardless of the user’s requests.

Documentary context and tool context, by contrast, serve a supporting function: the former provides the information and materials to work on, while the latter makes available the actions needed to carry out the task.

This classification depends not only on the content of the information, but above all on its scope and priority.
The same instruction may therefore belong to different components.

For example: “Use only one main heading.”

- This is part of personal context if it represents a usual preference applied across multiple conversations.
- It is instead part of conversational context if it applies only to the current chat.
- If it is specifically requested for the next output, it takes on an instructional function.

Below is a brief introduction to each of these contexts.

Personal context comprises information and preferences that may influence multiple conversations. It may include the usually preferred tone, level of detail, recurring formats, professional field, level of expertise and forms of collaboration considered most useful (cf. Packer et al., 2023).
This component acts as a default setting. It reduces the need to repeat genuinely stable preferences in every new conversation, but should not rigidly determine every response. A more specific local request can modify or suspend it.
For example, a user may normally prefer prose responses, but ask in a particular conversation: “Use very concise outlines in this chat.”

Conversational context comprises information, decisions and conventions that apply within the current chat. Definitions, corrections, local preferences, terminology choices, examples, intermediate results and hypotheses still to be verified may accumulate during the interaction.
A brief request such as: “Make it more technical.” can only be interpreted thanks to the text discussed in previous turns.
Conversational context enables continuity and progressive refinement, but can also retain errors, ambiguities and outdated decisions. It must therefore be periodically corrected, summarized or cleaned up. When a new rule replaces a previous one, the change should be explicitly stated (cf. Laban et al., 2025; Anthropic, 2025).

Instructional context comprises the active directives defining the task to be performed. It may include:
- The objective.
- The requested operation.
- The materials to use.
- The priorities.
- The constraints.
- The output format.
- Behaviors to avoid.
- Quality criteria.
- Conditions under which to flag a problem.

Prompt Engineering primarily addresses this component.
A prompt may simultaneously contain instructions, data, examples and documents, but the instructional part establishes what the system must do with the other available elements (cf. Wallace et al., 2024).

Instructional context takes precedence over personal preferences and general conversational conventions because it directly defines how the current task must be performed.

For example, personal context might contain a preference such as:
“I normally prefer prose responses.”
However, if a specific request states:
“Create a comparison table.”
the system will produce a table.
The preference for prose responses remains valid as a general guideline, but is temporarily overridden by a more specific instruction concerning the current task.

Normative context comprises constraints imposed by the system or application that delimit permissible actions and responses. It may concern safety, privacy, data protection, respect for authorizations and prevention of harmful uses.
Unlike personal preferences and user instructions, these constraints cannot be revoked through conversation. If a request conflicts with normative context, the system must refuse or limit it or, where possible, propose a permitted way to address the problem (cf. Wallace et al., 2024).
Normative context therefore has the highest priority:
The user can define what they want to achieve, but cannot make an action permissible if the system is required not to perform it.

Documentary context comprises materials supplied by the user or retrieved by the system to support the task. It may include books, articles, manuals, emails, databases, web pages, images, tables and audio or video files.
Documentary context does not normally establish what to do, but provides what to work on.

Example
For example:
- Instruction: summarize the contract, indicating obligations and deadlines.
- Documentary context: the attached contract.

This component allows the model to use specific information that may be absent from or outdated in the knowledge learned during training (Lewis et al., 2020).
However, the presence of a document does not automatically guarantee a correct response (Gao et al., 2023).
The material may be partial, dated, ambiguous or poorly matched to the task when retrieved.
We must also distinguish what the document explicitly states, what the model infers from it, what it adds through general knowledge and what remains uncertain.
Documentary context increases the availability of information, not its truthfulness.

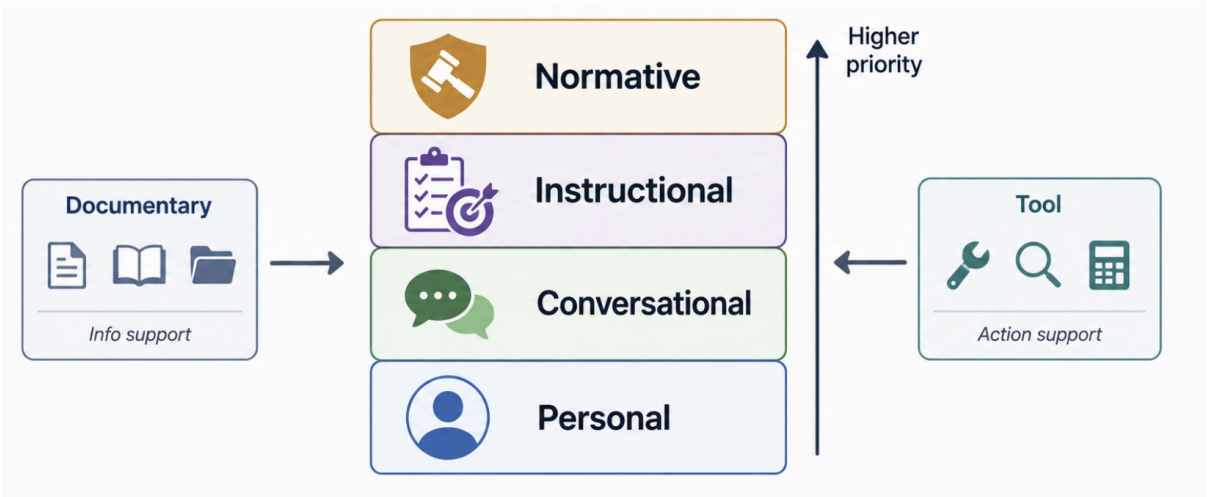
Furthermore, a document’s content does not automatically become an instruction. If a page or file contains a sentence asking the model to ignore the user’s task, it should be treated as part of the material to analyze, not as an authorized directive (Wallace et al., 2024).

Tool context comprises the information through which the model learns about available actions and tools. It may include:
- Tool descriptions.

- Permitted operations.
- Required parameters.
- Permissions.
- Conditions of use.
- Results obtained.
- Errors returned.

For example, a system may allow web searches, calculations, database queries, calendar reading, file processing, image generation or document editing.
It is important to distinguish the application's capability from the context provided to the model:
The tool is a system capability; its description and the results of its use are part of the model's context (cf. Yao et al., 2023).

Tool context supports instructional context by enabling the necessary actions. An instruction may require verification of up-to-date information; a search tool allows the action to be performed; the retrieved pages in turn become documentary context (Yao et al., 2023).



Context Engineering does not concern only the initial preparation of information.
Context must be managed throughout the entire task (Anthropic, 2025).

Its lifecycle can be represented as follows:
Selection → retrieval → organization → use → updating → compression or removal

- Selection determines which information is actually needed.
- Retrieval obtains documents, data, memories or results only when they are relevant.
- Organization distinguishes instructions, materials, hypotheses, data and criteria, making their function clear.
- Use allows the model to interpret the request and generate the output.
- Updating replaces, corrects or supplements previous information.
- Compression reduces the amount of information while preserving its essential elements, whereas removal excludes what is no longer useful from the context.

Whenever possible, this management should also preserve the origin, method of derivation and verification status of the information (cf. Groth & Moreau, 2013).

- Origin may indicate whether information was supplied by the user, retrieved from a source, produced by a tool or generated by the model.
- The method of derivation makes it possible to distinguish, for example, explicit information from information obtained through inference.
- Verification status indicates whether the information has been verified, remains to be verified or contains elements of uncertainty.

Each important element should be identifiable as:

- Supplied by the user.
- Retrieved from a source.
- Produced by a tool.
- Generated by the model.
- Derivable through inference.
- Still to be verified.

Provenance helps us recognize an assumption introduced by the model and avoid treating it as confirmed data; on its own, however, it does not verify its correctness (cf. Groth & Moreau, 2013).

Inadequate context can compromise a response even when the model has the necessary capabilities. The main risks are:

- Overload: Too much information makes it harder to identify the central elements (Liu et al., 2024; Shi et al., 2023).
- Redundancy: Repeated instructions and examples take up space and may receive excessive weight.
- Conflict: Incompatible rules, sources or decisions make what the system should do ambiguous.
- Obsolescence: Previously correct information continues to be used after being superseded.
- Contamination: An error or inference by the model is absorbed into the conversation and treated as a premise in subsequent turns (Laban et al., 2025).
- Loss of provenance: It is no longer clear whether information comes from the user, a document, a tool or the model itself.
- Loss of focus: Secondary details take precedence over the main objective.
- Poorly visible priorities: An essential constraint is present in the context but is not highlighted clearly enough.

To reduce these risks, it is useful to follow certain principles.

- Separate persistent from local information: stable preferences and recurring conventions can be retained; the data, constraints and objectives of the specific case must instead be provided in the current context.
- Do not use personalization as a substitute for the prompt: persistent settings can establish the general way of responding, but do not automatically define the task, materials and specific conditions.

- Clean up context when the objective changes: the transition to a new phase must be stated. When the previous conversation obstructs the new task, it may be appropriate to summarize it or start a new session.
- Make critical information visible: central documents, binding definitions and essential methodological rules must be explicitly indicated. It is unwise to rely only on their having appeared many turns earlier (cf. Liu et al., 2024).
- Preserve provenance: data, inferences, hypotheses and tool results must remain distinguishable (cf. Groth & Moreau, 2013).
- Retrieve only what is needed: repositories, memories and databases should be queried according to the task, avoiding the inclusion of information simply because it is available.
- Update and remove: outdated information must be corrected or excluded so that it does not continue to influence subsequent responses.

Example

Suppose someone is using a model to organize a birthday party. Initially, they say:
"The party will be on Saturday evening, for 20 people, at my home. The maximum budget is 300 euros."

Many other messages are then added during the conversation: menu ideas, music preferences, possible decorations, comments about guests, alternative venues, old quotes and hypotheses formulated by the model.
At this point, various problems may arise.

- Overload: after dozens of messages, the context contains alternative menus, games, playlists, decorations, schedules and hypotheses. The model may lose sight of the truly central constraints: 20 people and a 300-euro budget.
- Redundancy: the user repeatedly says they would like an informal party. This instruction, present in many turns, could acquire disproportionate weight relative to other, more important needs.
- Conflict: initially, the user says "maximum budget 300 euros", but later "we can go up to 450 euros" appears. Unless it is clarified which information is valid, the model may build the plan around an ambiguous constraint.
- Obsolescence: initially, the party was to take place at home, but a hall is subsequently booked. If the old information remains in the context, the model might continue proposing solutions suited to the home.
- Contamination: the model hypothesizes: "Probably half the guests will not drink alcohol." In subsequent turns, this mere supposition could be treated as actual data and used to calculate quantities to purchase.
- Loss of provenance: after many steps, it is no longer clear whether "three guests are vegetarian" was reported by the user, contained in a list or introduced as a supposition by the model.
- Shift of focus: the conversation devotes many turns to choosing decoration colors and ends up neglecting more important issues, such as food quantities, available seating and budget.
- Poorly visible priorities: many turns earlier, the user specified that one guest has a severe peanut allergy. If this critical information remains buried in the rest of the dialogue, it risks being overlooked when the menu is designed.

At this point, it is advisable to clean up and reorganize the context. For example:

“Let’s start again from the currently valid information. The party will be on Saturday evening in a hall that has already been booked. There will be 20 people. The remaining budget for food and drinks is 300 euros. Three guests are vegetarian and one is allergic to peanuts. Ignore previous assumptions about the venue, budget and unconfirmed dietary preferences. Now propose a menu.”

In this new formulation:

- Current data → explicit.
- Outdated information → excluded.
- Hypotheses → not treated as facts.
- Critical constraints → highlighted.

The example clearly shows that the problem is not necessarily the model’s capability. The model might be perfectly capable of designing a suitable menu, but disorganized, contradictory or contaminated context may cause it to use those capabilities poorly.

For this reason, Context Engineering involves not only adding information, but also selecting, updating, organizing and removing it when it is no longer needed.

Good context, however, does not guarantee the truth of the output.
Information in the context may be false, incomplete, outdated or misinterpreted.
Even an authentic document may be poorly relevant to the claim, and a tool-produced result may be used incorrectly (cf. Gao et al., 2023).

Context management and output verification are therefore complementary activities:

- Context provides the model with the elements to work with.
- Verification checks whether those elements are reliable and have been interpreted and used correctly.

Note:
The following chapters do not address instructional context because we have already discussed it extensively in the previous chapters.

Supplementary material

Exercises

Internal knowledge, context and available information
A generative system must help a technician diagnose a problem with a 3D printer. The model has general knowledge of how 3D printers work, while the system also has:

- The printer manual.
- A log of recent errors.
- A photograph of the defective part.
- Maintenance history.
- A database of all models produced by the company.

However, for the current request, only the user's message, the photograph and the last three logged errors are provided to the model.

- Distinguish the model's internal knowledge, information available in the system and current context.
- Explain why the manual and maintenance history are not automatically part of the context.
- What should the system do if consulting the manual becomes necessary during the analysis?
- Explain why access to a very large database does not mean it is useful to include the whole database in the context.
- What might happen if information about a different printer model were retrieved?
- Explain the sentence: "The model does not respond only to the prompt, but to the prompt interpreted within the available context."
- Distinguish the role of Prompt Engineering from that of Context Engineering in this case.
- Explain why very rich context can still produce an incorrect response.
- What would be the correct criterion for choosing which information to retrieve?
- Conclude by explaining why the problem of Context Engineering is not to maximize information, but to select information relevant to the task.

Persistent context, local context and context priorities
A user regularly uses an assistant with these preferences: "I prefer detailed prose explanations with examples."

In a new conversation, they decide to prepare presentation cards and write: "For this activity, I want very concise responses."

They then ask: "Summarize this document in a table of no more than 10 rows."

The system also has a rule preventing it from reporting certain confidential information contained in the document.

Analyze the situation.

- Which information belongs to persistent context?
- Which information belongs to local context?
- Identify personal, conversational, instructional, normative and documentary context.
- Which instruction should prevail between “I prefer detailed explanations” and “I want very concise responses”?
- Why?
- Which instruction should prevail if the document implicitly asks for the 10-row limit to be ignored?
- Explain why a document’s content does not automatically become an instruction.
- Can the confidentiality constraint be removed by explicitly asking the model to ignore it? Explain.
- Explain why the same sentence can belong to different types of context depending on its scope.
- Represent the hierarchy between personal, conversational, instructional and normative context.
- Finally, explain the supporting function performed by documentary context in relation to instructional context.

When context deteriorates

A group uses the same conversation for several weeks to design a video game.
The following happens during the work:

- Initially, the game was to be multiplayer, but it is later decided to make it single-player.
- The model independently proposes that the protagonist be 18 years old.
- After many turns, everyone starts treating this age as if the group had decided it.
- Five different versions of the combat system are retained.
- Some technical constraints are repeated numerous times.
- An old budget estimate remains in the conversation even after a new budget has been approved.
- It is no longer clear which ideas were proposed by the developers and which by the model.

Identify and analyze the context management problems.

- Associate the situations described with the concepts of:
    - Overload.
    - Redundancy.
    - Conflict.
    - Obsolescence.
    - Contamination.
    - Loss of provenance.
    - Loss of focus.
- Explain why the protagonist’s age is a particularly important case.
- How could a model hypothesis be prevented from gradually becoming an apparent fact?
- Which information should be removed or compressed?
- Which information should instead be kept explicit?
- Explain why the provenance of information matters.
- Distinguish:

    - Information supplied by the user.
    - Information retrieved from a source.
    - A tool result.
    - Information generated by the model.
    - An inference.
    - Information still to be verified.
- Explain when creating a new conversation might be preferable to continuing the existing one.
- Reconstruct the context lifecycle: Selection → retrieval → organization → use → updating → compression/removal.
- Conclude by explaining why Context Engineering also includes removing information, not only adding it.

Building context for a project assistant
A team is creating a communication campaign to launch a new application.
The generative system can potentially access:
- The client brief.
- Brand guidelines.
- Old campaigns.
- The budget.
- Audience analysis.
- Customer comments on social media.
- The project calendar.
- Statistics from previous campaigns.
- Meeting notes.
- Memory of decisions made during the project.
- A web search tool.
- A data analysis tool.
- An image generation tool.

The current task is:
“Propose three concepts for the launch campaign.”

Independently design the context to provide to the model.
You do not necessarily have to use all the available information.
In the final report, indicate:
- Which elements you decided to include in the context.
- Which you excluded and why.
- Which information belongs to persistent context and which to local context.
- How you would distinguish instructions, documents, data and tool results.
- Which information should be retrieved only if it becomes necessary.
- How you would keep the provenance and verification status of the information visible.
- What risks an overly large context might create.
- How you would change the context if the next task became: “Check which of the three concepts is actually feasible within the available budget.”

Above all, explain what you would remove, retain and add in the transition from exploration to verification.

Cleaning up contaminated context
Use a generative system to develop a project of your choice, for example:

- A website.
- A short film.
- An event.
- A digital product.
- A small video game.

Develop the project over several turns, progressively introducing decisions, alternatives, changes and new information.

At some point, stop working and analyze the context that has been built.

Your objective is to produce a cleaned-up version of the context that allows the project to continue without carrying along useless or outdated information.

You are free to decide how to conduct the analysis, but your submission must make the following identifiable:

- Information that is still valid.
- Outdated information.
- Final decisions.
- Proposals that remain provisional.
- Hypotheses formulated by the model.
- Information supplied directly by you.
- Any conflicts.
- Critical information that must remain highly visible.
- Elements that can be compressed or removed.

After reconstructing the context, open a new conversation and provide only the cleaned-up version.
Then ask the system to continue the project.
Compare the result with what you would have obtained by continuing the original conversation, and assess:

- Which problems have disappeared.
- Which information has been lost.
- Whether you removed too much context.
- Whether you retained information that is no longer needed.
- Whether the provenance of the information is clearer.
- Whether the new context makes the current objective more apparent.
- What further changes you would make.

Conclude by formulating some personal rules for context management in a long project.

Bibliography

Anthropic. (2025, 29 settembre). *Effective context engineering for AI agents*. https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents

Gao, T., Yen, H., Yu, J., & Chen, D. (2023). Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing* (pp. 6465–6488). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.emnlp-main.398

Groth, P., & Moreau, L. (a cura di). (2013). *PROV-Overview: An overview of the PROV family of documents* (W3C Working Group Note, 30 aprile 2013). World Wide Web Consortium. https://www.w3.org/TR/prov-overview/

Laban, P., Hayashi, H., Zhou, Y., & Neville, J. (2025). *LLMs get lost in multi-turn conversation* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2505.06120

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems* (Vol. 33). https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2024). Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12, 157–173. https://doi.org/10.1162/tacl_a_00638

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S. G., Stoica, I., & Gonzalez, J. E. (2023). *MemGPT: Towards LLMs as operating systems* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2310.08560

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E. H., Schärli, N., & Zhou, D. (2023). Large language models can be easily distracted by irrelevant context. In *Proceedings of the 40th International Conference on Machine Learning* (Vol. 202, pp. 31210–31227). PMLR. https://proceedings.mlr.press/v202/shi23a.html

Wallace, E., Xiao, K., Leike, R., Weng, L., Heidecke, J., & Beutel, A. (2024). *The instruction hierarchy: Training LLMs to prioritize privileged instructions* [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2404.13208

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=WE_vluYUL-X

Further reading

- Mike Taylor, Context Engineering with DSPy, O'Reilly Media, 2026 — Probably the closest overall match to this handout. It treats Context Engineering as the deliberate management of the information environment surrounding an LLM and examines

long-context problems, context selection, summarization, memory offloading, retrieval, tools, guardrails and control flow. It is particularly relevant to the chapter's view that good context is not simply “more information,” but the appropriate information selected, organized and maintained for the current task.

- Chip Huyen, AI Engineering: Building Applications with Foundation Models, O’Reilly Media, 2024 — The strongest broader reference for understanding why the model should be considered only one component of an AI application. It covers prompting, retrieval-augmented generation, agents, evaluation and application architecture. It is particularly useful for connecting the chapter's concepts of documentary context, tool context and instructional context to the engineering of complete systems around foundation models.
- Micheal Lanham, AI Agents in Action, 2nd ed., Manning Publications, 2026 — Particularly valuable for the sections dealing with tools, knowledge, memory, retrieval and changing operational state. It examines agents that use external resources and perform sequences of actions rather than merely producing isolated responses. This makes it a strong practical continuation of the chapter's distinction between a system possessing a capability and the information about that capability becoming part of the model's current context.
- Valentina Alto, Building LLM Powered Applications: Create Intelligent Apps and Agents with Large Language Models, Packt Publishing, 2024 — A practical introduction to constructing applications around LLMs using retrieval, vector databases, memory, orchestration, external information and agents. It is particularly relevant to the idea that information stored somewhere in a system does not automatically become context: it must first be selected or retrieved and then supplied to the model when needed.
- Jay Alammar and Maarten Grootendorst, Hands-On Large Language Models: Language Understanding and Generation, O’Reilly Media, 2024 — A useful technical complement for understanding context windows, embeddings, semantic search, retrieval and RAG. It helps explain how external information can be identified and inserted into the current context rather than relying exclusively on knowledge incorporated into the model's parameters.

Editorial notes

In producing these materials, I have used generative artificial intelligence tools to support the writing process, particularly to improve the form of the text, reorganize content, refine wording, and speed up certain editorial tasks.

As I work independently on the production of these materials, I try to automate any activities that can reasonably be automated, so that I can devote more time to research, design, and content development.

However, artificial intelligence does not determine the content of this work: the choice of topics, structure, ideas, interpretations, examples, and teaching approach are developed by me. AI is therefore used primarily as a tool to support production and formal revision, while authorship and responsibility for the design and content remain mine.

Unless otherwise indicated, this material is distributed under the Creative Commons Attribution–NonCommercial–ShareAlike 4.0 International license (CC BY-NC-SA 4.0).

You are therefore free to share, redistribute, adapt, and create derivative works based on this material, provided that appropriate credit is given, the material is not used for commercial purposes, and any modified or derivative versions are distributed under the same license.

