Prompt Techniques and Structuring

Written by Febogamedev

Prompt priming

Having distinguished the main dimensions of the prompt, we now introduce the concept of **prompt priming**.

Prompt priming consists in preparing the context before the main request by providing instructions, information, criteria or examples that guide how the task will be interpreted and performed (Liu et al., 2023; Reynolds & McDonell, 2021).

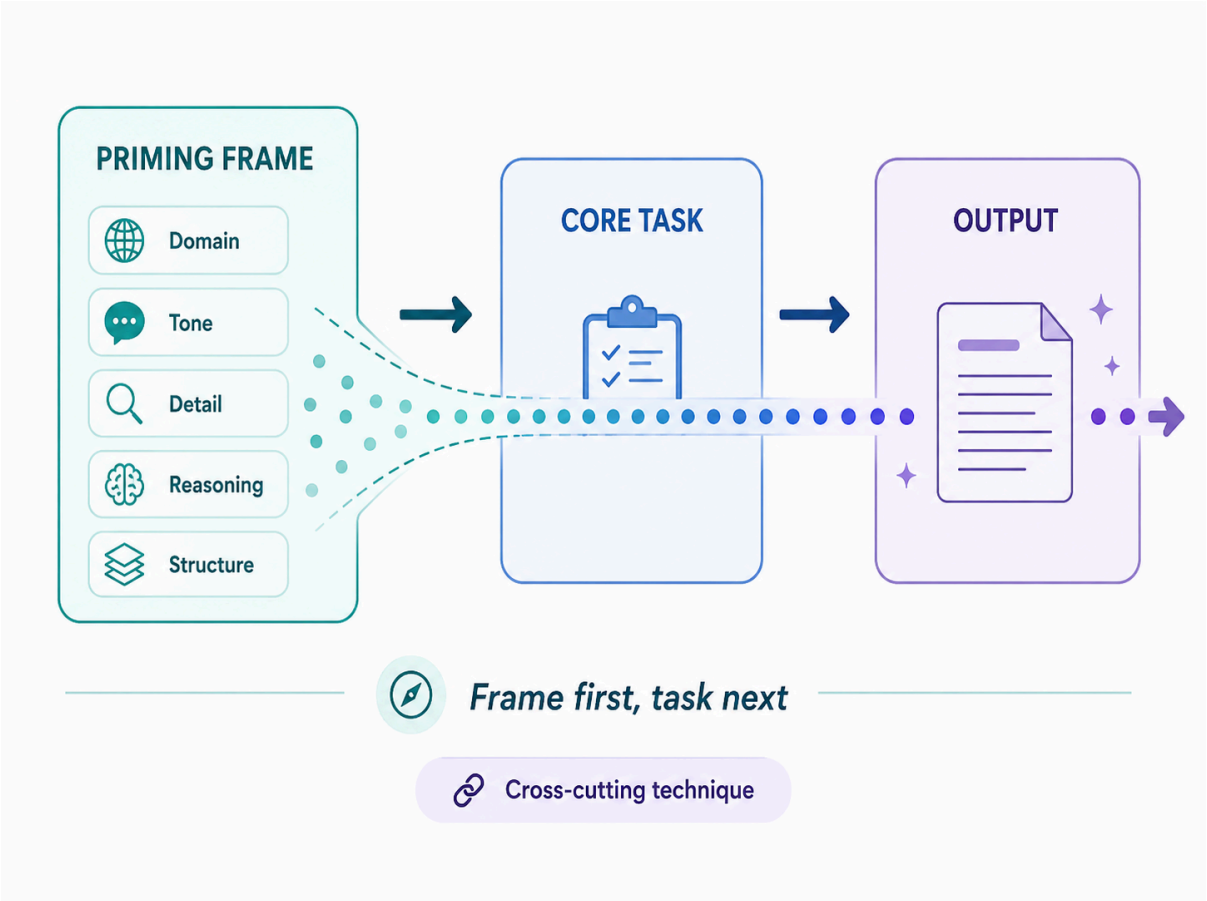
Instead of writing only “Explain what gravity is.”, a framework can be prepared: “You are preparing a physics handout for high school students. Use rigorous but accessible language, define technical terms and include a concrete example. Explain what gravity is.”

The main request is: Explain what gravity is.”
The preceding information prepares the context by indicating:

- The domain.
- The audience.
- The type of artifact.
- The language level.
- Some conceptual constraints.

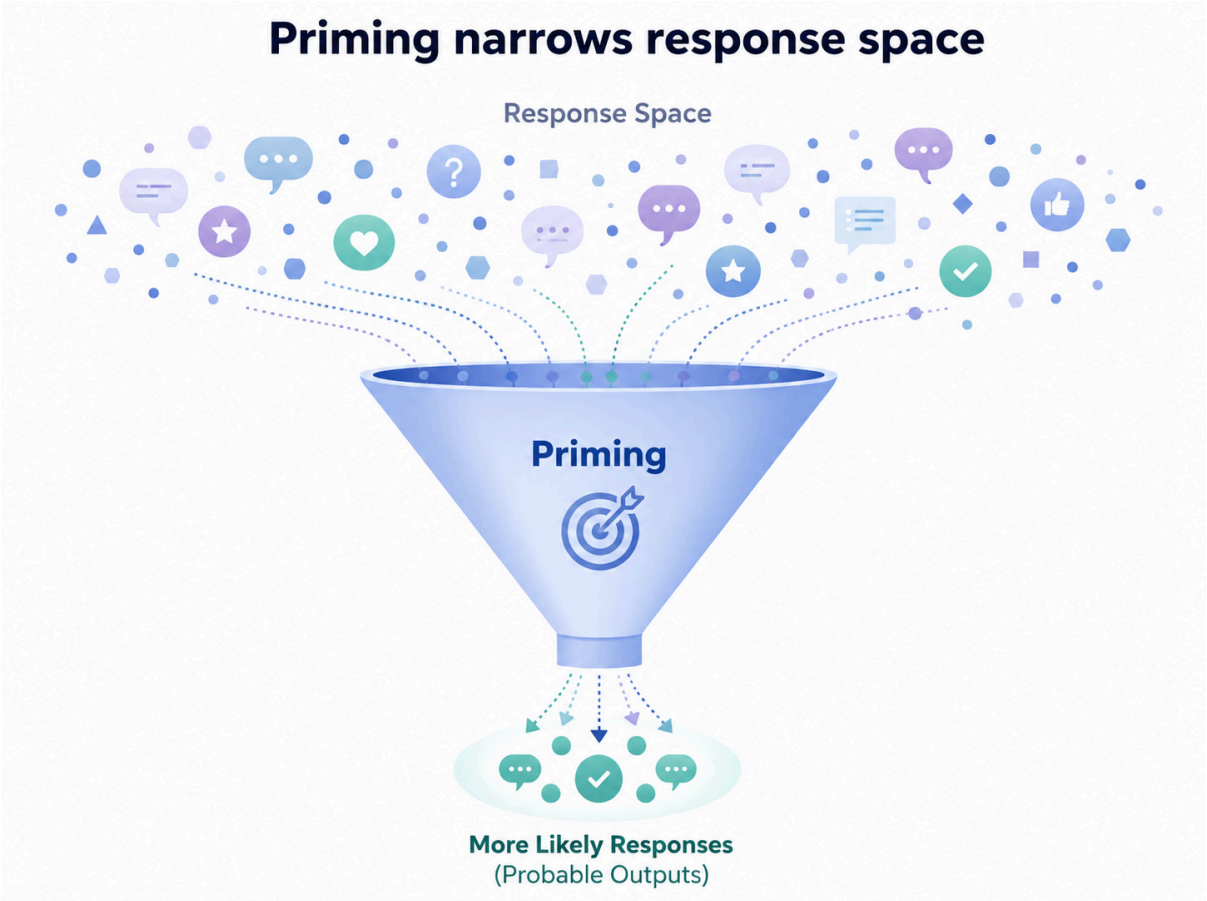
Priming is therefore not a separate dimension comparable to the request or the format.
It is a cross-cutting technique through which one or more dimensions are introduced before the operational core of the task.

A language model produces each token in relation to the tokens already present in the context (Brown et al., 2020).
Adding a framework before the request therefore changes the signals on which the continuation is generated.
The system is guided toward a particular domain, register, level of detail, type of reasoning or structure.



Priming does not mechanically force the model to respond in a particular way and does not permanently change the parameters learned during training (Brown et al., 2020; Liu et al., 2023).
It produces temporary conditioning through the available context.

Without priming, the system must infer many conditions from a potentially ambiguous request (Zamfirescu-Pereira et al., 2023).
With priming, the user can make all the dimensions listed earlier explicit in advance.
In design terms, priming narrows the region of the response space that is made more likely (Reynolds & McDonell, 2021).



For teaching purposes, priming can be distinguished by its scope as:

- Task-specific: primarily guides the current task.
- Local: guides a sequence of turns in the conversation.
- Persistent: can guide different interactions over time.

This classification does not describe three completely different techniques, but three different levels through which a framework can influence the interpretation of requests.

Task-specific priming is introduced primarily to guide a single task.
For example, consider this prompt:
“Consider the problem from the perspective of Interaction Design. Analyze this interface…”.
The framework precedes the operational request and makes a response consistent with that domain more likely (Brown et al., 2020).
Task-specific priming is designed mainly for the current task.
However, if the message remains available in the conversation history, it may also continue to influence subsequent turns indirectly (Brown et al., 2020).

Local priming, by contrast, establishes a framework explicitly intended to guide multiple requests within the current conversation.
For example, consider this prompt:
“In your next responses, bear in mind that I am writing a handout for graphic design students. Use examples related to visual production and define computing terms before using them”.

Subsequent requests can be interpreted in light of this framework for as long as it remains available and relevant in the conversational context.

Persistent priming instead derives from information or configurations that can be reused across different interactions.
It can be introduced, for example, through:

- Custom instructions.
- Profile preferences.
- Persistent memories.
- Assistant configurations.
- Recurring application instructions.

These elements do not coincide with the prompt written for an individual request. They belong to the broader operational context and can be applied or retrieved across different interactions.

It is nevertheless important to distinguish priming from context in general.
All priming changes the context, but not every element of the context necessarily serves a priming function.

A retrieved document, for example, can strongly influence a response and even guide its domain or vocabulary, but it remains primarily documentary context.
It becomes useful to speak of priming when that information specifically serves to predispose the model toward a particular interpretation or mode of response.

Example
A user configures their assistant by specifying:
“When you explain technical concepts to me, use clear language, avoid unnecessary jargon and always add a concrete example.”

This instruction does not concern a single request or only the current conversation.
If it is saved in custom instructions, the profile or a persistent memory, it can also influence future interactions.

For example, days later the user might ask: “What is a hash?”
The system might respond while already taking the persistent framework into account:
simple definition → accessible language → concrete example

In this case, the specific request does not contain those instructions, but the system retrieves them from a previous configuration.

When persistent, local and task-specific priming belong to the same instruction level and provide incompatible guidance, a framework more specific to the current task may temporarily replace a more general one.
For teaching purposes, increasing specificity can therefore be represented as:
Persistent → Local → Task-specific

- Persistent priming defines a customary and relatively general framework.
- Local priming adapts it to the ongoing conversation.

- Task-specific priming acts on the specific task and, precisely because it is more specific to the current request, may temporarily replace more general guidance.

However, this sequence does not represent a universal instruction hierarchy: actual priority also depends on the level from which instructions originate and the rules defined by the system architecture (as we will see when discussing context, normative context takes priority over everything).

Example
A user might have the persistent preference:
“Provide in-depth explanations.”
During a conversation, however, they might establish:
“Keep responses concise in this chat.”
And, for a single request, specify:
“Answer in one sentence.”

In that case, for that specific request, the task-specific framework takes precedence:
In-depth explanations → Concise responses → One sentence

This hierarchy does not mean that persistent priming is eliminated, but that it may be temporarily overridden by more local and specific instructions.

A common form of priming consists in defining the perspective from which the system should approach the task.
For example: “Act as a critical reviewer and identify the weaknesses in this argument.”

This formulation guides the response toward finding problems.
It can be useful, but it can also produce an excessively negative assessment.

A more balanced formulation would be:
“Analyze this argument by identifying strengths, weaknesses, implicit assumptions and possible alternative interpretations.”

In this way, priming does not impose a conclusion, but establishes an analytical procedure.
Priming should guide behavior without arbitrarily predetermining the judgment.

A similar problem can occur when the perspective is defined by assigning a role, for example “act as a teacher” or “respond as a reviewer”.
Assigning a role does not, however, give the model new knowledge or capabilities and does not necessarily improve the accuracy of the response.
For this reason, it is generally preferable to specify the desired behavior directly.

Instead of writing: “Act as an expert usability reviewer.”
it is more precise to write: “Analyze the interface by identifying navigation problems, ambiguities in actions and shortcomings in feedback. For each problem, describe its consequence for the user and propose a possible solution.”

A role remains useful when perspective, tone or simulation is a genuine part of the task, but it should not be used as a substitute for explicit instructions and criteria.

The general principles of prompt design also apply to priming: introductory information must be relevant, coherent and proportionate to the task.
Its purpose is not to lengthen the request, but to make explicit, before the main directive, the conditions that can usefully guide its interpretation.

Priming is therefore a cross-cutting technique through which some dimensions of the prompt are introduced in advance to prepare the context of the request (Liu et al., 2023).
It does not guarantee a particular result, but makes responses compatible with the provided framework more likely (Sclar et al., 2024).
Before adding a preamble, it may be useful to ask:

- What ambiguity am I trying to reduce?
- Which dimension of the prompt needs to be guided?
- Is this information necessary for the current request?
- Is the criterion observable?
- Are the instructions compatible with one another?
- Am I preparing the system to follow a method or to reach a predetermined conclusion?
- Could the instruction be expressed more simply?

Prompt template

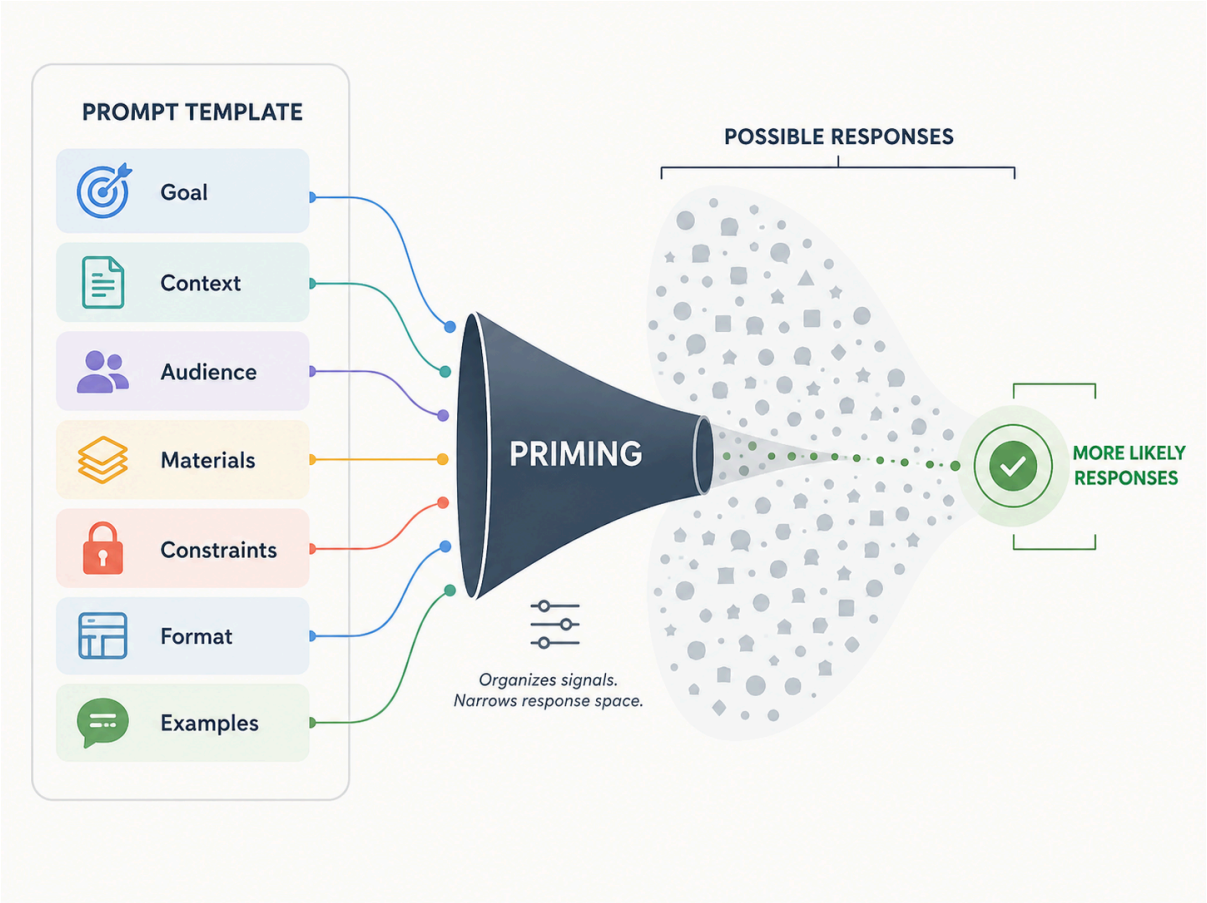
After analyzing the different dimensions of the prompt separately, they can be organized into a reusable structure.
This structure is called a **prompt template**.

A prompt template is a model that explicitly organizes the main components of a request, leaving the elements that change from one case to another variable (White et al., 2023).

For example, a general structure might include:
Objective → Audience → Context → Materials → Constraints → Format → Quality criteria

Not all these components need to be present in every prompt.
The template primarily serves as a design framework, useful for remembering which information may be relevant and preventing important aspects from being implicitly left to the model’s interpretation (White et al., 2023).

A prompt template can be interpreted as a structure based on the principle of priming: it explicitly organizes a series of signals—objective, context, audience, materials, constraints, format and examples—that progressively narrow the space of the most likely responses (Bsharat et al., 2024; White et al., 2023).



However, no universal template works for every model and every situation (Sclar et al., 2024).

- A simple request may require very little information.
- A complex or ambiguous task may benefit from a more elaborate structure.

Some general principles nevertheless remain useful:

- Formulate direct instructions.
- Clearly distinguish the task from the materials.
- Use a consistent structure.
- Specify the format only when it is actually necessary.

The template should therefore be considered a starting point to adapt and verify, not a formula capable of automatically guaranteeing the result (Bsharat et al., 2024; White et al., 2023).
Its function is to make the relevant parameters of the task explicit and organizable: what must be produced, why, for which audience, within which domain, based on what information and under which constraints.
When these conditions remain implicit, the system must infer them from the available context and may gravitate toward a generic response or one poorly suited to the purpose (Zamfirescu-Pereira et al., 2023).

Structuring the prompt therefore means consciously making decisions that would otherwise be left to the model's probabilistic behavior (Reynolds & McDonell, 2021).

- A linguistically elegant prompt can remain ambiguous.
- Conversely, a simply written but well-organized prompt can be easier to interpret, evaluate and modify.

From a design perspective, structure serves three main functions.
- Segmenting information.
- Establishing a hierarchy of information.
- Improving readability for design.

Segmenting information
Segmentation separates the parts of the prompt according to their function.
For example, it allows us to distinguish:
- The request.
- Contextual information.
- Materials to process.
- Output characteristics.

This distinction is particularly important when the prompt contains documents, data, examples or quoted texts (OpenAI, n.d.).
Without recognizable separation, part of the material might be interpreted as an instruction or confused with the main request (Anthropic, n.d.; OpenAI, n.d.).

Establishing a hierarchy of information
Not all information has the same importance.
Structure makes the main request and priority constraints more visible, preventing them from being dispersed among secondary details.

Establishing a hierarchy does not necessarily mean repeating the same instruction several times.
It means placing it in a recognizable section and distinguishing it from information that serves only a supporting function.

Improving readability for design
Structure also makes the prompt more readable for its designer (White et al., 2023).
When the output is unsatisfactory, an organized prompt makes it easier to identify the component to revise.

For example, one can ask whether the problem concerns the request, the audience, the available information, conceptual constraints or the format.
Structure also facilitates reuse, sharing and comparison between different versions of the same prompt (White et al., 2023).

Example
Below is a prompt:
“Rewrite this text about phishing for a handout intended for high school students. The students already know about email, but not social engineering. The text must be clear, no

more than 250 words, without lists, and must distinguish phishing from malware. Do not use an alarmist tone. The previous chapter discusses password security.

Phishing is a type of cyberattack that uses deceptive messages to persuade a person to disclose personal information. The message often contains a link to a fake website. Also explain that the sender may appear authentic and include some useful signs for recognizing the attempted scam. However, do not say that checking just one element is enough to be sure."

The prompt contains all the necessary information, but its functions are mixed together. The material to be rewritten also contains imperative phrases such as:

- "Also explain…"
- "However, do not say…"

It is therefore not immediately clear whether these phrases belong to the original text or are instructions directed at the model.

Below is the same prompt in segmented form.

```
## REQUEST
Rewrite the provided material as a brief educational
explanation of phishing.

## CONTEXT
The text will be included in a handout intended for high
school students.

The students:
- already know how email works;
- have not yet studied social engineering.

The previous chapter is devoted to password security.

## MATERIAL TO REWORK
[START OF MATERIAL]
Phishing is a type of cyberattack that uses deceptive messages
to persuade a person to disclose personal information.
The message often contains a link to a fake website.

Also explain that the sender may appear authentic and include
some useful signs for recognizing the attempted scam. However,
do not say that checking just one element is enough to be sure.
[END OF MATERIAL]

## CONCEPTUAL CONSTRAINTS
Priorities:
```

```
- Distinguish phishing from malware.
- Clarify that no single clue guarantees that a message is authentic.
- Avoid oversimplifications or alarmist statements.

## OUTPUT CHARACTERISTICS
- Continuous prose.
- No more than 250 words.
- No lists.
- Accessible but technically correct language.

## FINAL INSTRUCTION
Now write the final version of the text.
```

The prompt is segmented because it clearly distinguishes:

- The request, namely rewriting the text;
- The context, namely its educational purpose and the students' knowledge;
- The material, namely the content to be reworked;
- The conceptual constraints, namely what the meaning must respect;
- The output characteristics, namely the form of the response.

The delimiters [START OF MATERIAL] and [END OF MATERIAL] also clarify that the imperative phrases contained in the text are part of the material and are not necessarily independent instructions (Anthropic, n.d.; OpenAI, n.d.).

The conceptual constraints are arranged hierarchically: for example, the distinction between phishing and malware is made more important than the length of the text.
The hierarchy makes it visible that conceptual correctness takes priority over brevity. If it were not possible to satisfy every constraint perfectly, the designer has indicated which aspects should receive greater attention.

The prompt is also more readable: if the result were unsatisfactory, it would be easier to identify the part to modify.

- If the text were too technical, one could adjust the audience or the language level.
- If it were too long, the formal constraint could be modified.
- If it confused phishing and malware, the problem would concern the conceptual constraints.
- If the model reproduced the notes verbatim, the separation of the material to be processed could be revised.
- If the text did not connect well to the previous chapter, the contextual information could be clarified.

The following model organizes the dimensions of the prompt into three functional areas:

- Task definition.
- Context and materials.
- Output definition.

In more complex prompts, a brief final operational instruction can be added to these.

```
#TASK DEFINITION

##Request or directive:
What operation must the system perform?

##Purpose:
Why must it produce the output and what function will it serve?

##Audience:
Whom is the result intended for and what knowledge can it assume?

##Domain:
Within which disciplinary or semantic field must the task be
interpreted?

##Conceptual constraints:
What ideas, distinctions, relationships and criteria must the content
respect?

#CONTEXT AND MATERIALS

##Additional information:
What data, premises or conditions of the situation are needed?

##Materials to process:
What texts, documents, images, data or other content must the system
use?

##Examples, when needed:
What cases demonstrate the desired behavior or result in concrete terms?

#OUTPUT DEFINITION
##Format:
What type of artifact must be produced?

##Formal constraints:
How should the output appear?

##Behaviors to avoid, when relevant:
What specific results are considered inappropriate?

#FINAL OPERATIONAL INSTRUCTION
In brief, what action must the system perform after considering the
preceding instructions and materials?
```

It is not necessary to fill in every section.
For example:

- A simple translation may require only a directive and an indication of the language.
- A professional analysis or the production of educational content may instead require a more elaborate structure.

The template must be proportionate to the complexity of the task (Bsharat et al., 2024).
Let us examine the various parts of the model.

Task definition

The first part of the prompt clarifies the core of the activity.

- The request indicates the operation to perform.
- The purpose specifies the function of the output.
- The audience defines the reader's knowledge and needs.
- The domain defines the interpretive field.
- Conceptual constraints establish which contents, distinctions and criteria must be respected.

These elements do not necessarily have to be presented as separate fields. In a short prompt, they can be integrated into a single sentence.

Example

"Explain how DNS works to high school students who already know about IP addresses, with the aim of introducing the subsequent study of the Web."

This sentence already contains:

- the request: explain DNS.
- the audience: high school students.
- the prior knowledge: IP addresses.
- the purpose: prepare for studying the Web.
- the domain: computer networks.

Explicit segmentation becomes useful when these dimensions increase or need to be modified separately.

Context and materials

Additional information describes the concrete situation in which the task must be performed. Materials, by contrast, are the objects on which the system must operate: texts, documents, images, data, code or results obtained through external tools.

The distinction is important.

- In the request "Analyze the attached text", the text is the material to process.
- The statement "The text was written in 2018 and some technological references may now be outdated" contains additional information, because it describes a relevant condition for interpreting the material correctly.

When the prompt contains large amounts of data, it is useful to clearly delimit where the materials begin and end (Anthropic, n.d.).
Current guides suggest using headings, tags or delimiters consistently to separate instructions, context and input (Anthropic, n.d.; OpenAI, n.d.).
However, no type of delimiter is universally superior: it is more important that the chosen convention is clear and used consistently within the prompt (OpenAI, n.d.).

For example:

```
## MATERIAL TO ANALYZE
[Start of text]
...
[End of text]
```

or:

```
<material>
...
</material>
```

Delimiters do not activate a special function of the model (Liu et al., 2023).
They simply make the role of the different portions of context more recognizable.

Output definition
This part specifies what result must be produced.

- The format defines the type of artifact: continuous prose, a table, an outline, a list, code, JSON and so on.
- Formal constraints govern the signifier of the output, meaning how the content must manifest itself: length, structure, tone, number of sections, presence of formulas or terminological density.
- Any behaviors to avoid can be specified when they represent particularly relevant errors. Whenever possible, however, it is preferable to accompany the prohibition with a positive formulation (for example, “Use an analytical and neutral tone, avoiding promotional language”) (Bsharat et al., 2024)
    - The positive formulation indicates the direction to follow.
    - The complementary prohibition specifies the boundary that must not be crossed.

Example
Consider designing an educational chapter about *DNS* with a structured prompt using the template just presented:

```
#TASK DEFINITION

#Request:
Write an introductory chapter on how DNS works.
```

```
##Purpose:
Prepare students to study the Web, showing why domain names
must be associated with IP addresses.

##Audience:
High school students who already know the concept of an
IP address but have not yet studied the application layer.

##Domain:
Computer networks and the application layer.

##Conceptual constraints:
- Distinguish the domain name from the website.
- Distinguish the registrar, the registry and the DNS service.
- Explain that DNS does not transfer web pages, but provides
  information needed to locate the server.
- Include a complete example of domain name resolution.
- Clarify the function of the resolver and authoritative servers.
- Keep the DNS resolution process separate from the subsequent
  HTTP communication.

#CONTEXT AND MATERIALS

##Additional information:
- The chapter follows the one devoted to IP addresses.
- Students tend to confuse domain, hosting and website.
- The text will be read independently, without an immediate oral
explanation.

#OUTPUT DEFINITION

##Format:
Educational prose with headings and paragraphs.

##Formal constraints:
- Approximately 800 to 1,000 words in length.
- Technical but accessible language.
- Conclude with a five-point summary.
- Do not use mathematical formulas.

##Behaviors to avoid:
Use technical, concrete descriptions, avoiding presenting DNS
simply as an "Internet phone book" without explaining its limits.

#FINAL INSTRUCTION
```

```
Now write the chapter following the preceding instructions.
```

In this example, the structure does not magically add new capabilities to the model. However, it makes the different conditions of the task recognizable and allows them to be modified separately (White et al., 2023).

- If the text were too technical, one could adjust the audience or the formal constraints.
- If it confused *DNS* and *HTTP*, the relevant conceptual constraint could be revised.
- If it were unsuitable for its position in the handout, the additional information could be modified.

The template is not useful only before generation (White et al., 2023).
It can also be used after observing the output.

- An overly generic response may suggest that the domain, audience or purpose was not defined precisely enough.
- A response in the wrong format may indicate that formal constraints were insufficiently clear or that the requested format should be enforced through a dedicated tool.
- A correct but unhelpful result may show that the concrete context or conceptual constraints were incomplete.

Structure therefore makes the prompt easier to analyze, because it allows hypotheses about the cause of the problem to be formulated and targeted changes to be made (White et al., 2023; Zamfirescu-Pereira et al., 2023).

In the current state of generative systems, no template should be considered effective merely because it appears theoretically well constructed (Sclar et al., 2024).
Quality must be verified on:

- A specific model.
- A specific version.
- A representative set of cases.
- Success criteria defined in advance.

Providers' guidance emphasizes that prompt engineering remains an iterative process (Anthropic, n.d.; OpenAI, n.d.).

- Anthropic recommends establishing specific, measurable criteria.
- OpenAI recommends using pinned model versions and systematic evaluations to maintain more consistent behavior over time (OpenAI, n.d.).

As is clear, it is advisable to use structured language in the prompt.
Some languages inherently make text easier to structure.
One of the most widely used is **Markdown**.

In this context, Markdown is particularly useful because it allows explicit structure to be added to text using just a few characters, without turning it into a complex format (CommonMark Group, 2024).

For example, a request can be organized like this:

```
# Objective
Analyze the following text.
## Criteria
- Clarity.
- Correctness.
- Concision.
## Text
The text to analyze...
```

Headings, lists, code blocks and separators make the different parts of the request more recognizable and can help the model distinguish instructions, data, examples and content to process (OpenAI, n.d.).

For the same reason, *LLM*s can produce responses directly in Markdown.
The application hosting the model can subsequently render the text, transforming Markdown markers into graphically displayed headings, lists, tables, links and code blocks.

```
## Result
- First item.
- Second item.
- Third item.
```

Markdown is therefore a particularly convenient format for interaction between people and LLMs because it simultaneously has three characteristics:

- It is text-based.
- It is easily readable by people.
- It allows the structure of the content to be represented simply.

This is why it is often used both to organize prompts supplied to models and to structure the responses they generate (OpenAI, n.d.).

The order of information

The order of information can influence the system's behavior, but there is no universally optimal sequence (Lu et al., 2022; Sclar et al., 2024).
The most suitable organization depends on:

- The model used.
- The length of the context.
- The type of task.
- The presence of documents.
- The level at which the instruction is placed.
- The structure provided by the application.

It is useful to distinguish two cases.

- General and priority instructions: stable rules, priorities and general behaviors should be placed in system instructions or, when this is not possible, in a clearly visible position at the beginning of the prompt. This part defines the general framework within which the task must be interpreted.
- Long contexts and documents: when users provide documents or large amounts of data, it may instead be useful to present the material first and place the specific question immediately afterward, accompanied by a transition sentence (e.g., “Based on the preceding material, compare the two positions and identify the main points of disagreement.”) (Anthropic, n.d.)

The most recent guides recommend providing documents first in very long contexts and placing the specific request at the end, while keeping priority behavioral instructions at the system level or at the beginning (Anthropic, n.d.).
One possible organization is therefore:

- General instructions and priorities.
- Materials and supporting information.
- Specific request.
- Format and essential output constraints.

This is a design strategy, not an absolute rule (Sclar et al., 2024).
In very extensive contexts, different pieces of information are not necessarily used equally effectively (Liu et al., 2024).

The study known as **Lost in the Middle** observed, in multi-document question answering and information retrieval tasks, often better performance when relevant information was placed at the beginning or end of the context, and worse performance when it was in the middle (Liu et al., 2024).
However, this phenomenon is not a law that applies equally to every model and every activity (Liu et al., 2024; Sclar et al., 2024).
The design implication is not that every piece of information must be repeated at the beginning and end, but that essential instructions should not be buried in large amounts of secondary material (Liu et al., 2024).

In long prompts, it can be useful to:

- Remove irrelevant information.
- Group content by function.
- Use descriptive headings.
- Place the request close to where the material ends.
- Briefly reiterate the essential constraints in the final instruction.

Furthermore, when the prompt contains a great deal of information, it may be useful to conclude with a sentence that clearly signals the transition from preparation to execution (Anthropic, n.d.):
“Now write the chapter following the preceding instructions.”
Or:
“Based on the document provided, produce the requested comparison table.”
A label can also be used:

“OUTPUT: Write the final text in continuous prose.”

The word OUTPUT is a textual signal that makes the final action more recognizable.
A final instruction is particularly useful when the prompt ends with data, notes or lists.
Without an explicit transition, the immediately preceding structure can influence the form of the continuation (Sclar et al., 2024).
However, there is no need to add a final line to every request. In short prompts, it would only create redundancy.

Underspecification and default responses

The dimensions of the prompt analyzed in previous chapters are useful design categories for describing and organizing a request.
However, they do not correspond to separate internal components that the model explicitly recognizes as request, purpose, audience, domain or constraints.

An *LLM* does not have a checklist for checking whether all these dimensions have been included.
It receives context as a sequence of tokens and progressively generates a continuation based on the available signals (Brown et al., 2020).
Headings, labels and structures can make the function of different information more recognizable, but do not imply that the model processes it through rigid, isolated categories.
The distinction between request, context, audience and format belongs primarily to the design model humans use to construct and analyze the prompt (Liu et al., 2023).

When a relevant dimension is not specified, the model does not necessarily identify it as “missing” in the human sense of the term.
It may try to infer it through:

- The wording of the request.
- The preceding conversation.
- The application’s instructions.
- The information available in the context.
- The regularities learned during training.

When these signals are insufficient, several interpretations of the task remain plausible.
This phenomenon can be called **prompt underspecification.**

A prompt is underspecified when it does not make explicit certain conditions necessary to guide the result toward the user’s actual purpose (Zamfirescu-Pereira et al., 2023).
Consider a request such as: “Explain what compression is.”

The request indicates the general operation to perform, but leaves many decisions open.
For example:

- Is it about data, image, audio or signal compression?
- Is the explanation intended for a general or specialist audience?
- Should it be introductory or in-depth?
- Is it for a handout, a presentation or revision?

- Should it focus on theory or practical applications?
- Should it distinguish lossy and lossless compression?

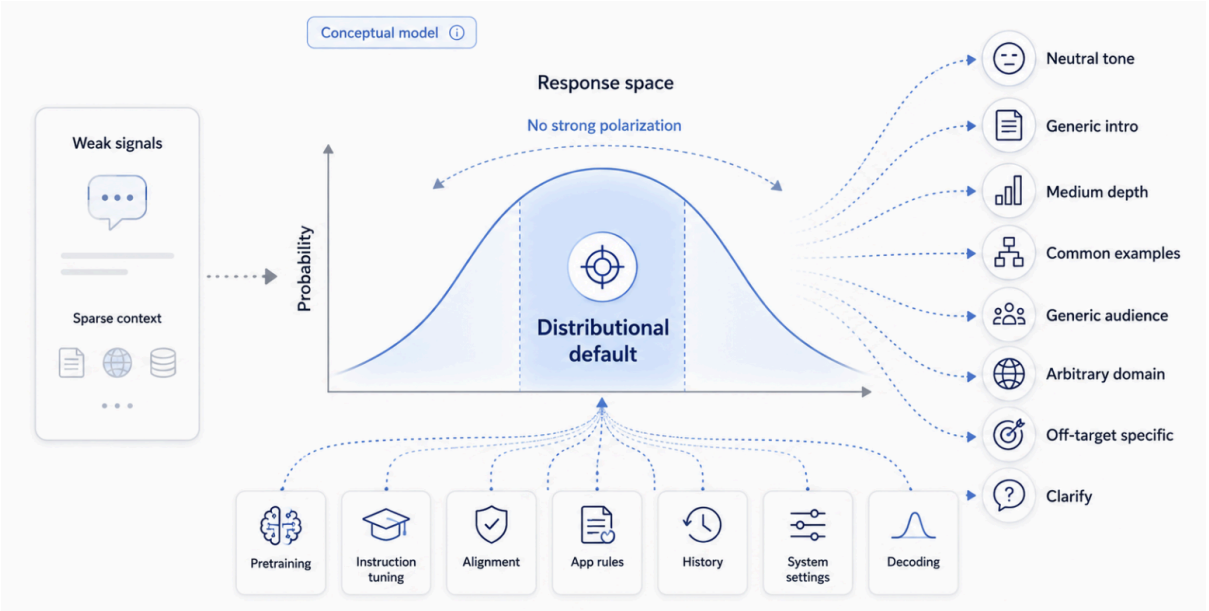
The model might produce a generally correct explanation, but not necessarily one suited to the user's concrete objective (Zamfirescu-Pereira et al., 2023).
Underspecification therefore does not imply that the request is incomprehensible.
It means that multiple different responses remain compatible with what was asked.

In this text, we will use the expression **distributional default** to refer to the set of behaviors toward which the system tends when the prompt and the rest of the context do not provide sufficient signals to guide the response in a more specific direction.
It is not an internal function formally named in this way, nor a predefined response identical across all systems.
It is a conceptual model useful for describing what can happen when certain conditions of the task remain implicit.

Default behavior does not depend only on the data used during pretraining (Ouyang et al., 2022).
It is also influenced by:
- Subsequent instruction training.
- Preferences learned during alignment.
- The application's internal instructions.
- Conversation history.
- System settings.
- The strategy used to generate the response.

For this reason, different models and applications may respond differently to the same underspecified prompt (Ouyang et al., 2022; Sclar et al., 2024).
A vague request can produce a neutral explanation for a general audience, but it can also lead the system to:
- Implicitly assume an audience.
- Arbitrarily choose a domain.
- Interpret the task in an unwanted direction.
- Provide a very specific but off-target response.
- Ask for further clarification.
- Produce different results from one run to another.

The distributional default therefore does not necessarily coincide with a generic response.
More generally, it refers to reliance on behaviors and assumptions that are most compatible with the signals already present in the system.



This image indicates that, when the prompt lacks sufficiently specific guidance, the system will tend to give the most likely responses, which often coincide with the most neutral and generic responses

When an important dimension remains implicit, three main consequences may occur.

- The number of compatible interpretations increases.
- Implicit assumptions are adopted.
- General behaviors become more likely.

The number of compatible interpretations increases

The more open the prompt, the greater the number of responses that can be considered consistent with the request.
The system therefore has fewer signals for favoring a specific direction.
The result can become more variable and depend more heavily on the model used, the application configuration or small differences in wording (Sclar et al., 2024).

For example, the request “Analyze this image.” can be interpreted in many ways. The system might describe its content, assess its composition, analyze its colors, discuss its symbolic meaning or identify any technical problems.
All these operations are compatible with the verb “analyze”, but do not necessarily address the same need.

Implicit assumptions are adopted

To complete the task, the system may independently assume certain unstated conditions (Zamfirescu-Pereira et al., 2023).
For example, it may choose:

- A general audience.
- An intermediate depth of explanation.
- A neutral tone.
- A structure aimed at a general audience.

- A particular interpretation of the domain.
- An implicit balance between concision and completeness.

These assumptions may be reasonable, but they remain hypotheses formulated by the system based on the available context.

- If they coincide with the user's intentions, the prompt may seem perfectly adequate even though it was underspecified.
- If they do not coincide, the response may appear formally correct but unhelpful.

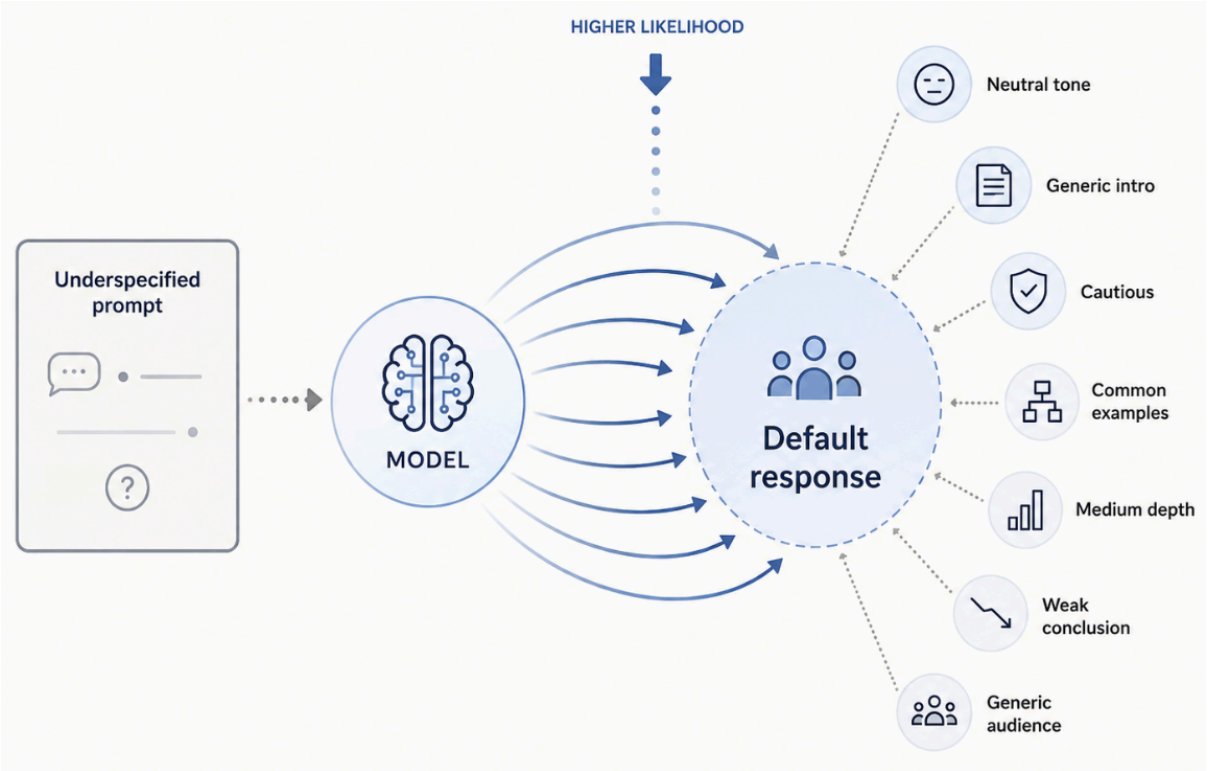
General behaviors become more likely

In the absence of specific guidance, responses compatible with the general behaviors learned by the model and with the assistant's configuration become more likely.

The result may therefore feature:

- A neutral tone.
- General introductions.
- Cautious explanations.
- Common examples.
- Intermediate depth.
- Unremarkable conclusions.
- Few assumptions about the audience's level.

This is not a conscious choice by the model, nor a deliberate strategy to avoid risk. It is the effect of the available signals and the configurations most compatible with them (Ouyang et al., 2022).



The problem of underspecification is particularly insidious because it does not necessarily produce obvious errors (Zamfirescu-Pereira et al., 2023).
A response can be:

- Grammatically correct.
- Well organized.
- Plausible.
- Generally correct.
- Apparently complete.

Yet it may be poorly aligned with the actual purpose.
The request “Explain what compression is.”
Could generate a general definition of reducing data size, accompanied by the distinction between lossy and lossless compression.
The response would not necessarily be wrong.
However, if the objective were to prepare graphic design students to understand the JPEG format, it would be too broad and insufficiently focused.

A more specific formulation could be:
“Explain lossy compression to graphic design students who already know about pixels and color depth. Use JPEG as the main example and clarify the relationship between size reduction, visual quality and the appearance of artifacts.”

This version makes several conditions explicit:

- Domain: digital graphics and image compression.
- Audience: graphic design students.
- Prior knowledge: pixels and color depth.
- Request: explain lossy compression.
- Conceptual constraints: use JPEG as an example, clarify the relationship between size, quality and artifacts.

The second request is not better simply because it contains more words.
It is more appropriate because it makes explicit information capable of significantly changing the result (White et al., 2023).

Thus, leaving a dimension implicit does not necessarily mean leaving it empty.
The system can assign it a value through the available signals.
For example:

- If the audience is not specified, the model may assume a general audience.
- If the format is not indicated, it may produce continuous prose.
- If the domain is not clarified, it may choose the interpretation that appears most likely in the context.
- If the purpose is not defined, it may simply explain the topic without adapting it to a concrete use.

These inferences are not necessarily wrong.
They become problematic when they concern important decisions and do not coincide with the user’s intentions (Zamfirescu-Pereira et al., 2023).

The principle can be summarized as follows:
What is not specified does not necessarily remain undefined: it may be completed through the assumptions available in the system.

Avoiding underspecification does not mean always filling in a complete template (Bsharat et al., 2024).
For a request such as “Translate the word ‘Buongiorno’ into English”, it is not necessary to define the audience, domain, purpose, additional information and conceptual criteria in detail.
The operation is simple and the possible interpretations are limited.
Even a more elaborate request can leave some dimensions implicit when:

- They are already clear from the conversation.
- They do not significantly change the result.
- They can be inferred without important consequences.
- Making them explicit would only introduce redundancy.

The problem is therefore not the presence of implicit elements, but leaving implicit the conditions that actually affect the quality and suitability of the output (Zamfirescu-Pereira et al., 2023).
Design consists in finding a balance between underspecification and overspecification (Sclar et al., 2024).

Supplementary material

Exercises

Priming: guiding without imposing

Consider these three prompts.

- Prompt A: “Analyze this interface.”
- Prompt B: “Consider the problem from the perspective of Interaction Design. Analyze this interface.”
- Prompt C: “Analyze this interface by identifying navigation problems, ambiguities in actions and shortcomings in feedback. For each problem, describe its consequence for the user and propose a possible solution.”

Answer the following questions:

- Explain what prompt priming means.
- Identify which part of Prompt B serves a priming function.
- Explain why priming is not an independent dimension of the prompt, but a cross-cutting technique.
- Explain how priming can narrow the space of the most likely responses.
- Explain why priming does not permanently change the model’s parameters.
- Distinguish task-specific, local and persistent priming.
- Provide an original example for each of the three levels.
- Explain why local priming can influence multiple requests in the same conversation.
- Explain how persistent priming can instead influence different conversations.
- Compare Prompt B and Prompt C: which defines the desired behavior more precisely?
- Explain why assigning a role such as “Act as an expert” does not give the model new knowledge.
- Explain why it may be preferable to describe actions and criteria directly rather than simply assigning a role.
- Analyze this instruction: “Act as a critical reviewer and find every problem.”. Explain what bias it might introduce.
- Rewrite it to guide the analytical method without predetermining the conclusion.
- Conclude by explaining the difference between guiding behavior and arbitrarily directing the judgment.

Underspecification and distributional default

Consider this request: “Explain what compression is.”
The system produces a general explanation of data compression, distinguishing lossy and lossless compression.

Answer the following questions.

- Explain why the prompt can be considered underspecified.
- Identify at least five important conditions that remained implicit.
- Explain why a response generated from an underspecified prompt can be correct yet unhelpful.

- Explain what the expression distributional default means in the handout.
- Clarify why it is not an internal function of the model actually named in this way.
- Indicate what assumptions the system might automatically adopt regarding audience, domain, tone, depth and format.
- Explain why two different models might respond differently to the same underspecified prompt.
- Describe the three main consequences indicated in the handout:
- An increase in compatible interpretations.
- The adoption of implicit assumptions.
- A higher likelihood of general behaviors.
- A teacher actually wanted to explain JPEG to graphic design students. Rewrite the request, making explicit only the conditions that are genuinely relevant.
- Explain why the new request is better not simply because it is longer, but because it reduces specific uncertainties.
- Provide an example of a request for which filling in a complex template would instead be unnecessary.
- Conclude by explaining why designing a prompt means finding a balance between underspecification and overspecification.

Testing prompt priming experimentally
Open a new conversation with a generative system, enter “Explain what a neural network is” and save the response.

Then open a second new conversation and use task-specific priming:
“You are preparing a short handout for graphic design students who use generative AI tools but do not know programming. Use rigorous but accessible language and connect at least one example to image generation.
Explain what a neural network is.”

Then carry out a third test using different priming:
“The explanation is intended for university computer science students. Use technical terminology and connect the concept to parameters, training and inference.
Explain what a neural network is.”

Compare the three responses.
- Analyze the vocabulary.
- Analyze the depth of explanation.
- Analyze the examples chosen.
- Identify what knowledge is assumed.
- Check whether the central content remains substantially the same.
- Identify which characteristics changed primarily as a result of priming.
- Explain which dimensions of the prompt were introduced in advance in the two frameworks.
- Identify any priming instructions that did not produce noticeable effects.
- Change only one component of the priming and repeat the test.
- Conclude by explaining experimentally in what sense priming produces temporary conditioning of the context rather than a change to the model.

From an unorganized prompt to a structured prompt
Use this deliberately poorly structured prompt:
“Rewrite this text about phishing for a handout intended for high school students. They already know about email but not social engineering. It must be no more than 250 words, with no lists, clear and technical. The previous chapter concerns passwords. Phishing uses deceptive messages to obtain personal information. Also explain that the sender may appear authentic. Do not say that a single clue is enough. Distinguish it from malware. Rewrite it.”

First run it as it is and save the output.
Then reorganize it using this template:

```
# TASK DEFINITION

## Request
...

## Purpose
...

## Audience
...

## Conceptual constraints
...

# CONTEXT AND MATERIALS

## Additional information
...

## Material to process
[START OF MATERIAL]
...
[END OF MATERIAL]

# OUTPUT DEFINITION

## Format
...

## Formal constraints
...

## Behaviors to avoid
...

# FINAL INSTRUCTION
```

```
...
```

Then:

- Place each piece of information in the appropriate section.
- Clearly distinguish instructions from material to process.
- Use consistent delimiters for the material.
- Identify which constraints have priority.
- Prioritize conceptual constraints over purely formal ones when necessary.
- Run the new prompt.
- Compare the output with that obtained from the unstructured version.
- Check whether the text better respects the format, audience and conceptual criteria.
- Now imagine that the result is too technical: indicate which section of the template you would modify.
- If it instead confused phishing and malware, indicate which section you would modify.
- If it reproduced the imperative phrases in the material verbatim, identify what structural problem might be present.
- Conclude by explaining the three main functions of structuring: segmentation, establishing a hierarchy and improving readability for design.

Information order and long context

You must ask a system to analyze a very long document.
The final purpose is: Identify the three main issues with the project described in the document and propose a possible solution for each.
Prepare three versions of the same prompt.

Version A

- Place the request at the beginning.
- Then insert the document.
- Do not add a final instruction.

Version B

- Place the general instructions at the beginning.
- Insert the document.
- Place the specific request immediately after the material.

Version C

- Insert the general instructions.
- Insert the document
- Add at the end:

“Based exclusively on the preceding material, now identify the three main issues and propose a solution for each. If the document does not contain sufficient information to support a conclusion, state this explicitly.”

For each version:

- Save the prompt.
- Save the output.

- Check whether the main request is followed.
- Check whether information actually present in the document is used.
- Compare how much the three responses stray from the task.
- Check whether the final instruction improves the connection between the material and the task.
- Identify any important information that appears to have been ignored.
- Remove some irrelevant information from the document and repeat the test.
- Group the material under descriptive headings and repeat the test again.
- Explain why, in a very long context, it is inadvisable to bury an essential constraint in the middle of large amounts of text.
- Connect what you observe to the problem described in the handout as Lost in the Middle, remembering that it is not a universal law applicable to every model and every task.
- Conclude by proposing the organization you consider most effective among: General instructions → materials → specific request → essential output constraints.

Bibliography

Anthropic. (n.d.). *Prompting best practices*. Claude Platform Docs. Retrieved September 6, 2026, from https://platform.claude.com/docs/en/build-with-claude/prompt-engineering/claude-prompting-best-practices

Bsharat, S. M., Myrzakhan, A., & Shen, Z. (2024). *Principled instructions are all you need for questioning LLaMA-1/2, GPT-3.5/4*. arXiv. https://doi.org/10.48550/arXiv.2312.16171

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language models are few-shot learners*. Advances in Neural Information Processing Systems, 33, 1877–1901. https://arxiv.org/abs/2005.14165

CommonMark Group. (2024). *CommonMark spec (version 0.31.2)*. https://spec.commonmark.org/0.31.2/

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

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., & Neubig, G. (2023). *Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing*. ACM Computing Surveys, 55(9), Article 195, 1–35. https://doi.org/10.1145/3560815

Lu, Y., Bartolo, M., Moore, A., Riedel, S., & Stenetorp, P. (2022). *Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity*. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 8086–8098). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.556

OpenAI. (n.d.). *Prompt engineering*. OpenAI API. Retrieved September 6, 2026, from https://developers.openai.com/api/docs/guides/prompt-engineering

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). *Training language models to follow instructions with human feedback*. Advances in Neural Information Processing Systems, 35, 27730–27744. https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html

Reynolds, L., & McDonell, K. (2021). *Prompt programming for large language models: Beyond the few-shot paradigm*. arXiv. https://doi.org/10.48550/arXiv.2102.07350

Sclar, M., Choi, Y., Tsvetkov, Y., & Suhr, A. (2024). *Quantifying language models' sensitivity to spurious features in prompt design or: How I learned to start worrying about prompt formatting*. In The Twelfth International Conference on Learning Representations. https://openreview.net/forum?id=RIu5lyNXjT

White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., & Schmidt, D. C. (2023). *A prompt pattern catalog to enhance prompt engineering with ChatGPT*. arXiv. https://doi.org/10.48550/arXiv.2302.11382

Zamfirescu-Pereira, J. D., Wong, R. Y., Hartmann, B., & Yang, Q. (2023). *Why Johnny can't prompt: How non-AI experts try (and fail) to design LLM prompts*. In Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (Article 437, pp. 1–21). Association for Computing Machinery. https://doi.org/10.1145/3544548.3581388

Further reading

- James Phoenix and Mike Taylor, Prompt Engineering for Generative AI: Future-Proof Inputs for Reliable AI Outputs, O'Reilly Media, 2024 — Probably the best overall companion to this handout. Its core principles include giving direction, specifying formats, providing examples, evaluating quality, and decomposing work. These ideas map very closely onto the handout's treatment of priming, prompt templates, explicit task structure, examples, and output constraints. oreilly.com
- John Berryman and Albert Ziegler, Prompt Engineering for LLMs: The Art and Science of Building Large Language Model-Based Applications, O'Reilly Media, 2024 — Particularly useful for understanding why wording, ordering, context and structure can influence probabilistic generation without deterministically controlling it. The book connects practical prompting with tokenization, autoregressive generation, probabilities, context and LLM behavior, which makes it a strong theoretical complement to the sections on priming and underspecification. oreilly.com
- Jay Alammar and Maarten Grootendorst, Hands-On Large Language Models: Language Understanding and Generation, O'Reilly Media, 2024 — A very strong reference for prompt ingredients, in-context learning, examples, chain prompting, constrained output and output verification. Its dedicated Prompt Engineering chapter

makes it particularly appropriate for the parts of the handout that explain how examples, formatting and task decomposition can guide generation. oreilly.com

- Mike Taylor, Context Engineering with DSPy, O'Reilly Media, 2026 — The closest current reference to the long-context sections of the handout. It explicitly covers context underspecification, context distraction, context overflow, context fragmentation and Lost in the Middle, as well as context editing, summarization, prompt optimization, few-shot learning and retrieval. It is therefore especially relevant to the final third of the chapter. oreilly.com
- Chip Huyen, AI Engineering: Building Applications with Foundation Models, O'Reilly Media, 2025 — A broader complement that is especially useful for moving from isolated prompts toward systematic prompt design, evaluation and context management inside real AI applications. Its structure includes dedicated treatments of evaluation, Prompt Engineering, RAG, agents and AI application architecture.

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.

