Gemini 2.0 Flash is now production ready!

Learn more (https://developers.googleblog.com/en/gemini-2-family-expands/)

Prompt design strategies

This page introduces you to some general prompt design strategies that you can employ when designing prompts.

Large language models (LLM) are trained on vast amounts of text data to learn the patterns and relationships between language. When given some text (the prompt), language models can predict what is likely to come next, like a sophisticated autocompletion tool.

Google AI Studio hosts a simple prompt gallery (https://aistudio.google.com/app/gallery) meant to interactively showcase many of the concepts shared in this guide. In the rest of this guide, you'll learn about the following prompt design strategies:

- Give clear and specific instructions (#give-clear-and-specific-instructions)
- Include few-shot examples (#include-few-shot-examples)
- Add contextual information (#add-contextual-information)
- Add prefixes (#add-prefixes)
- Let the model complete partial input (#let-the-model-complete-partial-input)

Give clear and specific instructions

Giving the model instructions on what to do is an effective and efficient way to customize model behavior. Ensure that the instructions you give are clear and specific. Instructions can be as simple as a list of step-by-step instructions or as complex as mapping out a user's experience and mindset.

Define the task to perform

Describe in detail the task that you want the model to perform. The task can be as simple as a step-by-step list of instructions or as complex as mapping out a user's experience and mindset. The following prompt tells the model to summarize a block of text:

:

```
rize this text:
 A quantum computer exploits quantum mechanical phenomena to perform calculations exponentially
r than any modern traditional computer. At very tiny scales, physical matter acts as both
cles and as waves, and quantum computing uses specialized hardware to leverage this behavior.
perating principles of quantum devices is beyond the scope of classical physics. When deployed
ale, quantum computers could be used in a wide variety of applications such as: in
security to break existing encryption methods while helping researchers create new ones, in
rology to develop better weather forecasting etc. However, the current state of the art quantum
ters are still largely experimental and impractical.
```

se:

```
um computers leverage the wave-particle duality of matter at tiny scales to perform
lations exponentially faster than traditional computers. They harness quantum phenomena,
 are beyond the scope of classical physics, to achieve this speed advantage. While
sing for applications like cybersecurity, weather forecasting, and beyond, current
um computers are still experimental and not yet practical for widespread use.
```

(gemini-1.

Specify any constraints

Specify any constraints on reading the prompt or generating a response. You can tell the model what to do and not to do. From the previous example, the summarization is shorter than the original text, but maybe you want the summary to be shorter. In this case, you can specify a constraint in the prompt on how long you want the summary to be:

:

```
rize this text in two sentences:
 A quantum computer exploits quantum mechanical phenomena to perform calculations exponentially
r than any modern traditional computer. At very tiny scales, physical matter acts as both
cles and as waves, and quantum computing uses specialized hardware to leverage this behavior.
perating principles of quantum devices is beyond the scope of classical physics. When deployed
ale, quantum computers could be used in a wide variety of applications such as: in
security to break existing encryption methods while helping researchers create new ones, in
rology to develop better weather forecasting etc. However, the current state of the art quantum
ters are still largely experimental and impractical.
```

se:

```
um computers harness the wave-particle duality of matter at the quantum level to perform
tations exponentially faster than traditional computers. While these machines hold
se potential for fields like cybersecurity and weather forecasting, they are currently
eir early stages of development and not yet practical for widespread use.
```

(gemini-1.

Define the format of the response

You can also give instructions that specify the format of the response. For example, you can ask for the response to be formatted as a table, bulleted list, elevator pitch, keywords, sentence, or paragraph. The following system message instructs the model to be more conversational in its response:

message

```
|uestions should be answered comprehensively with details, unless the user requests a concise respc
.fically.
```

```
 is a smart way to make a business that sells DVD's in 2025?
```

se:

```
 you're looking at a real uphill battle here. The DVD market is significantly diminished due to
iming services, digital downloads, and Blu-ray. However, that doesn't necessarily mean it's impossi
iild a DVD-based business in 2025. It just requires a hyper-focused strategy and a deep understandi
ie remaining market. Here's a breakdown of smart approaches:

iderstanding the Challenges and Opportunities
.enges:
iance of Streaming: Netflix, Amazon Prime Video, Disney+, and countless other services offer vast
iries on-demand.
:al Downloads/Rentals: Platforms like iTunes, Google Play, and Vudu make purchasing or renting digi
.ons easy.
-ay & 4K Ultra HD: Offer superior picture and sound quality, often preferred by home theater
isiasts.
```

(gemini-2.

Another example which we used in the past is:

message

```
|uestions should be answered comprehensively with details, unless the user requests a
.se response specifically. Respond in the same language as the query.
```

```
 is a smart way to create a lemonade business?
```

se:

```
:hing a Successful Lemonade Stand: A Smart Approach
:ing a lemonade business, whether a simple stand or a larger operation, requires strategic thinkinç
s a breakdown of how to approach it smartly:

:fine Your Business:
:: Are you aiming for a small, seasonal lemonade stand, a food truck, a delivery service, or someth
 entirely?
```

```
:t Audience: Who are you selling to? Families, office workers, tourists?
ıe Selling Proposition: What sets your lemonade apart? Special flavors, organic ingredients, a unic
ıe?

.ng: Research competitor prices and determine a profitable pricing structure.
:ion: Consider foot traffic, accessibility, and regulations for your chosen location.
```

(gemini-1.

Summary

- Give the models instructions on what to do.
- Make the instructions clear and specific.
- Specify any constraints or formatting requirements for the output.

Include few-shot examples

You can include examples in the prompt that show the model what getting it right looks like. The model attempts to identify patterns and relationships from the examples and applies them when generating a response. Prompts that contain a few examples are called *few-shot* prompts, while prompts that provide no examples are called *zero-shot prompts*. Few-shot prompts are often used to regulate the formatting, phrasing, scoping, or general patterning of model responses. Use specific and varied examples to help the model narrow its focus and generate more accurate results.

We recommend to always include few-shot examples in your prompts. Prompts without few-shot examples are likely to be less effective. In fact, you can remove instructions from your prompt if your examples are clear enough in showing the task at hand.

Zero-shot vs few-shot prompts

The following zero-shot prompt asks the model to choose the best explanation.

:

```
;e choose the best explanation to the question:

:ion: How is snow formed?
ınation1: Snow is formed when water vapor in the air freezes into ice crystals in the
;phere, which can combine and grow into snowflakes as they fall through the atmosphere and
ıulate on the ground.
ınation2: Water vapor freezes into ice crystals forming snow.
:r:
```

se:

```
ınation 1 is the best explanation.
```

(gemini-1.

If your use case requires the model to produce concise responses, you can include examples in the prompt that give preference to concise responses.

The following prompt provides two examples that show preference to the shorter explanations. In the response, you can see that the examples guided the model to choose the shorter explanation (Explanation2) as opposed to the longer explanation (Explanation1) like it did previously.

:

```
 are some examples showing a question, explanation, and answer format:

ion: Why is sky blue?
nation1: The sky appears blue because of Rayleigh scattering, which causes shorter blue
engths of light to be scattered more easily than longer red wavelengths, making the sky look

nation2: Due to Rayleigh scattering effect.
r: Explanation2

ion: What is the cause of earthquakes?
nation1: Sudden release of energy in the Earth's crust.
nation2: Earthquakes happen when tectonic plates suddenly slip or break apart, causing a
se of energy that creates seismic waves that can shake the ground and cause damage.
r: Explanation1

Answer the following question given the example formats above:

ion: How is snow formed?
nation1: Snow is formed when water vapor in the air freezes into ice crystals in the
phere, which can combine and grow into snowflakes as they fall through the atmosphere and
ulate on the ground.
nation2: Water vapor freezes into ice crystals forming snow.
r:
```

se:

```
r: Explanation2
```

(gemini-1.

Find the optimal number of examples

You can experiment with the number of examples to provide in the prompt for the most desired results. Models like Gemini can often pick up on patterns using a few examples, though you may need to experiment with what number of examples lead to the desired results. At the same time, if you include too many examples, the model may start to overfit (https://developers.google.com/machine-learning/glossary#overfitting) the response to the examples.

Use examples to show patterns instead of antipatterns

Using examples to show the model a pattern to follow is more effective than using examples to show the model an antipattern to avoid.

⛔ **Negative pattern**:

```
e


 end haikus with a question:
 are fun
rt and simple poem
 you enjoy them?
```

✅ **Positive pattern**:

```
e


s end haikus with an assertion:
 are fun
rt and simple poem
 to write
```

Use consistent formatting across examples

Make sure that the structure and formatting of few-shot examples are the same to avoid responses with undesired formats. One of the primary objectives of adding few-shot examples in prompts is to show the model the response format. Therefore, it is essential to ensure a consistent format across all examples, especially paying attention to XML tags, white spaces, newlines, and example splitters.

Summary

- Including prompt-response examples in the prompt helps the model learn how to respond.
- Give the model examples of the patterns to follow instead of examples of patterns to avoid.
- Experiment with the number of prompts to include. Depending on the model, too few examples are ineffective at changing model behavior. Too many examples cause the model to overfit.
- Use consistent formatting across examples

Add contextual information

You can include in the prompt instructions and information that the model needs to solve a problem instead of assuming that the model has all of the required information. This contextual information helps the model understand the constraints and details of what you're asking for it to do.

The following example asks the model to give troubleshooting guidance for a router:

:

```
should I do to fix my disconnected wifi? The light on my Google Wifi router is yellow and
ing slowly.
```

se:

```
low blinking light on your Google Wifi router usually indicates a problem with the internet
ction. Here are some steps to troubleshoot and fix the issue:
start your router and modem:
lug your modem from the power outlet and wait for 30 seconds.
g your modem back in and wait for it to fully boot up.
lug your Google Wifi router and wait for 30 seconds.
g your Google Wifi router back in and wait for it to fully boot up.
```

(gemini-1.

The response looks like generic troubleshooting information that's not specific to the router or the status of the LED indicator lights.

To customize the response for the specific router, you can add to the prompt the router's troubleshooting guide as context for it to refer to when providing a response.

:

```
r the question using the text below. Respond with only the text provided.
ion: What should I do to fix my disconnected wifi? The light on my Google Wifi router is yellow ar
ing slowly.


: Slowly pulsing yellow
it means: There is a network error.
to do:
that the Ethernet cable is connected to both your router and your modem and both devices are turr
ou might need to unplug and plug in each device again.

: Fast blinking yellow
it means: You are holding down the reset button and are factory resetting this device.
to do:
u keep holding down the reset button, after about 12 seconds, the light will turn solid yellow. Or
solid yellow, let go of the factory reset button.

: Solid yellow
it means: Router is factory resetting.
to do:
can take up to 10 minutes. When it's done, the device will reset itself and start pulsing white,
ng you know it's ready for setup.

: Solid red
it means: Something is wrong.
```

```
 to do:
.cal failure. Factory reset the router. If the light stays red, contact Wifi customer support.
```

se:

```
: that the Ethernet cable is connected to both your router and your modem and both devices are turr
'ou might need to unplug and plug in each device again.
```

(gemini-1.

Summary

- Include information (context) in the prompt that you want the model to use when generating a response.
- Give the model instructions on how to use the contextual information.

Add prefixes

A prefix is a word or phrase that you add to the prompt content that can serve several purposes, depending on where you put the prefix:

- **Input prefix:** Adding a prefix to the input signals semantically meaningful parts of the input to the model. For example, the prefixes "English:" and "French:" demarcate two different languages.
- **Output prefix:** Even though the output is generated by the model, you can add a prefix for the output in the prompt. The output prefix gives the model information about what's expected as a response. For example, the output prefix "JSON:" signals to the model that the output should be in JSON format.
- **Example prefix:** In few-shot prompts, adding prefixes to the examples provides labels that the model can use when generating the output, which makes it easier to parse output content.

In the following example, "Text:" is the input prefix and "The answer is:" is the output prefix.

:

```
;ify the text as one of the following categories.
'ge
ıll
 Rhino
ınswer is: large
 Mouse
ınswer is: small
 Snail
ınswer is: small
 Elephant
ınswer is:
```

se:

```
nswer is: large
```

(gemini-1.

Let the model complete partial input

Generative language models work like an advanced autocompletion tool. When you provide partial content, the model can provide the rest of the content or what it thinks is a continuation of that content as a response. When doing so, if you include any examples or context, the model can take those examples or context into account.

The following example provides a prompt with an instruction and an entity input:

:

```
he given order, return a JSON object that has the fields cheeseburger, hamburger, fries, or
, with the value being the quantity.

: A burger and a drink.
```

se:

```
urger": 1,
k": 1
```

(gemini-1.

While the model did as prompted, writing out the instructions in natural language can sometimes be challenging and it leaves a lot to the model's interpretation. For example, a restaurants menu might contain many items. To reduce the size of the JSON response, you probably want to omit the items that weren't ordered. In this case, you can give an example and a response prefix and let the model complete it:

:

```
 fields are cheeseburger, hamburger, fries, and drink.
: Give me a cheeseburger and fries
t:


seburger": 1,
s": 1
```

```
: I want two burgers, a drink, and fries.
t:
```

se:

```
urger": 2,
k": 1,
s": 1
```

(gemini-1.

Notice how "cheeseburger" was excluded from the output because it wasn't a part of the order.

Prompt the model to format its response

The completion strategy can also help format the response. The following example prompts the model to create an essay outline:

:

```
e an outline for an essay about hummingbirds.
```

se:

```
ngbirds: A Symphony of Speed and Color
troduction
mmingbirds are fascinating creatures known for their dazzling colors, lightning-fast flight, and
e feeding habits.
ief overview of the diversity of hummingbird species and their geographic distribution.
esis statement: Hummingbirds are an embodiment of resilience, adaptation, and the wonders of the
al world.
```

(gemini-1.

The prompt didn't specify the format for the outline and the model chose a format for you. To get the model to return an outline in a specific format, you can add text that represents the start of the outline and let the model complete it based on the pattern that you initiated.

:

```
e an outline for an essay about hummingbirds.
troduction
```

se:

```
ngbirds: Jewels of the Air
 Introduction**
ook:** Begin with a captivating image or anecdote about hummingbirds, emphasizing their vibrant
s, rapid flight, and unique feeding habits.
ackground:** Briefly introduce hummingbirds as the smallest birds in the world and their fascinati
ations for a nectar-rich diet.
hesis:** State the main argument or focus of the essay. Examples:
he incredible adaptations of hummingbirds enable them to thrive in diverse ecosystems.**
ummingbirds play a crucial role in pollination and biodiversity.**
he beauty and grace of hummingbirds make them captivating subjects for both scientific study and
tic appreciation.**
```

(gemini-1.

Summary

- If you give the model a partial input, the model completes that input based on any available examples or context in the prompt.
- Having the model complete an input may sometimes be easier than describing the task in natural language.
- Adding a partial answer to a prompt can guide the model to follow a desired pattern or format.

Break down prompts into simple components

For use cases that require complex prompts, you can help the model manage this complexity by breaking things down into simpler components.

Break down instructions

Instead of having many instructions in one prompt, create one prompt per instruction. You can choose which prompt to process based on the user's input.

Chain prompts

For complex tasks that involve multiple sequential steps, make each step a prompt and chain the prompts together in a sequence. In this sequential chain of prompts, the output of one prompt in the sequence becomes the input of the next prompt. The output of the last prompt in the sequence is the final output.

Aggregate responses

Aggregation is when you want to perform different parallel tasks on different portions of the data and aggregate the results to produce the final output. For example, you can tell the model to perform one operation on the first part of the data, perform another operation on the rest of the data and aggregate the results.

Summary

- Break down complex instructions into a prompt for each instruction and decide which prompt to apply based on the user's input.
- Break down multiple sequential steps into separate prompts and chain them such that the output on the preceding prompt becomes the input of the following prompt.
- Break down parallel tasks and aggregate the responses to produce the final output.

Experiment with different parameter values

Each call that you send to a model includes parameter values that control how the model generates a response. The model can generate different results for different parameter values. Experiment with different parameter values to get the best values for the task. The parameters available for different models may differ. The most common parameters are the following:

- Max output tokens
- Temperature
- Top-K
- Top-P

Max output tokens

Maximum number of tokens that can be generated in the response. A token is approximately four characters. 100 tokens correspond to roughly 20 words.

Specify a lower value for shorter responses and a higher value for longer responses.

Temperature

The temperature is used for sampling during response generation, which occurs when `topP` and `topK` are applied. Temperature controls the degree of randomness in token selection. Lower temperatures are good for prompts that require a more deterministic and less open-ended or creative response, while higher temperatures can lead to more diverse or creative results. A temperature of `0` is deterministic, meaning that the highest probability response is always selected.

For most use cases, try starting with a temperature of `0.2`. If the model returns a response that's too generic, too short, or the model gives a fallback response, try increasing the temperature.

Top-K

Top-K changes how the model selects tokens for output. A top-K of `1` means the next selected token is the most probable among all tokens in the model's vocabulary (also called greedy decoding), while a top-K of `3` means that the next token is selected from among the three most probable tokens by using temperature.

For each token selection step, the top-K tokens with the highest probabilities are sampled. Then tokens are further filtered based on top-P with the final token selected using temperature sampling.

Specify a lower value for less random responses and a higher value for more random responses. The default top-K is `40`.

Top-P

Top-P changes how the model selects tokens for output. Tokens are selected from the most (see top-K) to least probable until the sum of their probabilities equals the top-P value. For example, if tokens A, B, and C have a probability of 0.3, 0.2, and 0.1 and the top-P value is `0.5`, then the model will select either A or B as the next token by using temperature and excludes C as a candidate.

Specify a lower value for less random responses and a higher value for more random responses. The default top-P is `0.95`.

Prompt iteration strategies

Prompt design is an iterative process that often requires a few iterations before you get the desired response consistently. This section provides guidance on some things you can try when iterating on your prompts.

Use different phrasing

Using different words or phrasing in your prompts often yields different responses from the model even though they all mean the same thing. If you're not getting the expected results from your prompt, try rephrasing it.

```
on 1:
o I bake a pie?

on 2:
st a recipe for a pie.

on 3:
 s a good pie recipe?
```

Switch to an analogous task

If you can't get the model to follow your instructions for a task, try giving it instructions for an analogous task that achieves the same result.

This prompt tells the model to categorize a book by using predefined categories.

:

```
 category does The Odyssey belong to:
ler
i
logy
aphy
```

se:

```
dyssey belongs to the category of **mythology**.

s why:

ythology:** The Odyssey tells the story of Odysseus, a hero from Greek mythology, and his
ey home after the Trojan War. It features gods, monsters, and supernatural events common to
 mythology.
```

(gemini-1.

The response is correct, but the model didn't stay within the bounds of the options. You also want to model to just respond with one of the options instead of in a full sentence. In this case, you can rephrase the instructions as a multiple choice question and ask the model to choose an option.

:

```
ple choice problem: Which of the following options describes the book The Odyssey?
ns:
iller
-fi
hology
graphy
```

se:

```
orrect answer is **mythology**.
```

(gemini-1.

Change the order of prompt content

The order of the content in the prompt can sometimes affect the response. Try changing the content order and see how that affects the response.

```
Version 1:
[examples]
[context]
[input]

Version 2:
[input]
[examples]
[context]

Version 3:
[examples]
```

```
[input]
[context]
```

Fallback responses

A fallback response is a response returned by the model when either the prompt or the response triggers a safety filter. An example of a fallback response is "I'm not able to help with that, as I'm only a language model."

If the model responds with a fallback response, try increasing the temperature.

Things to avoid

- Avoid relying on models to generate factual information.
- Use with care on math and logic problems.

Next steps

- Now that you have a deeper understanding of prompt design, try writing your own prompts using Google AI Studio (http://aistudio.google.com).
- To learn about multimodal prompting, see Prompting with media files (/gemini-api/docs/prompting_with_media).

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the Apache 2.0 License (https://www.apache.org/licenses/LICENSE-2.0). For details, see the Google Developers Site Policies (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-02-25 UTC.