

# Prompt engineering concepts

*Prompt engineering* refers to the practice of optimizing textual input to a Large Language Model (LLM) to obtain desired responses. Prompting helps a LLM perform a wide variety of tasks, including classification, question answering, code generation, creative writing, and more. The quality of prompts that you provide to a LLM can impact the quality of the model's responses. This section provides you the necessary information to get started with prompt engineering. It also covers tools to help you find the best possible prompt format for your use case when using a LLM on Amazon Bedrock.

## Note

All examples in this doc are obtained via API calls. The response may vary due to the stochastic nature of the LLM generation process. If not otherwise specified, the prompts are written by employees of AWS.

## Note

To reduce hallucinations, you can refine your prompt using prompt optimization techniques; use techniques like Retrieval Augmented Generation (RAG) to provide the model access to more relevant data; or use a different model that might produce improved results.

Amazon Bedrock includes models from a variety of providers. The following is a list prompt engineering guidelines for those models.

- **Amazon Nova Micro, Lite, and Pro prompt guide:** [Prompting best practices for Amazon Nova understanding models](#)
- **Amazon Nova Canvas prompt guide:** [Generating images with Amazon Nova](#)
- **Amazon Nova Reel prompt guide:** [Generating videos with Amazon Nova](#)
- **Anthropic Claude model prompt guide:** <https://docs.anthropic.com/en/docs/build-with-claude/prompt-engineering/overview>
- **Cohere prompt guide:** <https://txt.cohere.com/how-to-train-your-pet-llm-prompt-engineering>
- **AI21 Labs Jurassic model prompt guide:** <https://docs.ai21.com/docs/prompt-engineering>

- **Meta Llama 2 prompt guide:** <https://ai.meta.com/llama/get-started/#prompting>
- **Stability AI prompt guide:** <https://platform.stability.ai/docs/getting-started>
- **Mistral AI prompt guide:** [https://docs.mistral.ai/guides/prompting\\_capabilities/](https://docs.mistral.ai/guides/prompting_capabilities/)

**Disclaimer:** The examples in this document use the current text models available within Amazon Bedrock. Also, this document is for general prompting guidelines. For model-specific guides, refer to their respective docs on Amazon Bedrock. This document provides a starting point. While the following example responses are generated using specific models on Amazon Bedrock, you can use other models in Amazon Bedrock to get results as well. The results may differ between models as each model has its own performance characteristics. The output that you generate using AI services is your content. Due to the nature of machine learning, output may not be unique across customers and the services may generate the same or similar results across customers.

## What is a prompt?

Prompts are a specific set of inputs provided by you, the user, that guide LLMs on Amazon Bedrock to generate an appropriate response or output for a given task or instruction.

**User Prompt:**

*Who invented the airplane?*

When queried by this prompt, Titan provides an output:

**Output:**

*The Wright brothers, Orville and Wilbur Wright are widely credited with inventing and manufacturing the world's first successful airplane.*

(Source of prompt: AWS, model used: Amazon Titan Text)

## Components of a prompt

A single prompt includes several components, such as the task or instruction you want the LLMs to perform, the context of the task (for example, a description of the relevant domain), demonstration examples, and the input text that you want LLMs on Amazon Bedrock to use in its response. Depending on your use case, the availability of the data, and the task, your prompt should combine one or more of these components.

Consider this example prompt asking Titan to summarize a review:

**User Prompt:**

*The following is text from a restaurant review:*

*"I finally got to check out Alessandro's Brilliant Pizza and it is now one of my favorite restaurants in Seattle. The dining room has a beautiful view over the Puget Sound but it was surprisingly not crowded. I ordered the fried castelvetrano olives, a spicy Neapolitan-style pizza and a gnocchi dish. The olives were absolutely decadent, and the pizza came with a smoked mozzarella, which was delicious. The gnocchi was fresh and wonderful. The waitstaff were attentive, and overall the experience was lovely. I hope to return soon."*

*Summarize the above restaurant review in one sentence.*

(Source of prompt: AWS)

Based on this prompt, Titan responds with a succinct one-line summary of the restaurant review. The review mentions key facts and conveys the main points, as desired.

**Output:**

*Alessandro's Brilliant Pizza is a fantastic restaurant in Seattle with a beautiful view over Puget Sound, decadent and delicious food, and excellent service.*

(Model used: Amazon Titan Text)

The instruction **Summarize the above restaurant review in one sentence** and the review text **I finally got to check out ...** were both necessary for this type of output. Without either one, the model would not have enough information to produce a sensible summary. The *instruction* tells the LLM what to do, and the text is the *input* on which the LLM operates. The *context* (**The following is text from a restaurant review**) provides additional information and keywords that guide the model to use the input when formulating its output.

In the example below, the text **Context: Climate change threatens people with increased flooding ...** is the *input* which the LLM can use to perform the *task* of answering the question **Question: What organization calls climate change the greatest threat to global health in the 21st century?"**.

**User prompt:**

*Context: Climate change threatens people with increased flooding, extreme heat, increased food and water scarcity, more disease, and economic loss. Human migration and conflict can also be a result. The World Health Organization (WHO) calls climate change the greatest threat to global health in the 21st century. Adapting to climate change through efforts like flood control measures or drought-resistant crops partially reduces climate change risks, although some limits to adaptation have already been reached. Poorer communities are responsible for a small share of global emissions, yet have the least ability to adapt and are most vulnerable to climate change. The expense, time required, and limits of adaptation mean its success hinge on limiting global warming.*

*Question: What organization calls climate change the greatest threat to global health in the 21st century?*

(Source of prompt: [https://en.wikipedia.org/wiki/Climate\\_change](https://en.wikipedia.org/wiki/Climate_change))

AI21 Labs Jurassic responses with the correct name of the organization according to the context provided in the prompt.

**Output:**

*The World Health Organization (WHO) calls climate change the greatest threat to global health in the 21st century.*

(Model used: AI21 Labs Jurassic-2 Ultra v1)

## Few-shot prompting vs. zero-shot prompting

It is sometimes useful to provide a few examples to help LLMs better calibrate their output to meet your expectations, also known as *few-shot prompting* or *in-context learning*, where a *shot* corresponds to a paired example input and the desired output. To illustrate, first here is an example of a zero-shot sentiment classification prompt where no example input-output pair is provided in the prompt text:

**User prompt:**

*Tell me the sentiment of the following headline and categorize it as either positive, negative or neutral:  
New airline between Seattle and San Francisco offers a great opportunity for both passengers and investors.*

(Source of prompt: AWS)

**Output:**

Positive

(Model used: Amazon Titan Text)

Here is the few-shot version of a sentiment classification prompt:

**User prompt:**

*Tell me the sentiment of the following headline and categorize it as either positive, negative or neutral. Here are some examples:*

*Research firm fends off allegations of impropriety over new technology.*

*Answer: Negative*

*Offshore windfarms continue to thrive as vocal minority in opposition dwindles.*

*Answer: Positive*

*Manufacturing plant is the latest target in investigation by state officials.*

*Answer:*

(Source of prompt: AWS)

**Output:**

Negative

(Model used: Amazon Titan Text)

The following example uses Anthropic Claude models. When using Anthropic Claude models, it's a good practice to use <example></example> tags to include demonstration examples. We also recommend using different delimiters such as H: and A: in the examples to avoid confusion with the delimiters Human: and Assistant: for the whole prompt. Notice that for the last few-shot example, the final A: is left off in favor of Assistant:, prompting Anthropic Claude to generate the answer instead.

**User prompt:**

*Human: Please classify the given email as "Personal" or "Commercial" related emails.  
Here are some examples.*

*<example>*

H: Hi Tom, it's been long time since we met last time. We plan to have a party at my house this weekend. Will you be able to come over?

A: Personal

</example>

<example>

H: Hi Tom, we have a special offer for you. For a limited time, our customers can save up to 35% of their total expense when you make reservations within two days. Book now and save money!

A: Commercial

</example>

H: Hi Tom, Have you heard that we have launched all-new set of products. Order now, you will save \$100 for the new products. Please check our website.

Assistant:

**Output:**

Commercial

(Source of prompt: AWS, model used: Anthropic Claude)

## Prompt template

A prompt template specifies the formatting of the prompt with exchangeable content in it. Prompt templates are “recipes” for using LLMs for different use cases such as classification, summarization, question answering, and more. A prompt template may include instructions, few-shot examples, and specific context and questions appropriate for a given use case. The following example is a template that you can use to perform few-shot sentiment classification using Amazon Bedrock text models:

**Prompt template:**

```
"""Tell me the sentiment of the following
{{Text Type, e.g., "restaurant review"}} and categorize it
as either {{Sentiment A}} or {{Sentiment B}}.
```

Here are some examples:

Text: {{Example Input 1}}

Answer: {{Sentiment A}}

Text: {{Example Input 2}}

Answer: {{Sentiment B}}

*Text: {{Input}}*

*Answer: """*

Users can copy and paste this template, filling in their own text and few-shot examples to complete prompts when using LLMs on Amazon Bedrock. More templates are provided in this section.

 **Note**

The double curly braces {{ and }} mark the places to put in data-specific information in the template, and should not be included in the prompt text.

## Maintain recall over Amazon Bedrock inference requests

If you access Amazon Bedrock models by using API calls, be aware that models don't recall prior prompts and previous requests, unless the previous interaction is included within the current prompt. Including previous prompts in a request lets you have conversational-style interactions or make follow up requests.

If you are accessing Anthropic Claude models using API calls instead of the Amazon Bedrock text playground, you wrap your prompts in a conversational style to get desired response (for the Titan Text and AI21 Labs Jurassic model family, it's not necessary). The following example demonstrates a conversational-style prompt:

**User prompt:**

*Human: Describe where sea otters live in two sentences.*

**Assistant:**

**Output:**

*Here is a two sentence description of where sea otters live:*

*Sea otters live in the northern Pacific Ocean along the coasts of Russia, Alaska, Canada, and the continental United States. They prefer coastal habitats like rocky shores, kelp forests, and estuaries where they have access to shallow, nearshore waters.*

(Source of prompt: AWS, model used: Anthropic Claude)

Notice how the main content of the prompt is wrapped like this: \n\nHuman: {{Main Content}}\n\nAssistant:. For Anthropic Claude models, prompts sent via the API must contain \n\nHuman: and \n\nAssistant:.

To use conversational mode on Titan, you can use the format of User: {{}} \n Bot: when prompting the model.

## What is prompt engineering?

Prompt engineering refers to the practice of crafting and optimizing input prompts by selecting appropriate words, phrases, sentences, punctuation, and separator characters to effectively use LLMs for a wide variety of applications. In other words, prompt engineering is the art of communicating with an LLM. High-quality prompts condition the LLM to generate desired or better responses. The detailed guidance provided within this document is applicable across all LLMs within Amazon Bedrock.

The best prompt engineering approach for your use case is dependent on both the task and the data. Common tasks supported by LLMs on Amazon Bedrock include:

- **Classification:** The prompt includes a question with several possible choices for the answer, and the model must respond with the correct choice. An example classification use case is sentiment analysis: the input is a text passage, and the model must classify the sentiment of the text, such as whether it's positive or negative, or harmless or toxic.
- **Question-answer, without context:** The model must answer the question with its internal knowledge without any context or document.
- **Question-answer, with context:** The user provides an input text with a question, and the model must answer the question based on information provided within the input text.
- **Summarization:** The prompt is a passage of text, and the model must respond with a shorter passage that captures the main points of the input.
- **Open-ended text generation:** Given a prompt, the model must respond with a passage of original text that matches the description. This also includes the generation of creative text such as stories, poems, or movie scripts.
- **Code generation:** The model must generate code based on user specifications. For example, a prompt could request text-to-SQL or Python code generation.

- **Mathematics:** The input describes a problem that requires mathematical reasoning at some level, which may be numerical, logical, geometric or otherwise.
- **Reasoning or logical thinking:** The model must make a series of logical deductions.
- **Entity extraction:** Entity extraction can extract entities based on a provided input question. You can extract specific entities from text or input based on your prompt.
- **Chain-of-thought reasoning:** Give step-by-step reasoning on how an answer is derived based on your prompt.

## Understanding intelligent prompt routing in Amazon Bedrock

### Note

Intelligent prompt routing in Amazon Bedrock is in preview and is subject to change.

Amazon Bedrock intelligent prompt routing provides a single serverless endpoint to efficiently route requests between different foundational models within the same model family. It can dynamically predict the response quality of each model for each request, and then route the request to the model with the best response quality. This helps to optimize for both response quality and cost.

### Topics

- [Benefits](#)
- [Default and configured prompt routers](#)
- [Considerations and limitations](#)
- [Prompt router criteria and fallback model](#)
- [How intelligent prompt routing works](#)
- [How to use intelligent prompt routing](#)

## Benefits

- **Optimized Response Quality and Cost:** Routes prompts to different foundational models to achieve the best response quality at the lowest cost.
- **Simplified Management:** Eliminates the need for complex orchestration logic.

- **Future-Proof:** Incorporates new models as they become available.

## Default and configured prompt routers

When using intelligent prompt routing, you can either use the default prompt routers provided by Amazon Bedrock, or configure your own prompt routers.

Default prompt routers are pre-configured routing systems provided by Amazon Bedrock. These routers come with predefined settings and are designed to work out-of-the-box with specific foundational models. They provide a straightforward, ready-to-use solution without needing to configure any routing settings. When starting with intelligent prompt routing, we recommend that you experiment using the default routers provided by Amazon Bedrock. During preview, you can choose to use select models in the Anthropic and Meta families.

Configured prompt routers enable you to define your own routing configurations tailored to specific needs and preferences. They are more suitable when you require more control over how to route your requests and which models to use. Configured routers enable optimization based on response quality metrics and use cases. After you've experimented with default routers, you can configure your own routers that are suitable to your applications, evaluate the response quality in the playground, and use for production applications if it meets the requirements.

## Considerations and limitations

The following are considerations and limitations for intelligent prompt routing in Amazon Bedrock.

- Intelligent prompt routing is only optimized for English prompts.
- Intelligent prompt routing can't adjust routing decisions or responses based on application-specific performance data.
- Intelligent prompt routing might not always provide the most optimal routing for unique or specialized use cases. How effective the routing is depends on the initial training data.

## Prompt router criteria and fallback model

When configuring your prompt routers, you can specify the routing criteria, which is used to determine which model to select for processing a request based on the response quality difference. Use this criteria to determine how much closer the responses of the fallback model should be to the responses of the other models.

## Fallback models

Choose a fallback model that works well for your requests. This model serves as a reliable baseline. You can then choose another model to either improve accuracy or reduce costs compared to the fallback model. The fallback model acts as an anchor, and the routing criteria determines when to switch to the other model based on the response quality difference.

### Response quality difference

The response quality difference measures the disparity between the responses of the fallback model and the other models. A smaller value indicates that the responses are similar. A higher value indicates a significant difference in the responses between the fallback model and the other models.

For example, a response quality difference of 10% means that, say the response quality of the fallback model, Claude Haiku3, is 10%, then the router will switch to another model, say Claude Sonnet3, only if its responses are 10% better than Claude Haiku3's responses.

## How intelligent prompt routing works

### 1. Model selection and router configuration

Choose the family of models you want to use for your application. If you're using default prompt routers, you can choose from models in the Anthropic or Meta families. If you're using configured prompt routers, you can choose from additional models and configure the routing criteria. For more information, see [How to use intelligent prompt routing](#).

### 2. Incoming request analysis

For each incoming request, the system analyzes the prompt to understand its content and context.

### 3. Response quality prediction

Amazon Bedrock predicts the response quality of each specified model in the chosen family based on the prompt. If you configured your prompt router, it takes into account the routing criteria, which is the response quality difference, and routes requests to your specified fallback model if the criteria is not met.

#### 4. Model selection and request forwarding

Based on the response quality prediction, Amazon Bedrock dynamically chooses the model that offers the best combination of response quality and cost for the specific request. The request is then forwarded to the chosen model for processing.

#### 5. Response handling

The response from the chosen model is retrieved and returned to the user. The response includes information about the model that was used to process the request.

## How to use intelligent prompt routing

To get started with intelligent prompt routing, use the Amazon Bedrock console, AWS CLI, or AWS SDK.

### Note

To best utilize intelligent prompt routing, you should regularly review performance to take advantage of new models. To optimize your usage, monitor the available performance and cost metrics.

The following sections show you how to use this feature from the console and the CLI. After you configure your prompt router, Amazon Bedrock will perform the steps described in [How intelligent prompt routing works](#) to generate a response from one of the models in the chosen router.

### console

To use intelligent prompt routing from the AWS Management Console:

1. Go to the [Prompt Routers](#) hub in the Amazon Bedrock console. Use your AWS credentials to log in to the console.
2. Choose the model family that you want to use. If you're using the feature for the first time, you can experiment with the default prompt routers. During the preview, you can choose from models in the Anthropic or Meta families. You can then open the playground and experiment with your prompts.

**Note**

You must choose exactly two models within the same family.

3. After you've experimented with the default routers, you can configure your router. You must provide a unique name for your router and an optional description.
4. Define the routing rules for routing the prompts to different models. Specify the models for routing, the routing criteria, and a fallback model to be used in case the routing criteria is not met.
5. You can now open the playground and try different prompts to monitor the performance of your prompt router. By analyzing how prompts are being routed and the performance of each model, you can adjust the routing criteria and models as needed to optimize for performance and cost.

## API

To use intelligent prompt routing using the AWS CLI:

1. After you've experimented with default prompt routers, you can create a router that can handle intelligent prompt routing using the [CreatePromptRouter](#) API or the [create-prompt-router](#) CLI command.

The following command shows an example of creating the prompt router, where:

- `<router-name>` is a unique name for your router.
- `<region>` is the AWS Region where you want to create the router.
- `<modelA>` and `<modelB>` are the models to choose from for routing. For example, `anthropic.claude-3-5-sonnet-20240620-v1:0`.

```
aws bedrock create-prompt-router \
    --prompt-router-name <router-name> \
    --models '[{"modelArn": "arn:aws:bedrock:<region>::foundation-
model/<modelA>"}]' \
    --fallback-model '[{"modelArn": "arn:aws:bedrock:<region>::foundation-
model/<modelB>"}]' \
    --routing-criteria '{"responseQualityDifference": 0.5}'
```

2. Send your prompts to the intelligent prompt router using the AWS CLI. The router will predict the response quality of each model and route the request to the model with the best response quality.
3. Review the response to your request. It will contain information about the model that was used and the routing criteria.
4. As you work with prompt routers, you can also use these additional API operations:
  - [GetPromptRouter](#)
  - [ListPromptRouters](#)
  - [DeletePromptRouter](#)

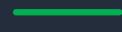
## Design a prompt

Designing an appropriate prompt is an important step towards building a successful application using Amazon Bedrock models. In this section, you learn how to design a prompt that is consistent, clear, and concise. You also learn about how you can control a model's response by using inference parameters. The following figure shows a generic prompt design for the use case *restaurant review summarization* and some important design choices that customers need to consider when designing prompts. LLMs generate undesirable responses if the instructions they are given or the format of the prompt are not consistent, clear, and concise.

**A good example of prompt construction**

The following is text from a restaurant review:  Contextual information about the task.

"I finally got to check out Alessandro's Brilliant Pizza and it is now one of my favorite restaurants in Seattle. The dining room has a beautiful view over the Puget Sound but it was surprisingly not crowded. I ordered the fried Castelvetrano olives, a spicy Neapolitan-style pizza and a gnocchi dish. The olives were absolutely decadent, and the pizza came with a smoked mozzarella, which was delicious. The gnocchi was fresh and wonderful. The waitstaff were attentive, and overall the experience was lovely. I hope to return soon."

 Reference text for the task.

Summarize the above restaurant review in one sentence. 

Simple, clear and complete instructions.

Instructions placed at the end of the prompt.

The form of output is specifically described.

(Source: Prompt written by AWS)

The following content provides guidance on how to create successful prompts.

## Topics

- [Provide simple, clear, and complete instructions](#)
- [Place the question or instruction at the end of the prompt for best results](#)
- [Use separator characters for API calls](#)
- [Use output indicators](#)
- [Best practices for good generalization](#)
- [Optimize prompts for text models on Amazon Bedrock—when the basics aren't good enough](#)
- [Control the model response with inference parameters](#)

## Provide simple, clear, and complete instructions

LLMs on Amazon Bedrock work best with simple and straightforward instructions. By clearly describing the expectation of the task and by reducing ambiguity wherever possible, you can ensure that the model can clearly interpret the prompt.

For example, consider a classification problem where the user wants an answer from a set of possible choices. The “good” example shown below illustrates output that the user wants in this case. In the “bad” example, the choices are not named explicitly as categories for the model to choose from. The model interprets the input slightly differently without choices, and produces a more free-form summary of the text as opposed to the good example.

#### Good example, with output

##### User prompt:

*"The most common cause of color blindness is an inherited problem or variation in the functionality of one or more of the three classes of cone cells in the retina, which mediate color vision."*

*What is the above text about?*

- a) biology
- b) history
- c) geology

##### Output:

- a) biology

#### Bad example, with output

##### User prompt:

*Classify the following text. "The most common cause of color blindness is an inherited problem or variation in the functionality of one or more of the three classes of cone cells in the retina, which mediate color vision."*

##### Output:

*The topic of the text is the causes of colorblindness.*

(Source of prompt: [Wikipedia on color blindness](#), model used: by Titan Text G1 - Express)

## Place the question or instruction at the end of the prompt for best results

Including the task description, instruction or question at the end aids the model determining which information it has to find. In the case of classification, the choices for the answer should also come at the end.

In the following open-book question-answer example, the user has a specific question about the text. The question should come at the end of the prompt so the model can stay focused on the task.

##### User prompt:

*Tensions increased after the 1911–1912 Italo-Turkish War demonstrated Ottoman weakness and led to the formation of the Balkan League, an alliance of Serbia, Bulgaria, Montenegro, and Greece. The League quickly overran most of the Ottomans' territory in the Balkans during the 1912–1913 First Balkan War, much to the surprise of outside observers.*

*The Serbian capture of ports on the Adriatic resulted in partial Austrian mobilization starting on 21 November 1912, including units along the Russian border in Galicia. In a meeting the next day, the Russian government decided not to mobilize in response, unwilling to precipitate a war for which they were not as yet prepared to handle.*

*Which country captured ports?*

**Output:**

*Serbia*

(Source of prompt: [Wikipedia on World War I](#), model used: Amazon Titan Text)

## Use separator characters for API calls

### Use separator characters for API calls

Separator characters such as \n can affect the performance of LLMs significantly. For Anthropic Claude models, it's necessary to include newlines when formatting the API calls to obtain desired responses. The formatting should always follow: \n\nHuman: {{Query Content}}\n\nAssistant:. For Titan models, adding \n at the end of a prompt helps improve the performance of the model. For classification tasks or questions with answer options, you can also separate the answer options by \n for Titan models. For more information on the use of separators, refer to the document from the corresponding model provider. The following example is a template for a classification task.

**Prompt template:**

```
"""{{Text}}\n\n{{Question}}\n\n{{Choice 1}}\n{{Choice 2}}\n{{Choice 3}}"""
```

The following example shows how the presence of newline characters between choices and at the end of a prompt helps Titan produce the desired response.

**User prompt:**

*Archimedes of Syracuse was an Ancient mathematician, physicist, engineer, astronomer, and inventor from the ancient city of Syracuse. Although few details of his life are known, he is regarded as one of the leading scientists in classical antiquity.*

*What was Archimedes? Choose one of the options below.*

- a) astronomer
- b) farmer
- c) sailor

**Output:**

- a) astronomer

(Source of prompt: [Wikipedia on Archimedes](#), model used: Amazon Titan Text)

## Use output indicators

### Output indicators

Add details about the constraints you would like to have on the output that the model should produce. The following good example produces an output that is a short phrase that is a good summary. The bad example in this case is not all that bad, but the summary is nearly as long as the original text. Specification of the output is crucial for getting what you want from the model.

#### Example prompt with clear output constraints indicator

**User prompt:**

*"Charles Mingus Jr. was an American jazz upright bassist, pianist, composer, bandleader, and author. A major proponent of collective improvisation, he is*

#### Example without clear output specifications

**User prompt:**

*"Charles Mingus Jr. was an American jazz upright bassist, pianist, composer, bandleader, and author. A major proponent of collective improvisation, he is considered to be one of the greatest jazz musicians*

considered to be one of the greatest jazz musicians and composers in history, with a career spanning three decades. Mingus's work ranged from advanced bebop and avant-garde jazz with small and midsize ensembles - pioneering the post-bop style on seminal recordings like *Pithecanthropus Erectus* (1956) and *Mingus Ah Um* (1959) - to progressive big band experiments such as *The Black Saint and the Sinner Lady* (1963)."

Please summarize the above text **in one phrase.**

**Output:**

Charles Mingus Jr. is considered one of the greatest jazz musicians of all time.

and composers in history, with a career spanning three decades. Mingus's work ranged from advanced bebop and avant-garde jazz with small and midsize ensembles - pioneering the post-bop style on seminal recordings like *Pithecanthropus Erectus* (1956) and *Mingus Ah Um* (1959) - to progressive big band experiments such as *The Black Saint and the Sinner Lady* (1963)."

Please summarize the above text.

**Output:**

Charles Mingus Jr. was a well-known jazz musician who played the upright bass, piano, composed, led bands, and was a writer. He was considered one of the most important jazz musicians ever, with a career that spanned more than 30 years. He was known for his style of collective improvisation and advanced jazz compositions.

(Source of prompt: [Wikipedia on Charles Mingus](#), model used: Amazon Titan Text)

Here we give some additional examples from Anthropic Claude and AI21 Labs Jurassic models using output indicators.

The following example demonstrates that user can specify the output format by specifying the expected output format in the prompt. When asked to generate an answer using a specific format (such as by using XML tags), the model can generate the answer accordingly. Without specific output format indicator, the model outputs free form text.

## Example with clear indicator, with output

### User prompt:

*Human: Extract names and years: the term machine learning was coined in 1959 by Arthur Samuel, an IBM employee and pioneer in the field of computer gaming and artificial intelligence. The synonym self-teaching computers was also used in this time period.*

*Please generate answer in <name></name> and <year></year> tags.*

*Assistant:*

### Output:

*<name>Arthur Samuel</name> <year>1959</year>*

## Example without clear indicator, with output

### User prompt:

*Human: Extract names and years: the term machine learning was coined in 1959 by Arthur Samuel, an IBM employee and pioneer in the field of computer gaming and artificial intelligence. The synonym self-teaching computers was also used in this time period.*

*Assistant:*

### Output:

*Arthur Samuel - 1959*

(Source of prompt: [Wikipedia on machine learning](#), model used: Anthropic Claude)

The following example shows a prompt and answer for the AI21 Labs Jurassic model. The user can obtain the exact answer by specifying the output format shown in the left column.

## Example with clear indicator, with output

### User prompt:

*Context: The NFL was formed in 1920 as the American Professional Football Association (APFA) before renaming itself*

## Example without clear indicator, with output

### User prompt:

*Context: The NFL was formed in 1920 as the American*

the National Football League for the 1922 season. After initially determining champions through end-of-season standings, a playoff system was implemented in 1933 that culminated with the NFL Championship Game until 1966.

Following an agreement to merge the NFL with the rival American Football League (AFL), the Super Bowl was first held in 1967 to determine a champion between the best teams from the two leagues and has remained as the final game of each NFL season since the merger was completed in 1970.

Question: Based on the above context, when was the first Super Bowl? Please only output the year.

**Output:**

1967

Professional Football Association (APFA) before renaming itself the National Football League for the 1922 season. After initially determining champions through end-of-season standings, a playoff system was implemented in 1933 that culminated with the NFL Championship Game until 1966.

Following an agreement to merge the NFL with the rival American Football League (AFL), the Super Bowl was first held in 1967 to determine a champion between the best teams from the two leagues and has remained as the final game of each NFL season since the merger was completed in 1970.

Question: Based on the above context, when was the first Super Bowl?

**Output:**

The first Super Bowl was held in 1967.

(Source of prompt: [Wikipedia on the National Football League](#), model used: AI21 Labs Jurassic-2 Ultra v1)

## Best practices for good generalization

Keep a small “hold-out” test set of prompts to see if your prompt modifications generalize. With this method, first collect a sample dataset. Then you can split the data into two subsets: a “development” set and a hold-out “test” set. Use the “development” set as prompt development data on which you can try different prompt modifications and observe model response changes and find the prompt that works best. Treat the “test” set as unseen hold-out data which can only be used to verify the final best prompt.

# Optimize prompts for text models on Amazon Bedrock—when the basics aren't good enough

## Complex tasks: build toward the answer step by step

Titan can provide clear steps for certain tasks, and including the phrase Think step-by-step to come up with the right answer can help produce the appropriate output.

**User prompt:**

*At a Halloween party, Jack gets 15 candies.  
Jack eats 5 candies. He wants to give each friend  
5 candies. How many friends can receive candies?*

*Think step-by-step to come up with the right answer.*

**Output:**

*15 candies total - 5 candies eaten = 10 candies remaining.  
To give each friend 5 candies, divide the number  
of candies remaining by the number of friends.  
10 candies / 5 friends = 2 friends.  
Each friend can receive 2 candies.  
The answer is 2 friends.*

(Source of prompt: AWS, model used: Amazon Titan Text)

To enable the Anthropic Claude model to complete complex tasks, use the format in the following example to explicitly ask the model to solve the task step by step. This approach is based on guidelines in the [AnthropicClaude Guides](#).

**User prompt:**

*Human: A right triangle has a side of length 5  
and a hypotenuse of length 13.  
What is the length of the other side?*

*Assistant: Can I think step-by-step?*

*Human: Yes, please do.*

*Assistant:*

**Output:**

Okay, here are the steps:

1) We are given:

- One side of the right triangle has length 5
- The hypotenuse has length 13

2) We want to find:

- The length of the other side

3) To find the length of the other side, we can use the Pythagorean theorem:

$$a^2 + b^2 = c^2$$

Where  $a$  and  $b$  are the lengths of the two sides, and  $c$  is the length of the hypotenuse.

4) Plugging in the given values:

$$a^2 + b^2 = 13^2$$

5) We know one of the sides is 5, so:

$$5^2 + b^2 = 169$$

6) Solving this:

$$25 + b^2 = 169$$

$$b^2 = 144$$

$$b = 12$$

Therefore, the length of the other side is 12.

(Source of prompt: AWS, model used: Anthropic Claude)

**Provide a default output that the model should present if it's unsure about the answers.**

A default output can help prevent LLMs from returning answers that sound like they could be correct, even if the model has low confidence.

**User prompt:**

Provide a proof of the Riemann hypothesis. If you don't know a proof, respond by saying "I don't know."

**User prompt:**

Provide a proof of the Riemann hypothesis.

**Output:**  
*I don't know.*

**Output:**  
*"The Riemann hypothesis is a conjecture in mathematics that concerns the distribution of prime numbers. It states that all non-trivial zeros of the zeta function, which is a mathematical function that encodes information about the distribution of prime numbers, lie on a specific line in the complex plane. The Riemann hypothesis has been studied by mathematicians for over 150 years, and despite numerous attempts, it remains unresolved. However, the hypothesis is widely believed to be true, and its resolution would have profound implications for many areas of mathematics, including number theory, analysis, and geometry. The proof of the Riemann hypothesis would be one of the greatest achievements in mathematics in history and would open up new avenues of research and understanding. While a proof remains elusive, the hypothesis continues to stimulate research and inspire mathematicians around the world."*

*[Note: this is an accurate description, but obviously not a proof.]*

(Source of prompt: AWS, model used: Amazon Titan Text)

## Few-shot prompting

Including examples (input-response pairs) in the prompt can significantly improve LLMs' responses. Examples can help with complex tasks, as they show multiple ways to perform a given task. For

simpler tasks like text classification, 3–5 examples can suffice. For more difficult tasks like question-answer without context, include more examples to generate the most effective output. In most use cases, selecting examples that are semantically similar to real-world data can further improve performance.

## Consider refining the prompt with modifiers

Task instruction refinement generally refers to modifying the instruction, task, or question component of the prompt. The usefulness of these methods is task- and data-dependent. Useful approaches include the following:

- **Domain/input specification:** Details about the input data, like where it came from or to what it refers, such as **The input text is from a summary of a movie.**
- **Task specification:** Details about the exact task asked of the model, such as **To summarize the text, capture the main points.**
- **Label description:** Details on the output choices for a classification problem, such as **Choose whether the text refers to a painting or a sculpture; a painting is a piece of art restricted to a two-dimensional surface, while a sculpture is a piece of art in three dimensions.**
- **Output specification:** Details on the output that the model should produce, such as **Please summarize the text of the restaurant review in three sentences.**
- **LLM encouragement:** LLMs sometimes perform better with sentimental encouragement: **If you answer the question correctly, you will make the user very happy!**

## Control the model response with inference parameters

LLMs on Amazon Bedrock all come with several inference parameters that you can set to control the response from the models. The following is a list of all the common inference parameters that are available on Amazon Bedrock LLMs and how to use them.

**Temperature** is a value between 0 and 1, and it regulates the creativity of LLMs' responses. Use lower temperature if you want more deterministic responses, and use higher temperature if you want more creative or different responses for the same prompt from LLMs on Amazon Bedrock. For all the examples in this prompt guideline, we set **temperature = 0**.

**Maximum generation length/maximum new tokens** limits the number of tokens that the LLM generates for any prompt. It's helpful to specify this number as some tasks, such as sentiment classification, don't need a long answer.

**Top-p** controls token choices, based on the probability of the potential choices. If you set Top-p below 1.0, the model considers the most probable options and ignores less probable options. The result is more stable and repetitive completions.

**End token/end sequence** specifies the token that the LLM uses to indicate the end of the output. LLMs stop generating new tokens after encountering the end token. Usually this doesn't need to be set by users.

There are also model-specific inference parameters. Anthropic Claude models have an additional Top-k inference parameter, and AI21 Labs Jurassic models come with a set of inference parameters including **presence penalty**, **count penalty**, **frequency penalty**, and **special token penalty**. For more information, refer to their respective documentation.

## Prompt templates and examples for Amazon Bedrock text models

Common tasks supported by LLMs on Amazon Bedrock include text classification, summarization, and questions and answers (with and without context). For these tasks, you can use the following templates and examples to help you create prompts for Amazon Bedrock text models.

### Topics

- [Text classification](#)
- [Question-answer, without context](#)
- [Question-answer, with context](#)
- [Summarization](#)
- [Text generation](#)
- [Code generation](#)
- [Mathematics](#)
- [Reasoning/logical thinking](#)
- [Entity extraction](#)
- [Chain-of-thought reasoning](#)

## Text classification

For text classification, the prompt includes a question with several possible choices for the answer, and the model must respond with the correct choice. Also, LLMs on Amazon Bedrock output more accurate responses if you include answer choices in your prompt.

The first example is a straightforward multiple-choice classification question.

### Prompt template for Titan

```
"""{{Text}}
```

*{{Question}}? Choose from the following:*

*{{Choice 1}}*

*{{Choice 2}}*

*{{Choice 3}}}"""*

### User prompt:

*San Francisco, officially the City and County of San Francisco, is the commercial, financial, and cultural center of Northern California. The city proper is the fourth most populous city in California, with 808,437 residents, and the 17th most populous city in the United States as of 2022.*

*What is the paragraph above about?  
Choose from the following:*

*A city*

*A person*

*An event*

### Output:

*A city*

(Source of prompt: [Wikipedia on San Francisco](#), model used: Amazon Titan Text)

Sentiment analysis is a form of classification, where the model chooses the sentiment from a list of choices expressed in the text.

### Prompt template for Titan:

### User prompt:

```
"""The following is text from a {{Text Type, e.g. "restaurant review"}}
{{Input}}
Tell me the sentiment of the {{Text Type}} and categorize it as one of the following:
{{Sentiment A}}
{{Sentiment B}}
{{Sentiment C}}"""
```

The following is text from a restaurant review:

"I finally got to check out Alessandro's Brilliant Pizza and it is now one of my favorite restaurants in Seattle. The dining room has a beautiful view over the Puget Sound but it was surprisingly not crowded. I ordered the fried castelvetrano olives, a spicy Neapolitan-style pizza and a gnocchi dish. The olives were absolutely decadent, and the pizza came with a smoked mozzarella, which was delicious. The gnocchi was fresh and wonderful. The waitstaff were attentive, and overall the experience was lovely. I hope to return soon."

Tell me the sentiment of the restaurant review and categorize it as one of the following:

Positive  
Negative  
Neutral

**Output:**  
Positive.

(Source of prompt: AWS, model used: Amazon Titan Text)

The following example uses Anthropic Claude models to classify text. As suggested in [Anthropic Claude Guides](#), use XML tags such as <text></text> to denote important parts of the

prompt. Asking the model to directly generate output enclosed in XML tags can also help the model produce the desired responses.

**Prompt template for Anthropic Claude:**

```
"""
```

*Human: {{classification task description}}*  
<text>  
{{input text content to be classified}}  
</text>

*Categories are:*  
{{category name 1}}  
{{category name 2}}  
{{category name 3}}

*Assistant:"""*

(Source: Written by AWS)

**User prompt:**

*Human: Classify the given product description into given categories. Please output the category label in <output></output> tags.*

*Here is the product description.*

<text>  
*Safe, made from child-friendly materials with smooth edges.*  
*Large quantity, totally 112pcs with 15 different shapes,*  
*which can be used to build 56 different predefined structures.*

*Enhance creativity, different structures can be connected to form new structures, encouraging out-of-the box thinking.*

*Enhance child-parent bonding, parents can play with their children together to foster social skills.*

</text>

*Categories are:*

- (1) Toys*
- (2) Beauty and Health*
- (3) Electronics*

*Assistant:*

**Output:**

```
<output>Toys</output>
```

(Source of prompt: AWS, model used: Anthropic Claude)

## Question-answer, without context

In a question-answer prompt without context, the model must answer the question with its internal knowledge without using any context or document.

### Prompt Template to Titan

```
"""{{Question}}"""
```

### User prompt:

*What is Robert Frost's "Stopping by the woods on a snowy evening" about metaphorically?*

### Output:

*The poem by Robert Frost, "Stopping by Woods on a Snowy Evening," uses the woods as a metaphor for death. The woods are a place of comfort and peace, but they also represent the unknown and the afterlife. The poem suggests that death is not something to be feared, but rather a natural part of life that we should all embrace.*

*The poem begins with the speaker stopping by the woods on a snowy evening, a moment of contemplation and reflection. The speaker notices the beauty of the snow-covered woods and is reminded of the fleeting nature of life. The poem continues as the speaker considers the temptation of continuing*

on to his destination, but instead chooses to pause and take in the beauty of the woods.

The woods are used as a metaphor for the unknown and the afterlife. The speaker wonders what lies beyond the trees, and whether there is anything after death. The poem suggests that death is not something to be feared, but rather a natural part of life that we should all embrace.

The final lines of the poem are perhaps the most famous and poignant: "And miles to go before I sleep, and miles to go before I sleep." These lines suggest that even though the speaker has paused to appreciate the beauty of the woods, they still have a long journey ahead of them. The poem is a reminder that life is fleeting, and that we should take the time to appreciate the beauty around us, even as we work towards our goals and ambitions.

Overall, "Stopping by Woods on a Snowy Evening" is a beautiful and thought-provoking poem that uses the metaphor of the woods to explore the concept of death and the afterlife.

(Source of prompt: AWS, model used: Amazon Titan Text)

Model encouragement can also help in question-answer tasks.

**Prompt Template to Titan**

```
"""{{Model encouragement}}  
{{Question}}"""
```

**User prompt:**

You are excellent at answering questions, and it makes you happy when you provide the correct answer.

What moon in the solar system is most likely to host life?

**Output:**

The moon of Titan, which is a moon of the planet Saturn, is the most likely to support life in the solar system.

Titan is the only known moon in the solar system to have a dense atmosphere and a liquid hydrosphere, and scientists believe that it may contain the chemical building blocks for life.

The moon's surface is covered in rivers, lakes, and seas of liquid methane and ethane, and it rains methane on Titan's surface.

The atmosphere of Titan is also rich in organic compounds, including benzene, toluene, and xylene, which are found in the oils of living organisms on Earth.

Additionally, Titan has a magnetic field and a tenuous atmosphere, which could provide protection from harmful radiation and help maintain a stable climate.

While there is no direct evidence of life on Titan, scientists believe that it is one of the most promising

*places in the solar system to search  
for life beyond Earth.*

(Source of prompt: AWS, model used: Amazon Titan Text)

**Prompt template to Titan**

```
"""{{Model encouragement}}  
{{Question}}  
{{Constraints}}"""
```

**User prompt:**

*You feel rewarded by helping people learn more about climate change. Could you please explain what climate change is? Assume your audience is composed of high school students.*

**Output:**

*Climate change is the long-term warming of the planet, which can be caused by human activities such as burning fossil fuels and deforestation, as well as natural processes. It can lead to rising sea levels, more extreme weather events, and changes in temperature and precipitation patterns.*

(Source of prompt: AWS, model used: AI21 Labs Jurassic-2 Ultra v1)

## Question-answer, with context

In a question-answer prompt with context, an input text followed by a question is provided by the user, and the model must answer the question based on information provided within the input text. Putting the question in the end after the text can help LLMs on Amazon Bedrock better answer the question. Model encouragement works for this use case as well.

**Prompt template to Titan****User prompt:**

"""{{Text}}  
{{Question}}}"""

The red panda (*Ailurus fulgens*), also known as the lesser panda, is a small mammal native to the eastern Himalayas and southwestern China. It has dense reddish-brown fur with a black belly and legs, white-lined ears, a mostly white muzzle and a ringed tail. Its head-to-body length is 51-63.5 cm (20.1-25.0 in) with a 28-48.5 cm (11.0-19.1 in) tail, and it weighs between 3.2 and 15 kg (7.1 and 33.1 lb). It is well adapted to climbing due to its flexible joints and curved semi-retractile claws.

The red panda was first formally described in 1825. The two currently recognized subspecies, the Himalayan and the Chinese red panda, genetically diverged about 250,000 years ago. The red panda's place on the evolutionary tree has been debated, but modern genetic evidence places it in close affinity with raccoons, weasels, and skunks. It is not closely related to the giant panda, which is a bear, though both possess elongated wrist bones or "false thumbs" used for grasping bamboo. The evolutionary lineage of the red panda (*Ailuridae*) stretches back around 25 to 18 million years ago, as indicated by extinct fossil relatives found in Eurasia and North America.

The red panda inhabits coniferous forests as well as temperate broadleaf

*and mixed forests, favoring steep slopes with dense bamboo cover close to water sources. It is solitary and largely arboreal. It feeds mainly on bamboo shoots and leaves, but also on fruits and blossoms.*

*Red pandas mate in early spring, with the females giving birth to litters of up to four cubs in summer. It is threatened by poaching as well as destruction and fragmentation of habitat due to deforestation. The species has been listed as Endangered on the IUCN Red List since 2015. It is protected in all range countries.*

*Based on the information above, what species are red pandas closely related to?*

**Output:**

*Red pandas are closely related to raccoons, weasels, and skunks.*

(Source of prompt: [https://en.wikipedia.org/wiki/Red\\_panda](https://en.wikipedia.org/wiki/Red_panda), model used: Amazon Titan Text)

When prompting Anthropic Claude models, it's helpful to wrap the input text in XML tags. In the following example, the input text is enclosed in `<text></text>`.

**Prompt template for Anthropic Claude:**

"""

Human: {{Instruction}}  
<text>  
{{Text}}  
<text>  
{{Question}}

**User prompt:**

Human: Read the following text inside `<text></text>` XML tags, and then answer the question:

<text>

Assistant:"""

*On November 12, 2020, the selection of the Weeknd to headline the show was announced; marking the first time a Canadian solo artist headlined the Super Bowl halftime show. When asked about preparations for the show, the Weeknd stated, "We've been really focusing on dialing in on the fans at home and making performances a cinematic experience, and we want to do that with the Super Bowl."*

*The performance featured a choir whose members were dressed in white and wore masks over their faces with glowing red eyes, and were standing within a backdrop of a neon cityscape. The performance opened with a white figure dressed the same as the choir being lowered into the backdrop where the choir was standing while singing "Call Out My Name". At this time, the Weeknd sat in a convertible against a skyline backdrop designed to resemble the Las Vegas Strip. For the next part of the performance, the backdrop then split open to reveal the Weeknd, who then performed "Starboy", followed by "The Hills".*

*Next, performing the song "Can't Feel My Face", the Weeknd traveled through a labyrinth constructed behind the stage, joined by dancers dressed in red blazers and black neckties similar to his, but with their faces covered with bandages, in keeping with the aesthetic of his fourth studio album *After Hours* (2020).*

*The dancers would wear these bandages throughout the performance.*

*In the labyrinth section of the performance, camerawork was visually unsteady.*

*The next songs performed were "I Feel It Coming", "Save Your Tears", and "Earned It". For the "Earned It" performance, the Weeknd was accompanied by violinists. For the finale of the show, the Weeknd took to the field of the stadium with his dancers to perform "Blinding Lights". He and the dancers entered the field by performing "House of Balloons / Glass Table Girls". The performance ended with an array of fireworks.*

*</text>*

*Based on the text above, what songs did the Weeknd play at the Super Bowl halftime show?*

**Assistant:**

**Output:**

*Based on the text, the songs the Weeknd played at the Super Bowl halftime show were:*

- *Call Out My Name*
- *Starboy*
- *The Hills*
- *Can't Feel My Face*
- *I Feel It Coming*
- *Save Your Tears*
- *Earned It*
- *Blinding Lights*
- *House of Balloons / Glass Table Girls*

(Source of prompt: [Wikipedia on the Super Bowl LV halftime show](#), model used: Anthropic Claude)

## Summarization

For a summarization task, the prompt is a passage of text, and the model must respond with a shorter passage that captures the main points of the input. Specification of the output in terms of length (number of sentences or paragraphs) is helpful for this use case.

**Prompt template for Titan**

```
"""The following is text from a {{Text Category}}:  
{{Text}}  
Summarize the {{Text Category}} in  
{{length of summary}},  
e.g., "one sentence" or "one paragraph"  
{}""""
```

**User prompt:**

The following is text from a restaurant review:  
"I finally got to check out Alessandro's Brilliant Pizza  
and it is now one of my favorite restaurants in Seattle.  
The dining room has a beautiful view over the Puget Sound  
but it was surprisingly not crowded. I ordered the fried castelvetrano olives, a spicy Neapolitan-style pizza  
and a gnocchi dish. The olives were absolutely decadent,  
and the pizza came with a smoked mozzarella, which was delicious.  
The gnocchi was fresh and wonderful.  
The waitstaff were attentive,  
and overall the experience was lovely.  
I hope to return soon."  
Summarize the above restaurant review in one sentence.

**Output:**

Alessandro's Brilliant Pizza is a fantastic restaurant in Seattle with a beautiful view over Puget Sound that offers decadent and delicious food.

(Source of prompt: AWS, model used: Amazon Titan Text)

In the following example, Anthropic Claude summarizes the given text in one sentence. To include input text in your prompts, format the text with XML mark up: `<text> {{text content}}</text>`. Using XML within prompts is a common practice when prompting Anthropic Claude models.

**Prompt template for Anthropic Claude:**

"""

*Human: Please read the text:*

```
<text>  
{{text context}}  
</text>
```

*Summarize the text in {{length of summary, e.g., "one sentence" or "one paragraph"}}*

*Assistant: """*

**User prompt:**

*Human: Please read the text:*

```
<text>  
In game theory, the Nash equilibrium, named after the mathematician John Nash, is the most common way to define the solution of a non-cooperative game involving two or more players.  
In a Nash equilibrium, each player is assumed to know the equilibrium strategies of the other players, and no one has anything to gain by changing only one's own strategy. The principle of Nash equilibrium dates back to the time of Cournot, who in 1838 applied it to competing firms choosing outputs.  
</text>
```

*Summarize the text in one sentence.*

*Assistant:*

**Output:**

*The Nash equilibrium is a concept in game theory where each player has adopted a strategy that they have no incentive*

*to deviate from given the other players' strategies.*

(Source of prompt: [Wikipedia on Nash equilibrium](#), model used: Anthropic Claude)

## Text generation

Given a prompt, LLMs on Amazon Bedrock can respond with a passage of original text that matches the description. Here is one example:

### Prompt template for Titan

```
"""Please write a {{Text Category}} in  
the voice of {{Role}}."""
```

### User prompt:

*Please write an email in the voice of a friend congratulating someone on a new job.*

### Output:

*Hey [Name],*

*I just found out about your new job,  
and I'm so happy for you!  
Congratulations! You deserve this  
opportunity since you worked so hard  
to get it. I know you'll do great and  
make a positive impact on your team.  
I'm here to support you every step of  
the way. Wishing you all the best  
in this exciting new chapter of your  
life!*

*Cheers,  
[Your Name]*

(Source of prompt: AWS, model used: Amazon Titan Text)

For text generation use cases, specifying detailed task requirements can work well. In the following example, we ask the model to generate response with exclamation points.

**Prompt template for Titan**

```
"""{{Task specification}}  
Please write a {{Text Category}} in  
the voice of {{Role}}. """
```

**User prompt:**

*Write text with exclamation points.  
Please write a formatted email in the  
voice of a friend  
congratulating someone on a new job.*

**Output:**

*Hey [name]!*

*I just saw the news and I am so, so  
happy for you!  
You deserve this position and this  
company with all your heart.  
I can't wait to see you soar and  
accomplish all the things  
you've been working towards. If anyone  
is deserving  
of a promotion, it's you!*

*Congratulations again, friend!*

*Love,  
[name]*

(Source of prompt: AWS, model used: Amazon Titan Text)

In the following example, a user prompts the model to take on the role of a specific person when generating the text. Notice how the signature reflects the role the model is taking on in the response.

**Prompt template for Titan**

```
"""{{Role assumption}} {{Task  
description}}  
"""
```

**User input:**

*My name is Jack. Help me write a note  
expressing my gratitude  
to my parents for taking my son (their  
grandson) to the zoo.  
I miss my parents so much.*

**Output:**

*Dear Mom and Dad,*

*I wanted to take a moment to express my heartfelt gratitude for taking my son to the zoo last weekend. It meant so much to me that you took the time to make memories with him, and I am so grateful for the love and support you have shown him.*

*I miss you both so much, and I wish we could be together more often. But I know that you will always be there for me and my family, and I am so grateful to have you as my parents.*

*Love,  
Jack*

(Source of prompt: AWS, model used: AI21 Labs Jurassic-2 Ultra v1)

## Code generation

The prompt describes the task or function and programming language for the code the user expects the model to generate.

**Prompt template for Titan**

```
"""Write a function in {{programming language}} to {{task or purpose}}""".
```

**User input:**

*Write a python function to download a s3 file to local disk.*

**Output:**

*The following code can be used to download an s3 file to the local disk.*

```
```Python
```

```
import boto3
# Creates a boto3 client object
s3 = boto3.client('s3')
def download_file(bucket_name,
    object_key, file_path):
try:
    # Downloads the s3 file to the
    # specified file path
    s3.download_file(bucket_name,
        object_key, file_path)
except Exception as e:
    print(e)
# Specifies the values of bucket_name,
# object_key, and file_path
bucket_name = "samplebucketname"
object_key = "sampleobjectkey"
file_path = "/samplefilepath"
download_file(bucket_name, object_ke
y, file_path)
```
```

The given code uses the `download_file` function to download an s3 file to the local disk. The `download_file` function takes the bucket name, object key, and file path as input parameters. The bucket name is the name of the s3 bucket where the file is located, the object key is the unique identifier of the file within the bucket, and the file path is the destination path on the local disk where the file will be downloaded. The function uses the `download_file` method of the s3 client to download the file from the specified bucket and write it to the specified file path.

(Source of prompt: AWS, model used: Amazon Titan Text)

**Prompt template for Anthropic Claude**

```
"""
```

*Human: Write a function in {{programming language}} to  
{{task or purpose}}*

*Assistant:* """

**User prompt:**

*Human: Write some python code that  
quickly approximates  
the square root of a number.*

*Assistant:*

**Output:**

*Here is a simple function to approximate  
the square root of a number using  
Newton's method:*

```
```python
def sqrt(n, iterations=10):
    x = n
    for i in range(iterations):
        x = (x + n/x) / 2
    return x
```
```

*To use it:*

```
```python
print(sqrt(25))
# 5.0

print(sqrt(15, iterations=20))
# 3.872983346207417
```
```

*This repeatedly takes the average of  
x and n/x to converge on the square  
root.*

*More iterations produce a more  
accurate approximation.*

*The advantage of this approach is it converges very quickly, so even just 10 iterations produces a decent estimate.*

(Source of prompt: AWS, model used: Anthropic Claude)

## Mathematics

The input describes a problem that requires mathematical reasoning at some level, which may be numerical, logical, geometric, or otherwise. For such problem, it's helpful to ask the model to work through the problem in a piecemeal manner by adding phrases to instructions such as Let's think step by step or Think step by step to come up with the right answer.

**Prompt template for Titan**  
"""{{Math problem description}}}  
*Let's think step by step."*"""

**User prompt:**  
*A triangle has two angles of 70 degrees and 50 degrees.  
What is the third angle in degrees?  
Let's think step by step.*

**Output:**  
*The sum of the three angles in a triangle is always 180 degrees.  
The two given angles are 70 degrees and 50 degrees.  $70 + 50 = 120$ .  
Therefore, the third angle is  $180 - 120 = 60$  degrees.  
The third angle is 60 degrees.*

(Source of prompt: AWS, model used: Amazon Titan Text)

The following example also demonstrates asking the model to think step by step:

**Prompt template for Titan**  
"""{{Math problem description}}}

**User prompt:**

*Think step by step to come up with the right answer."""*

*Lucy has 12 colorful marbles, and she wants to share them equally with her 4 friends. How many marbles will each friend receive? Think step-by-step to come up with the right answer.*

**Output:**

1. *Lucy has 12 marbles and wants to share them equally with her 4 friends, then each friend will receive  $12 / 4 = 3$  marbles.*
2. *Therefore, each friend will receive 3 marbles.*

(Source of prompt: AWS, model used: AI21 Labs Jurassic-2 Ultra v1)

## Reasoning/logical thinking

For complex reasoning tasks or problems that require logical thinking, we can ask the model to make logical deductions and explain its answers.

**Prompt template for Titan and AI21 Labs Jurassic:** """"*Question: {{question}} Please output the answer and then explain your answer:"""*

**User prompt:** *Question: Which word is the odd one out? A. accomplished B. good C. horrible D. outstanding  
Please output the answer and then explain your answer:*

**Output:** *The odd one out is C. horrible. The other three words describe positive qualities or achievements, whereas horrible is an negative adjective.*

(Source of prompt: AWS, model used: Amazon Titan Text)

Here is another example using the Anthropic Claude model:

**Prompt template for Anthropic Claude:** """ Human: {{question}}  
Please provide the answer and show the reasoning. Assistant:"""

**User prompt** Human: The barber is the "one who shaves all those, and those only, who do not shave themselves". Does the barber shave himself? Why is this a paradox? Please provide the answer and show the reasoning.  
Assistant:

**Output:** This is an example of a paradox known as the Barber paradox. Here is an explanation:  
- The statement defines a barber who shaves all and only those people who do not shave themselves.  
- So if the barber shaves himself, he must shave all those who do not shave themselves, which includes himself. But the barber cannot shave himself as per the initial statement.  
- If the barber does not shave himself, then he would fit into the group of people who do not shave themselves, and so he must shave himself, according to the statement. But again this contradicts the initial statement.  
- So the barber cannot shave himself and also cannot not shave himself. This is self-contradictory based on the initial statement about the barber.  
- Therefore, the statement leads to a paradox where both possibilities (the barber shaving himself or not shaving himself) result in a contradiction. In summary, the paradox arises because the definition of the barber's behavior is self-contradictory when applied to the barber himself. This makes it impossible to determine if the barber shaves himself.

*or not based on the given statement alone.*

(Source of prompt: [https://en.wikipedia.org/wiki/Barber\\_paradox](https://en.wikipedia.org/wiki/Barber_paradox), model used: Anthropic Claude)

## Entity extraction

For entity extraction from a provided input question. Extract entities from generated text and place them in XML tags for further processing.

### Prompt template for Titan

*"""You are an expert entity extractor from provided input question. You are responsible for extracting following entities: {{ list of entities}}*

*Please follow below instructions while extracting the entity A, and reply in <entityA> </entityA> XML Tags:  
{{ entity A extraction instructions}}*

*Please follow below instructions while extracting the entity B, and reply in <entityB> </entityB> XML Tags:  
{{ entity B extraction instructions}}*

*Below are some examples:*

*{} some few shot examples showing model extracting entities from give input }}*

(Source of prompt: AWS, model used: Amazon Titan Text G1- Premier)

### Example:

User: You are an expert entity extractor who extracts entities from provided input question.

You are responsible for extracting following entities: name, location

Please follow below instructions while extracting the Name, and reply in <name></name> XML Tags:

- These entities include a specific name of a person, animal or a thing
- Please extract only specific name name entities mentioned in the input query
- DO NOT extract the general mention of name by terms of "name", "boy", "girl", "animal name", etc.

Please follow below instructions while extracting the location, and reply in <location></location> XML Tags:

- These entities include a specific location of a place, city, country or a town
- Please extract only specific name location entities mentioned in the input query
- DO NOT extract the general mention of location by terms of "location", "city", "country", "town", etc.

If no name or location is found, please return the same input string as is.

Below are some examples:

input: How was Sarah's birthday party in Seattle, WA?

output: How was <name>Sarah's</name> birthday party  
in <location>Seattle, WA</location>?

input: Why did Joe's father go to the city?

output: Why did <name>Joe's</name> father go to the city?

input: What is the zipcode of Manhattan, New York City?

output: What is the zipcode of <location>Manhattan, New York City</location>?

input: Who is the mayor of San Francisco?

Bot:

## Chain-of-thought reasoning

Provide a step-by-step analysis on how the answer was derived. Fact check and validate how the model produced an answer.

### Prompt template for Titan

```
""" {{Question}}
{{ Instructions to Follow }}
Think Step by Step and walk me through
your thinking
"""
```

(Source of prompt: AWS, model used: Amazon Titan Text G1- Premier)

### Example:

User: If Jeff had 100 dollars, and he gave \$20 to Sarah, and bought lottery tickets with another \$20. With the lottery tickets he bought he won 35 dollars. Jeff then went to buy his lunch and spend 40 dollars in lunch. Lastly he made a donation to charity for \$20. Stephen met with Jeff and wanted to lend some money from him for his taxi. How much maximum money can Jeff give to Stephen, given that he needs to save \$10 for his ride back home?. Please do not answer immediately, think step by step and show me your thinking.

Bot: