Understand and count tokens

✓ Python JavaScript Go

Gemini and other generative AI models process input and output at a granularity called a *token*.

About tokens

Tokens can be single characters like `z` or whole words like `cat`. Long words are broken up into several tokens. The set of all tokens used by the model is called the vocabulary, and the process of splitting text into tokens is called *tokenization*.

For Gemini models, a token is equivalent to about 4 characters. 100 tokens is equal to about 60-80 English words.

When billing is enabled, the cost of a call to the Gemini API (/pricing) is determined in part by the number of input and output tokens, so knowing how to count tokens can be helpful.

Try out counting tokens in a Colab

You can try out counting tokens by using a Colab.

Try a Colab notebook (https://colab.research.google.com/github/google-gemini/cookbook/blob/main/qu

View notebook on GitHub (https://github.com/google-gemini/cookbook/blob/main/quickstarts/Counting

Context windows

The models available through the Gemini API have context windows that are measured in tokens. The context window defines how much input you can provide and how much output

the model can generate. You can determine the size of the context window by calling the getModels endpoint (/api/rest/v1/models/get) or by looking in the models documentation (/gemini-api/docs/models/gemini).

In the following example, you can see that the `gemini-1.5-flash` model has an input limit of about 1,000,000 tokens and an output limit of about 8,000 tokens, which means a context window is 1,000,000 tokens.

```
from google import genai

client = genai.Client()
model_info = client.models.get(model="gemini-2.0-flash")
print(f"{model_info.input_token_limit=}")
print(f"{model_info.output_token_limit=}")
# ( e.g., input_token_limit=30720, output_token_limit=2048 )
```

ini/api-examples/blob/16520aea9a9f109d1f616ba3400e87c1ece94cac/python/count_tokens.py#L25-L31)

Count tokens

All input to and output from the Gemini API is tokenized, including text, image files, and other non-text modalities.

You can count tokens in the following ways:

- **Call `count_tokens` (/api/rest/v1/models/countTokens) with the input of the request.**
  This returns the total number of tokens in *the input only*. You can make this call before sending the input to the model to check the size of your requests.
- **Use the `usage_metadata` attribute on the `response` object after calling `generate_content`.**
  This returns the total number of tokens in *both the input and the output*: `total_token_count`.
  It also returns the token counts of the input and output separately: `prompt_token_count` (input tokens) and `candidates_token_count` (output tokens).

Count text tokens

If you call `count_tokens` with a text-only input, it returns the token count of the text in *the input only* (`total_tokens`). You can make this call before calling `generate_content` to check the size of your requests.

Another option is calling `generate_content` and then using the `usage_metadata` attribute on the `response` object to get the following:

- The separate token counts of the input (`prompt_token_count`) and the output (`candidates_token_count`)
- The total number of tokens in *both the input and the output* (`total_token_count`)

```
from google import genai

client = genai.Client()
prompt = "The quick brown fox jumps over the lazy dog."

# Count tokens using the new client method.
total_tokens = client.models.count_tokens(
    model="gemini-2.0-flash", contents=prompt
)
print("total_tokens: ", total_tokens)
# ( e.g., total_tokens: 10 )

response = client.models.generate_content(
    model="gemini-2.0-flash", contents=prompt
)

# The usage_metadata provides detailed token counts.
print(response.usage_metadata)
# ( e.g., prompt_token_count: 11, candidates_token_count: 73, total_token_count:
```

ini/api-examples/blob/16520aea9a9f109d1f616ba3400e87c1ece94cac/python/count_tokens.py#L36-L54)

Count multi-turn (chat) tokens

If you call `count_tokens` with the chat history, it returns the total token count of the text from each role in the chat (`total_tokens`).

Another option is calling `send_message` and then using the `usage_metadata` attribute on the `response` object to get the following:

- The separate token counts of the input (`prompt_token_count`) and the output (`candidates_token_count`)
- The total number of tokens in *both the input and the output* (`total_token_count`)

To understand how big your next conversational turn will be, you need to append it to the history when you call `count_tokens`.

```
from google import genai
from google.genai import types

client = genai.Client()

chat = client.chats.create(
    model="gemini-2.0-flash",
    history=[
        types.Content(
            role="user", parts=[types.Part(text="Hi my name is Bob")]
        ),
        types.Content(role="model", parts=[types.Part(text="Hi Bob!")]),
    ],
)
# Count tokens for the chat history.
print(
    client.models.count_tokens(
        model="gemini-2.0-flash", contents=chat.get_history()
    )
)
# ( e.g., total_tokens: 10 )

response = chat.send_message(
    message="In one sentence, explain how a computer works to a young child."
)
print(response.usage_metadata)
# ( e.g., prompt_token_count: 25, candidates_token_count: 21, total_token_count:

# You can count tokens for the combined history and a new message.
extra = types.UserContent(
    parts=[
        types.Part(
            text="What is the meaning of life?",
        )
    ]
```

```
)
history = chat.get_history()
history.append(extra)
print(client.models.count_tokens(model="gemini-2.0-flash", contents=history))
# ( e.g., total_tokens: 56 )
```

ini/api-examples/blob/16520aea9a9f109d1f616ba3400e87c1ece94cac/python/count_tokens.py#L59-L98)

Count multimodal tokens

All input to the Gemini API is tokenized, including text, image files, and other non-text modalities. Note the following high-level key points about tokenization of multimodal input during processing by the Gemini API:

- With Gemini 2.0, image inputs with both dimensions <=384 pixels are counted as 258 tokens. Images larger in one or both dimensions are cropped and scaled as needed into tiles of 768x768 pixels, each counted as 258 tokens. Prior to Gemini 2.0, images used a fixed 258 tokens.
- Video and audio files are converted to tokens at the following fixed rates: video at 263 tokens per second and audio at 32 tokens per second.

Image files

If you call `count_tokens` with a text-and-image input, it returns the combined token count of the text and the image in *the input only* (`total_tokens`). You can make this call before calling `generate_content` to check the size of your requests. You can also optionally call `count_tokens` on the text and the file separately.

Another option is calling `generate_content` and then using the `usage_metadata` attribute on the `response` object to get the following:

- The separate token counts of the input (`prompt_token_count`) and the output (`candidates_token_count`)
- The total number of tokens in *both the input and the output* (`total_token_count`)

Note: You'll get the same token count if you use a file uploaded using the File API or you provide the file as inline data.

Example that uses an uploaded image from the File API:

```
from google import genai

client = genai.Client()
prompt = "Tell me about this image"
your_image_file = client.files.upload(file=media / "organ.jpg")

print(
    client.models.count_tokens(
        model="gemini-2.0-flash", contents=[prompt, your_image_file]
    )
)
# ( e.g., total_tokens: 263 )

response = client.models.generate_content(
    model="gemini-2.0-flash", contents=[prompt, your_image_file]
)
print(response.usage_metadata)
# ( e.g., prompt_token_count: 264, candidates_token_count: 80, total_token_count
```

/api-examples/blob/16520aea9a9f109d1f616ba3400e87c1ece94cac/python/count_tokens.py#L127-L144)

Example that provides the image as inline data:

```
from google import genai
import PIL.Image

client = genai.Client()
prompt = "Tell me about this image"
your_image_file = PIL.Image.open(media / "organ.jpg")

# Count tokens for combined text and inline image.
print(
    client.models.count_tokens(
        model="gemini-2.0-flash", contents=[prompt, your_image_file]
    )
)
# ( e.g., total_tokens: 263 )

response = client.models.generate_content(
```

```
    model="gemini-2.0-flash", contents=[prompt, your_image_file]
)
print(response.usage_metadata)
# ( e.g., prompt_token_count: 264, candidates_token_count: 80, total_token_count
```
/api-examples/blob/16520aea9a9f109d1f616ba3400e87c1ece94cac/python/count_tokens.py#L103-L122)

Video or audio files

Audio and video are each converted to tokens at the following fixed rates:

- Video: 263 tokens per second
- Audio: 32 tokens per second

If you call `count_tokens` with a text-and-video/audio input, it returns the combined token count of the text and the video/audio file in *the input only* (`total_tokens`). You can make this call before calling `generate_content` to check the size of your requests. You can also optionally call `count_tokens` on the text and the file separately.

Another option is calling `generate_content` and then using the `usage_metadata` attribute on the `response` object to get the following:

- The separate token counts of the input (`prompt_token_count`) and the output (`candidates_token_count`)
- The total number of tokens in *both the input and the output* (`total_token_count`)

Note: You'll get the same token count if you use a file uploaded using the File API or you provide the file as inline data.

```
from google import genai
import time

client = genai.Client()
prompt = "Tell me about this video"
your_file = client.files.upload(file=media / "Big_Buck_Bunny.mp4")

# Poll until the video file is completely processed (state becomes ACTIVE).
while not your_file.state or your_file.state.name != "ACTIVE":
    print("Processing video...")
```

```
    print("File state:", your_file.state)
    time.sleep(5)
    your_file = client.files.get(name=your_file.name)

print(
    client.models.count_tokens(
        model="gemini-2.0-flash", contents=[prompt, your_file]
    )
)
# ( e.g., total_tokens: 300 )

response = client.models.generate_content(
    model="gemini-2.0-flash", contents=[prompt, your_file]
)
print(response.usage_metadata)
# ( e.g., prompt_token_count: 301, candidates_token_count: 60, total_token_count
```

/api-examples/blob/16520aea9a9f109d1f616ba3400e87c1ece94cac/python/count_tokens.py#L149-L174)

System instructions and tools

System instructions and tools also count towards the total token count for the input.

If you use system instructions, the `total_tokens` count increases to reflect the addition of `system_instruction`.

If you use function calling, the `total_tokens` count increases to reflect the addition of `tools`.

Last updated 2025-05-12 UTC.