skillcrush

OpenAI Chat Completions API Endpoint Cheatsheet

Contents:

For more information, refer to OpenAI's API documentation.

API Request Code Example

A Chat Completions API call is formatted as an object, known as the request body. This is an example of a Chat Completions API request with the most commonly used parameters.

```
response = client.chat.completions.create(
     model = "gpt-3.5-turbo",
     messages = [
          {"role": "system", "content": "You are a helpful assistant."},
          {"role": "user", "content": "Hello"}
     ],
     max_tokens = 250,
     stop = ["\n"],
     temperature = 0.3,
     frequency_penalty = 0.6,
     presence_penalty = 0.3,
     n = 1,
     response_form = "prompt",
     seed = 12345,
     stream = false,
     user = "579275"
)
```

API Request Breakdown

Let's take a look at how those parameters are actually used including their types (string, array, etc.), default values, and whether they're required or optional parameters.

model (string) *required*
A string that represents the ID of the model to use for the request.

messages (array) *required*
An array of objects, with each object containing two required fields formatted as a dictionary: role and content. Role defines the author of the content, and content defines the content being passed to the model.

max_tokens (integer) *optional, default = null*
Controls the length of the response generated by the model. For example, setting this value to 50 means the model will not exceed 50 tokens when generating a response. Shorter **max_tokens** values will produce more concise responses, and larger values will produce longer and more detailed responses.

stop (string / array) *optional, default = null*
Defines a list of strings that signal to the model to stop generating further text. In the code example above, if the model encounters a newline (`"\n"`), it will stop generating text. This is useful for controlling the amount of text generated by the model.

temperature (number) *optional, default = 1*
Controls the randomness of the model's generated text. A low temperature means the model is more likely to generate accurate text. A high temperature means the model becomes much more exploratory and can generate very random and even nonsensical text.

frequency_penalty (number) *optional, default = 0*
Adjusts the likelihood that the model will repeatedly give similar or identical responses. A higher value reduces repetitive responses.

presence_penalty (number) *optional, default = 0*
Adjusts the likelihood that the model will generate responses with varying words and phrases. A higher value encourages the model to generate more diverse responses.

n (integer) *optional, default = 1*
The number of completions or conversation responses to generate for each prompt. For example, if **n** is set to 5, the model will return five different responses to the prompt. Keeping this set to the default of 1 will keep API usage costs lower.

response_format (object) *optional, default = text*
Defines the format in which the model should return the response. This can be set to either text or json_object.

seed (integer) *optional, default = null*
Controls the randomness of the model's generated text. Multiple requests with the same seed value and parameters will produce consistent responses. If you want each request to generate a different response, you can use a different seed value each time or omit the seed parameter altogether.

stream (boolean) *optional, default = false*
Controls whether or not the API should stream the response as it's generated. If set to true, the response will display in a way that mimics a human being typing a response.

user (string) *optional, default = null*
A user ID. A good use for this is to personalize the model's responses for different users.

Additional Parameters

A few parameters are for advanced use or for very specific use cases that won't be covered in our AI curriculum. You can refer to OpenAI's API documentation for more information.

logprobs (boolean) *optional, default = false*
Logarithms (fancy math calculations) used by the model for decision-making during text generation. Setting this to true will include details about logprobs in the response. Setting it to false will not include these details. This is typically only used in very specific situations where you need to do advanced analysis of the model's output.

logit_bias (map) *optional, default = null*

A dictionary specifying how much to bias the model's output toward certain tokens (words or parts of words). In the code example above, the model is more likely to generate text that includes the word “Hello”. This is useful for guiding the model's response towards specific topics or conversational styles. However, it should be used carefully, as overuse can lead to unnatural-sounding text.

top_logprobs (integer) *optional, default = null*

Defines how many choices the model considers when trying to predict the next word in a sentence. **logprobs** must be set to true in order to use this parameter.

top_p (number) *optional, default = 1*

This is an alternative method to **temperature** to control the randomness of the model’s generated text. In general you should set either this parameter *or* **temperature**, but not both.

tools (array) *optional, default = null*

This can be set to a list of predefined functions that the model can use as it processes the request and generates output. The predefined functions are proprietary

tool_choice (string) *optional, default = null*

Defines which (if any) function is called by the model by providing the specific function name to be called. If no function is specified, the model will decide whether and which function is to be called.

API Response Code Example

After making an API request, the API sends a response in the form of a response object. This is an example of what you might see returned by a Chat Completions API request.

```
{
      "id": "chatcmpl-123",
      "object": "chat.completion",
      "created": 1677652288,
      "model": "gpt-3.5-turbo-0125",
      "system_fingerprint": "fp_44709d6fcb",
      "choices": [{
            "index": 0,
            "message": {
                  "role": "assistant",
                  "content": "\n\nHello there, how may I assist you
            today?",
            },
            "logprobs": null,
            "finish_reason": "stop"
      }],
      "usage": {
            "prompt_tokens": 9,
            "completion_tokens": 12,
            "total_tokens": 21
      }
}
```

API Response Breakdown

Let's look at a breakdown of the properties within a Chat Completions API response including their types, values, and how to reference them using Python.

id (string)

The unique identifier for the chat completion object.

`completion.id`

object (string)

The object type, which is always chat.completion. This is handy for when you have multiple requests going to different types of AI models and need to make sure you're working with the right response objects in your code.

`completion.object`

choices (array)

A list of choices, which are simply the conversational responses generated by the model from the prompt in the API request. By default the model will return one choice, but you can specify how many possible responses you want by using the `n` parameter in the request.

`completion.choices` will return all choices in the array

`completion.choices[0]` will return the first choice in the array

`completion.choices[0].message.content` will return the text generated by the model in response to the prompt in the API request

created (integer)

The Unix timestamp of when the response was created. This looks like a long string of numbers and it refers to the number of seconds that have elapsed since 1/1/1970. Unix timestamps are a way for developers all over the world to refer to the exact same time regardless of location.

`completion.created`

model (string)

The model used for the response. In almost *all* cases this will be the model you specified in your request, but it will also include the exact version number. In the example above, you'll see the exact version of the gpt-3.5-turbo model used is 0125.

`completion.model`

system_fingerprint (string)

The system fingerprint represents the backend configuration that the model runs with. You can think of this as sort of like a specific version of an app running on a specific version of an operating system, like the latest version of the Instagram app running on the latest version of iOS. The app needs to know what version of iOS it's running on in order to work correctly for that specific version. This system_fingerprint is a record of that kind of relationship for the API request.

`completion.system_fingerprint`

usage (object)

Token usage information for both the API request and the API response.

`completion.usage.prompt_tokens` will return the total number of tokens used by the API request

`completion.usage.completion_tokens` will return the total number of tokens used by the API response

`completion.usage.total_tokens` will return the sum of the two