Package: oaii (via r-universe)

October 10, 2024

Type Package

Title 'OpenAI' API R Interface

Version 0.5.0

Description A comprehensive set of helpers that streamline data transmission and processing, making it effortless to interact with the 'OpenAI' API.

License MIT + file LICENSE

URL https://github.com/cezarykuran/oaii

Encoding UTF-8

Imports base64enc, checkmate, httr, magrittr, utils

Suggests htmltools, testthat (>= 3.0.0)

Config/testthat/edition 3

RoxygenNote 7.3.0

NeedsCompilation no

Author Cezary Kuran [aut, cre]

Maintainer Cezary Kuran <cezary.kuran@gmail.com>

Date/Publication 2024-03-13 05:00:02 UTC

Repository https://cezarykuran.r-universe.dev

RemoteUrl https://github.com/cran/oaii

RemoteRef HEAD

RemoteSha 2d6f4e762b8c3af4847283d216957b80400b1d0a

Contents

api_get_key *Get the OpenAI API key from environment variable*

Description

Get the OpenAI API key from environment variable

Usage

```
api_get_key()
```

Value

API key string

See Also

api_set_key

api_set_key *Store the OpenAI API key as environment variable*

Description

Store the OpenAI API key as environment variable

Usage

```
api_set_key(api_key)
```

Arguments

api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

API key string (‘api_key‘)

See Also

api_get_key

Examples

```
## Not run:
api_set_key("my-secret-api-key-string")

## End(Not run)
```

api_upload_file *Create "uploaded file" object*

Description

See upload_file

Usage

```
api_upload_file(f, type = NULL)
```

Arguments

f	string/raw, content of file or path to the file
type	mime type of path. If not supplied, will be guess by `mime::guess_type()` when needed.

Value

NULL if ‘f‘ was NULL otherwise "uploaded_file" object

assistants_create_assistant_request *API assistants: create assistant*

Description

Create an assistant with a model and instructions. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/createAssistant https://platform.openai.com/docs/assistants

Usage

```
assistants_create_assistant_request(
  model,
  name = NULL,
  description = NULL,
  instructions = NULL,
  tools = NULL,
  file_ids = NULL,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

model	string, ID of the model to use. You can use the List models API to see all of your available models, or see our model overview for descriptions of them.
name	NULL/string, the name of the assistant. The maximum length is 256 characters.
description	NULL/string, the description of the assistant. The maximum length is 512 characters.
instructions	NULL/string, the system instructions that the assistant uses. The maximum length is 32768 characters.
tools	NULL/list, a list of tool enabled on the assistant. There can be a maximum of 128 tools per assistant. Tools can be of types code_interpreter, retrieval, or function.
file_ids	NULL/character vector, a list of file IDs attached to this assistant. There can be a maximum of 20 files attached to the assistant. Files are ordered by their creation date in ascending order.
metadata	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maxium of 512 characters long.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

---

assistants_create_file_request

API assistants: create assistant file

Description

Create an assistant file by attaching a file (https://platform.openai.com/docs/api-reference/files) to an assistant (https://platform.openai.com/docs/api-reference/assistants). To get more details, visit https://platform.openai.com/docs/api-reference/assistants/createAssistantFile https://platform.openai.com/docs/assistants

Usage

```
assistants_create_file_request(assistant_id, file_id, api_key = api_get_key())
```

Arguments

assistant_id	string, the ID of the assistant for which to create a File.
file_id	string, a file ID (with purpose="assistants") that the assistant should use. Useful for tools like retrieval and code_interpreter that can access files.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`assistants_delete_assistant_file_request`
API assistants: delete assistant file

Description

Delete an AssistantFile. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/deleteAssistantFi
https://platform.openai.com/docs/assistants

Usage

```
assistants_delete_assistant_file_request(
  assistant_id,
  file_id,
  api_key = api_get_key()
)
```

Arguments

`assistant_id`	string, the ID of the assistant who the file belongs to.
`file_id`	string, the ID of the file to delete
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`assistants_delete_assistant_request`
API assistants: delete assistant

Description

Delete an assistant. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/deleteAssistant
https://platform.openai.com/docs/assistants

Usage

```
assistants_delete_assistant_request(assistant_id, api_key = api_get_key())
```

Arguments

assistant_id	string, the ID of the assistant to delete
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

---

assistants_list_asistants_request

API assistants: list assistants

Description

Returns a list of assistants. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/listAssistants https://platform.openai.com/docs/assistants

Returns a list of assistant files. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/listAssistantFiles https://platform.openai.com/docs/assistants

Usage

```
assistants_list_asistants_request(
  assistant_id,
  limit = NULL,
  order = NULL,
  after = NULL,
  before = NULL,
  api_key = api_get_key()
)

assistants_list_asistants_request(
  assistant_id,
  limit = NULL,
  order = NULL,
  after = NULL,
  before = NULL,
  api_key = api_get_key()
)
```

Arguments

assistant_id	string, the ID of the assistant the file belongs to.
limit	NULL/integer, a limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 20.

order	NULL/string, sort order by the created_at timestamp of the objects. asc for ascending order and desc for descending order. Defaults to desc
after	NULL/string, a cursor for use in pagination. after is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include after=obj_foo in order to fetch the next page of the list.
before	NULL/string, a cursor for use in pagination. before is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include before=obj_foo in order to fetch the previous page of the list.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`assistants_modify_assistant_request`

API assistants: modify assistant

Description

Modifies an assistant. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/modifyAssistant https://platform.openai.com/docs/assistants

Usage

```
assistants_modify_assistant_request(
  assistant_id,
  model = NULL,
  name = NULL,
  description = NULL,
  instructions = NULL,
  tools = NULL,
  file_ids = NULL,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

assistant_id	string, the ID of the assistant to modify
model	string, ID of the model to use. You can use the List models API to see all of your available models, or see our model overview (https://platform.openai.com/docs/models/overview) for descriptions of them.

name	NULL/string, the name of the assistant. The maximum length is 256 characters.
description	NULL/string, the description of the assistant. The maximum length is 512 characters.
instructions	NULL/string, the system instructions that the assistant uses. The maximum length is 32768 characters.
tools	NULL/list, a list of tool enabled on the assistant. There can be a maximum of 128 tools per assistant. Tools can be of types code_interpreter, retrieval, or function.
file_ids	NULL/character vector, a list of file IDs attached to this assistant. There can be a maximum of 20 files attached to the assistant. Files are ordered by their creation date in ascending order.
metadata	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maxium of 512 characters long.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`assistants_retrieve_assistant_file_request`
API assistants: retrieve assistant file

Description

Retrieves an AssistantFile. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/getAssistantFi
https://platform.openai.com/docs/assistants

Usage

```
assistants_retrieve_assistant_file_request(
  assistant_id,
  file_id,
  api_key = api_get_key()
)
```

Arguments

assistant_id	string, the ID of the assistant who the file belongs to.
file_id	string, the ID of the file we’re getting.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`assistants_retrieve_assistant_request` *API assistants: retrieve assistant*

Description

Retrieves an assistant. To get more details, visit https://platform.openai.com/docs/api-reference/assistants/getAssistant https://platform.openai.com/docs/assistants

Usage

```
assistants_retrieve_assistant_request(assistant_id, api_key = api_get_key())
```

Arguments

`assistant_id`	string, the ID of the assistant to retrieve.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`audio_speech_request` *API audio: text to speech request*

Description

Generates audio from the input text. To get more details, visit https://platform.openai.com/docs/api-reference/audio/createSpeech https://platform.openai.com/docs/guides/speech-to-text

Usage

```
audio_speech_request(
  model,
  input,
  voice,
  response_format = NULL,
  speed = NULL,
  api_key = api_get_key()
)
```

Arguments

model	string, one of the available TTS models: 'tts-1' or 'tts-1-hd'
input	string, the text to generate audio for. The maximum length is 4096 characters.
voice	string, the voice to use when generating the audio. Supported voices are alloy, echo, fable, onyx, nova, and shimmer. Previews of the voices are available in the Text to speech guide - https://platform.openai.com/docs/guides/text-to-speech/voice-options
response_format	string, the format to audio in. Supported formats are mp3 (default), opus, aac, and flac
speed	double, the speed of the generated audio. Select a value from 0.25 to 4.0, 1.0 is the default.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: 'status_code' (response$status_code) and 'message_long' (built on response content)

Examples

```
## Not run:
res_content <- audio_speech_request(
  "tts-1",
  "When the power of love overcomes the love of power, the world will know peace.",
  "nova"
)
if (!is_error(res_content)) {
  writeBin(res_content, "peace.mp3")
}

## End(Not run)
```

---

audio_transcription_request

API audio: speech to text (transcryption)

Description

Transcribes audio into the input language. To get more details, visit https://platform.openai.com/docs/api-reference/audio/createTranscription https://platform.openai.com/docs/guides/speech-to-text

Usage

```
audio_transcription_request(
  file,
  model,
  language = NULL,
  prompt = NULL,
  response_format = NULL,
  temperature = NULL,
  file_type = NULL,
  api_key = api_get_key()
)
```

Arguments

file	string/raw, content of the audio file or path to the audio file to transcribe, in one of these formats: flac, mp3, mp4, mpeg, mpga, m4a, ogg, wav, or webm.
model	string, ID of the model to use. Only 'whisper-1' is currently available.
language	NULL/string, the language of the input audio. Supplying the input language in ISO-639-1 format will improve accuracy and latency. See https://en.wikipedia.org/wiki/List_of_ISO_639_
prompt	NULL/string, an optional text to guide the model's style or continue a previous audio segment. The prompt (https://platform.openai.com/docs/guides/speech-to-text/prompting) should match the audio language.
response_format	NULL/string, The format of the transcript output, in one of these options: json (default), text, srt, verbose_json, or vtt.
temperature	NULL/double, the sampling temperature, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. If set to 0, the model will use log probability to automatically increase the temperature until certain thresholds are hit. 0 is default.
file_type	NULL/string mime type of file (e.g. "audio/mpeg"). If NULL (default), will be guess by mime::guess_type() when needed.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: 'status_code' (response$status_code) and 'message_long' (built on response content)

Examples

```
## Not run:
res_content <- audio_speech_request(
  "path/to/audio/file.mp3",
  "whisper-1",
  "en",
  response_format = "text"
```

```
)
if (!is_error(res_content)) {
  message(res_content)
}

## End(Not run)
```

---

audio_translation_request *API audio: translate audio file into English text*

Description

Translates audio into English. To get more details, visit https://platform.openai.com/docs/api-reference/audio/createTranslation https://platform.openai.com/docs/guides/speech-to-text

Usage

```
audio_translation_request(
  file,
  model,
  prompt = NULL,
  response_format = NULL,
  temperature = NULL,
  api_key = api_get_key()
)
```

Arguments

file	string/raw, content of the input audio file or path to the input audio file to translate, in one of these formats: flac, mp3, mp4, mpeg, mpga, m4a, ogg, wav, or webm
model	string, ID of the model to use. Only 'whisper-1' is currently available.
prompt	string, An optional text to guide the model's style or continue a previous audio segment. The prompt should be in English.
response_format	string, the format of the transcript output, in one of these options: json (default), text, srt, verbose_json, or vtt.
temperature	double, the sampling temperature, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. If set to 0, the model will use log probability to automatically increase the temperature until certain thresholds are hit. 0 is default.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
res_content <- audio_translation_request(
  "path/to/audio/file.mp3",
  "whisper-1",
  response_format = "text"
)
if (!is_error(res_content)) {
  message(res_content)
}

## End(Not run)
```

---

browseable_audio *Create browseable HTML audio*

Description

Create browseable HTML audio

Usage

```
browseable_audio(data, format = "mp3")
```

Arguments

data	audio data
format	audio format

Value

HTML audio

chat_fetch_messages *Fetch messages from response content*

Description

Fetch messages (dialog data.frame with chat messages) from response content

Usage

```
chat_fetch_messages(res_content)
```

Arguments

res_content	response object returned by chat_request

Value

Messages from response as dialog data.frame (see dialog_df)

Examples

```
## Not run:
  question <- dialog_df("hi")
  res_content <- chat_request(
    messages = question,
    model = "gpt-3.5-turbo"
  )
  if (!is_error(res_content)) {
    answer <- chat_fetch_messages(res_content)
    conversation <- merge_dialog_df(question, answer)
    print(conversation)
  }

## End(Not run)
```

chat_request *API chat: send create (chat) request*

Description

Creates a model response for the given chat conversation. To get more details, visit https://platform.openai.com/docs/api-reference/chat/create https://platform.openai.com/docs/guides/text-generation

Usage

```
chat_request(
  messages,
  model,
  frequency_penalty = NULL,
  logit_bias = NULL,
  logprobs = NULL,
  top_logprobs = NULL,
  max_tokens = NULL,
  n = NULL,
  presence_penalty = NULL,
  response_format = NULL,
  seed = NULL,
  stop = NULL,
  stream = NULL,
  temperature = NULL,
  top_p = NULL,
  tools = NULL,
  tool_choice = NULL,
  user = NULL,
  api_key = api_get_key()
)
```

Arguments

messages	data.frame, data.frame with messages comprising the conversation so far
model	string, ID of the model to use. See the model endpoint compatibility table https://platform.openai.com/docs/models/model-endpoint-compatibility for details on which models work with the Chat API.
frequency_penalty	NULL/double, number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim. More at https://platform.openai.com/docs/guides/text-generation/parameter-details
logit_bias	NULL/list, modify the likelihood of specified tokens appearing in the completion. Accepts a list that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token. See https://platform.openai.com/tokenizer
logprobs	NULL/flag, whether to return log probabilities of the output tokens or not. If true, returns the log probabilities of each output token returned in the content of message. This option is currently not available on the gpt-4-vision-preview model. Defaults to false.
top_logprobs	NULL/int, an integer between 0 and 5 specifying the number of most likely tokens to return at each token position, each with an associated log probability.

logprobs must be set to true if this parameter is used.

`max_tokens` NULL/int, the maximum number of tokens to generate in the chat completion

`n` NULL/int, how many chat completion choices to generate for each input message.

`presence_penalty`
NULL/double, number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics. See https://platform.openai.com/docs/guides/text-generation/parameter-details

`response_format`
NULL/list, an object specifying the format that the model must output. Compatible with gpt-4-1106-preview and gpt-3.5-turbo-1106. Setting to list(type = "json_object") enables JSON mode, which guarantees the message the model generates is valid JSON. Important: when using JSON mode, you must also instruct the model to produce JSON yourself via a system or user message. Without this, the model may generate an unending stream of whitespace until the generation reaches the token limit, resulting in a long-running and seemingly "stuck" request. Also note that the message content may be partially cut off if finish_reason="length", which indicates the generation exceeded max_tokens or the conversation exceeded the max context length. Text is default response format.

`seed` NULL/int, this feature is in Beta. If specified, our system will make a best effort to sample deterministically, such that repeated requests with the same seed and parameters should return the same result. Determinism is not guaranteed, and you should refer to the system_fingerprint response parameter to monitor changes in the backend.

`stop` NULL/character vector, up to 4 sequences where the API will stop generating further tokens.

`stream` NULL/flag, if set, partial message deltas will be sent, like in ChatGPT. Tokens will be sent as data-only server-sent events as they become available, with the stream terminated by a data: [DONE] message. Defaults to false

`temperature` NULL/double, what sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

`top_p` NULL/double, an alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10 probability mass are considered. We generally recommend altering this or temperature but not both. Defaults to 1

`tools` NULL/list, a "list" of tools the model may call. Currently, only functions are supported as a tool. Use this to provide a list of functions the model may generate JSON inputs for. Example value:

```
list(
  # string (required), the type of the tool. Currently, only
  # 'function' is supported
```

```
                    type = "function",

                    # list (required)
                    function = list(
                      # string (optional)
                      description = "some description",

                      # string (required), the name of the function to be called.
                      # Must be a-z, A-Z, 0-9, or contain underscores and dashes,
                      # with a maximum length of 64
                      name = "functionname",

                      # list (optional), the parameters the functions accepts,
                      # described as a JSON Schema object. Omitting parameters
                      # defines a function with an empty parameter list.
                      parameters = list()
                    )
                  )
```

tool_choice — NULL/string/list, controls which (if any) function is called by the model. 'none' means the model will not call a function and instead generates a message. 'auto' means the model can pick between generating a message or calling a function. Specifying a particular function via list 'list(type = "function", function": list(name: "my_function"))' forces the model to call that function. 'none' is the default when no functions are present, 'auto' is the default if functions are present.

user — NULL/string, a unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. See https://platform.openai.com/docs/guides/safety-best-practices/end-user-ids

api_key — string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: 'status_code' (response$status_code) and 'message_long' (built on response content)

Examples

```
## Not run:
  question <- dialog_df("hi")
  res_content <- chat_request(
    messages = question,
    model = "gpt-3.5-turbo"
  )
  if (!is_error(res_content)) {
    answer <- chat_fetch_messages(res_content)
    conversation <- merge_dialog_df(question, answer)
    print(conversation)
  }
```

```
## End(Not run)
```

`completions_fetch_text` *Fetch completions text from response content*

Description

Fetch completions text from response content (completions_request) as dialog data.frame

Usage

```
completions_fetch_text(res_content, role = "ai", ltrim = TRUE)
```

Arguments

`res_content`	response object returned by completions_request
`role`	string, dialog role (phrase owner)
`ltrim`	flag, trim left white space character(s) from text

Value

dialog data.frame

Examples

```
## Not run:
  prompt <- "x=1, y=2, z=x*y, z=?"
  res_content <- completions_request(
    model = "text-davinci-003",
    prompt = prompt
  )
  if (!is_error(res_content)) {
    answer <- completions_fetch_text(res_content)
    print(answer)
  }

## End(Not run)
```

`completions_request` *API completions: create request*

Description

To get more details, visit https://platform.openai.com/docs/api-reference/completions/create

Usage

```
completions_request(
  model,
  prompt,
  suffix = NULL,
  max_tokens = NULL,
  temperature = NULL,
  top_p = NULL,
  n = NULL,
  stream = NULL,
  logprobs = NULL,
  echo = NULL,
  stop = NULL,
  presence_penalty = NULL,
  frequency_penalty = NULL,
  best_of = NULL,
  user = NULL,
  api_key = api_get_key()
)
```

Arguments

`model`	string, ID of the model to use. You can use the list models (https://platform.openai.com/docs/api-reference/models/list) API to see all of your available models, or see our model overview (https://platform.openai.com/docs/models/overview) for descriptions of them.
`prompt`	API endpoint parameter
`suffix`	string/NULL, the suffix that comes after a completion of inserted text.
`max_tokens`	integer, the maximum number of tokens (https://platform.openai.com/tokenizer) to generate in the completion. The token count of your prompt plus max_tokens cannot exceed the model's context length.
`temperature`	double, what sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.
`top_p`	double, an alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered.

n	integer, How many completions to generate for each prompt. Note: Because this parameter generates many completions, it can quickly consume your token quota. Use carefully and ensure that you have reasonable settings for ‘max_tokens‘ and ‘stop‘.
stream	flag, Whether to stream back partial progress. If set, tokens will be sent as data-only server-sent events as they become available, with the stream terminated by a data: ‘[DONE]‘ message.
logprobs	integer, Include the log probabilities on the logprobs most likely tokens, as well the chosen tokens. For example, if logprobs is 5, the API will return a list of the 5 most likely tokens. The API will always return the logprob of the sampled token, so there may be up to logprobs+1 elements in the response.
echo	logical, echo back the prompt in addition to the completion
stop	string or array, up to 4 sequences where the API will stop generating further tokens. The returned text will not contain the stop sequence.
presence_penalty	double, Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.
frequency_penalty	double, Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.
best_of	integer, Generates best_of completions server-side and returns the "best" (the one with the highest log probability per token). Results cannot be streamed. When used with n, best_of controls the number of candidate completions and n specifies how many to return – best_of must be greater than n.
user	string, A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
  prompt <- "x=1, y=2, z=x*y, z=?"
  res_content <- completions_request(
    model = "text-davinci-003",
    prompt = prompt
  )
  if (!is_error(res_content)) {
    answer <- completions_fetch_text(res_content)
    print(answer)
  }
```

```
## End(Not run)
```

`csv_to_dialog_df` *Read csv file as dialog data.frame*

Description

Read csv file as dialog data.frame

Usage

```
csv_to_dialog_df(datapath)
```

Arguments

`datapath`	string, csv file path

Value

Content of the input csv file as dialog data.frame, SimpleError when an error occurs

---

`df_col_dt_format` *Replace unix timestamp column(s) to formated dt string*

Description

Replace unix timestamp column(s) to formated dt string

Usage

```
df_col_dt_format(
  df,
  col,
  format = "%Y-%m-%d %H:%M:%S",
  tz = "",
  on_missing_col = "warn"
)
```

Arguments

df	data.frame, input data.frame
col	character vector, column names of the df that will be modified
format	A character string. The default for the format methods is "%Y-%m-%d %H:%M:%S" if any element has a time component which is not midnight, and "%Y-%m-%d" otherwise. If options("digits.secs") is set, up to the specified number of digits will be printed for seconds.
tz	A character string specifying the time zone to be used for the conversion. System-specific (see as.POSIXlt), but "" is the current time zone, and "GMT" is UTC. Invalid values are most commonly treated as UTC, on some platforms with a warning.
on_missing_col	string, behavior for missing column(s): "warn" - log warning, "skip" - skip missing column(s), "stop" - throw error

Value

Modified input data.frame

Examples

```
df <- data.frame(
  x = c("a", "b"),
  dt = c(1687868601, 1688417643)
)
df_col_dt_format(df, "dt")
df_col_dt_format(df, "dt", "%H:%M")
```

---

df_col_obj_implode *Change to string nested lists in a given data.frame*

Description

Change to string nested lists in a given data.frame

Usage

```
df_col_obj_implode(
  df,
  col,
  obj_prop = NULL,
  nested = TRUE,
  cell_header = "",
  objs_glue = "----\n",
  cell_footer = "",
  obj_header = "",
```

```
  props_glue = "\n",
  obj_footer = "",
  prop_fmt = "%s: %s",
  null_prop_str = "[null]",
  on_missing_col = "warn"
)
```

Arguments

df	data.frame, input data.frame
col	character vector, df column names containing objects
obj_prop	NULL/character vector, object properties (NULL means all)
nested	flag, whether the rows of the columns contain multiple objects
cell_header	string/NULL, cell header
objs_glue	string, how to combine objects
cell_footer	string/NULL, cell footer
obj_header	string/NULL, object header
props_glue	string, how to combine properties
obj_footer	string/NULL, object footer
prop_fmt	string, sprintf fmt parameter with two ‘%s‘ fields (property
null_prop_str	string, value for NULL object property name, value)
on_missing_col	string, behavior for missing column(s): "warn" - log warning, "skip" - skip missing column(s), "stop" - throw error

Value

Modified input data.frame

Examples

```
df <- as.data.frame(do.call(cbind, list(
  a = list(list(x = 1, y = 2), list(x = 3, y = 4)),
  b = list("z", "z")
)))
df_col_obj_implode(df, "a", c("x", "y"), nested = FALSE, props_glue = ", ")
```

df_exclude_col *Remove columns from data.frame*

Description

Remove columns from data.frame

Usage

```
df_exclude_col(df, col, on_missing_col = "warn")
```

Arguments

df	data.frame, input data.frame
col	character vector, column name(s) to be deleted
on_missing_col	string, behavior for missing column(s): "warn" - log warning, "skip" - skip missing column(s), "stop" - throw error

Value

Modified input data.frame

Examples

```
df <- data.frame(a = 1:3, b = 1:3, c = 1:3)
df_exclude_col(df, "b")
df_exclude_col(df, c("a", "c"))
```

df_null_replace *Replace all NULL values in given data.frame*

Description

Replace all NULL values in given data.frame

Usage

```
df_null_replace(df, replacement = "")
```

Arguments

df	data.frame, input data.frame
replacement	string, replacement for NULL

Value

Modified input data.frame

df_order_by_col *Sort data.frame by column name*

Description

Sort data.frame by column name

Usage

```
df_order_by_col(df, col, decreasing = FALSE, on_missing_col = "warn")
```

Arguments

df	data.frame, input data.frame
col	string, column name as sort source
decreasing	flag, should the sort order be increasing or decreasing?
on_missing_col	string, behavior for missing column(s): "warn" - log warning, "skip" - skip missing column(s), "stop" - throw error

Value

Modified input data.frame

Examples

```
df <- data.frame(
  a = c("a", "b", "c"),
  b = c(1, 3, 2),
  c = c(3, 2, 1)
)
df_order_by_col(df, "b", decreasing = TRUE)
df_order_by_col(df, "c")
```

dialog_df *Create dialog data.frame*

Description

Create dialog data.frame

Usage

```
dialog_df(content, role = "user", finish_reason = "stop")
```

Arguments

content	string, message content
role	string, message role ("owner")
finish_reason	see https://platform.openai.com/docs/guides/gpt/chat-completions-response-format

Value

A one-row data.frame with columns: ‘content‘, ‘role‘ and ‘finish_reason‘

Examples

```
dialog_df("some text message")
dialog_df("some another text message", role = "assistant")
```

---

dialog_df_to_csv *Save dialog data.frame as csv file*

Description

Save dialog data.frame as csv file

Usage

```
dialog_df_to_csv(dialog_df, file)
```

Arguments

dialog_df	data.frame, dialog data.frame to save in csv file
file	string, csv file path

Value

write.table return value or SimpleError

`embeddings_create_request` *API embeddings: create embeddings*

Description

Creates an embedding vector representing the input text. To get more details, visit https://platform.openai.com/docs/api-reference/embeddings/create https://platform.openai.com/docs/guides/embeddings

Usage

```
embeddings_create_request(
  input,
  model,
  encoding_format = NULL,
  user = NULL,
  api_key = api_get_key()
)
```

Arguments

`input`	character vector, input text to embed, encoded as a string or array of tokens. To embed multiple inputs in a single request, pass an array of strings or array of token arrays. The input must not exceed the max input tokens for the model (8192 tokens for text-embedding-ada-002), cannot be an empty string, and any array must be 2048 dimensions or less.
`model`	string, ID of the model to use. You can use the list models API (https://platform.openai.com/docs/api-reference/models/list) to see all of your available models, or see our model overview (https://platform.openai.com/docs/models/overview) for descriptions of them.
`encoding_format`	string, he format to return the embeddings in. Can be either float (default) or base64.
`user`	string, a unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. To learn more visit https://platform.openai.com/docs/guides/safety-best-practices/end-user-ids
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

embeddings_object_request

API embeddings: get embedding object

Description

Represents an embedding vector returned by embedding endpoint. To get more details, visit https://platform.openai.com/docs/
reference/embeddings/object https://platform.openai.com/docs/guides/embeddings

Usage

```
embeddings_object_request(index, embedding, object, api_key = api_get_key())
```

Arguments

index	integer, The index of the embedding in the list of embeddings
embedding	double vector, the embedding vector, which is a "list of floats". The length of vector depends on the model as listed in the embedding guide.
object	string, the object type, which is always "embedding"
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

feedback

Feedback - ask chat and receive reply

Description

Simple chat_request wrapper - send text to chat and get response.

Usage

```
feedback(question, model = "gpt-3.5-turbo", max_tokens = NULL, print = TRUE)
```

Arguments

question	string, question text
model	string, ID of the model to use. See the model endpoint compatibility table https://platform.openai.com/docs/models/model-endpoint-compatibility for details on which models work with the Chat API.
max_tokens	NULL/int, the maximum number of tokens to generate in the chat completion
print	flag, If TRUE, print the answer on the console

Value

string, chat answer

`files_delete_request` *API files: delete file request*

Description

Delete a file. To get more details, visit https://platform.openai.com/docs/api-reference/files/delete

Usage

```
files_delete_request(file_id, api_key = api_get_key())
```

Arguments

`file_id`	string, id of the uploaded file
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`files_fetch_list` *Extract files list as data.frame from response object*

Description

Extract files list as data.frame from response object

Usage

```
files_fetch_list(res_content)
```

Arguments

`res_content`	response object returned by files_list_request

Value

Files list as data.frame

Examples

```
## Not run:
res_content <- files_list_request()
if (!is_error(res_content)) {
  files_list_df <- files_fetch_list(res_content)
  print(files_list_df)
}

## End(Not run)
```

`files_list_request` *API files: get list request*

Description

Returns a list of files that belong to the user's organization. To get more details, visit: https://platform.openai.com/docs/api-reference/files/list

Usage

```
files_list_request(purpose = NULL, api_key = api_get_key())
```

Arguments

`purpose`	NULL/string, only return files with the given purpose
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: 'status_code' (response$status_code) and 'message_long' (built on response content)

Examples

```
## Not run:
res_content <- files_list_request()
if (!is_error(res_content)) {
  files_list_df <- files_fetch_list(res_content)
  print(files_list_df)
}

## End(Not run)
```

files_retrieve_content_request

API files: retrieve content request

Description

To get more details, visit https://platform.openai.com/docs/api-reference/files/retrieve-content

Usage

```
files_retrieve_content_request(file_id, api_key = api_get_key())
```

Arguments

file_id	string, id of the uploaded file
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
res_content <- files_retrieve_content_request("some-file-id")
if (!is_error(res_content)) {
  writeBin(res_content, "some-file.jsonl")
}

## End(Not run)
```

files_retrieve_request

API files: retrieve file request

Description

Returns information about a specific file. To get more details, visit: https://platform.openai.com/docs/api-reference/files/retrieve

Usage

```
files_retrieve_request(file_id, api_key = api_get_key())
```

Arguments

file_id	string, id of the uploaded file
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

---

files_upload_request *API files: upload request*

Description

Upload a file that can be used across various endpoints. The size of all the files uploaded by one organization can be up to 100 GB. The size of individual files can be a maximum of 512 MB or 2 million tokens for Assistants. See the Assistants Tools guide (https://platform.openai.com/docs/assistants/tools) to learn more about the types of files supported. The Fine-tuning API only supports .jsonl files. To get more details, visit: https://platform.openai.com/docs/api-reference/files/upload

Usage

```
files_upload_request(file, purpose, file_type = NULL, api_key = api_get_key())
```

Arguments

file	string/raw, path or content of the JSON Lines file to be uploaded
purpose	string, the intended purpose of the uploaded documents. Use "fine-tune" for Fine-tuning.
file_type	NULL/string, mime type of ‘file‘. See api_upload_file
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

fine_tuning_cancel_job_request

API fine-tuning: cancel fine-tuning job request

Description

Immediately cancel a fine-tune job. To get more details, visit https://platform.openai.com/docs/guides/fine-tuning https://platform.openai.com/docs/api-reference/fine-tuning/cancel

Usage

```
fine_tuning_cancel_job_request(fine_tuning_job_id, api_key = api_get_key())
```

Arguments

fine_tuning_job_id	string, the ID of the fine-tuning job to cancel
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
res_content <- fine_tuning_cancel_job_request("job-id")
if (!is_error(res_content)) {
  message("job canceled")
}

## End(Not run)
```

fine_tuning_create_job_request

API fine-tuning: create job (model) request

Description

Creates a fine-tuning job which begins the process of creating a new model from a given dataset. To get more details, visit https://platform.openai.com/docs/guides/fine-tuning https://platform.openai.com/docs/api-reference/fine-tuning/create

Usage

```
fine_tuning_create_job_request(
  model,
  training_file,
  hyperparameters = NULL,
  suffix = NULL,
  validation_file = NULL,
  api_key = api_get_key()
)
```

Arguments

model	string, the name of the base model to fine-tune. You can select one of the supported models: gpt-3.5-turbo-1106 (recommended), gpt-3.5-turbo-0613, babbage-002, davinci-002, gpt-4-0613 (experimental)
training_file	string, the ID of an uploaded file that contains training data. See files_upload_request.
hyperparameters	list/NULL, the hyperparameters used for the fine-tuning job. 'hyperparameters$batch_size' string/integer/NULL defaults to "auto", number of examples in each batch. A larger batch size means that model parameters are updated less frequently, but with lower variance. 'hyperparameters$learning_rate_multiplier' string/number/NULL defaults to "auto", scaling factor for the learning rate. A smaller learning rate may be useful to avoid overfitting. 'hyperparameters$n_epochs' string/integer/NULL, defaults to "auto", the number of epochs to train the model for. An epoch refers to one full cycle through the training dataset.
suffix	string/NULL, A string of up to 18 characters that will be added to your fine-tuned model name. For example, a suffix of "custom-model-name" would produce a model name like ft:gpt-3.5-turbo:openai:custom-model-name:7p4lURel
validation_file	string/NULL, the ID of an uploaded file that contains validation data. If you provide this file, the data is used to generate validation metrics periodically during fine-tuning. These metrics can be viewed in the fine-tuning results file. The same data should not be present in both train and validation files. Your dataset must be formatted as a JSONL file. You must upload your file with the purpose fine-tune.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: 'status_code' (response$status_code) and 'message_long' (built on response content)

`fine_tuning_events_list_request` *API fine-tuning: list events request*

Description

Get status updates for a fine-tuning job. To get more details, visit https://platform.openai.com/docs/guides/fine-tuning https://platform.openai.com/docs/api-reference/fine-tuning/list-events

Usage

```
fine_tuning_events_list_request(
  fine_tuning_job_id,
  after = NULL,
  limit = NULL,
  api_key = api_get_key()
)
```

Arguments

`fine_tuning_job_id`	string, the ID of the fine-tuning job to get events for
`after`	string/NULL, identifier for the last event from the previous pagination request.
`limit`	integer/NULL, number of events to retrieve (defaults to 20)
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
res_content <- fine_tuning_events_list_request("job-id")
if (!is_error(res_content)) {
  fine_tuning_events_df <- fine_tuning_fetch_events_list(res_content)
  print(fine_tuning_events_df)
}

## End(Not run)
```

fine_tuning_fetch_events_list

API fine-tuning: job list from response object

Description

Extract fine-tuning job list as data.frame from response object

Usage

```
fine_tuning_fetch_events_list(res_content)
```

Arguments

res_content	response object returned by fine_tuning_events_list_request

Value

fine-tuning events list as data.frame

Examples

```
## Not run:
res_content <- fine_tuning_events_list_request("job-id")
if (!is_error(res_content)) {
  fine_tuning_events_df <- fine_tuning_fetch_events_list(res_content)
  print(fine_tuning_events_df)
}

## End(Not run)
```

fine_tuning_fetch_jobs_list

API fine-tuning: extract fine-tuning jobs list from response object

Description

Extract fine-tuning jobs list as data.frame from response object

Usage

```
fine_tuning_fetch_jobs_list(res_content)
```

Arguments

res_content	response object returned by fine_tuning_jobs_list_request

Value

fine-tuning list models as data.frame

Examples

```
## Not run:
res_content <- fine_tuning_jobs_list_request()
if (!is_error(res_content)) {
  fine_tuning_jobs_df <- fine_tuning_fetch_jobs_list(res_content)
  print(fine_tuning_jobs_df)
}

## End(Not run)
```

---

fine_tuning_fetch_retrived_job

API fine-tuning: fetch retrived job object from response object

---

Description

Extract fine-tuning job object from response object

Usage

```
fine_tuning_fetch_retrived_job(res_content)
```

Arguments

res_content	response object returned by fine_tuning_retrive_job_request

Value

fine-tuning job object

Examples

```
## Not run:
res_content <- fine_tuning_retrive_job_request("job-id")
if (!is_error(res_content)) {
  fine_tuning_events_df <- fine_tuning_fetch_events_list(res_content)
  print(fine_tuning_events_df)
}

## End(Not run)
```

`fine_tuning_jobs_list_request` *API fine-tuning: list jobs request*

Description

List your organization's fine-tuning jobs. To get more details, visit https://platform.openai.com/docs/guides/fine-tuning https://platform.openai.com/docs/api-reference/fine-tuning/list

Usage

```
fine_tuning_jobs_list_request(
  after = NULL,
  limit = NULL,
  api_key = api_get_key()
)
```

Arguments

`after`	NULL/string, identifier for the last job from the previous pagination request
`limit`	NULL/integer, number of fine-tuning jobs to retrieve (default 20)
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: 'status_code' (response$status_code) and 'message_long' (built on response content)

Examples

```
## Not run:
res_content <- fine_tuning_jobs_list_request()
if (!is_error(res_content)) {
  fine_tuning_jobs_df <- fine_tuning_fetch_jobs_list(res_content)
  print(fine_tuning_jobs_df)
}

## End(Not run)
```

fine_tuning_retrive_job_request *API fine-tuning: retrieve fine-tuning job request*

Description

Get info about a fine-tuning job. To get more details, visit https://platform.openai.com/docs/guides/fine-tuning https://platform.openai.com/docs/api-reference/fine-tuning/retrieve

Usage

```
fine_tuning_retrive_job_request(fine_tuning_job_id, api_key = api_get_key())
```

Arguments

fine_tuning_job_id	string, the ID of the fine-tuning job to get events for
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
res_content <- fine_tuning_retrive_job_request("job-id")
if (!is_error(res_content)) {
  fine_tuning_events_df <- fine_tuning_fetch_events_list(res_content)
  print(fine_tuning_events_df)
}

## End(Not run)
```

images_edit_request *API images: edit request*

Description

Creates an edited or extended image given an original image and a prompt. To get more details, visit https://platform.openai.com/docs/api-reference/images/edits

Usage

```
images_edit_request(
  image,
  prompt,
  mask = NULL,
  model = NULL,
  n = NULL,
  size = NULL,
  response_format = NULL,
  user = NULL,
  api_key = api_get_key()
)
```

Arguments

image	string/raw, the image to edit. Must be a valid PNG file, less than 4MB, and square. If mask is not provided, image must have transparency, which will be used as the mask.
prompt	string, a text description of the desired image(s). The maximum length is 1000 characters.
mask	NULL/string/raw, an additional image whose fully transparent areas (e.g. where alpha is zero) indicate where image should be edited. Must be a valid PNG file, less than 4MB, and have the same dimensions as ‘image‘.
model	NULL/string, the model to use for image generation. Only dall-e-2 is supported at this time.
n	NULL/int, the number of images to generate. Must be between 1 (default) and 10.
size	NULL/string, the size of the generated images. Must be one of 256x256, 512x512, or 1024x1024 (default).
response_format	NULL/string, the format in which the generated images are returned. Must be one of "url" or "b64_json".
user	string a unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

images_fech_set *Fetch image set from response content*

Description

To get more details, visit https://platform.openai.com/docs/api-reference/images/create https://platform.openai.com/docs/api-reference/images/edits

Usage

```
images_fech_set(res_content, prompt = NULL, size = NULL)
```

Arguments

res_content	response object returned by images_generator_request or images_edit_request
prompt	NULL/string additional info put into the image set object
size	NULL/string additional info put into the image set object

Value

Image set as a list consisting of three elements: ‘data‘, ‘prompt‘ and ‘size‘

images_generator_request *API images: create (generator) request*

Description

To get more details, visit https://platform.openai.com/docs/api-reference/images/create

Usage

```
images_generator_request(
  prompt,
  model = NULL,
  n = NULL,
  quality = NULL,
  response_format = NULL,
  size = NULL,
  style = NULL,
  user = NULL,
  api_key = api_get_key()
)
```

Arguments

prompt	string, a text description of the desired image(s). The maximum length is 1000 characters for dall-e-2 and 4000 characters for dall-e-3.
model	NULL/string, the model to use for image generation. Defaults to ‘dall-e-2‘
n	NULL/int, the number of images to generate. Must be between 1 and 10. For dall-e-3, only n=1 is supported.
quality	NULL/string, the quality of the image that will be generated. ‘hd‘ creates images with finer details and greater consistency across the image. This param is only supported for dall-e-3. Defaults to ‘standard‘.
response_format	NULL/string, the format in which the generated images are returned. Must be one of "url" or "b64_json".
size	NULL/string, the size of the generated images. Must be one of 256x256, 512x512, or 1024x1024 for dall-e-2. Must be one of 1024x1024, 1792x1024, or 1024x1792 for dall-e-3 models. 1024x1024 is default.
style	NULL/string, the style of the generated images. Must be one of ‘vivid‘ (default) or ‘natural‘. Vivid causes the model to lean towards generating hyper-real and dramatic images. Natural causes the model to produce more natural, less hyper-real looking images. This param is only supported for dall-e-3.
user	string a unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

---

images_merge_sets *Merge image set/sets*

Description

Merge given image set/sets into single images sets object (list with image sets). Have a look at images_fech_set.

Usage

```
images_merge_sets(...)
```

Arguments

...	images set(s), NULL also allowed

Value

List of image set(s)

`images_variation_request` *API images: create image variation request*

Description

Creates a variation of a given image. To get more details, visit https://platform.openai.com/docs/api-reference/images/createVariation

Usage

```
images_variation_request(
  image,
  model = NULL,
  n = NULL,
  response_format = NULL,
  size = NULL,
  user = NULL,
  api_key = api_get_key()
)
```

Arguments

`image`	string/raw, the image to edit. Must be a valid PNG file, less than 4MB, and square
`model`	NULL/string, the model to use for image generation. Only dall-e-2 is supported at this time.
`n`	NULL/int, the number of images to generate. Must be between 1 (default) and 10.
`response_format`	NULL/string, the format in which the generated images are returned. Must be one of "url" or "b64_json".
`size`	NULL/string, the size of the generated images. Must be one of 256x256, 512x512, or 1024x1024 (default).
`user`	string a unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

is_browseable *Test if RStudio Viewer (build in browser) is available*

Description

Test if RStudio Viewer (build in browser) is available

Usage

```
is_browseable()
```

Value

TRUE/FALSE

is_error *Test if object belongs to "error" class*

Description

Test if object belongs to "error" class

Usage

```
is_error(x)
```

Arguments

x	R variable

Value

TRUE/FALSE

Examples

```
is_error(FALSE)
is_error(simpleError("test"))
```

is_image_set *Test if x is a image set*

Description

Test if x is a image set - a list consisting of three elements: data, prompt and size

Usage

```
is_image_set(x)
```

Arguments

x	R variable to test

Value

TRUE/FALSE

merge_dialog_df *Merge multiple dialog data.frame*

Description

Merge multiple dialog data.frame

Usage

```
merge_dialog_df(...)
```

Arguments

...	dialog data.frame or NULL

Value

data.frame containing all input dialogs

Examples

```
d1 <- dialog_df("message 1")
d2 <- dialog_df("message 2")
print(
  merge_dialog_df(
    d1,
    merge_dialog_df(d1, d2),
    NULL,
    d2
  )
)
```

`messages_create_message_request` *API messages: create message*

Description

Create a message. To get more details, visit https://platform.openai.com/docs/api-reference/messages/createMessage https://platform.openai.com/docs/assistants

Usage

```
messages_create_message_request(
  thread_id,
  role,
  content,
  file_ids = NULL,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread to create a message for.
`role`	string, the role of the entity that is creating the message. Currently only user is supported.
`content`	string, the content of the message.
`file_ids`	NULL/character vector, a list of File IDs that the message should use. There can be a maximum of 10 files attached to a message. Useful for tools like retrieval and code_interpreter that can access and use files.
`metadata`	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maxium of 512 characters long.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`messages_list_messages_request`
API messages: list messages

Description

Returns a list of messages for a given thread. To get more details, visit https://platform.openai.com/docs/api-reference/messages/listMessages https://platform.openai.com/docs/assistants

Usage

```
messages_list_messages_request(
  thread_id,
  limit = NULL,
  order = NULL,
  after = NULL,
  before = NULL,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread the messages belong to
`limit`	NULL/integer, a limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 20.
`order`	NULL/string, sort order by the created_at timestamp of the objects. asc for ascending order and desc for descending order. Defaults to desc
`after`	NULL/string, a cursor for use in pagination. after is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include after=obj_foo in order to fetch the next page of the list.
`before`	NULL/string, a cursor for use in pagination. before is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include before=obj_foo in order to fetch the previous page of the list.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`messages_list_message_files_request`
API messages: list message files

Description

Returns a list of message files. To get more details, visit https://platform.openai.com/docs/api-reference/messages/listMessageFiles https://platform.openai.com/docs/assistants

Usage

```
messages_list_message_files_request(
  thread_id,
  message_id,
  limit = NULL,
  order = NULL,
  after = NULL,
  before = NULL,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread the messages belong to
`message_id`	string, the ID of the message that the files belongs to
`limit`	NULL/integer, a limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 20.
`order`	NULL/string, sort order by the created_at timestamp of the objects. asc for ascending order and desc for descending order. Defaults to desc
`after`	NULL/string, a cursor for use in pagination. after is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include after=obj_foo in order to fetch the next page of the list.
`before`	NULL/string, a cursor for use in pagination. before is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include before=obj_foo in order to fetch the previous page of the list.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`messages_modify_message_request`
API messages: modify message

Description

Modifies a message. To get more details, visit https://platform.openai.com/docs/api-reference/messages/modifyMessage https://platform.openai.com/docs/assistants

Usage

```
messages_modify_message_request(
  thread_id,
  message_id,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread to which this message belongs
`message_id`	string, the ID of the message to modify
`metadata`	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maximum of 512 characters long.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`messages_retrieve_message_file_request`
API messages: retrieve message file

Description

Retrieve a message file. To get more details, visit https://platform.openai.com/docs/api-reference/messages/getMessageFile https://platform.openai.com/docs/assistants

Usage

```
messages_retrieve_message_file_request(
  thread_id,
  message_id,
  file_id,
  api_key = api_get_key()
)
```

Arguments

thread_id	string, the ID of the thread the messages belong to
message_id	string, the ID of the message that the files belongs to
file_id	string, the ID of the file being retrieved
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

---

messages_retrieve_message_request

API messages: retrieve message

Description

Retrieve a message. To get more details, visit https://platform.openai.com/docs/api-reference/messages/getMessage https://platform.openai.com/docs/assistants

Usage

```
messages_retrieve_message_request(
  thread_id,
  message_id,
  api_key = api_get_key()
)
```

Arguments

thread_id	string, the ID of the thread the messages belong to
message_id	string, the ID of the message that the files belongs to
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

models_delete_request *API models: delete model request*

Description

Delete a fine-tuned model. You must have the Owner role in your organization to delete a model. To get more details, visit https://platform.openai.com/docs/models https://platform.openai.com/docs/api-reference/fine-tunes/delete-model

Usage

```
models_delete_request(model, api_key = api_get_key())
```

Arguments

model	string, the model to delete
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

models_fetch_list *Extract models from response object*

Description

Extract models list as data.frame from response object

Usage

```
models_fetch_list(res_content)
```

Arguments

res_content	response object returned by models_list_request

Value

List of available models as data.frame

Examples

```
## Not run:
res_content <- models_list_request()
if (!is_error(res_content)) {
  models_list_df <- models_fetch_list(res_content)
  print(models_list_df)
}

## End(Not run)
```

`models_list_request` *API models: list request*

Description

Lists the currently available models, and provides basic information about each one such as the owner and availability. To get more details, visit: https://platform.openai.com/docs/models https://platform.openai.com/docs/reference/models/list

Usage

```
models_list_request(api_key = api_get_key())
```

Arguments

`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
res_content <- models_list_request()
if (!is_error(res_content)) {
  models_list_df <- models_fetch_list(res_content)
  print(models_list_df)
}

## End(Not run)
```

`models_retrieve_request`

API models: retrieve model request

Description

Retrieves a model instance, providing basic information about the model such as the owner and permissioning. To get more details, visit: https://platform.openai.com/docs/models https://platform.openai.com/docs/api-reference/models/list

Usage

```
models_retrieve_request(model, api_key = api_get_key())
```

Arguments

`model`	string, the ID of the model to use for this request
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`moderation_create_request`

API moderations: create moderation

Description

Given a input text, outputs if the model classifies it as violating OpenAI's content policy. To get more details, visit https://platform.openai.com/docs/api-reference/moderations/create https://platform.openai.com/docs/guide

Usage

```
moderation_create_request(input, model = NULL, api_key = api_get_key())
```

Arguments

`input`	string, the input text to classify
`model`	string, two content moderations models are available: ‘text-moderation-stable‘ and ‘text-moderation-latest‘. The default is ‘text-moderation-latest‘ which will be automatically upgraded over time. This ensures you are always using our most accurate model. If you use ‘text-moderation-stable‘, we will provide advanced notice before updating the model. Accuracy of ’text-moderation-stable’ may be slightly lower than for ‘text-moderation-latest‘.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`print.oaii_content_audio`

Class oaii_content_audio print S3 method

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `class`es.

Usage

```
## S3 method for class 'oaii_content_audio'
print(x, ...)
```

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

`print.oaii_content_audio_aac`

Class oaii_content_audio_aac print S3 method

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `class`es.

Usage

```
## S3 method for class 'oaii_content_audio_aac'
print(x, ...)
```

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

print.oaii_content_audio_flac

Class oaii_content_audio_flac print S3 method

Description

print prints its argument and returns it *invisibly* (via invisible(x)). It is a generic function which means that new printing methods can be easily added for new classes.

Usage

```
## S3 method for class 'oaii_content_audio_flac'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

print.oaii_content_audio_mp3

Class oaii_content_audio_mp3 print S3 method

Description

print prints its argument and returns it *invisibly* (via invisible(x)). It is a generic function which means that new printing methods can be easily added for new classes.

Usage

```
## S3 method for class 'oaii_content_audio_mp3'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

`print.oaii_content_audio_opus`

Class oaii_content_audio_opus print S3 method

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `class`es.

Usage

```
## S3 method for class 'oaii_content_audio_opus'
print(x, ...)
```

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

`print.oaii_content_images`

Class oaii_content_images print S3 method

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `class`es.

Usage

```
## S3 method for class 'oaii_content_images'
print(x, ...)
```

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

print.oaii_files_df *print S3 method for oaii_files_df class*

Description

print prints its argument and returns it *invisibly* (via invisible(x)). It is a generic function which means that new printing methods can be easily added for new classes.

Usage

```
## S3 method for class 'oaii_files_df'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

print.oaii_fine_tuning_events_df *print S3 method for oaii_fine_tuning_events_df class*

Description

print prints its argument and returns it *invisibly* (via invisible(x)). It is a generic function which means that new printing methods can be easily added for new classes.

Usage

```
## S3 method for class 'oaii_fine_tuning_events_df'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

`print.oaii_fine_tuning_job` *print S3 method for oaii_fine_tuning_job class*

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `class`es.

Usage

```
## S3 method for class 'oaii_fine_tuning_job'
print(x, ...)
```

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

`print.oaii_fine_tuning_jobs_df` *print S3 method for oaii_fine_tuning_jobs_df class*

Description

`print` prints its argument and returns it *invisibly* (via `invisible(x)`). It is a generic function which means that new printing methods can be easily added for new `class`es.

Usage

```
## S3 method for class 'oaii_fine_tuning_jobs_df'
print(x, ...)
```

Arguments

`x`	an object used to select a method.
`...`	further arguments passed to or from other methods.

print.oaii_models_df *print S3 method for oaii_models_df class*

Description

print prints its argument and returns it *invisibly* (via invisible(x)). It is a generic function which means that new printing methods can be easily added for new classes.

Usage

```
## S3 method for class 'oaii_models_df'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

print.oaii_res_se *Class oaii_res_se print S3 method*

Description

print prints its argument and returns it *invisibly* (via invisible(x)). It is a generic function which means that new printing methods can be easily added for new classes.

Usage

```
## S3 method for class 'oaii_res_se'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

request *API request*

Description

To get more details, visit https://platform.openai.com/docs/api-reference/making-requests

Usage

```
request(
  endpoint,
  api_key = api_get_key(),
  body = NULL,
  query = NULL,
  encode = "json",
  method = "POST",
  content_class = NULL
)
```

Arguments

endpoint	string, API endpoint url
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)
body	One of the following: • FALSE: No body. This is typically not used with POST, PUT, or PATCH, but can be useful if you need to send a bodyless request (like GET) with VERB(). • NULL: An empty body • "": A length 0 body • upload_file("path/"): The contents of a file. The mime type will be guessed from the extension, or can be supplied explicitly as the second argument to upload_file() • A character or raw vector: sent as is in body. Use content_type() to tell the server what sort of data you are sending. • A named list: See details for encode.
query	NULL/list, query string elements as list(name1 = value1, name2 = value2)
encode	If the body is a named list, how should it be encoded? Can be one of form (application/x-www-form-urlencoded), multipart, (multipart/form-data), or json (application/json). For "multipart", list elements can be strings or objects created by upload_file(). For "form", elements are coerced to strings and escaped, use I() to prevent double-escaping. For "json", parameters are automatically "unboxed" (i.e. length 1 vectors are converted to scalars). To preserve a length 1 vector as a vector, wrap in I(). For "raw", either a character or raw vector. You'll need to make sure to set the content_type() yourself.

method	string, request method
content_class	NULL/character vector, NULL or additional class name(s) (S3) appended to the response content

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`runs_cancel_run_request`

API runs: cancel a run

Description

Cancels a run that is "in_progress". To get more details, visit https://platform.openai.com/docs/api-reference/runs/cancelRun https://platform.openai.com/docs/assistants

Usage

```
runs_cancel_run_request(thread_id, run_id, api_key = api_get_key())
```

Arguments

thread_id	string, the ID of the thread (https://platform.openai.com/docs/api-reference/threads) to which this run belongs
run_id	string, the ID of the run to cancel
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`runs_create_run_request`

API runs: create run

Description

Create a run. To get more details, visit https://platform.openai.com/docs/api-reference/runs/createRun https://platform.openai.com/docs/assistants

Usage

```
runs_create_run_request(
  thread_id,
  assistant_id,
  model = NULL,
  instructions = NULL,
  additional_instructions = NULL,
  tools = NULL,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

thread_id	string, the ID of the thread to run
assistant_id	string, the ID of the assistant to use to execute this run
model	NULL/string, the ID of the model (https://platform.openai.com/docs/api-reference/models) to be used to execute this run. If a value is provided here, it will override the model associated with the assistant. If not, the model associated with the assistant will be used.
instructions	NULL/string, overrides the instructions (https://platform.openai.com/docs/api-reference/assistants/createAssistant) of the assistant. This is useful for modifying the behavior on a per-run basis.
additional_instructions	NULL/string, appends additional instructions at the end of the instructions for the run. This is useful for modifying the behavior on a per-run basis without overriding other instructions.
tools	NULL/named list, override the tools the assistant can use for this run. This is useful for modifying the behavior on a per-run basis. Example:

```
# code interpreter tool
list(
  type = "code_interpreter"
)
# or retrieval tool
list(
  type = "retrieval"
)
# or function tool
list(
  type = "retrieval",
  function = list(
    # string (optional), a description of what the function
    # does, used by the model to choose when and how to call
    # the function.
    description =
    # string (required), the name of the function to be called.
```

```
    # Must be a-z, A-Z, 0-9, or contain underscores and dashes,
    # with a maximum length of 64.
    name =
    # list (optional), the parameters the functions accepts.
    # See the guide
  # (https://platform.openai.com/docs/guides/text-generation/function-calling)
    # for examples. Omitting parameters defines a function
    # with an empty parameter list.
    parameters = list (
    )
 )
)
```

metadata	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maximum of 512 characters long.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

Examples

```
## Not run:
runs_create_run_request(
  thread_id = "thread_abc123",
  assistant_id = "asst_abc123"
)

## End(Not run)
```

---

runs_create_thread_and_run_request

API runs: create thread and run

Description

Create a thread and run it in one request. To get more details, visit https://platform.openai.com/docs/api-reference/runs/createThreadAndRun https://platform.openai.com/docs/assistants

Usage

```
runs_create_thread_and_run_request(
  assistant_id,
  thread,
  model = NULL,
  instructions = NULL,
  tools = NULL,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

assistant_id	string, the ID of the assistant to use to execute this run
thread	NULL/list,

```
list(
  # messages "array" (list of list(s))
  messages = list(
    list(
    # string (required), the role of the entity that is creating
    # the message. Currently only user is supported.
    role =
    # string (required), the content of the message.
    content =
    # character vector (optional), a list of File IDs that
    # the message should use. There can be a maximum of 10
   # files attached to a message. Useful for tools like retrieval
    # and code_interpreter that can access and use files.
    file_ids =
    # named list (optional), set of 16 key-value pairs that
    # can be attached to an object. This can be useful for
    # storing additional information about the object in a
    # structured format. Keys can be a maximum of 64 characters
    # long and values can be a maximum of 512 characters long.
    metadata = list (
      meta1 = "value1"
    )
  )
 ),
 # named list (optional), set of 16 key-value pairs that
 # can be attached to an object. This can be useful for
# storing additional information about the object in a structured
 # format. Keys can be a maximum of 64 characters long
 # and values can be a maximum of 512 characters long.
 metadata = list(
   metaX = "value y"
 )
)
```

`model` NULL/string, the ID of the model (https://platform.openai.com/docs/api-reference/models) to be used to execute this run. If a value is provided here, it will override the model associated with the assistant. If not, the model associated with the assistant will be used.

`instructions` NULL/string, overrides the instructions (https://platform.openai.com/docs/api-reference/assistants/createAssistant) of the assistant. This is useful for modifying the behavior on a per-run basis.

`tools` NULL/named list, override the tools the assistant can use for this run. This is useful for modifying the behavior on a per-run basis. Example:

```
# code interpreter tool
list(
  type = "code_interpreter"
)
# or retrieval tool
list(
  type = "retrieval"
)
# or function tool
list(
  type = "retrieval",
  function = list(
   # string (optional), a description of what the function does, used by the model to cho
    # the function.
    description =
   # string (required), the name of the function to be called. Must be a-z, A-Z, 0-9, or c
    # dashes, with a maximum length of 64.
    name =
   # list (optional), the parameters the functions accepts. See the guide
   # (https://platform.openai.com/docs/guides/text-generation/function-calling) for e
    # defines a function with an empty parameter list.
    parameters = list (
    )
 )
)
```

`metadata` NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maximum of 512 characters long.

`api_key` string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`runs_list_runs_request` *API runs: list runs*

Description

Returns a list of runs belonging to a thread. To get more details, visit https://platform.openai.com/docs/api-reference/runs/listRuns https://platform.openai.com/docs/assistants

Usage

```
runs_list_runs_request(
  thread_id,
  limit = NULL,
  order = NULL,
  after = NULL,
  before = NULL,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread the run belongs to
`limit`	NULL/integer, a limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 20.
`order`	NULL/string, sort order by the created_at timestamp of the objects. asc for ascending order and desc for descending order. Defaults to desc
`after`	NULL/string, a cursor for use in pagination. after is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include after=obj_foo in order to fetch the next page of the list.
`before`	NULL/string, a cursor for use in pagination. before is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include before=obj_foo in order to fetch the previous page of the list.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`runs_list_run_steps_request` *API runs: list run steps*

Description

Returns a list of runs belonging to a thread. To get more details, visit https://platform.openai.com/docs/api-reference/runs/listRuns https://platform.openai.com/docs/assistants

Usage

```
runs_list_run_steps_request(
  thread_id,
  run_id,
  limit = NULL,
  order = NULL,
  after = NULL,
  before = NULL,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread the run belongs to
`run_id`	string, the ID of the run the run steps belong to
`limit`	NULL/integer, a limit on the number of objects to be returned. Limit can range between 1 and 100, and the default is 20.
`order`	NULL/string, sort order by the created_at timestamp of the objects. asc for ascending order and desc for descending order. Defaults to desc
`after`	NULL/string, a cursor for use in pagination. after is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include after=obj_foo in order to fetch the next page of the list.
`before`	NULL/string, a cursor for use in pagination. before is an object ID that defines your place in the list. For instance, if you make a list request and receive 100 objects, ending with obj_foo, your subsequent call can include before=obj_foo in order to fetch the previous page of the list.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`runs_modify_run_request`

API runs: modify run

Description

Modifies a run. To get more details, visit https://platform.openai.com/docs/api-reference/runs/modifyRun https://platform.openai.com/docs/assistants

Usage

```
runs_modify_run_request(
  thread_id,
  run_id,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread (https://platform.openai.com/docs/api-reference/threads) that was run
`run_id`	string, the ID of the run to modify
`metadata`	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maximum of 512 characters long.
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`runs_retrieve_run_request`

API runs: retrieve run

Description

Retrieves a thread. To get more details, visit https://platform.openai.com/docs/api-reference/threads/getThread https://platform.openai.com/docs/assistants

Usage

```
runs_retrieve_run_request(thread_id, run_id, api_key = api_get_key())
```

Arguments

thread_id	string, The ID of the thread (https://platform.openai.com/docs/api-reference/threads) that was run
run_id	string, the ID of the run to retrieve
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

---

runs_retrieve_run_step_request

API runs: retrieve run step

Description

Retrieves a run step. To get more details, visit https://platform.openai.com/docs/api-reference/runs/getRunStep https://platform.openai.com/docs/assistants

Usage

```
runs_retrieve_run_step_request(
  thread_id,
  run_id,
  step_id,
  api_key = api_get_key()
)
```

Arguments

thread_id	string, the ID of the thread (https://platform.openai.com/docs/api-reference/threads) to which the run and run step belongs
run_id	string, the ID of the run to which the run step belongs
step_id	string, the ID of the run step to retrieve
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`runs_submit_tool_outputs_request` *API runs: submit tool outputs to run*

Description

When a run has the status: "requires_action" and required_action.type is submit_tool_outputs, this endpoint can be used to submit the outputs from the tool calls once they're all completed. All outputs must be submitted in a single request. To get more details, visit https://platform.openai.com/docs/api-reference/runs/submitToolOutputs https://platform.openai.com/docs/assistants

Usage

```
runs_submit_tool_outputs_request(
  thread_id,
  run_id,
  tool_outputs,
  api_key = api_get_key()
)
```

Arguments

`thread_id`	string, the ID of the thread (https://platform.openai.com/docs/api-reference/threads) to which this run belongs
`run_id`	string, the ID of the run that requires the tool output submission
`tool_outputs`	list, a list of tools for which the outputs are being submitted. `list( # string (optional), the ID of the tool call in the required_action # object within the run object the output is being submitted for. tool_call_id = # string (optional), the output of the tool call to be # submitted to continue the run output = )`
`api_key`	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: 'status_code' (response$status_code) and 'message_long' (built on response content)

set_logger *Set log functions used by 'oaii' package*

Description

Set log functions used by 'oaii' package

Usage

```
set_logger(...)
```

Arguments

...	parameters in form log_level = function

Value

invisible(NULL)

Examples

```
## Not run:
logger <- log4r::logger("DEBUG")
log_error <- function(...) log4r::error(logger, ...)
log_warning <- function(...) log4r::warn(logger, ...)
log_info <- function(...) log4r::info(logger, ...)
log_debug <- function(...) log4r::debug(logger, ...)

oaii::set_logger(
  error = log_error,
  warning = log_warning,
  info = log_info,
  debug = log_debug
)

## End(Not run)
```

threads_create_thread_request *API threads: create thread*

Description

Create threads that assistants can interact with. To get more details, visit https://platform.openai.com/docs/api-reference/threads/createThread https://platform.openai.com/docs/assistants

Usage

```
threads_create_thread_request(
  messages = NULL,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

messages	NULL/list, a list of messages to start the thread with. The message "object" description:

```
list(
  list(
    # string (required), the role of the entity that is
    # creating the message. Currently only 'user' is supported.
    role = "user",
    # string (required), the content of the message.
    content =
    # character vector (optional), a list of File IDs that
    # the message should use. There can be a maximum of 10
    # files attached to a message. Useful for tools like
    # retrieval and code_interpreter that can access and
    # use files.
    file_ids =
    # named list (optional), set of 16 key-value pairs that
    # can be attached to an object. This can be useful for
    # storing additional information about the object in a
    # structured format. Keys can be a maximum of 64 characters
    # long and values can be a maximum of 512 characters long.
    metadata = list(
      meta1 = "value 2"
    )
  )
)
```

metadata	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maximum of 512 characters long.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

threads_delete_thread_request

API threads: delete thread

Description

Delete a thread. To get more details, visit https://platform.openai.com/docs/api-reference/threads/deleteThread https://platform.openai.com/docs/assistants

Usage

```
threads_delete_thread_request(thread_id, api_key = api_get_key())
```

Arguments

thread_id	string, the ID of the thread to delete
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

threads_modify_thread_request

API threads: modify thread

Description

Modifies a thread. To get more details, visit https://platform.openai.com/docs/api-reference/threads/modifyThread https://platform.openai.com/docs/assistants

Usage

```
threads_modify_thread_request(
  thread_id,
  metadata = NULL,
  api_key = api_get_key()
)
```

Arguments

thread_id	string, the ID of the thread to modify. Only the ‘metadata‘ can be modified.
metadata	NULL/list, set of 16 key-value pairs that can be attached to an object. This can be useful for storing additional information about the object in a structured format. Keys can be a maximum of 64 characters long and values can be a maximum of 512 characters long.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

---

threads_retrieve_thread_request

API threads: retrieve thread

Description

Retrieves a thread. To get more details, visit https://platform.openai.com/docs/api-reference/threads/getThread https://platform.openai.com/docs/assistants

Usage

```
threads_retrieve_thread_request(thread_id, api_key = api_get_key())
```

Arguments

thread_id	string, the ID of the thread to retrieve.
api_key	string, OpenAI API key (see https://platform.openai.com/account/api-keys)

Value

content of the httr response object or SimpleError (conditions) enhanced with two additional fields: ‘status_code‘ (response$status_code) and ‘message_long‘ (built on response content)

`timestap_dt_str` *Convert unix timestamp to formated date/time string*

Description

Convert unix timestamp to formated date/time string

Usage

```
timestap_dt_str(
  timestamp,
  format = "%Y-%m-%d %H:%M:%S",
  tz = "",
  usetz = FALSE
)
```

Arguments

`timestamp`	int, unix timestamp value
`format`	A character string. The default for the `format` methods is `"%Y-%m-%d %H:%M:%S"` if any element has a time component which is not midnight, and `"%Y-%m-%d"` otherwise. If `options("digits.secs")` is set, up to the specified number of digits will be printed for seconds.
`tz`	A character string specifying the time zone to be used for the conversion. System-specific (see `as.POSIXlt`), but `""` is the current time zone, and `"GMT"` is UTC. Invalid values are most commonly treated as UTC, on some platforms with a warning.
`usetz`	logical. Should the time zone abbreviation be appended to the output? This is used in printing times, and more reliable than using `"%Z"`.

Value

The `format` methods and `strftime` return character vectors representing the time. `NA` times are returned as `NA_character_`.

`strptime` turns character representations into an object of class `"POSIXlt"`. The time zone is used to set the `isdst` component and to set the `"tzone"` attribute if `tz != ""`. If the specified time is invalid (for example `'"2010-02-30 08:00"'`) all the components of the result are `NA`. (NB: this does means exactly what it says – if it is an invalid time, not just a time that does not exist in some time zone.)

Index