Package 'openaistream'

January 23, 2024

Type Package

Title Streaming Capabilities for 'OpenAI API' Interactions

Version 0.2.0

Maintainer Li Gen <libingfei@sina.com>

Description
Based on the 'httr2' framework, the 'OpenAI' interface supports streaming calls and model training. For more details on the API methods implemented, see the 'OpenAI' platform documentation at <https://platform.openai.com/docs/api-reference>.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.2.0), jsonlite (>= 1.8.7), httr2 (>= 0.2.3), R6 (>= 2.5.1), iterators (>= 1.0.14), curl (>= 5.1.0)

Suggests png (>= 0.1-8), testthat (>= 3.0.0), covr

RoxygenNote 7.3.0

URL https://github.com/libingfei/openaistream

BugReports https://github.com/libingfei/openaistream/issues

Config/testthat/edition 3

NeedsCompilation no

Author Li Gen [aut, cre]

Repository CRAN

Date/Publication 2024-01-23 07:50:02 UTC

R topics documented:

api_config 2
assistants 3
audio 6
base_api 8
chat 9
DataStream 10

embeddings 11
files 12
fine_tuning 14
images 16
messages 17
models 19
moderations 21
openai 22
runs 23
threads 26

Index **28**

`api_config` *api_config Class*

Description

api_config Class

api_config Class

Details

A R6 class to manage config.

Methods

Public methods:

- `api_config$new()`
- `api_config$set_api_key()`
- `api_config$set_proxy()`
- `api_config$get_proxy()`
- `api_config$get_api_key()`
- `api_config$get_api_endpoints()`
- `api_config$clone()`

Method `new()`**:** Initialize the api_config object

Usage:

```
api_config$new()
```

Method `set_api_key()`**:** Configure the api_key settings.

Usage:

```
api_config$set_api_key(api_key)
```

Arguments:

`api_key` your openai_key

Method `set_proxy()`**:** Configure the proxy settings.

Usage:

```
api_config$set_proxy(proxy_ip, proxy_port)
```

Arguments:

`proxy_ip` character Required. The IP address of the proxy.

`proxy_port` character Required. The port number of the proxy.

Method `get_proxy()`**:** Configure the proxy gettings.

Usage:

```
api_config$get_proxy()
```

Method `get_api_key()`**:** Api key gettings.

Usage:

```
api_config$get_api_key()
```

Method `get_api_endpoints()`**:** Endpoints gettings.

Usage:

```
api_config$get_api_endpoints()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
api_config$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`assistants` *assistants Class*

Description

assistants Class

assistants Class

Super class

`openaistream::openai` -> `assistants`

Methods

Public methods:

- assistants$create()
- assistants$retrieve()
- assistants$modify()
- assistants$delete()
- assistants$list()
- assistants$file_create()
- assistants$file_retrieve()
- assistants$file_delete()
- assistants$file_list()
- assistants$clone()

Method `create()`**:** Create an assistant with a model and instructions.

Usage:

```
assistants$create(model, ..., verbosity = 0)
```

Arguments:

`model` character Required. 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.

`...` Additional parameters as required by the OpenAI API.For example:name;description;instructions;tools;file_ids;meta

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: An assistant object.

Method `retrieve()`**:** Retrieves an assistant.

Usage:

```
assistants$retrieve(assistant_id, verbosity = 0)
```

Arguments:

`assistant_id` character Required. The ID of the assistant to retrieve.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The assistant object matching the specified ID.

Method `modify()`**:** Modifies an assistant.

Usage:

```
assistants$modify(assistant_id, ..., verbosity = 0)
```

Arguments:

`assistant_id` character Required. The ID of the assistant to modify.

`...` Additional parameters as required by the OpenAI API.For example:model;name;description;instructions;tools;file_id

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The assistant object matching the specified ID.

Method `delete()`**:** Delete an assistant.

Usage:

```
assistants$delete(assistant_id, verbosity = 0)
```

Arguments:

`assistant_id` character Required. The ID of the assistant to delete.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Deletion status

Method `list()`**:** Returns a list of assistants

Usage:

```
assistants$list(..., verbosity = 0)
```

Arguments:

`...` Additional parameters as required by the OpenAI API.For example:limit;order;after;before;

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: A list of assistant objects.

Method `file_create()`**:** Create an assistant file by attaching a File to an assistant.

Usage:

```
assistants$file_create(assistant_id, file_id, verbosity = 0)
```

Arguments:

`assistant_id` character Required. The ID of the assistant for which to create a File.

`file_id` character Required. A File ID (with purpose="assistants") that the assistant should use. Useful for tools like retrieval and code_interpreter that can access files.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: An assistant file object.

Method `file_retrieve()`**:** Retrieves an AssistantFile.

Usage:

```
assistants$file_retrieve(assistant_id, file_id, verbosity = 0)
```

Arguments:

`assistant_id` character Required. The ID of the assistant who the file belongs to.

`file_id` character Required. The ID of the file we're getting.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The assistant file object matching the specified ID.

Method `file_delete()`**:** Delete an assistant file.

Usage:

```
assistants$file_delete(assistant_id, file_id, verbosity = 0)
```

Arguments:

`assistant_id` character Required. The ID of the assistant who the file belongs to.

`file_id` character Required. The ID of the file we're getting.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Deletion status

Method `file_list()`**:** Retrieve a list of files from OpenAI.

Usage:

```
assistants$file_list(assistant_id, ..., verbosity = 0)
```

Arguments:

`assistant_id` character Required. The ID of the assistant who the file belongs to.

`...` Additional parameters as required by the OpenAI API.For example:limit,order,after,before

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: A list of files.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
assistants$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`audio` *audio Class*

Description

To turn audio into text or text into audio

Super class

`openaistream::openai` -> `audio`

Methods

Public methods:

- `audio$speech()`
- `audio$transcription()`
- `audio$translation()`
- `audio$clone()`

Method `speech()`**:** Generates audio from the input text.

Usage:

```
audio$speech(
  model = "tts-1",
  input,
  voice = "alloy",
  stream = F,
  num = 100,
  ...,
  verbosity = 0
)
```

Arguments:

`model` character Required. One of the available TTS models: tts-1 or tts-1-hd

`input` character Required. The text to generate audio for. The maximum length is 4096 characters.

`voice` character Required. The voice to use when generating the audio. Supported voices are alloy, echo, fable, onyx, nova, and shimmer.

`stream` logical. Using the stream call, it will return raw data of the specified length, which can be saved in the set format such as mp3, etc. For details, please see the examples.

`num` The num parameter controls the number of raw entries returned by a stream in one go. Note that this is different from the n parameter, which specifies the number of results returned. For detailed information on the n parameter, please refer to OpenAI's API documentation.

`...` Additional parameters as required by the OpenAI API.For example:response_format;speed....

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The audio file content.

Method `transcription()`**:** Transcribes audio into the input language.

Usage:

```
audio$transcription(path, model = "whisper-1", ..., verbosity = 0)
```

Arguments:

`path` character Required. The audio file object (not file name) to transcribe, in one of these formats: flac, mp3, mp4, mpeg, mpga, m4a, ogg, wav, or webm.

`model` character Required. ID of the model to use. Only whisper-1 is currently available.

`...` Additional parameters as required by the OpenAI API.For example:language;prompt;response_format;temperature...

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The transcribed text.

Method `translation()`**:** Translates audio into English.

Usage:

```
audio$translation(path, model = "whisper-1", ..., verbosity = 0)
```

Arguments:

`path` character Required. The audio file object (not file name) to transcribe, in one of these formats: flac, mp3, mp4, mpeg, mpga, m4a, ogg, wav, or webm.

`model` character Required. ID of the model to use. Only whisper-1 is currently available.

`...` Additional parameters as required by the OpenAI API.For example:prompt;response_format;temperature....

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The transcribed text.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
audio$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`base_api` *base_api Class*

Description

base_api Class

base_api Class

Methods

Public methods:

- `base_api$new()`
- `base_api$clone()`

Method `new()`**:** Initialize the OpenAI API interface with the provided API key and other.

Usage:

```
base_api$new(etc)
```

Arguments:

`etc` Config.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
base_api$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

chat *chat Class*

Description

chat Class

chat Class

Super class

`openaistream::openai` -> `chat`

Methods

Public methods:

- `chat$create()`
- `chat$clone()`

Method `create()`**:** Creates a model response for the given chat conversation.

Usage:

```
chat$create(messages, model, stream = F, n = 1, num = 2, verbosity = 0, ...)
```

Arguments:

`messages` list Required. A list of messages comprising the conversation so far.

`model` character Required. The model to use for generating chat completions.

`stream` logical. Whether to stream back partial progress. If set, tokens will be sent as data-only server-sent events as they become available.

`n` integer. How many chat completion choices to generate for each input message. Note that you will be charged based on the number of generated tokens across all of the choices. Keep n as 1 to minimize costs. NOTE: The parameter sometimes fails to work when 'num' is not assigned a value, and the reason for this is currently unclear. When failure occurs, try assigning any integer greater than 0 to the 'num' parameter.

`num` The num parameter controls the number of text entries returned by a stream in one go. Note that this is different from the n parameter, which specifies the number of results returned. For detailed information on the n parameter, please refer to OpenAI's API documentation.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

`...` Additional parameters as required by the OpenAI API.For example:max_tokens;n;stop;temperature.....

Returns: Returns a chat completion object, or a streamed sequence of chat completion chunk objects if the request is streamed.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
chat$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

DataStream *DataStream Class*

Description

DataStream Class

DataStream Class

Details

A R6 class to manage data streams.

Active bindings

`next_value` The next data value from the DataStream or an error message.

Methods

Public methods:

- `DataStream$new()`
- `DataStream$close()`
- `DataStream$get_state()`
- `DataStream$clone()`

Method `new()`**:** Initialize the DataStream object

Usage:

```
DataStream$new(requery, num = 2)
```

Arguments:

`requery` The requery object, usually of class 'curl'

`num` Numeric. Defines the number of lines to read from the data source.

Method `close()`**:** Close the DataStream

This method tries to destroy the object, and closes the requery if opened.

Usage:

```
DataStream$close()
```

Returns: A character message indicating the status of the close operation.

Method `get_state()`**:** Get the status of the DataStream

Usage:

```
DataStream$get_state()
```

Returns: A character string indicating the current status of the DataStream.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
DataStream$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`embeddings` *embeddings Class*

Description

embeddings Class

embeddings Class

Super class

`openaistream::openai` -> `embeddings`

Methods

Public methods:

- `embeddings$create()`
- `embeddings$clone()`

Method `create()`**:** Creates an embedding vector representing the input text.

Usage:

```
embeddings$create(model, input, ..., verbosity = 0)
```

Arguments:

`model` character Required. 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.

`input` character Required. 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.

`...` Additional parameters as required by the OpenAI API.For example:encoding_format;user....

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Embeddings for the input data.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
embeddings$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`files` *Files Class*

Description

Files Class

Files Class

Super class

`openaistream::openai` -> `files`

Methods

Public methods:

- `files$upload()`
- `files$list()`
- `files$delete()`
- `files$retrieve()`
- `files$retrieve_content()`
- `files$clone()`

Method `upload()`**:** 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 to learn more about the types of files supported. The Fine-tuning API only supports .jsonl files.

Usage:

```
files$upload(path = NULL, verbosity = 0, purpose = "fine-tune")
```

Arguments:

`path` character Required. Path to the file that needs to be uploaded.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

`purpose` The intended purpose of the uploaded file. Use "fine-tune" for Fine-tuning and "assistants" for Assistants and Messages. This allows us to validate the format of the uploaded file is correct for fine-tuning.

Returns: The uploaded File object.

Method `list()`**:** Returns a list of files that belong to the user's organization.

Usage:

```
files$list(..., verbosity = 0)
```

Arguments:

`...` Additional parameters as required by the OpenAI API.For example:purpose

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: A list of File objects.

Method `delete()`**:** Delete a file.

Usage:

```
files$delete(file_id, verbosity = 0)
```

Arguments:

`file_id` character Required. The ID of the file to use for this request.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Deletion status.

Method `retrieve()`**:** Returns information about a specific file.

Usage:

```
files$retrieve(file_id, verbosity = 0)
```

Arguments:

`file_id` character Required. The ID of the file to retrieve details for.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The File object matching the specified ID.

Method `retrieve_content()`**:** Returns information about a specific file content.

Usage:

```
files$retrieve_content(file_id, verbosity = 0)
```

Arguments:

`file_id` character Required. The ID of the file to retrieve details for.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The File object matching the specified ID.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
files$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

fine_tuning *DataStream Class*

Description

DataStream Class

DataStream Class

Super class

openaistream::openai -> fine_tuning

Methods

Public methods:

- fine_tuning$create()
- fine_tuning$list()
- fine_tuning$retrieve()
- fine_tuning$cancel()
- fine_tuning$events()
- fine_tuning$clone()

Method `create()`**:** Creates a job that fine-tunes a specified model from a given dataset. Response includes details of the enqueued job including job status and the name of the fine-tuned models once complete.

Usage:

```
fine_tuning$create(
  model,
  training_file,
  hyperparameters = list(n_epochs = 1),
  ...,
  verbosity = 0
)
```

Arguments:

`model` character Required. The model ID.

`training_file` character Required. The file used for training.

`hyperparameters` list. The hyperparameters used for the fine-tuning job. include batch_size;learning_rate_multiplier;n_

`...` Additional parameters as required by the OpenAI API.For example:validation_file.....

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Response indicating the success or failure of the fine-tuning job creation.

Method `list()`**:** List your organization's fine-tuning jobs

Usage:

```
fine_tuning$list(verbosity = 0, ...)
```

Arguments:

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

`...` Additional parameters as required by the OpenAI API. For example:after,limit.

Returns: A list of paginated fine-tuning job objects.

Method `retrieve()`**:** Get info about a fine-tuning job.

Usage:

```
fine_tuning$retrieve(job_id, verbosity = 0)
```

Arguments:

`job_id` character Required. The ID of the fine-tuning job.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The fine-tuning object with the given ID.

Method `cancel()`**:** Immediately cancel a fine-tune job.

Usage:

```
fine_tuning$cancel(job_id, verbosity = 0)
```

Arguments:

`job_id` character Required. The ID of the fine-tuning job to cancel.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The cancelled fine-tuning object.

Method `events()`**:** Get status updates for a fine-tuning job.

Usage:

```
fine_tuning$events(job_id, ..., verbosity = 0)
```

Arguments:

`job_id` character Required. The ID of the fine-tuning job to get events for.

`...` Additional parameters as required by the OpenAI API. For example:after,limit.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: A list of fine-tuning event objects.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
fine_tuning$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`images` *images Class*

Description

images Class

images Class

Super class

`openaistream::openai` -> `images`

Methods

Public methods:

- `images$create()`
- `images$edit()`
- `images$variation()`
- `images$clone()`

Method `create()`**:** Creates an image given a prompt.

Usage:

```
images$create(prompt, ..., verbosity = 0)
```

Arguments:

`prompt` character Required. 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

`...` Additional parameters as required by the OpenAI API.For example:n;quality;response_format...

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Returns a list of image objects.

Method `edit()`**:** Creates an edited or extended image given an original image and a prompt.

Usage:

```
images$edit(image, prompt, ..., verbosity = 0)
```

Arguments:

`image` character Required. 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` character Required. A text description of the desired image(s). The maximum length is 1000 characters.

`...` Additional parameters as required by the OpenAI API.For example:mask;model;n...

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Returns a list of image objects.

Method `variation()`**:** Creates a variation of a given image.

Usage:

```
images$variation(image, ..., verbosity = 0)
```

Arguments:

`image` character Required. The image to use as the basis for the variation(s). Must be a valid PNG file, less than 4MB, and square.

`...` Additional parameters as required by the OpenAI API.For example:model;n;response_format

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Returns a list of image objects.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
images$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`messages` *messages Class*

Description

messages Class

messages Class

Super class

`openaistream::openai` -> `messages`

Methods

Public methods:

- `messages$create()`
- `messages$retrieve()`
- `messages$modify()`
- `messages$list()`
- `messages$file_retrieve()`
- `messages$file_list()`
- `messages$clone()`

Method `create()`**:** Create a message.

Usage:

```
messages$create(thread_id, role, content, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread to create a message for.

`role` character Required. The role of the entity that is creating the message. Currently only user is supported.

`content` character Required. The content of the message.

`...` Additional parameters as required by the OpenAI API. For example:file_ids,metadata

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: A message object.

Method `retrieve()`**:** Retrieve a message.

Usage:

```
messages$retrieve(thread_id, message_id, verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread the message belongs to.

`message_id` character Required. The ID of the message to retrieve.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The message object matching the specified ID.

Method `modify()`**:** Modifies a message.

Usage:

```
messages$modify(thread_id, message_id, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread the message belongs to.

`message_id` character Required. The ID of the message to retrieve.

`...` Additional parameters as required by the OpenAI API. For example:metadata

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The modified message object.

Method `list()`**:** Returns a list of messages for a given thread.

Usage:

```
messages$list(thread_id, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread the messages belong to.

`...` Additional parameters as required by the OpenAI API.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: A list of message objects.

Method `file_retrieve()`**:** Retrieves a message file.

Usage:

```
messages$file_retrieve(thread_id, message_id, file_id, verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread the message belongs to.

`message_id` character Required. The ID of the message the file belongs to.

`file_id` character Required The ID of the file being retrieved.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers;

Returns: The message file object.

Method `file_list()`**:** Returns a list of message files.

Usage:

```
messages$file_list(thread_id, message_id, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread the message belongs to.

`message_id` character Required. The ID of the message the file belongs to.

`...` Additional parameters as required by the OpenAI API.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers;

Returns: A list of message file objects.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
messages$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`models` *Models Class*

Description

Models Class

Models Class

Super class

`openaistream::openai` -> `models`

Methods

Public methods:

- `models$list()`
- `models$retrieve()`
- `models$delete()`
- `models$clone()`

Method `list()`**:** Lists the currently available models, and provides basic information about each one such as the owner and availability.

Usage:

```
models$list(verbosity = 0)
```

Arguments:

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: A list of model objects.

Method `retrieve()`**:** Retrieves a model instance, providing basic information about the model such as the owner and permissioning.

Usage:

```
models$retrieve(model, verbosity = 0)
```

Arguments:

`model` character Required. The ID of the model to use for this request

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: The model object matching the specified ID.

Method `delete()`**:** Delete a fine-tuned model. You must have the Owner role in your organization to delete a model.

Usage:

```
models$delete(model, verbosity = 0)
```

Arguments:

`model` character Required. The model to delete

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Deletion status.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
models$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`moderations` *moderations Class*

Description

moderations Class

moderations Class

Super class

`openaistream::openai` -> `moderations`

Methods

Public methods:

- `moderations$create()`
- `moderations$clone()`

Method `create()`**:** Classifies if text violates OpenAI's Content Policy.

Usage:

```
moderations$create(input, ..., verbosity = 0)
```

Arguments:

`input` character Required. The input text to classify.

`...` Additional parameters as required by the OpenAI API.For example:model...

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.).

Returns: Returns a list of image objects.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
moderations$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

openai *An R6 Class Interface to OpenAI API*

Description

Provides methods to interact with OpenAI API including fetching model details, generating completions, managing files, and more. Always ensure that the API key is kept private.

Active bindings

`models` class

`files` class

`fine_tuning` class

`chat` class

`audio` class

`embeddings` class

`images` class

`moderations` class

`assistants` class

`threads` class

`messages` class

`runs` class

Methods

Public methods:

- `openai$new()`
- `openai$set_proxy()`
- `openai$clone()`

Method `new()`**:** Initialize the OpenAI API interface with the provided API key.

Usage:

```
openai$new(api_key)
```

Arguments:

`api_key` The OpenAI API key.

Method `set_proxy()`**:** Configure the proxy settings.

Usage:

```
openai$set_proxy(proxy_ip, proxy_port)
```

Arguments:

`proxy_ip` character Required. The IP address of the proxy.

`proxy_port` character Required. The port number of the proxy.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
openai$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
Sys.setenv(OPENAI_KEY="you openai key")
aaa <- openai$new(Sys.getenv("OPENAI_KEY"))
#if need proxy
#aaa$set_proxy("127.0.0.1", 10890)
# List model
aaa$models$list()
```

`runs` *runs Class*

Description

runs Class

runs Class

Super class

`openaistream::openai` -> `runs`

Methods

Public methods:

- `runs$create()`
- `runs$retrieve()`
- `runs$modify()`
- `runs$list()`
- `runs$submit_tool_outputs()`
- `runs$cancel()`
- `runs$create_tread()`
- `runs$steps_retrieve()`
- `runs$steps_list()`
- `runs$clone()`

Method `create()`**:** Create a run.

Usage:

```
runs$create(thread_id, assistant_id, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread to run.

`assistant_id` character Required. The ID of the assistant to use to execute this run.

`...` Additional parameters as required by the OpenAI API.For example:model,instructions,tools,metadata

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: A run object.

Method `retrieve()`**:** Retrieves a run.

Usage:

```
runs$retrieve(thread_id, run_id, verbosity = 0)
```

Arguments:

`thread_id` character Required The ID of the thread the run belongs to.

`run_id` character Required The ID of the run to retrieve.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The run object matching the specified ID.

Method `modify()`**:** Modifies a run.

Usage:

```
runs$modify(thread_id, run_id, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required The ID of the thread the run belongs to.

`run_id` character Required The ID of the run to retrieve.

`...` Additional parameters as required by the OpenAI API.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The modified run object matching the specified ID.

Method `list()`**:** Returns a list of runs for a given thread.

Usage:

```
runs$list(thread_id, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required The ID of the thread the run belongs to.

`...` Additional parameters as required by the OpenAI API.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: A list of run objects.

Method `submit_tool_outputs()`**:** 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.

Usage:

```
runs$submit_tool_outputs(thread_id, run_id, tool_outputs, verbosity = 0)
```

Arguments:

`thread_id` character Required The ID of the thread the run belongs to.

`run_id` character Required The ID of the run to retrieve.

`tool_outputs` character Required. A list of tools for which the outputs are being submitted.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The modified run object matching the specified ID.

Method `cancel()`**:** Cancels a run that is in_progress.

Usage:

```
runs$cancel(thread_id, run_id, verbosity = 0)
```

Arguments:

`thread_id` character Required The ID of the thread the run belongs to.

`run_id` character Required The ID of the run to retrieve.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The modified run object matching the specified ID.

Method `create_tread()`**:** Create a thread and run it in one request.

Usage:

```
runs$create_tread(assistant_id, ..., verbosity = 0)
```

Arguments:

`assistant_id` character Required The ID of the assistant to use to execute this run.

`...` Additional parameters as required by the OpenAI API.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: A run object.

Method `steps_retrieve()`**:** Retrieves a run step.

Usage:

```
runs$steps_retrieve(thread_id, run_id, step_id, verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread to which the run and run step belongs.

`run_id` character Required. The ID of the run the step belongs to.

`step_id` character Required. The ID of the step to retrieve.

`verbosity` numeric. Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The run step object matching the specified ID.

Method `steps_list()`**:** Returns a list of run steps belonging to a run.

Usage:

```
runs$steps_list(thread_id, run_id, ..., verbosity = 0)
```

Arguments:

`thread_id` character Required The ID of the thread the run belongs to.

`run_id` character Required The ID of the run the step belongs to.

`...` Additional parameters as required by the OpenAI API.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: A list of run step objects.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
runs$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`threads` *threads Class*

Description

threads Class

threads Class

Super class

`openaistream::openai` -> `threads`

Methods

Public methods:

- `threads$create()`
- `threads$retrieve()`
- `threads$modify()`
- `threads$delete()`
- `threads$clone()`

Method `create()`**:** Create a thread.

Usage:

```
threads$create(..., verbosity = 0)
```

Arguments:

`...` Additional parameters as required by the OpenAI API.For example:messages;name;metadata

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: A thread object.

Method `retrieve()`**:** Retrieves a thread.

Usage:

```
threads$retrieve(thread_id, verbosity = 0)
```

Arguments:

`thread_id` character Required. The ID of the thread to retrieve.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The thread object matching the specified ID.

Method `modify()`**:** Modifies a thread.

Usage:

```
threads$modify(thread_id, ..., verbosity = 0)
```

Arguments:

`thread_id` The ID of the thread to modify. Only the metadata can be modified.

`...` Additional parameters as required by the OpenAI API.For example:metadata

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: The modified thread object matching the specified ID.

Method `delete()`**:** Delete a thread.

Usage:

```
threads$delete(thread_id, verbosity = 0)
```

Arguments:

`thread_id` character Required The ID of the thread to delete.

`verbosity` numeric Verbosity level for the API call(0:no output;1:show headers; 2:show headers and bodies;3: show headers, bodies, and curl status messages.)

Returns: Deletion status.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
threads$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index

api_config, 2
assistants, 3
audio, 6

base_api, 8

chat, 9

DataStream, 10

embeddings, 11

files, 12
fine_tuning, 14

images, 16

messages, 17
models, 19
moderations, 21

openai, 22
openaistream::openai, *3*, *6*, *9*, *11*, *12*, *14*, *16*, *17*, *19*, *21*, *23*, *26*

runs, 23

threads, 26