

Package ‘tidyllm’

December 8, 2024

Title Tidy Integration of Large Language Models

Version 0.3.0

Description A tidy interface for integrating large language model (LLM) APIs such as 'Claude', 'Openai', 'Groq', 'Mistral' and local models via 'Ollama' into R workflows. The package supports text and media-based interactions, interactive message history, batch request APIs, and a tidy, pipeline-oriented interface for streamlined integration into data workflows. Web services are available at <https://www.anthropic.com>, <https://openai.com>, <https://groq.com>, <https://mistral.ai/> and <https://ollama.com>.

License MIT + file LICENSE

URL <https://edubruell.github.io/tidyllm/>

BugReports <https://github.com/edubruell/tidyllm/issues>

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Suggests knitr, rmarkdown, testthat (>= 3.0.0), tidyverse, httpptest2, httpuv

Imports S7 (>= 0.2.0), base64enc, glue, jsonlite, curl, httr2, lubridate, purrr, rlang, stringr, grDevices, pdftools, tibble, cli, png, lifecycle

Collate 'tidyllm-package.R' 'utilites.R' 'LLMMessage.R' 'APIProvider.R' 'llm_message.R' 'llm_verbs.R' 'media.R' 'message_retrieval.R' 'perform_api_requests.R' 'rate_limits.R' 'tidyllm_schema.R' 'pdfbatch.R' 'api_openai.R' 'api_claude.R' 'api_gemini.R' 'api_ollama.R' 'api_azure_openai.R' 'api_groq.R' 'api_mistral.R' 'api_perplexity.R' 'zzz.R'

Depends R (>= 4.2.0)

Config/testthat/edition 3

NeedsCompilation no

Author Eduard Brüll [aut, cre],
Jia Zhang [ctb]

Maintainer Eduard Brüll <eduard.brue11@zew.de>

Repository CRAN

Date/Publication 2024-12-08 22:20:02 UTC

Contents

azure_openai	3
azure_openai_chat	4
azure_openai_embedding	6
cancel_openai_batch	7
chat	7
chatgpt	9
check_batch	10
check_claude_batch	10
check_mistral_batch	11
check_openai_batch	12
claude	13
claude_chat	13
df_llm_message	15
embed	15
fetch_batch	17
fetch_claude_batch	18
fetch_mistral_batch	19
fetch_openai_batch	19
gemini	20
gemini_chat	21
gemini_delete_file	22
gemini_embedding	23
gemini_file_metadata	23
gemini_list_files	24
gemini_upload_file	24
get_metadata	25
get_reply	26
get_reply_data	26
get_user_message	27
groq	28
groq_chat	28
groq_transcribe	30
list_batches	31
list_claude_batches	32
list_mistral_batches	32
list_openai_batches	33
LLMMessage	34
llm_message	34
mistral	36
mistral_chat	36
mistral_embedding	38

ollama	38
ollama_chat	39
ollama_download_model	41
ollama_embedding	42
ollama_list_models	42
openai	43
openai_chat	43
openai_embedding	45
pdf_page_batch	46
perplexity	47
perplexity_chat	47
rate_limit_info	49
send_batch	49
send_claude_batch	51
send_mistral_batch	52
send_openai_batch	53
tidyllm_schema	55

Index**57**

azure_openai	<i>Azure-OpenAI Endpoint Provider Function</i>
--------------	--

Description

The `azure_openai()` function acts as an interface for interacting with the Azure OpenAI API through main `tidyllm` verbs.

Usage

```
azure_openai(..., .called_from = NULL)
```

Arguments

<code>...</code>	Parameters to be passed to the Azure OpenAI API specific function, such as model configuration, input text, or API-specific options.
<code>.called_from</code>	An internal argument that specifies which action (e.g., <code>chat</code>) the function is being invoked from. This argument is automatically managed and should not be modified by the user.

Details

`azure_openai()` currently routes messages only to `azure_openai_chat()` when used with `chat()`.

Value

The result of the requested action, depending on the specific function invoked (currently, only an updated `LLMMessage` object for `azure_openai_chat()`).

azure_openai_chat	<i>Send LLM Messages to an OpenAI Chat Completions endpoint on Azure</i>
-------------------	--

Description

This function sends a message history to the Azure OpenAI Chat Completions API and returns the assistant's reply. This function is work in progress and not fully tested

Usage

```
azure_openai_chat(  
    .llm,  
    .endpoint_url = Sys.getenv("AZURE_ENDPOINT_URL"),  
    .deployment = "gpt-4o-mini",  
    .api_version = "2024-08-01-preview",  
    .max_completion_tokens = NULL,  
    .frequency_penalty = NULL,  
    .logit_bias = NULL,  
    .logprobs = FALSE,  
    .top_logprobs = NULL,  
    .presence_penalty = NULL,  
    .seed = NULL,  
    .stop = NULL,  
    .stream = FALSE,  
    .temperature = NULL,  
    .top_p = NULL,  
    .timeout = 60,  
    .verbose = FALSE,  
    .json_schema = NULL,  
    .dry_run = FALSE,  
    .max_tries = 3  
)
```

Arguments

<code>.llm</code>	An LLMMessage object containing the conversation history.
<code>.endpoint_url</code>	Base URL for the API (default: <code>Sys.getenv("AZURE_ENDPOINT_URL")</code>).
<code>.deployment</code>	The identifier of the model that is deployed (default: <code>"gpt-4o-mini"</code>).
<code>.api_version</code>	Which version of the API is deployed (default: <code>"2024-08-01-preview"</code>)
<code>.max_completion_tokens</code>	An upper bound for the number of tokens that can be generated for a completion, including visible output tokens and reasoning tokens.
<code>.frequency_penalty</code>	Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far.

<code>.logit_bias</code>	A named list modifying the likelihood of specified tokens appearing in the completion.
<code>.logprobs</code>	Whether to return log probabilities of the output tokens (default: FALSE).
<code>.top_logprobs</code>	An integer between 0 and 20 specifying the number of most likely tokens to return at each token position.
<code>.presence_penalty</code>	Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far.
<code>.seed</code>	If specified, the system will make a best effort to sample deterministically.
<code>.stop</code>	Up to 4 sequences where the API will stop generating further tokens.
<code>.stream</code>	If set to TRUE, the answer will be streamed to console as it comes (default: FALSE).
<code>.temperature</code>	What sampling temperature to use, between 0 and 2. Higher values make the output more random.
<code>.top_p</code>	An alternative to sampling with temperature, called nucleus sampling.
<code>.timeout</code>	Request timeout in seconds (default: 60).
<code>.verbose</code>	Should additional information be shown after the API call (default: FALSE).
<code>.json_schema</code>	A JSON schema object as R list to enforce the output structure (If defined has precedence over JSON mode).
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object (default: FALSE).
<code>.max_tries</code>	Maximum retries to perform request

Value

A new `LLMMessage` object containing the original messages plus the assistant's response.

Examples

```
## Not run:
# Basic usage
msg <- llm_message("What is R programming?")
result <- azure_openai_chat(msg)

# With custom parameters
result2 <- azure_openai_chat(msg,
  .deployment = "gpt-4o-mini",
  .temperature = 0.7,
  .max_tokens = 1000)

## End(Not run)
```

 azure_openai_embedding

Generate Embeddings Using OpenAI API on Azure

Description

Generate Embeddings Using OpenAI API on Azure

Usage

```
azure_openai_embedding(
  .input,
  .deployment = "text-embedding-3-small",
  .endpoint_url = Sys.getenv("AZURE_ENDPOINT_URL"),
  .api_version = "2023-05-15",
  .truncate = TRUE,
  .timeout = 120,
  .dry_run = FALSE,
  .max_tries = 3
)
```

Arguments

<code>.input</code>	A character vector of texts to embed or an LLMMessageobject
<code>.deployment</code>	The embedding model identifier (default: "text-embedding-3-small").
<code>.endpoint_url</code>	Base URL for the API (default: Sys.getenv("AZURE_ENDPOINT_URL")).
<code>.api_version</code>	What API-Version othe Azure OpenAI API should be used (default: "2023-05-15")
<code>.truncate</code>	Whether to truncate inputs to fit the model's context length (default: TRUE).
<code>.timeout</code>	Timeout for the API request in seconds (default: 120).
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object.
<code>.max_tries</code>	Maximum retry attempts for requests (default: 3).

Value

A tibble with two columns: `input` and `embeddings`. The `input` column contains the texts sent to embed, and the `embeddings` column is a list column where each row contains an embedding vector of the sent input.

cancel_openai_batch *Cancel an In-Progress OpenAI Batch*

Description

This function cancels an in-progress batch created through the OpenAI API. The batch will be moved to a "cancelling" state and, eventually, "cancelled."

Usage

```
cancel_openai_batch(.batch_id, .dry_run = FALSE, .max_tries = 3, .timeout = 60)
```

Arguments

.batch_id	Character; the unique identifier for the batch to cancel.
.dry_run	Logical; if TRUE, returns the constructed request without executing it (default: FALSE).
.max_tries	Integer; maximum number of retries if the request fails (default: 3).
.timeout	Integer; request timeout in seconds (default: 60).

Value

A list containing the response from the OpenAI API about the cancellation status.

chat *Chat with a Language Model*

Description

The chat() function sends a message to a language model via a specified provider and returns the response. It routes the provided LLMMessage object to the appropriate provider-specific chat function, while allowing for the specification of common arguments applicable across different providers.

Usage

```
chat(  
  .llm,  
  .provider = getOption("tidyllm_chat_default"),  
  .dry_run = NULL,  
  .stream = NULL,  
  .temperature = NULL,  
  .timeout = NULL,  
  .top_p = NULL,  
  .max_tries = NULL,
```

```

.model = NULL,
.verbose = NULL,
.json_schema = NULL,
.seed = NULL,
.stop = NULL,
.frequency_penalty = NULL,
.presence_penalty = NULL
)

```

Arguments

<code>.llm</code>	An LLMMessage object containing the message or conversation history to send to the language model.
<code>.provider</code>	A function or function call specifying the language model provider and any additional parameters. This should be a call to a provider function like <code>openai()</code> , <code>claude()</code> , etc. You can also set a default provider function via the <code>tidyllm_chat_default</code> option.
<code>.dry_run</code>	Logical; if TRUE, simulates the request without sending it to the provider. Useful for testing.
<code>.stream</code>	Logical; if TRUE, streams the response from the provider in real-time.
<code>.temperature</code>	Numeric; controls the randomness of the model's output (0 = deterministic).
<code>.timeout</code>	Numeric; the maximum time (in seconds) to wait for a response.
<code>.top_p</code>	Numeric; nucleus sampling parameter, which limits the sampling to the top cumulative probability p .
<code>.max_tries</code>	Integer; the maximum number of retries for failed requests.
<code>.model</code>	Character; the model identifier to use (e.g., "gpt-4").
<code>.verbose</code>	Logical; if TRUE, prints additional information about the request and response.
<code>.json_schema</code>	List; A JSON schema object as R list to enforce the output structure
<code>.seed</code>	Integer; sets a random seed for reproducibility.
<code>.stop</code>	Character vector; specifies sequences where the model should stop generating further tokens.
<code>.frequency_penalty</code>	Numeric; adjusts the likelihood of repeating tokens (positive values decrease repetition).
<code>.presence_penalty</code>	Numeric; adjusts the likelihood of introducing new tokens (positive values encourage novelty).

Details

The `chat()` function provides a unified interface for interacting with different language model providers. Common arguments such as `.temperature`, `.model`, and `.stream` are supported by most providers and can be passed directly to `chat()`. If a provider does not support a particular argument, an error will be raised.

Advanced provider-specific configurations can be accessed via the provider functions.

Value

An updated LLMMessage object containing the response from the language model.

Examples

```
## Not run:
# Basic usage with OpenAI provider
llm_message("Hello World") |>
  chat(ollama(.ollama_server = "https://my-ollama-server.de"), .model="mixtral")

  chat(mistral, .model="mixtral")

# Use streaming with Claude provider
llm_message("Tell me a story") |>
  chat(claude(), .stream=TRUE)

## End(Not run)
```

 chatgpt

Alias for the OpenAI Provider Function

Description

The chatgpt function is an alias for the openai() provider function. It provides a convenient way to interact with the OpenAI API for tasks such as sending chat messages, generating embeddings, and handling batch operations using tidyllm verbs like chat(), embed(), and send_batch().

Usage

```
chatgpt(..., .called_from = NULL)
```

Arguments

...	Parameters to be passed to the appropriate OpenAI-specific function, such as model configuration, input text, or other API-specific options.
.called_from	An internal argument that specifies the context (e.g., chat, embed, send_batch) in which the function is being invoked. This is automatically managed and should not be modified by the user.

Value

The result of the requested action, depending on the specific function invoked (e.g., an updated LLMMessage object for chat(), or a matrix for embed()).

check_batch	<i>Check Batch Processing Status</i>
-------------	--------------------------------------

Description

This function retrieves the processing status and other details of a specified batchid or a list of LLMMessage objects with batch attribute. It routes the input to the appropriate provider-specific batch API function.

Usage

```
check_batch(
  .llms,
  .provider = getOption("tidyllm_cbatch_default"),
  .dry_run = NULL,
  .max_tries = NULL,
  .timeout = NULL
)
```

Arguments

.llms	A list of LLMMessage objects or a character vector with a batch ID.
.provider	A function or function call specifying the language model provider and any additional parameters. This should be a call to a provider function like openai(), claude(), etc. You can also set a default provider function via the tidyllm_cbatch_default option.
.dry_run	Logical; if TRUE, returns the prepared request object without executing it
.max_tries	Maximum retries to perform the request
.timeout	Integer specifying the request timeout in seconds

Value

A tibble with information about the status of batch processing.

check_claude_batch	<i>Check Batch Processing Status for Claude API</i>
--------------------	---

Description

This function retrieves the processing status and other details of a specified Claude batch ID from the Claude API.

Usage

```

check_claude_batch(
    .llms = NULL,
    .batch_id = NULL,
    .api_url = "https://api.anthropic.com/",
    .dry_run = FALSE,
    .max_tries = 3,
    .timeout = 60
)

```

Arguments

.llms	A list of LLMMMessage objects
.batch_id	A manually set batchid
.api_url	Character; base URL of the Claude API (default: "https://api.anthropic.com/").
.dry_run	Logical; if TRUE, returns the prepared request object without executing it (default: FALSE).
.max_tries	Maximum retries to perform request
.timeout	Integer specifying the request timeout in seconds (default: 60).

Value

A tibble with information about the status of batch processing

check_mistral_batch	<i>Check Batch Processing Status for Mistral Batch API</i>
---------------------	--

Description

This function retrieves the processing status and other details of a specified Mistral batch ID from the Mistral Batch API.

Usage

```

check_mistral_batch(
    .llms = NULL,
    .batch_id = NULL,
    .dry_run = FALSE,
    .max_tries = 3,
    .timeout = 60
)

```

Arguments

<code>.llms</code>	A list of LLMMessage objects.
<code>.batch_id</code>	A manually set batch ID.
<code>.dry_run</code>	Logical; if TRUE, returns the prepared request object without executing it (default: FALSE).
<code>.max_tries</code>	Maximum retries to perform the request (default: 3).
<code>.timeout</code>	Integer specifying the request timeout in seconds (default: 60).

Value

A tibble with information about the status of batch processing.

<code>check_openai_batch</code>	<i>Check Batch Processing Status for OpenAI Batch API</i>
---------------------------------	---

Description

This function retrieves the processing status and other details of a specified OpenAI batch ID from the OpenAI Batch API.

Usage

```
check_openai_batch(
  .llms = NULL,
  .batch_id = NULL,
  .dry_run = FALSE,
  .max_tries = 3,
  .timeout = 60
)
```

Arguments

<code>.llms</code>	A list of LLMMessage objects.
<code>.batch_id</code>	A manually set batch ID.
<code>.dry_run</code>	Logical; if TRUE, returns the prepared request object without executing it (default: FALSE).
<code>.max_tries</code>	Maximum retries to perform the request (default: 3).
<code>.timeout</code>	Integer specifying the request timeout in seconds (default: 60).

Value

A tibble with information about the status of batch processing.

claude	<i>Provider Function for Claude models on the Anthropic API</i>
--------	---

Description

The `claude()` function acts as an interface for interacting with the Anthropic API through main `tidyLLM` verbs such as `chat()`, `embed()`, and `send_batch()`. It dynamically routes requests to Claude-specific functions like `claude_chat()` and `send_claude_batch()` based on the context of the call.

Usage

```
claude(..., .called_from = NULL)
```

Arguments

<code>...</code>	Parameters to be passed to the appropriate OpenAI-specific function, such as model configuration, input text, or API-specific options.
<code>.called_from</code>	An internal argument that specifies which action (e.g., <code>chat</code> , <code>send_batch</code>) the function is being invoked from. This argument is automatically managed and should not be modified by the user.

Value

The result of the requested action, depending on the specific function invoked (e.g., an updated `LLMMessage` object for `chat()`, or a matrix for `embed()`).

<code>claude_chat</code>	<i>Interact with Claude AI models via the Anthropic API</i>
--------------------------	---

Description

Interact with Claude AI models via the Anthropic API

Usage

```
claude_chat(  
  .llm,  
  .model = "claude-3-5-sonnet-20241022",  
  .max_tokens = 1024,  
  .temperature = NULL,  
  .top_k = NULL,  
  .top_p = NULL,  
  .metadata = NULL,  
  .stop_sequences = NULL,  
)
```

```

    .tools = NULL,
    .api_url = "https://api.anthropic.com/",
    .verbose = FALSE,
    .max_tries = 3,
    .timeout = 60,
    .stream = FALSE,
    .dry_run = FALSE
  )

```

Arguments

<code>.llm</code>	An LLMMessage object containing the conversation history and system prompt.
<code>.model</code>	Character string specifying the Claude model version (default: "claude-3-5-sonnet-20241022").
<code>.max_tokens</code>	Integer specifying the maximum number of tokens in the response (default: 1024).
<code>.temperature</code>	Numeric between 0 and 1 controlling response randomness.
<code>.top_k</code>	Integer controlling diversity by limiting the top K tokens.
<code>.top_p</code>	Numeric between 0 and 1 for nucleus sampling.
<code>.metadata</code>	List of additional metadata to include with the request.
<code>.stop_sequences</code>	Character vector of sequences that will halt response generation.
<code>.tools</code>	List of additional tools or functions the model can use.
<code>.api_url</code>	Base URL for the Anthropic API (default: "https://api.anthropic.com/").
<code>.verbose</code>	Logical; if TRUE, displays additional information about the API call (default: FALSE).
<code>.max_tries</code>	Maximum retries to perform request
<code>.timeout</code>	Integer specifying the request timeout in seconds (default: 60).
<code>.stream</code>	Logical; if TRUE, streams the response piece by piece (default: FALSE).
<code>.dry_run</code>	Logical; if TRUE, returns the prepared request object without executing it (default: FALSE).

Value

A new LLMMessage object containing the original messages plus Claude's response.

Examples

```

## Not run:
# Basic usage
msg <- llm_message("What is R programming?")
result <- claude_chat(msg)

# With custom parameters
result2 <- claude_chat(msg,
  .temperature = 0.7,

```

```
.max_tokens = 1000)  
  
## End(Not run)
```

df_llm_message *Convert a Data Frame to an LLMMessage Object*

Description

This function converts a data frame into an LLMMessage object representing a conversation history. The data frame must have specific columns (role and content), with each row representing a message.

Usage

```
df_llm_message(.df)
```

Arguments

.df A data frame with at least two rows and columns role and content. The role column should contain "user", "assistant", or "system". The content column should contain the corresponding message text.

Value

An LLMMessage object representing the structured conversation.

See Also

[llm_message\(\)](#)

Other Message Creation Utilities: [llm_message\(\)](#)

embed *Generate text embeddings*

Description

The embed() function allows you to embed a text via a specified provider. It routes the input to the appropriate provider-specific embedding function.

Usage

```
embed(
  .input,
  .provider = getOption("tidyllm_embed_default"),
  .model = NULL,
  .truncate = NULL,
  .timeout = NULL,
  .dry_run = NULL,
  .max_tries = NULL
)
```

Arguments

<code>.input</code>	A character vector of texts to embed or an LLMMessage object
<code>.provider</code>	A function or function call specifying the language model provider and any additional parameters. This should be a call to a provider function like <code>openai()</code> , <code>ollama()</code> , etc. You can also set a default provider function via the <code>tidyllm_embed_default</code> option.
<code>.model</code>	The embedding model to use
<code>.truncate</code>	Whether to truncate inputs to fit the model's context length
<code>.timeout</code>	Timeout for the API request in seconds
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object.
<code>.max_tries</code>	Maximum retry attempts for requests

Value

A tibble with two columns: `input` and `embeddings`. The `input` column contains the texts sent to `embed`, and the `embeddings` column is a list column where each row contains an embedding vector of the sent input.

Examples

```
## Not run:
c("What is the meaning of life, the universe and everything?",
  "How much wood would a woodchuck chuck?",
  "How does the brain work?") |>
embed(gemini)

## End(Not run)
```

`fetch_batch`*Fetch Results from a Batch API*

Description

This function retrieves the results of a completed batch and updates the provided list of `LLMMessage` objects with the responses. It aligns each response with the original request using the `custom_ids` generated in `send_batch()`.

Usage

```
fetch_batch(  
    .llms,  
    .provider = getOption("tidyllm_fbatch_default"),  
    .dry_run = NULL,  
    .max_tries = NULL,  
    .timeout = NULL  
)
```

Arguments

<code>.llms</code>	A list of <code>LLMMessage</code> objects containing conversation histories.
<code>.provider</code>	A function or function call specifying the language model provider and any additional parameters. This should be a call to a provider function like <code>openai()</code> , <code>claude()</code> , etc. You can also set a default provider function via the <code>tidyllm_fbatch_default</code> option.
<code>.dry_run</code>	Logical; if <code>TRUE</code> , returns the constructed request without executing it
<code>.max_tries</code>	Integer; maximum number of retries if the request fails
<code>.timeout</code>	Integer; request timeout in seconds

Details

The function routes the input to the appropriate provider-specific batch API function.

Value

A list of updated `LLMMessage` objects, each with the assistant's response added if successful.

fetch_claude_batch	<i>Fetch Results for a Claude Batch</i>
--------------------	---

Description

This function retrieves the results of a completed Claude batch and updates the provided list of LLMMessage objects with the responses. It aligns each response with the original request using the custom_ids generated in send_claude_batch().

Usage

```
fetch_claude_batch(
    .llms,
    .batch_id = NULL,
    .api_url = "https://api.anthropic.com/",
    .dry_run = FALSE,
    .max_tries = 3,
    .timeout = 60
)
```

Arguments

.llms	A list of LLMMessage objects that were part of the batch. The list should have names (custom IDs) set by send_claude_batch() to ensure correct alignment.
.batch_id	Character; the unique identifier for the batch. By default this is NULL and the function will attempt to use the batch_id attribute from .llms.
.api_url	Character; the base URL for the Claude API (default: "https://api.anthropic.com/").
.dry_run	Logical; if TRUE, returns the constructed request without executing it (default: FALSE).
.max_tries	Integer; maximum number of retries if the request fails (default: 3).
.timeout	Integer; request timeout in seconds (default: 60).

Value

A list of updated LLMMessage objects, each with the assistant's response added if successful.

fetch_mistral_batch *Fetch Results for an Mistral Batch*

Description

This function retrieves the results of a completed Mistral batch and updates the provided list of LLMMessage objects with the responses. It aligns each response with the original request using the custom_ids generated in send_mistral_batch().

Usage

```
fetch_mistral_batch(  
    .llms,  
    .batch_id = NULL,  
    .dry_run = FALSE,  
    .max_tries = 3,  
    .timeout = 60  
)
```

Arguments

.llms	A list of LLMMessage objects that were part of the batch.
.batch_id	Character; the unique identifier for the batch. By default this is NULL and the function will attempt to use the batch_id attribute from .llms.
.dry_run	Logical; if TRUE, returns the constructed request without executing it (default: FALSE).
.max_tries	Integer; maximum number of retries if the request fails (default: 3).
.timeout	Integer; request timeout in seconds (default: 60).

Value

A list of updated LLMMessage objects, each with the assistant's response added if successful.

fetch_openai_batch *Fetch Results for an OpenAI Batch*

Description

This function retrieves the results of a completed OpenAI batch and updates the provided list of LLMMessage objects with the responses. It aligns each response with the original request using the custom_ids generated in send_openai_batch().

Usage

```

fetch_openai_batch(
    .llms,
    .batch_id = NULL,
    .dry_run = FALSE,
    .max_tries = 3,
    .timeout = 60
)

```

Arguments

<code>.llms</code>	A list of LLMMessage objects that were part of the batch.
<code>.batch_id</code>	Character; the unique identifier for the batch. By default this is NULL and the function will attempt to use the <code>batch_id</code> attribute from <code>.llms</code> .
<code>.dry_run</code>	Logical; if TRUE, returns the constructed request without executing it (default: FALSE).
<code>.max_tries</code>	Integer; maximum number of retries if the request fails (default: 3).
<code>.timeout</code>	Integer; request timeout in seconds (default: 60).

Value

A list of updated LLMMessage objects, each with the assistant's response added if successful.

gemini

Google Gemini Provider Function

Description

The `gemini()` function acts as a provider interface for interacting with the Google Gemini API through `tidyllm`'s main verbs such as `chat()` and `embed()`. It dynamically routes requests to Gemini-specific functions like `gemini_chat()` and `gemini_embedding()` based on the context of the call.

Usage

```
gemini(..., .called_from = NULL)
```

Arguments

<code>...</code>	Parameters to be passed to the appropriate Gemini-specific function, such as model configuration, input text, or API-specific options.
<code>.called_from</code>	An internal argument specifying which action (e.g., <code>chat</code> , <code>embed</code>) the function is invoked from. This argument is automatically managed by the <code>tidyllm</code> verbs and should not be modified by the user.

Details

Some functions, such as `gemini_upload_file()` and `gemini_delete_file()`, are specific to Gemini and do not have general verb counterparts.

Value

The result of the requested action, depending on the specific function invoked (e.g., an updated `LLMMessage` object for `chat()`).

<code>gemini_chat</code>	<i>Send LLMMessage to Gemini API</i>
--------------------------	--------------------------------------

Description

Send LLMMessage to Gemini API

Usage

```
gemini_chat(
    .llm,
    .model = "gemini-1.5-flash",
    .fileid = NULL,
    .temperature = NULL,
    .max_output_tokens = NULL,
    .top_p = NULL,
    .top_k = NULL,
    .grounding_threshold = NULL,
    .presence_penalty = NULL,
    .frequency_penalty = NULL,
    .stop_sequences = NULL,
    .safety_settings = NULL,
    .json_schema = NULL,
    .timeout = 120,
    .dry_run = FALSE,
    .max_tries = 3,
    .verbose = FALSE,
    .stream = FALSE
)
```

Arguments

<code>.llm</code>	An existing <code>LLMMessage</code> object or an initial text prompt.
<code>.model</code>	The model identifier (default: "gemini-1.5-flash").
<code>.fileid</code>	Optional vector of file IDs uploaded via <code>gemini_upload_file()</code> (default: <code>NULL</code>).
<code>.temperature</code>	Controls randomness in generation (default: <code>NULL</code> , range: 0.0-2.0).

<code>.max_output_tokens</code>	Maximum tokens in the response (default: NULL).
<code>.top_p</code>	Controls nucleus sampling (default: NULL, range: 0.0-1.0).
<code>.top_k</code>	Controls diversity in token selection (default: NULL, range: 0 or more).
<code>.grounding_threshold</code>	A grounding threshold between 0 and 1. With lower grounding thresholds Gemini will use Google to search for relevant information before answering. (default: NULL).
<code>.presence_penalty</code>	Penalizes new tokens (default: NULL, range: -2.0 to 2.0).
<code>.frequency_penalty</code>	Penalizes frequent tokens (default: NULL, range: -2.0 to 2.0).
<code>.stop_sequences</code>	Optional character sequences to stop generation (default: NULL, up to 5).
<code>.safety_settings</code>	A list of safety settings (default: NULL).
<code>.json_schema</code>	A JSON schema object as R list to enforce the output structure
<code>.timeout</code>	When should our connection time out (default: 120 seconds).
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object.
<code>.max_tries</code>	Maximum retries to perform request (default: 3).
<code>.verbose</code>	Should additional information be shown after the API call.
<code>.stream</code>	Should the response be streamed (default: FALSE).

Value

Returns an updated LLMMessage object.

<code>gemini_delete_file</code>	<i>Delete a File from Gemini API</i>
---------------------------------	--------------------------------------

Description

Deletes a specific file from the Gemini API using its file ID.

Usage

```
gemini_delete_file(.file_name)
```

Arguments

<code>.file_name</code>	The file ID (e.g., "files/abc-123") to delete.
-------------------------	--

Value

Invisibly returns NULL. Prints a confirmation message upon successful deletion.

gemini_embedding *Generate Embeddings Using the Google Gemini API*

Description

Generate Embeddings Using the Google Gemini API

Usage

```
gemini_embedding(  
    .input,  
    .model = "text-embedding-004",  
    .truncate = TRUE,  
    .timeout = 120,  
    .dry_run = FALSE,  
    .max_tries = 3  
)
```

Arguments

.input	A character vector of texts to embed or an LLMMessage object
.model	The embedding model identifier (default: "text-embedding-3-small").
.truncate	Whether to truncate inputs to fit the model's context length (default: TRUE).
.timeout	Timeout for the API request in seconds (default: 120).
.dry_run	If TRUE, perform a dry run and return the request object.
.max_tries	Maximum retry attempts for requests (default: 3).

Value

A matrix where each column corresponds to the embedding of a message in the message history.

gemini_file_metadata *Retrieve Metadata for a File from Gemini API*

Description

Retrieves metadata for a specific file uploaded to the Gemini API.

Usage

```
gemini_file_metadata(.file_name)
```

Arguments

.file_name	The file ID (e.g., "files/abc-123") to retrieve metadata for.
------------	---

Value

A tibble containing metadata fields such as name, display name, MIME type, size, and URI.

gemini_list_files	<i>List Files in Gemini API</i>
-------------------	---------------------------------

Description

Lists metadata for files uploaded to the Gemini API, supporting pagination.

Usage

```
gemini_list_files(.page_size = 10, .page_token = NULL)
```

Arguments

.page_size	The maximum number of files to return per page (default: 10, maximum: 100).
.page_token	A token for fetching the next page of results (default: NULL).

Value

A tibble containing metadata for each file, including fields such as name, display name, MIME type, and URI.

gemini_upload_file	<i>Upload a File to Gemini API</i>
--------------------	------------------------------------

Description

Uploads a file to the Gemini API and returns its metadata as a tibble.

Usage

```
gemini_upload_file(.file_path)
```

Arguments

.file_path	The local file path of the file to upload.
------------	--

Value

A tibble containing metadata about the uploaded file, including its name, URI, and MIME type.

get_metadata	<i>Retrieve Metadata from Assistant Replies</i>
--------------	---

Description

Retrieves metadata from assistant replies within an LLMMessage object. It returns the metadata as a tibble.

Usage

```
get_metadata(.llm, .index = NULL)
```

```
last_metadata(.llm)
```

Arguments

.llm	An LLMMessage object containing the message history.
.index	A positive integer specifying which assistant reply's metadata to extract. If NULL (default), metadata for all replies is returned.

Details

Metadata columns may include:

- `model`: The model used for generating the reply.
- `timestamp`: The time when the reply was generated.
- `prompt_tokens`: The number of tokens in the input prompt.
- `completion_tokens`: The number of tokens in the assistant's reply.
- `total_tokens`: The total number of tokens (prompt + completion).
- `api_specific`: A list column with API-specific metadata.

For convenience, [last_metadata\(\)](#) is provided to retrieve the metadata for the last message.

Value

A tibble containing metadata for the specified assistant reply or all replies.

See Also

[last_metadata\(\)](#)

get_reply	<i>Retrieve Assistant Reply as Text</i>
-----------	---

Description

Extracts the plain text content of the assistant's reply from an LLMMessage object. Use [get_reply_data\(\)](#) for structured replies in JSON format.

Usage

```
get_reply(.llm, .index = NULL)
```

```
last_reply(.llm)
```

Arguments

.llm	An LLMMessage object containing the message history.
.index	A positive integer indicating the index of the assistant reply to retrieve. Defaults to NULL, which retrieves the last reply.

Details

This function is the core utility for retrieving assistant replies by index. For convenience, [last_reply\(\)](#) is provided as a wrapper to retrieve the latest assistant reply.

Value

Returns a character string containing the assistant's reply, or NA_character_ if no reply exists.

See Also

[get_reply_data\(\)](#), [last_reply\(\)](#)

get_reply_data	<i>Retrieve Assistant Reply as Structured Data</i>
----------------	--

Description

Parses the assistant's reply as JSON and returns the corresponding structured data. If the reply is not marked as JSON, attempts to extract and parse JSON content from the text.

Usage

```
get_reply_data(.llm, .index = NULL)
```

```
last_reply_data(.llm)
```

Arguments

- .llm An LLMMessage object containing the message history.
- .index A positive integer indicating the index of the assistant reply to retrieve. Defaults to NULL, which retrieves the last reply.

Details

For convenience, [last_reply_data\(\)](#) is provided as a wrapper to retrieve the latest assistant reply's data.

Value

Returns the parsed data from the assistant's reply, or NULL if parsing fails.

See Also

[get_reply\(\)](#), [last_reply_data\(\)](#)

<code>get_user_message</code>	<i>Retrieve a User Message by Index</i>
-------------------------------	---

Description

Extracts the content of a user's message from an LLMMessage object at a specific index.

Usage

```
get_user_message(.llm, .index = NULL)
```

```
last_user_message(.llm)
```

Arguments

- .llm An LLMMessage object.
- .index A positive integer indicating which user message to retrieve. Defaults to NULL, which retrieves the last message.

Details

For convenience, [last_user_message\(\)](#) is provided as a wrapper to retrieve the latest user message without specifying an index.

Value

Returns the content of the user's message at the specified index. If no messages are found, returns `NA_character_`.

See Also

[last_user_message\(\)](#)

groq

Groq API Provider Function

Description

The `groq()` function acts as an interface for interacting with the Groq API through `tidyLLM`'s main verbs. Currently, Groq only supports `groq_chat()` for chat-based interactions and `groq_transcribe()` for transcription tasks.

Usage

```
groq(..., .called_from = NULL)
```

Arguments

<code>...</code>	Parameters to be passed to the Groq-specific function, such as model configuration, input text, or API-specific options.
<code>.called_from</code>	An internal argument that specifies which action (e.g., <code>chat</code>) the function is being invoked from. This argument is automatically managed and should not be modified by the user.

Details

Since `groq_transcribe()` is unique to Groq and does not have a general verb counterpart, `groq()` currently routes messages only to `groq_chat()` when used with verbs like `chat()`.

Value

The result of the requested action, depending on the specific function invoked (currently, only an updated `LLMMessage` object for `groq_chat()`).

groq_chat

Send LLM Messages to the Groq Chat API

Description

This function sends a message history to the Groq Chat API and returns the assistant's reply.

Usage

```

groq_chat(
  .llm,
  .model = "llama-3.2-11b-vision-preview",
  .max_tokens = 1024,
  .temperature = NULL,
  .top_p = NULL,
  .frequency_penalty = NULL,
  .presence_penalty = NULL,
  .stop = NULL,
  .seed = NULL,
  .api_url = "https://api.groq.com/",
  .json = FALSE,
  .timeout = 60,
  .verbose = FALSE,
  .stream = FALSE,
  .dry_run = FALSE,
  .max_tries = 3
)

```

Arguments

<code>.llm</code>	An LLMMessage object containing the conversation history.
<code>.model</code>	The identifier of the model to use (default: "llama-3.2-11b-vision-preview").
<code>.max_tokens</code>	The maximum number of tokens that can be generated in the response (default: 1024).
<code>.temperature</code>	Controls the randomness in the model's response. Values between 0 and 2 are allowed, where higher values increase randomness (optional).
<code>.top_p</code>	Nucleus sampling parameter that controls the proportion of probability mass considered. Values between 0 and 1 are allowed (optional).
<code>.frequency_penalty</code>	Number between -2.0 and 2.0. Positive values penalize repeated tokens, reducing likelihood of repetition (optional).
<code>.presence_penalty</code>	Number between -2.0 and 2.0. Positive values encourage new topics by penalizing tokens that have appeared so far (optional).
<code>.stop</code>	One or more sequences where the API will stop generating further tokens. Can be a string or a list of strings (optional).
<code>.seed</code>	An integer for deterministic sampling. If specified, attempts to return the same result for repeated requests with identical parameters (optional).
<code>.api_url</code>	Base URL for the Groq API (default: "https://api.groq.com/").
<code>.json</code>	Whether the response should be structured as JSON (default: FALSE).
<code>.timeout</code>	Request timeout in seconds (default: 60).
<code>.verbose</code>	If TRUE, displays additional information after the API call, including rate limit details (default: FALSE).

<code>.stream</code>	Logical; if TRUE, streams the response piece by piece (default: FALSE).
<code>.dry_run</code>	If TRUE, performs a dry run and returns the constructed request object without executing it (default: FALSE).
<code>.max_tries</code>	Maximum retries to perform request

Value

A new LLMMessage object containing the original messages plus the assistant's response.

Examples

```
## Not run:
# Basic usage
msg <- llm_message("What is Groq?")
result <- groq_chat(msg)

# With custom parameters
result2 <- groq_chat(msg,
  .model = "llama-3.2-vision",
  .temperature = 0.5,
  .max_tokens = 512)

## End(Not run)
```

groq_transcribe

Transcribe an Audio File Using Groq transcription API

Description

This function reads an audio file and sends it to the Groq transcription API for transcription.

Usage

```
groq_transcribe(
  .audio_file,
  .model = "whisper-large-v3",
  .language = NULL,
  .prompt = NULL,
  .temperature = 0,
  .api_url = "https://api.groq.com/openai/v1/audio/transcriptions",
  .dry_run = FALSE,
  .verbose = FALSE,
  .max_tries = 3
)
```

Arguments

<code>.audio_file</code>	The path to the audio file (required). Supported formats include flac, mp3, mp4, mpeg, mpga, m4a, ogg, wav, or webm.
<code>.model</code>	The model to use for transcription (default: "whisper-large-v3").
<code>.language</code>	The language of the input audio, in ISO-639-1 format (optional).
<code>.prompt</code>	A prompt to guide the transcription style. It should match the audio language (optional).
<code>.temperature</code>	Sampling temperature, between 0 and 1, with higher values producing more randomness (default: 0).
<code>.api_url</code>	Base URL for the API (default: "https://api.groq.com/openai/v1/audio/transcriptions").
<code>.dry_run</code>	Logical; if TRUE, performs a dry run and returns the request object without making the API call (default: FALSE).
<code>.verbose</code>	Logical; if TRUE, rate limiting info is displayed after the API request (default: FALSE).
<code>.max_tries</code>	Maximum retries to perform request

Value

A character vector containing the transcription.

Examples

```
## Not run:
# Basic usage
groq_transcribe(.audio_file = "example.mp3")

## End(Not run)
```

list_batches

List all Batch Requests on a Batch API

Description

List all Batch Requests on a Batch API

Usage

```
list_batches(.provider = getOption("tidyllm_lbatch_default"))
```

Arguments

<code>.provider</code>	A function or function call specifying the language model provider and any additional parameters. This should be a call to a provider function like <code>openai()</code> , <code>claude()</code> , etc. You can also set a default provider function via the <code>tidyllm_lbatch_default</code> option.
------------------------	---

Value

A tibble with information about the status of batch processing.

list_claude_batches *List Claude Batch Requests*

Description

Retrieves batch request details from the Claude API.

Usage

```
list_claude_batches(
  .api_url = "https://api.anthropic.com/",
  .limit = 20,
  .max_tries = 3,
  .timeout = 60
)
```

Arguments

.api_url	Base URL for the Claude API (default: "https://api.anthropic.com/").
.limit	Maximum number of batches to retrieve (default: 20).
.max_tries	Maximum retry attempts for requests (default: 3).
.timeout	Request timeout in seconds (default: 60).

Value

A tibble with batch details: batch ID, status, creation time, expiration time, and request counts (succeeded, errored, expired, canceled).

list_mistral_batches *List Mistral Batch Requests*

Description

Retrieves batch request details from the OpenAI Batch API.

Usage

```
list_mistral_batches(
  .limit = 100,
  .max_tries = 3,
  .timeout = 60,
  .status = NULL,
  .created_after = NULL
)
```


Arguments

.limit	Maximum number of batches to retrieve (default: 20).
.max_tries	Maximum retry attempts for requests (default: 3).
.timeout	Request timeout in seconds (default: 60).
.status	Filter by status. (default: NULL)
.created_after	created after a string specifying a date-time (default: NULL)

Value

A tibble with batch details for all batches fitting the request

list_openai_batches *List OpenAI Batch Requests*

Description

Retrieves batch request details from the OpenAI Batch API.

Usage

```
list_openai_batches(.limit = 20, .max_tries = 3, .timeout = 60)
```

Arguments

.limit	Maximum number of batches to retrieve (default: 20).
.max_tries	Maximum retry attempts for requests (default: 3).
.timeout	Request timeout in seconds (default: 60).

Value

A tibble with batch details: batch ID, status, creation time, expiration time, and request counts (total, completed, failed).

LLMMessage

Large Language Model Message Class

Description

LLMMessage is an S7 class for managing a conversation history intended for use with large language models (LLMs). Please use `llm_message()` to create or modify LLMMessage objects.

Usage

```
LLMMessage(message_history = list(), system_prompt = character(0))
```

Arguments

<code>message_history</code>	A list containing messages. Each message is a named list with keys like <code>role</code> , <code>content</code> , <code>media</code> , etc.
<code>system_prompt</code>	A character string representing the default system prompt used for the conversation.

Details

The LLMMessage class includes the following features:

- Stores message history in a structured format.
- Supports attaching media and metadata to messages.
- Provides generics like `add_message()`, `has_image()`, and `remove_message()` for interaction.
- Enables API-specific formatting through the `to_api_format()` generic.
- `message_history`: A list containing messages. Each message is a named list with keys like `role`, `content`, `media`, etc.
- `system_prompt`: A character string representing the default system prompt used for the conversation.

llm_message

Create or Update Large Language Model Message Object

Description

This function creates a new LLMMessage object or updates an existing one. It supports adding text prompts and various media types, such as images, PDFs, text files, or plots.

Usage

```
llm_message(  
  .llm = NULL,  
  .prompt = NULL,  
  .role = "user",  
  .system_prompt = "You are a helpful assistant",  
  .imagefile = NULL,  
  .pdf = NULL,  
  .textfile = NULL,  
  .capture_plot = FALSE,  
  .f = NULL  
)
```

Arguments

<code>.llm</code>	An existing LLMMessage object or an initial text prompt.
<code>.prompt</code>	Text prompt to add to the message history.
<code>.role</code>	The role of the message sender, typically "user" or "assistant".
<code>.system_prompt</code>	Default system prompt if a new LLMMessage needs to be created.
<code>.imagefile</code>	Path to an image file to be attached (optional).
<code>.pdf</code>	Path to a PDF file to be attached (optional). Can be a character vector of length one (file path), or a list with <code>filename</code> , <code>start_page</code> , and <code>end_page</code> .
<code>.textfile</code>	Path to a text file to be read and attached (optional).
<code>.capture_plot</code>	Boolean to indicate whether a plot should be captured and attached as an image (optional).
<code>.f</code>	An R function or an object coercible to a function via <code>rlang::as_function</code> , whose output should be captured and attached (optional).

Value

Returns an updated or new LLMMessage object.

See Also

[df_llm_message\(\)](#)

Other Message Creation Utilities: [df_llm_message\(\)](#)

mistral

Mistral Provider Function

Description

The `mistral()` function acts as an interface for interacting with the Mistral API through main `tidyllm` verbs such as `chat()` and `embed()`. It dynamically routes requests to Mistral-specific functions like `mistral_chat()` and `mistral_embedding()` based on the context of the call.

Usage

```
mistral(..., .called_from = NULL)
```

Arguments

<code>...</code>	Parameters to be passed to the appropriate Mistral-specific function, such as model configuration, input text, or API-specific options.
<code>.called_from</code>	An internal argument that specifies which action (e.g., <code>chat</code> , <code>embed</code> , <code>send_batch</code>) the function is being invoked from. This argument is automatically managed and should not be modified by the user.

Value

The result of the requested action, depending on the specific function invoked (e.g., an updated `LLMMessage` object for `chat()`, or a matrix for `embed()`).

mistral_chat

Send LLMMessage to Mistral API

Description

Send `LLMMessage` to Mistral API

Usage

```
mistral_chat(
  .llm,
  .model = "mistral-large-latest",
  .stream = FALSE,
  .seed = NULL,
  .json = FALSE,
  .temperature = 0.7,
  .top_p = 1,
  .stop = NULL,
  .safe_prompt = FALSE,
```

```

        .timeout = 120,
        .max_tries = 3,
        .max_tokens = 1024,
        .min_tokens = NULL,
        .dry_run = FALSE,
        .verbose = FALSE
    )

```

Arguments

<code>.llm</code>	An LLMMessage object.
<code>.model</code>	The model identifier to use (default: "mistral-large-latest").
<code>.stream</code>	Whether to stream back partial progress to the console. (default: FALSE).
<code>.seed</code>	The seed to use for random sampling. If set, different calls will generate deterministic results (optional).
<code>.json</code>	Whether the output should be in JSON mode(default: FALSE).
<code>.temperature</code>	Sampling temperature to use, between 0.0 and 1.5. Higher values make the output more random, while lower values make it more focused and deterministic (default: 0.7).
<code>.top_p</code>	Nucleus sampling parameter, between 0.0 and 1.0. The model considers tokens with top_p probability mass (default: 1).
<code>.stop</code>	Stop generation if this token is detected, or if one of these tokens is detected when providing a list (optional).
<code>.safe_prompt</code>	Whether to inject a safety prompt before all conversations (default: FALSE).
<code>.timeout</code>	When should our connection time out in seconds (default: 120).
<code>.max_tries</code>	Maximum retries to perform request
<code>.max_tokens</code>	The maximum number of tokens to generate in the completion. Must be >= 0 (default: 1024).
<code>.min_tokens</code>	The minimum number of tokens to generate in the completion. Must be >= 0 (optional).
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object (default: FALSE).
<code>.verbose</code>	Should additional information be shown after the API call? (default: FALSE)

Value

Returns an updated LLMMessage object.

mistral_embedding	<i>Generate Embeddings Using Mistral API</i>
-------------------	--

Description

Generate Embeddings Using Mistral API

Usage

```
mistral_embedding(
  .input,
  .model = "mistral-embed",
  .timeout = 120,
  .max_tries = 3,
  .dry_run = FALSE
)
```

Arguments

.input	A character vector of texts to embed or an LLMMessage object
.model	The embedding model identifier (default: "mistral-embed").
.timeout	Timeout for the API request in seconds (default: 120).
.max_tries	Maximum retries to perform request
.dry_run	If TRUE, perform a dry run and return the request object.

Value

A matrix where each column corresponds to the embedding of a message in the message history.

ollama	<i>Ollama API Provider Function</i>
--------	-------------------------------------

Description

The `ollama()` function acts as an interface for interacting with local AI models via the Ollama API. It integrates seamlessly with the main `tidyllm` verbs such as `chat()` and `embed()`.

Usage

```
ollama(..., .called_from = NULL)
```

Arguments

- ... Parameters to be passed to the appropriate Ollama-specific function, such as model configuration, input text, or API-specific options.
- .called_from An internal argument specifying the verb (e.g., chat, embed) the function is invoked from. This argument is automatically managed by tidyllm and should not be set by the user.

Details

Some functionalities, like `ollama_download_model()` or `ollama_list_models()` are unique to the Ollama API and do not have a general verb counterpart. These functions can be only accessed directly.

Supported Verbs:

- `chat()`: Sends a message to an Ollama model and retrieves the model's response.
- `embed()`: Generates embeddings for input texts using an Ollama model.

Value

The result of the requested action:

- For `chat()`: An updated `LLMMessage` object containing the model's response.
- For `embed()`: A matrix where each column corresponds to an embedding.

ollama_chat

Interact with local AI models via the Ollama API

Description

Interact with local AI models via the Ollama API

Usage

```
ollama_chat(
  .llm,
  .model = "gemma2",
  .stream = FALSE,
  .seed = NULL,
  .json_schema = NULL,
  .temperature = NULL,
  .num_ctx = 2048,
  .num_predict = NULL,
  .top_k = NULL,
  .top_p = NULL,
  .min_p = NULL,
  .mirostat = NULL,
```

```

.mirostat_eta = NULL,
.mirostat_tau = NULL,
.repeat_last_n = NULL,
.repeat_penalty = NULL,
.tfs_z = NULL,
.stop = NULL,
.ollama_server = "http://localhost:11434",
.timeout = 120,
.keep_alive = NULL,
.dry_run = FALSE
)

```

Arguments

<code>.llm</code>	An LLMMessage object containing the conversation history and system prompt.
<code>.model</code>	Character string specifying the Ollama model to use (default: "gemma2")
<code>.stream</code>	Logical; whether to stream the response (default: FALSE)
<code>.seed</code>	Integer; seed for reproducible generation (default: NULL)
<code>.json_schema</code>	A JSON schema object as R list to enforce the output structure (default: NULL)
<code>.temperature</code>	Float between 0-2; controls randomness in responses (default: NULL)
<code>.num_ctx</code>	Integer; sets the context window size (default: 2048)
<code>.num_predict</code>	Integer; maximum number of tokens to predict (default: NULL)
<code>.top_k</code>	Integer; controls diversity by limiting top tokens considered (default: NULL)
<code>.top_p</code>	Float between 0-1; nucleus sampling threshold (default: NULL)
<code>.min_p</code>	Float between 0-1; minimum probability threshold (default: NULL)
<code>.mirostat</code>	Integer (0,1,2); enables Mirostat sampling algorithm (default: NULL)
<code>.mirostat_eta</code>	Float; Mirostat learning rate (default: NULL)
<code>.mirostat_tau</code>	Float; Mirostat target entropy (default: NULL)
<code>.repeat_last_n</code>	Integer; tokens to look back for repetition (default: NULL)
<code>.repeat_penalty</code>	Float; penalty for repeated tokens (default: NULL)
<code>.tfs_z</code>	Float; tail free sampling parameter (default: NULL)
<code>.stop</code>	Character; custom stop sequence(s) (default: NULL)
<code>.ollama_server</code>	String; Ollama API endpoint (default: "http://localhost:11434")
<code>.timeout</code>	Integer; API request timeout in seconds (default: 120)
<code>.keep_alive</code>	Character; How long should the ollama model be kept in memory after request (default: NULL - 5 Minutes)
<code>.dry_run</code>	Logical; if TRUE, returns request object without execution (default: FALSE)

Details

The function provides extensive control over the generation process through various parameters:

- Temperature (0-2): Higher values increase creativity, lower values make responses more focused
- Top-k/Top-p: Control diversity of generated text
- Mirostat: Advanced sampling algorithm for maintaining consistent complexity
- Repeat penalties: Prevent repetitive text
- Context window: Control how much previous conversation is considered

Value

A new LLMMessage object containing the original messages plus the model's response

Examples

```
## Not run:
llm_message("user", "Hello, how are you?")
response <- ollama_chat(llm, .model = "gemma2", .temperature = 0.7)

# With custom parameters
response <- ollama_chat(
  llm,
  .model = "llama2",
  .temperature = 0.8,
  .top_p = 0.9,
  .num_ctx = 4096
)

## End(Not run)
```

ollama_download_model *Download a model from the Ollama API*

Description

This function sends a request to the Ollama API to download a specified model from Ollama's large online library of models.

Usage

```
ollama_download_model(.model, .ollama_server = "http://localhost:11434")
```

Arguments

`.model` The name of the model to download.
`.ollama_server` The base URL of the Ollama API (default is "http://localhost:11434").

ollama_embedding *Generate Embeddings Using Ollama API*

Description

Generate Embeddings Using Ollama API

Usage

```
ollama_embedding(
  .input,
  .model = "all-minilm",
  .truncate = TRUE,
  .ollama_server = "http://localhost:11434",
  .timeout = 120,
  .dry_run = FALSE
)
```

Arguments

<code>.input</code>	Aa character vector of texts to embed or an LLMMessage object
<code>.model</code>	The embedding model identifier (default: "all-minilm").
<code>.truncate</code>	Whether to truncate inputs to fit the model's context length (default: TRUE).
<code>.ollama_server</code>	The URL of the Ollama server to be used (default: "http://localhost:11434").
<code>.timeout</code>	Timeout for the API request in seconds (default: 120).
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object.

Value

A matrix where each column corresponds to the embedding of a message in the message history.

ollama_list_models *Retrieve and return model information from the Ollama API*

Description

This function connects to the Ollama API and retrieves information about available models, returning it as a tibble.

Usage

```
ollama_list_models(.ollama_server = "http://localhost:11434")
```

Arguments

`.ollama_server` The URL of the ollama server to be used

Value

A tibble containing model information, or NULL if no models are found.

openai

OpenAI Provider Function

Description

The `openai()` function acts as an interface for interacting with the OpenAI API through main `tidyllm` verbs such as `chat()`, `embed()`, and `send_batch()`. It dynamically routes requests to OpenAI-specific functions like `openai_chat()` and `openai_embedding()` based on the context of the call.

Usage

```
openai(..., .called_from = NULL)
```

Arguments

`...` Parameters to be passed to the appropriate OpenAI-specific function, such as model configuration, input text, or API-specific options.

`.called_from` An internal argument that specifies which action (e.g., `chat`, `embed`, `send_batch`) the function is being invoked from. This argument is automatically managed and should not be modified by the user.

Value

The result of the requested action, depending on the specific function invoked (e.g., an updated `LLMMessage` object for `chat()`, or a matrix for `embed()`).

`openai_chat`

Send LLM Messages to the OpenAI Chat Completions API

Description

This function sends a message history to the OpenAI Chat Completions API and returns the assistant's reply.

Usage

```

openai_chat(
  .llm,
  .model = "gpt-4o",
  .max_completion_tokens = NULL,
  .frequency_penalty = NULL,
  .logit_bias = NULL,
  .presence_penalty = NULL,
  .seed = NULL,
  .stop = NULL,
  .stream = FALSE,
  .temperature = NULL,
  .top_p = NULL,
  .api_url = "https://api.openai.com/",
  .timeout = 60,
  .verbose = FALSE,
  .json_schema = NULL,
  .max_tries = 3,
  .dry_run = FALSE,
  .compatible = FALSE,
  .api_path = "/v1/chat/completions"
)

```

Arguments

<code>.llm</code>	An LLMMessage object containing the conversation history.
<code>.model</code>	The identifier of the model to use (default: "gpt-4o").
<code>.max_completion_tokens</code>	An upper bound for the number of tokens that can be generated for a completion, including visible output tokens and reasoning tokens.
<code>.frequency_penalty</code>	Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far.
<code>.logit_bias</code>	A named list modifying the likelihood of specified tokens appearing in the completion.
<code>.presence_penalty</code>	Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far.
<code>.seed</code>	If specified, the system will make a best effort to sample deterministically.
<code>.stop</code>	Up to 4 sequences where the API will stop generating further tokens.
<code>.stream</code>	If set to TRUE, the answer will be streamed to console as it comes (default: FALSE).
<code>.temperature</code>	What sampling temperature to use, between 0 and 2. Higher values make the output more random.
<code>.top_p</code>	An alternative to sampling with temperature, called nucleus sampling.

<code>.api_url</code>	Base URL for the API (default: "https://api.openai.com/").
<code>.timeout</code>	Request timeout in seconds (default: 60).
<code>.verbose</code>	Should additional information be shown after the API call (default: FALSE).
<code>.json_schema</code>	A JSON schema object as R list to enforce the output structure (If defined has precedence over JSON mode).
<code>.max_tries</code>	Maximum retries to perform request
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object (default: FALSE).
<code>.compatible</code>	If TRUE, skip API and rate-limit checks for OpenAI compatible APIs (default: FALSE).
<code>.api_path</code>	The path relative to the base <code>.api_url</code> for the API (default: "/v1/chat/completions").

Value

A new LLMMessage object containing the original messages plus the assistant's response.

openai_embedding *Generate Embeddings Using OpenAI API*

Description

Generate Embeddings Using OpenAI API

Usage

```
openai_embedding(
  .input,
  .model = "text-embedding-3-small",
  .truncate = TRUE,
  .timeout = 120,
  .dry_run = FALSE,
  .max_tries = 3,
  .verbose = FALSE
)
```

Arguments

<code>.input</code>	An existing LLMMessage object (or a character vector of texts to embed)
<code>.model</code>	The embedding model identifier (default: "text-embedding-3-small").
<code>.truncate</code>	Whether to truncate inputs to fit the model's context length (default: TRUE).
<code>.timeout</code>	Timeout for the API request in seconds (default: 120).
<code>.dry_run</code>	If TRUE, perform a dry run and return the request object.
<code>.max_tries</code>	Maximum retry attempts for requests (default: 3).
<code>.verbose</code>	Should information about current ratelimits be printed? (default: FALSE)

Value

A tibble with two columns: `input` and `embeddings`. The `input` column contains the texts sent to embed, and the `embeddings` column is a list column where each row contains an embedding vector of the sent input.

pdf_page_batch	<i>Batch Process PDF into LLM Messages</i>
----------------	--

Description

This function processes a PDF file page by page. For each page, it extracts the text and converts the page into an image. It creates a list of `LLMMessage` objects with the text and the image for multimodal processing. Users can specify a range of pages to process and provide a custom function to generate prompts for each page.

Usage

```
pdf_page_batch(
  .pdf,
  .general_prompt,
  .system_prompt = "You are a helpful assistant",
  .page_range = NULL,
  .prompt_fn = NULL
)
```

Arguments

<code>.pdf</code>	Path to the PDF file.
<code>.general_prompt</code>	A default prompt that is applied to each page if <code>.prompt_fn</code> is not provided.
<code>.system_prompt</code>	Optional system prompt to initialize the <code>LLMMessage</code> (default is "You are a helpful assistant").
<code>.page_range</code>	A vector of two integers specifying the start and end pages to process. If <code>NULL</code> , all pages are processed.
<code>.prompt_fn</code>	An optional custom function that generates a prompt for each page. The function takes the page text as input and returns a string. If <code>NULL</code> , <code>.general_prompt</code> is used for all pages.

Value

A list of `LLMMessage` objects, each containing the text and image for a page.

perplexity	<i>Perplexity Provider Function</i>
------------	-------------------------------------

Description

The `perplexity()` function acts as a provider interface for interacting with the Perplexity API through `tidyLLM`'s `chat()` verb. It dynamically routes requests to Perplexity-specific function. At the moment this is only `perplexity_chat()`

Usage

```
perplexity(..., .called_from = NULL)
```

Arguments

<code>...</code>	Parameters to be passed to the appropriate Perplexity-specific function, such as model configuration, input text, or API-specific options.
<code>.called_from</code>	An internal argument specifying which action (e.g., <code>chat</code> , <code>embed</code>) the function is invoked from. This argument is automatically managed by the <code>tidyLLM</code> verbs and should not be modified by the user.

Value

The result of the requested action, depending on the specific function invoked (e.g., an updated `LLMMessage` object for `chat()`).

<code>perplexity_chat</code>	<i>Send LLM Messages to the Perplexity Chat API</i>
------------------------------	---

Description

This function sends a message history to the Perplexity Chat API and returns the assistant's reply.

Usage

```
perplexity_chat(  
  .llm,  
  .model = "llama-3.1-sonar-small-128k-online",  
  .max_tokens = 1024,  
  .temperature = NULL,  
  .top_p = NULL,  
  .frequency_penalty = NULL,  
  .presence_penalty = NULL,  
  .stop = NULL,  
  .search_domain_filter = NULL,
```

```

    .return_images = FALSE,
    .search_recency_filter = NULL,
    .api_url = "https://api.perplexity.ai/",
    .json = FALSE,
    .timeout = 60,
    .verbose = FALSE,
    .stream = FALSE,
    .dry_run = FALSE,
    .max_tries = 3
  )

```

Arguments

<code>.llm</code>	An LLMMessage object containing the conversation history.
<code>.model</code>	The identifier of the model to use (default: "llama-3.2-11b-vision-preview").
<code>.max_tokens</code>	The maximum number of tokens that can be generated in the response (default: 1024).
<code>.temperature</code>	Controls the randomness in the model's response. Values between 0 (exclusive) and 2 (exclusive) are allowed, where higher values increase randomness (optional).
<code>.top_p</code>	Nucleus sampling parameter that controls the proportion of probability mass considered. Values between 0 (exclusive) and 1 (exclusive) are allowed (optional).
<code>.frequency_penalty</code>	Number greater than 0. Values > 1.0 penalize repeated tokens, reducing the likelihood of repetition (optional).
<code>.presence_penalty</code>	Number between -2.0 and 2.0. Positive values encourage new topics by penalizing tokens that have appeared so far (optional).
<code>.stop</code>	One or more sequences where the API will stop generating further tokens. Can be a string or a list of strings (optional).
<code>.search_domain_filter</code>	A vector of domains to limit or exclude from search results. For exclusion, prefix domains with a "-" (optional, currently in closed beta).
<code>.return_images</code>	Logical; if TRUE, enables returning images from the model's response (default: FALSE, currently in closed beta).
<code>.search_recency_filter</code>	Limits search results to a specific time interval (e.g., "month", "week", "day", or "hour"). Only applies to online models (optional).
<code>.api_url</code>	Base URL for the Perplexity API (default: "https://api.perplexity.ai/").
<code>.json</code>	Whether the response should be structured as JSON (default: FALSE).
<code>.timeout</code>	Request timeout in seconds (default: 60).
<code>.verbose</code>	If TRUE, displays additional information after the API call, including rate limit details (default: FALSE).
<code>.stream</code>	Logical; if TRUE, streams the response piece by piece (default: FALSE).

.dry_run	If TRUE, performs a dry run and returns the constructed request object without executing it (default: FALSE).
.max_tries	Maximum retries to perform the request (default: 3).

Value

A new LLMMessage object containing the original messages plus the assistant's response.

rate_limit_info	<i>Get the current rate limit information for all or a specific API</i>
-----------------	---

Description

This function retrieves the rate limit details for the specified API, or for all APIs stored in the .tidyllm_rate_limit_env if no API is specified.

Usage

```
rate_limit_info(.api_name = NULL)
```

Arguments

.api_name	(Optional) The name of the API whose rate limit info you want to get. If not provided, the rate limit info for all APIs in the environment will be returned.
-----------	--

Value

A tibble containing the rate limit information.

send_batch	<i>Send a batch of messages to a batch API</i>
------------	--

Description

The send_batch() function allows you to send a list of LLMMessage objects to an API. It routes the input to the appropriate provider-specific batch API function.

Usage

```

send_batch(
  .llms,
  .provider = getOption("tidyllm_sbatch_default"),
  .dry_run = NULL,
  .temperature = NULL,
  .timeout = NULL,
  .top_p = NULL,
  .max_tries = NULL,
  .model = NULL,
  .verbose = NULL,
  .json_schema = NULL,
  .seed = NULL,
  .stop = NULL,
  .frequency_penalty = NULL,
  .presence_penalty = NULL,
  .id_prefix = NULL
)

```

Arguments

<code>.llms</code>	A list of <code>LLMMessage</code> objects containing conversation histories.
<code>.provider</code>	A function or function call specifying the language model provider and any additional parameters. This should be a call to a provider function like <code>openai()</code> , <code>claude()</code> , etc. You can also set a default provider function via the <code>tidyllm_sbatch_default</code> option.
<code>.dry_run</code>	Logical; if <code>TRUE</code> , simulates the request without sending it to the provider. Useful for testing.
<code>.temperature</code>	Numeric; controls the randomness of the model's output (0 = deterministic).
<code>.timeout</code>	Numeric; the maximum time (in seconds) to wait for a response.
<code>.top_p</code>	Numeric; nucleus sampling parameter, which limits the sampling to the top cumulative probability <code>p</code> .
<code>.max_tries</code>	Integer; the maximum number of retries for failed requests.
<code>.model</code>	Character; the model identifier to use (e.g., "gpt-4").
<code>.verbose</code>	Logical; if <code>TRUE</code> , prints additional information about the request and response.
<code>.json_schema</code>	List; A JSON schema object as R list to enforce the output structure
<code>.seed</code>	Integer; sets a random seed for reproducibility.
<code>.stop</code>	Character vector; specifies sequences where the model should stop generating further tokens.
<code>.frequency_penalty</code>	Numeric; adjusts the likelihood of repeating tokens (positive values decrease repetition).
<code>.presence_penalty</code>	Numeric; adjusts the likelihood of introducing new tokens (positive values encourage novelty).

`.id_prefix` Character string to specify a prefix for generating custom IDs when names in `.llms` are missing

Value

An updated and named list of `.llms` with identifiers that align with batch responses, including a `batch_id` attribute.

send_claude_batch *Send a Batch of Messages to Claude API*

Description

This function creates and submits a batch of messages to the Claude API for asynchronous processing.

Usage

```
send_claude_batch(
    .llms,
    .model = "claude-3-5-sonnet-20241022",
    .max_tokens = 1024,
    .temperature = NULL,
    .top_k = NULL,
    .top_p = NULL,
    .stop_sequences = NULL,
    .api_url = "https://api.anthropic.com/",
    .verbose = FALSE,
    .dry_run = FALSE,
    .overwrite = FALSE,
    .max_tries = 3,
    .timeout = 60,
    .id_prefix = "tidyllm_claude_req_"
)
```

Arguments

`.llms` A list of `LLMMessage` objects containing conversation histories.

`.model` Character string specifying the Claude model version (default: "claude-3-5-sonnet-20241022").

`.max_tokens` Integer specifying the maximum tokens per response (default: 1024).

`.temperature` Numeric between 0 and 1 controlling response randomness.

`.top_k` Integer for diversity by limiting the top K tokens.

`.top_p` Numeric between 0 and 1 for nucleus sampling.

`.stop_sequences` Character vector of sequences that halt response generation.

<code>.api_url</code>	Base URL for the Claude API (default: "https://api.anthropic.com/").
<code>.verbose</code>	Logical; if TRUE, prints a message with the batch ID (default: FALSE).
<code>.dry_run</code>	Logical; if TRUE, returns the prepared request object without executing it (default: FALSE).
<code>.overwrite</code>	Logical; if TRUE, allows overwriting an existing batch ID associated with the request (default: FALSE).
<code>.max_tries</code>	Maximum number of retries to perform the request.
<code>.timeout</code>	Integer specifying the request timeout in seconds (default: 60).
<code>.id_prefix</code>	Character string to specify a prefix for generating custom IDs when names in <code>.llms</code> are missing. Defaults to "tidyllm_claude_req_".

Value

An updated and named list of `.llms` with identifiers that align with batch responses, including a `batch_id` attribute.

send_mistral_batch *Send a Batch of Requests to the Mistral API*

Description

Send a Batch of Requests to the Mistral API

Usage

```
send_mistral_batch(
  .llms,
  .model = "mistral-small-latest",
  .endpoint = "/v1/chat/completions",
  .metadata = NULL,
  .temperature = 0.7,
  .top_p = 1,
  .max_tokens = 1024,
  .min_tokens = NULL,
  .seed = NULL,
  .stop = NULL,
  .dry_run = FALSE,
  .overwrite = FALSE,
  .max_tries = 3,
  .timeout = 60,
  .id_prefix = "tidyllm_mistral_req_"
)
```

Arguments

.llms	A list of LLMMessage objects containing conversation histories.
.model	The Mistral model version (default: "mistral-small-latest").
.endpoint	The API endpoint (default: "/v1/chat/completions").
.metadata	Optional metadata for the batch.
.temperature	Sampling temperature to use, between 0.0 and 1.5. Higher values make the output more random (default: 0.7).
.top_p	Nucleus sampling parameter, between 0.0 and 1.0 (default: 1).
.max_tokens	The maximum number of tokens to generate in the completion (default: 1024).
.min_tokens	The minimum number of tokens to generate (optional).
.seed	Random seed for deterministic outputs (optional).
.stop	Stop generation at specific tokens or strings (optional).
.dry_run	Logical; if TRUE, returns the prepared request without executing it (default: FALSE).
.overwrite	Logical; if TRUE, allows overwriting existing custom IDs (default: FALSE).
.max_tries	Maximum retry attempts for requests (default: 3).
.timeout	Request timeout in seconds (default: 60).
.id_prefix	Prefix for generating custom IDs (default: "tidyllm_mistral_req_").

Value

The prepared_llms list with the batch_id attribute attached.

send_openai_batch	<i>Send a Batch of Messages to OpenAI Batch API</i>
-------------------	---

Description

This function creates and submits a batch of messages to the OpenAI Batch API for asynchronous processing.

Usage

```
send_openai_batch(
  .llms,
  .model = "gpt-4o",
  .max_completion_tokens = NULL,
  .frequency_penalty = NULL,
  .logit_bias = NULL,
  .presence_penalty = NULL,
  .seed = NULL,
  .stop = NULL,
```

```

    .temperature = NULL,
    .top_p = NULL,
    .dry_run = FALSE,
    .overwrite = FALSE,
    .json_schema = NULL,
    .max_tries = 3,
    .timeout = 60,
    .verbose = FALSE,
    .id_prefix = "tidyllm_openai_req_"
  )

```

Arguments

<code>.llms</code>	A list of LLMMessage objects containing conversation histories.
<code>.model</code>	Character string specifying the OpenAI model version (default: "gpt-4o").
<code>.max_completion_tokens</code>	Integer specifying the maximum tokens per response (default: NULL).
<code>.frequency_penalty</code>	Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far.
<code>.logit_bias</code>	A named list modifying the likelihood of specified tokens appearing in the completion.
<code>.presence_penalty</code>	Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far.
<code>.seed</code>	If specified, the system will make a best effort to sample deterministically.
<code>.stop</code>	Up to 4 sequences where the API will stop generating further tokens.
<code>.temperature</code>	What sampling temperature to use, between 0 and 2. Higher values make the output more random.
<code>.top_p</code>	An alternative to sampling with temperature, called nucleus sampling.
<code>.dry_run</code>	Logical; if TRUE, returns the prepared request object without executing it (default: FALSE).
<code>.overwrite</code>	Logical; if TRUE, allows overwriting an existing batch ID associated with the request (default: FALSE).
<code>.json_schema</code>	A JSON schema object as R list to enforce the output structure (default: NULL).
<code>.max_tries</code>	Maximum number of retries to perform the request (default: 3).
<code>.timeout</code>	Integer specifying the request timeout in seconds (default: 60).
<code>.verbose</code>	Logical; if TRUE, additional info about the requests is printed (default: FALSE).
<code>.id_prefix</code>	Character string to specify a prefix for generating custom IDs when names in <code>.llms</code> are missing (default: "tidyllm_openai_req_").

Value

An updated and named list of `.llms` with identifiers that align with batch responses, including a `batch_id` attribute.

tidyllm_schema	<i>Create a JSON schema for structured outputs</i>
----------------	--

Description

This function creates a JSON schema suitable for use with the API functions in tidyllm.

Usage

```
tidyllm_schema(name = "tidyllm_schema", ...)
```

Arguments

name	A character vector specifying the schema name. This serves as an identifier for the schema (Default: "tidyllm_schema").
...	Named arguments where each name represents a field in the schema and each value specifies the type. Supported types include R data types: <ul style="list-style-type: none">• "character": Represents a character type• "string": Allowed shorthand for character type• "factor(...)": A string with specific allowable values, represented as enum in JSON. Specify options as factor(option1, option2).• "logical": Represents a boolean.• "numeric": Represents a number.• "type[]": Appending [] allows for vector of a given type, e.g., "character[]".

Details

The `tidyllm_schema()` function is designed to make defining JSON schemas for tidyllm more concise and user-friendly. It maps R-like types to JSON schema types and validates inputs to enforce tidy data principles. Nested structures are not allowed to maintain compatibility with tidy data conventions.

Value

A list representing the JSON schema with the specified fields and types, suitable for passing to `openai()`'s `json_schema` parameter.

Note

Factor types (`factor(...)`) are treated as enumerations in JSON and are limited to a set of allowable string values. Arrays of a given type can be specified by appending [] to the type.

Examples

```
## Not run:
# Define a schema with tidy data principles
json_schema <- tidyllm_schema(
  name = "DocumentAnalysisSchema",
  Title = "character",
  Authors = "character[]",
  SuggestedFilename = "character",
  Type = "factor(Policy, Research)",
  Answer_Q1 = "character",
  Answer_Q2 = "character",
  Answer_Q3 = "character",
  Answer_Q4 = "character",
  KeyCitations = "character[]"
)

# Pass the schema to openai()
result <- openai(
  .llm = msg,
  .json_schema = json_schema
)

## End(Not run)
```


Index

* Message Creation Utilities

- df_llm_message, 15
- llm_message, 34

- azure_openai, 3
- azure_openai_chat, 4
- azure_openai_embedding, 6

- cancel_openai_batch, 7
- chat, 7
- chatgpt, 9
- check_batch, 10
- check_claude_batch, 10
- check_mistral_batch, 11
- check_openai_batch, 12
- claude, 13
- claude_chat, 13

- df_llm_message, 15, 35
- df_llm_message(), 35

- embed, 15

- fetch_batch, 17
- fetch_claude_batch, 18
- fetch_mistral_batch, 19
- fetch_openai_batch, 19

- gemini, 20
- gemini_chat, 21
- gemini_delete_file, 22
- gemini_embedding, 23
- gemini_file_metadata, 23
- gemini_list_files, 24
- gemini_upload_file, 24
- get_metadata, 25
- get_reply, 26
- get_reply(), 27
- get_reply_data, 26
- get_reply_data(), 26
- get_user_message, 27

- groq, 28
- groq_chat, 28
- groq_transcribe, 30

- last_metadata (get_metadata), 25
- last_metadata(), 25
- last_reply (get_reply), 26
- last_reply(), 26
- last_reply_data (get_reply_data), 26
- last_reply_data(), 27
- last_user_message (get_user_message), 27
- last_user_message(), 27, 28
- list_batches, 31
- list_claude_batches, 32
- list_mistral_batches, 32
- list_openai_batches, 33
- llm_message, 15, 34
- llm_message(), 15
- LLMMessage, 34

- mistral, 36
- mistral_chat, 36
- mistral_embedding, 38

- ollama, 38
- ollama_chat, 39
- ollama_download_model, 41
- ollama_embedding, 42
- ollama_list_models, 42
- openai, 43
- openai_chat, 43
- openai_embedding, 45

- pdf_page_batch, 46
- perplexity, 47
- perplexity_chat, 47

- rate_limit_info, 49

- send_batch, 49
- send_claude_batch, 51

`send_mistral_batch`, [52](#)

`send_openai_batch`, [53](#)

`tidy11m_schema`, [55](#)