Package: tidyllm (via r-universe)

November 8, 2024

Title Tidy Integration of Large Language Models

Version 0.2.0

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 7.3.1

VignetteBuilder knitr

Suggests knitr, rmarkdown, testthat (>= 3.0.0), tidyverse, httptest2, httpuv

Imports R6, base64enc, glue, jsonlite, curl, httr2, lubridate, purrr, rlang, stringr, grDevices, pdftools, tibble, cli, png, lifecycle

Depends R (>= 4.2.0)

Config/testthat/edition 3

NeedsCompilation no

Author Eduard Brüll [aut, cre]

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

Repository CRAN

Date/Publication 2024-11-07 11:40:02 UTC

Config/pak/sysreqs libicu-dev libjpeg-dev libpng-dev libssl-dev libpoppler-cpp-dev

Contents

azure_openai *Send LLM Messages to an OpenAI Chat Completions endpoint on Azure*

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(
  .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 = FALSE,
  .json_schema = NULL,
  .dry_run = FALSE,
  .max_tries = 3
)
```

Arguments

.llm	An LLMMessage object containing the conversation history.
.endpoint_url	Base URL for the API (default: Sys.getenv("AZURE_ENDPOINT_URL")).
.deployment	The identifier of the model that is deployed (default: "gpt-4o-mini").
.api_version	Which version of the API is deployed (default: "2024-08-01-preview")
.max_completion_tokens	An upper bound for the number of tokens that can be generated for a completion, including visible output tokens and reasoning tokens.

.frequency_penalty
: Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far.

.logit_bias
: A named list modifying the likelihood of specified tokens appearing in the completion.

.logprobs
: Whether to return log probabilities of the output tokens (default: FALSE).

.top_logprobs
: An integer between 0 and 20 specifying the number of most likely tokens to return at each token position.

.presence_penalty
: Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far.

.seed
: If specified, the system will make a best effort to sample deterministically.

.stop
: Up to 4 sequences where the API will stop generating further tokens.

.stream
: If set to TRUE, the answer will be streamed to console as it comes (default: FALSE).

.temperature
: What sampling temperature to use, between 0 and 2. Higher values make the output more random.

.top_p
: An alternative to sampling with temperature, called nucleus sampling.

.timeout
: Request timeout in seconds (default: 60).

.verbose
: Should additional information be shown after the API call (default: FALSE).

.json
: Should output be in JSON mode (default: FALSE).

.json_schema
: A JSON schema object as R list to enforce the output structure (If defined has precedence over JSON mode).

.dry_run
: If TRUE, perform a dry run and return the request object (default: FALSE).

.max_tries
: 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(msg)

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

## End(Not run)
```

chatgpt *ChatGPT Wrapper (Deprecated)*

Description

Provides a wrapper for the `openai()` function to facilitate migration from the deprecated `chatgpt()` function. This ensures backward compatibility while allowing users to transition to the updated features.

Usage

```
chatgpt(
  .llm,
  .model = "gpt-4o",
  .max_tokens = 1024,
  .temperature = NULL,
  .top_p = NULL,
  .top_k = NULL,
  .frequency_penalty = NULL,
  .presence_penalty = NULL,
  .api_url = "https://api.openai.com/",
  .timeout = 60,
  .verbose = FALSE,
  .json = FALSE,
  .stream = FALSE,
  .dry_run = FALSE
)
```

Arguments

`.llm`	An `LLMMessage` (passed directly to the `openai()` function)
`.model`	A character string specifying the model to use.
`.max_tokens`	An integer specifying the maximum number of tokens (mapped to `.max_completion_tokens` in `openai()`)
`.temperature`	A numeric value for controlling randomness. This is
`.top_p`	A numeric value for nucleus sampling, indicating the top
`.top_k`	Currently unused, as it is not supported by `openai()`.
`.frequency_penalty`	A numeric value that penalizes new tokens based on their frequency so far.
`.presence_penalty`	A numeric value that penalizes new tokens based on whether they appear in the text so far.
`.api_url`	Character string specifying the API URL. Defaults to the OpenAI API endpoint.
`.timeout`	An integer specifying the request timeout in seconds. This is

.verbose	Will print additional information about the request (default: false)
.json	Should json-mode be used? (detault: false)
.stream	Should the response be processed as a stream (default: false)
.dry_run	Should the request is constructed but not actually sent. Useful for debugging and testing. (default: false)

Details

This function is deprecated and is now a wrapper around `openai()`. It is recommended to switch to using `openai()` directly in future code. The `chatgpt()` function remains available to ensure backward compatibility for existing projects.

Value

An `LLMMessage` object with the assistant's reply.

See Also

Use `openai()` instead.

Examples

```
## Not run:
# Using the deprecated chatgpt() function
result <- chatgpt(.llm = llm_message(), .prompt = "Hello, how are you?")

## End(Not run)
```

---

check_claude_batch *Check Batch Processing Status for Claude API*

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 LLMMessage 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 peform request
.timeout	Integer specifying the request timeout in seconds (default: 60).

Value

A tibble with information about the status of batch processing

---

check_openai_batch *Check Batch Processing Status for OpenAI Batch API*

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

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

Value

A tibble with information about the status of batch processing.

claude *Interact with Claude AI models via the Anthropic API*

Description

Interact with Claude AI models via the Anthropic API

Usage

```
claude(
  .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

.llm	An LLMMessage object containing the conversation history and system prompt.
.model	Character string specifying the Claude model version (default: "claude-3-5-sonnet-20241022").
.max_tokens	Integer specifying the maximum number of tokens in the response (default: 1024).
.temperature	Numeric between 0 and 1 controlling response randomness.
.top_k	Integer controlling diversity by limiting the top K tokens.
.top_p	Numeric between 0 and 1 for nucleus sampling.
.metadata	List of additional metadata to include with the request.
.stop_sequences	Character vector of sequences that will halt response generation.
.tools	List of additional tools or functions the model can use.
.api_url	Base URL for the Anthropic API (default: "https://api.anthropic.com/").
.verbose	Logical; if TRUE, displays additional information about the API call (default: FALSE).

.max_tries	Maximum retries to peform request
.timeout	Integer specifying the request timeout in seconds (default: 60).
.stream	Logical; if TRUE, streams the response piece by piece (default: FALSE).
.dry_run	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(msg)

# With custom parameters
result2 <- claude(msg,
                  .temperature = 0.7,
                  .max_tokens = 1000)

## End(Not run)
```

---

df_llm_message *Convert a Data Frame to an LLMMessage Object*

Description

This function takes a data frame and converts it into an LLMMessage object representing a conversation history. The data frame should contain specific columns (`role` and `content`) with each row representing a message in the conversation.

Usage

```
df_llm_message(.df)
```

Arguments

.df	A data frame with at least two rows and columns `role` and `content`. The column `role` should contain values from "user", "assistant", or "system", and `content` should be of type character.

Value

An LLMMessage object containing the structured messages as per the input data frame.

`fetch_claude_batch` *Fetch Results for a Claude Batch*

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_id`s 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_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_id`s generated in `send_openai_batch()`.

Usage

```
fetch_openai_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.

generate_callback_function *Generate API-Specific Callback Function for Streaming Responses*

Description

This function generates a callback function that processes streaming responses from different language model APIs. The callback function is specific to the API provided (`claude`, `ollama`, `"mistral"`, or `openai`) and processes incoming data streams, printing the content to the console and updating a global environment for further use.

Usage

```
generate_callback_function(.api)
```

Arguments

.api	A character string indicating the API type. Supported values are "claude", "ollama", "mistral", "groq" and "openai".

Details

- **For Claude API**: The function processes event and data lines, and handles the message_start and message_stop events to control streaming flow.
- **For Ollama API**: The function directly parses the stream content as JSON and extracts the message$content field.
- **For OpenAI, Mistral and Groq**: The function handles JSON data streams and processes content deltas. It stops processing when the [DONE] message is encountered.

Value

A function that serves as a callback to handle streaming responses from the specified API. The callback function processes the raw data, updates the .tidyllm_stream_env$stream object, and prints the streamed content to the console. The function returns TRUE if streaming should continue, and FALSE when streaming is finished.

---

get_reply *Get Assistant Reply as Text*

Description

Retrieves the assistant's reply as plain text from an LLMMessage object at a specified index.

Usage

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

Arguments

.llm	An LLMMessage object containing the message history.
.index	A positive integer for the assistant reply index to retrieve, defaulting to the last reply.

Value

Plain text content of the assistant's reply, or NA_character_ if no reply is available.

get_reply_data *Get Data from an Assistant Reply by parsing structured JSON responses*

Description

Retrieves and parses the assistant's reply as JSON from an LLMMessage object at a specified index. If the reply is not marked as JSON, attempts to extract JSON content from text.

Usage

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

Arguments

.llm	An LLMMessage object containing the message history.
.index	A positive integer for the assistant reply index to retrieve, defaulting to the last reply.

Value

Parsed data content of the assistant's reply, or NULL if parsing fails.

get_user_message *Retrieve a User Message by Index*

Description

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

Usage

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

Arguments

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

Value

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

groq *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(
  .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

.llm	An LLMMessage object containing the conversation history.
.model	The identifier of the model to use (default: "llama-3.2-11b-vision-preview").
.max_tokens	The maximum number of tokens that can be generated in the response (default: 1024).
.temperature	Controls the randomness in the model's response. Values between 0 and 2 are allowed, where higher values increase randomness (optional).
.top_p	Nucleus sampling parameter that controls the proportion of probability mass considered. Values between 0 and 1 are allowed (optional).
.frequency_penalty	Number between -2.0 and 2.0. Positive values penalize repeated tokens, reducing likelihood of repetition (optional).
.presence_penalty	Number between -2.0 and 2.0. Positive values encourage new topics by penalizing tokens that have appeared so far (optional).

.stop	One or more sequences where the API will stop generating further tokens. Can be a string or a list of strings (optional).
.seed	An integer for deterministic sampling. If specified, attempts to return the same result for repeated requests with identical parameters (optional).
.api_url	Base URL for the Groq API (default: "https://api.groq.com/").
.json	Whether the response should be structured as JSON (default: FALSE).
.timeout	Request timeout in seconds (default: 60).
.verbose	If TRUE, displays additional information after the API call, including rate limit details (default: FALSE).
.stream	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 peform 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(msg)

# With custom parameters
result2 <- groq(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

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

Value

A character vector containing the transcription.

Examples

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

## End(Not run)
```

initialize_api_env *Initialize or Retrieve API-specific Environment*

Description

This function initializes a named environment for storing rate limit information specific to an API. It ensures that each API's rate limit data is stored separately.

Usage

```
initialize_api_env(.api_name)
```

Arguments

.api_name	The name of the API for which to initialize or retrieve the environment

last_reply *Get the Last Assistant Reply as Text*

Description

A wrapper around `get_reply()` to retrieve the most recent assistant text reply.

Usage

```
last_reply(.llm)
```

Arguments

.llm	A `LLMMessage` object.

Value

Returns the content of the assistant's reply at the specified index, based on the following conditions:

last_reply_data *Get the Last Assistant Reply as Text*

Description

A wrapper around `get_reply_data()` to retrieve structured data from the most recent assistant reply.

Usage

```
last_reply_data(.llm)
```

Arguments

`.llm`	A `LLMMessage` object.

Value

Returns the content of the assistant's reply at the specified index, based on the following conditions:

last_user_message *Retrieve the Last User Message*

Description

A wrapper around `get_user_message()` to retrieve the most recent user message.

Usage

```
last_user_message(.llm)
```

Arguments

`.llm`	A `LLMMessage` object.

Value

The content of the last user message.

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_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

Large Language Model Message Class

Large Language Model Message Class

Details

This class manages a history of messages and media interactions intended for use with large language models. It allows for adding messages, converting messages for API usage, and printing the history in a structured format.

Public fields

`message_history` List to store all message interactions.

`system_prompt` The system prompt used for a conversation

Methods

Public methods:

- `LLMMessage$new()`
- `LLMMessage$clone_deep()`
- `LLMMessage$add_message()`
- `LLMMessage$to_api_format()`
- `LLMMessage$has_image()`
- `LLMMessage$remove_message()`
- `LLMMessage$print()`
- `LLMMessage$clone()`

Method `new()`**:** Initializes the LLMMessage object with an optional system prompt.

Usage:

```
LLMMessage$new(system_prompt = "You are a helpful assistant")
```

Arguments:

`system_prompt` A string that sets the initial system prompt.

Returns: A new LLMMessage object. Deep Clone of LLMMessage Object
This method creates a deep copy of the `LLMMessage` object. It ensures that all internal states, including message histories and settings, are copied so that the original object remains unchanged when mutations are applied to the copy. This is particularly useful for maintaining immutability in a tidyverse-like functional programming context where functions should not have side effects on their inputs.

Method `clone_deep()`**:**

Usage:

```
LLMMessage$clone_deep()
```

Returns: A new `LLMMessage` object that is a deep copy of the original. Add a message
Adds a message to the history. Optionally includes media.

Method `add_message()`**:**

Usage:

```
LLMMessage$add_message(role, content, media = NULL, json = FALSE)
```

Arguments:

`role` The role of the message sender (e.g., "user", "assistant").

`content` The textual content of the message.

`media` Optional; media content to attach to the message.

`json` Is the message a raw string that contains a json response? Convert to API format
Converts the message history to a format suitable for various API calls.

Method `to_api_format()`**:**

Usage:

```
LLMMessage$to_api_format(
  api_type,
  cgpt_image_detail = "auto",
  no_system = FALSE
)
```

Arguments:

`api_type` The type of API (e.g., "claude","groq","openai").

`cgpt_image_detail` Specific option for ChatGPT API (imagedetail - set to auto)

`no_system` Without system prompt (default: FALSE)

Returns: A message history in the target API format Simple helper function to determine whether the message history contains an image We check this function whenever we call models that do not support images so we can post a warning to the user that images were found but not sent to the model

Method `has_image()`**:**

Usage:

```
LLMMessage$has_image()
```

Returns: Returns TRUE if the message hisotry contains images Remove a Message by Index
Removes a message from the message history at the specified index.

Method `remove_message()`**:**

Usage:

```
LLMMessage$remove_message(index)
```

Arguments:

`index` A positive integer indicating the position of the message to remove.

Returns: The `LLMMessage` object, invisibly.

Method `print()`**:** Prints the current message history in a structured format.

Usage:

```
LLMMessage$print()
```

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

`llm_message` *Create or Update Large Language Model Message Object*

Description

This function allows the creation of a new LLMMessage object or the updating of an existing one. It can handle the addition of text prompts and various media types such as images, PDFs, text files, or plots. The function includes input validation to ensure that all provided parameters are in the correct format.

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

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

Value

Returns an updated or new LLMMessage object.

`mistral` *Send LLMMessage to Mistral API*

Description

Send LLMMessage to Mistral API

Usage

```
mistral(
  .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

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

Value

Returns an updated LLMMessage object.

---

mistral_embedding *Generate Embeddings Using Mistral API*

Description

Generate Embeddings Using Mistral API

Usage

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

Arguments

.llm	An existing LLMMessage object (or a character vector of texts to embed)
.model	The embedding model identifier (default: "mistral-embed").
.timeout	Timeout for the API request in seconds (default: 120).
.max_tries	Maximum retries to peform 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 *Interact with local AI models via the Ollama API*

Description

Interact with local AI models via the Ollama API

Usage

```
ollama(
  .llm,
  .model = "gemma2",
  .stream = FALSE,
  .seed = NULL,
  .json = FALSE,
  .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

.llm	An LLMMessage object containing the conversation history and system prompt.
.model	Character string specifying the Ollama model to use (default: "gemma2")
.stream	Logical; whether to stream the response (default: FALSE)
.seed	Integer; seed for reproducible generation (default: NULL)
.json	Logical; whether to format response as JSON (default: FALSE)
.temperature	Float between 0-2; controls randomness in responses (default: NULL)
.num_ctx	Integer; sets the context window size (default: 2048)
.num_predict	Integer; maximum number of tokens to predict (default: NULL)
.top_k	Integer; controls diversity by limiting top tokens considered (default: NULL)
.top_p	Float between 0-1; nucleus sampling threshold (default: NULL)
.min_p	Float between 0-1; minimum probability threshold (default: NULL)
.mirostat	Integer (0,1,2); enables Mirostat sampling algorithm (default: NULL)
.mirostat_eta	Float; Mirostat learning rate (default: NULL)
.mirostat_tau	Float; Mirostat target entropy (default: NULL)
.repeat_last_n	Integer; tokens to look back for repetition (default: NULL)
.repeat_penalty	Float; penalty for repeated tokens (default: NULL)
.tfs_z	Float; tail free sampling parameter (default: NULL)
.stop	Character; custom stop sequence(s) (default: NULL)
.ollama_server	String; Ollama API endpoint (default: "http://localhost:11434")
.timeout	Integer; API request timeout in seconds (default: 120)
.keep_alive	Character; How long should the ollama model be kept in memory after request (default: NULL - 5 Minutes)
.dry_run	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(llm, .model = "gemma2", .temperature = 0.7)

# With custom parameters
response <- ollama(
  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. It can operate in a streaming mode where it provides live updates of the download status and progress, or a single response mode.

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

Arguments

.llm	An existing LLMMessage object (or a charachter vector of texts to embed)
.model	The embedding model identifier (default: "all-minilm").
.truncate	Whether to truncate inputs to fit the model's context length (default: TRUE).
.ollama_server	The URL of the Ollama server to be used (default: "http://localhost:11434").
.timeout	Timeout for the API request in seconds (default: 120).
.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_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 *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(
  .llm,
  .model = "gpt-4o",
  .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,
  .api_url = "https://api.openai.com/",
  .timeout = 60,
  .verbose = FALSE,
  .json = FALSE,
  .json_schema = NULL,
  .max_tries = 3,
  .dry_run = FALSE,
  .compatible = FALSE,
  .api_path = "/v1/chat/completions"
)
```

Arguments

.llm	An LLMMessage object containing the conversation history.
.model	The identifier of the model to use (default: "gpt-4o").
.max_completion_tokens	An upper bound for the number of tokens that can be generated for a completion, including visible output tokens and reasoning tokens.
.frequency_penalty	Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far.

.logit_bias	A named list modifying the likelihood of specified tokens appearing in the completion.
.logprobs	Whether to return log probabilities of the output tokens (default: FALSE).
.top_logprobs	An integer between 0 and 20 specifying the number of most likely tokens to return at each token position.
.presence_penalty	Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far.
.seed	If specified, the system will make a best effort to sample deterministically.
.stop	Up to 4 sequences where the API will stop generating further tokens.
.stream	If set to TRUE, the answer will be streamed to console as it comes (default: FALSE).
.temperature	What sampling temperature to use, between 0 and 2. Higher values make the output more random.
.top_p	An alternative to sampling with temperature, called nucleus sampling.
.api_url	Base URL for the API (default: "https://api.openai.com/").
.timeout	Request timeout in seconds (default: 60).
.verbose	Should additional information be shown after the API call (default: FALSE).
.json	Should output be in JSON mode (default: FALSE).
.json_schema	A JSON schema object as R list to enforce the output structure (If defined has precedence over JSON mode).
.max_tries	Maximum retries to perform request
.dry_run	If TRUE, perform a dry run and return the request object (default: FALSE).
.compatible	If TRUE, skip API and rate-limit checks for OpenAI compatible APIs (default: FALSE).
.api_path	The path relative to the base .api_url 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(
  .llm,
  .model = "text-embedding-3-small",
  .truncate = TRUE,
  .timeout = 120,
  .dry_run = FALSE,
  .max_tries = 3
)
```

Arguments

.llm	An existing LLMMessage object (or a character vector of texts to embed)
.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.

---

parse_duration_to_seconds

This internal function parses duration strings as returned by the OpenAI API

Description

This internal function parses duration strings as returned by the OpenAI API

Usage

```
parse_duration_to_seconds(.duration_str)
```

Arguments

.duration_str	A duration string.

Value

A numeric number of seconds

pdf_page_batch *Batch Process PDF into LLM Messages*

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

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

Value

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

perform_api_request *Perform an API request to interact with language models*

Description

Perform an API request to interact with language models

Usage

```
perform_api_request(
  .request,
  .api,
  .stream = FALSE,
  .timeout = 60,
  .max_tries = 3,
  .parse_response_fn = NULL,
  .dry_run = FALSE
)
```

Arguments

.request	The httr2 request object.
.api	The API identifier (e.g., "claude", "openai").
.stream	Stream the response if TRUE.
.timeout	Request timeout in seconds.
.max_tries	Maximum retry attempts for requests (default: 3).
.parse_response_fn	A function to parse the assistant's reply.
.dry_run	If TRUE, perform a dry run and return the request object.

Value

A list containing the assistant's reply and response headers.

ratelimit_from_header *Extract rate limit information from API response headers*

Description

Extract rate limit information from API response headers

Usage

```
ratelimit_from_header(.response_headers, .api)
```

Arguments

`.response_headers`	Headers from the API response
`.api`	The API type ("claude", "openai","groq")

Value

A list containing rate limit information

---

rate_limit_info *Get the current rate limit information for all or a specific API*

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

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,
  .logprobs = FALSE,
  .top_logprobs = 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

.llms	A list of LLMMessage objects containing conversation histories.
.model	Character string specifying the OpenAI model version (default: "gpt-4o").
.max_completion_tokens	Integer specifying the maximum tokens per response (default: NULL).
.frequency_penalty	Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far.
.logit_bias	A named list modifying the likelihood of specified tokens appearing in the completion.
.logprobs	Whether to return log probabilities of the output tokens (default: FALSE).
.top_logprobs	An integer between 0 and 20 specifying the number of most likely tokens to return at each token position.

`.presence_penalty`
: Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far.

`.seed`
: If specified, the system will make a best effort to sample deterministically.

`.stop`
: Up to 4 sequences where the API will stop generating further tokens.

`.temperature`
: What sampling temperature to use, between 0 and 2. Higher values make the output more random.

`.top_p`
: An alternative to sampling with temperature, called nucleus sampling.

`.dry_run`
: Logical; if TRUE, returns the prepared request object without executing it (default: FALSE).

`.overwrite`
: Logical; if TRUE, allows overwriting an existing batch ID associated with the request (default: FALSE).

`.json_schema`
: A JSON schema object as R list to enforce the output structure (default: NULL).

`.max_tries`
: Maximum number of retries to perform the request (default: 3).

`.timeout`
: Integer specifying the request timeout in seconds (default: 60).

`.verbose`
: Logical; if TRUE, additional info about the requests is printed (default: FALSE).

`.id_prefix`
: Character string to specify a prefix for generating custom IDs when names in `.llms` 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` *Create a JSON schema for structured outputs*

Description

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

Usage

```
tidyllm_schema(name, ...)
```

Arguments

`name`
: A character vector specifying the schema name. This serves as an identifier for the schema.

`...`
: Named arguments where each name represents a field in the schema and each value specifies the type. Supported types include R data types:

- "character": Represents a charcater type
- "string": Allowed shorthand for charachter 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)
```

`update_rate_limit` *Update the standard API rate limit info in the hidden .tidyllm_rate_limit_env environment*

Description

This function initializes stores ratelimit information from API functions for future use

Usage

```
update_rate_limit(.api_name, .response_object)
```

Arguments

`.api_name`	The name of the API for which to initialize or retrieve the environment.
`.response_object`	A preparsed response object cotaining info on remaining requests, tokens and rest times

Index