Package 'tidyllm'

October 10, 2024

Title Tidy Integration of Large Language Models

Version 0.1.0

Description A tidy interface for integrating large language model (LLM) APIs such as 'Claude', 'ChatGPT', 'Groq', and local models via 'Ollama' into R workflows. The package supports text and media-based interactions, interactive message history, stateful rate limit handling, 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>, and <https://ollama.com>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

VignetteBuilder knitr

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

Imports R6, base64enc, glue, jsonlite, httr2, lubridate, pdftools, purrr, rlang, stringr, grDevices, tibble

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 2024-10-10 16:20:01 UTC

Contents

`chatgpt` *Call the OpenAI API to interact with ChatGPT or o-reasoning models*

Description

Call the OpenAI API to interact with ChatGPT or o-reasoning models

Usage

```
chatgpt(
  .llm,
  .model = "gpt-4",
  .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,
  .wait = TRUE,
  .min_tokens_reset = 0L,
  .stream = FALSE
)
```

Arguments

`.llm`	An existing LLMMessage object or an initial text prompt.
`.model`	The model identifier (default: "gpt-4o").
`.max_tokens`	The maximum number of tokens to generate (default: 1024).
`.temperature`	Control for randomness in response generation (optional).
`.top_p`	Nucleus sampling parameter (optional).
`.top_k`	Top k sampling parameter (optional).

`.frequency_penalty`
: Controls repetition frequency (optional).

`.presence_penalty`
: Controls how much to penalize repeating content (optional)

`.api_url`
: Base URL for the API (default: https://api.openai.com/v1/completions).

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

`.verbose`
: Should additional information be shown after the API call

`.wait`
: Should we wait for rate limits if necessary?

`.min_tokens_reset`
: How many tokens should be remaining to wait until we wait for token reset?

`.stream`
: Stream back the response piece by piece (default: FALSE).

Value

Returns an updated LLMMessage object.

`claude` *Call the Anthropic API to interact with Claude models*

Description

Call the Anthropic API to interact with Claude models

Usage

```
claude(
  .llm,
  .model = "claude-3-5-sonnet-20240620",
  .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,
  .wait = TRUE,
  .min_tokens_reset = 0L,
  .timeout = 60,
  .stream = FALSE
)
```

Arguments

.llm	An existing LLMMessage object or an initial text prompt.
.model	The model identifier (default: "claude-3-5-sonnet-20240620").
.max_tokens	The maximum number of tokens to generate (default: 1024).
.temperature	Control for randomness in response generation (optional).
.top_k	Top k sampling parameter (optional).
.top_p	Nucleus sampling parameter (optional).
.metadata	Additional metadata for the request (optional).
.stop_sequences	Sequences that stop generation (optional).
.tools	Additional tools used by the model (optional).
.api_url	Base URL for the API (default: "https://api.anthropic.com/v1/messages").
.verbose	Should additional information be shown after the API call
.wait	Should we wait for rate limits if necessary?
.min_tokens_reset	How many tokens should be remaining to wait until we wait for token reset?
.timeout	Request timeout in seconds (default: 60).
.stream	Stream back the response piece by piece (default: FALSE).

Value

Returns an updated LLMMessage object.

---

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.

Examples

```
# Example data frame with role and content
df_example <- data.frame(
 role = c("system", "user", "assistant","user"),
 content = c("You allways only answer with two words",
             "Why is the sky blue?",
             "Rayleigh scattering",
             "Why is the sun yellow?"),
 stringsAsFactors = FALSE
)
df_llm_message(df_example)
```

---

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`, or `chatgpt`) 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"`, and `"chatgpt"`.

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 ChatGPT API**: 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.

`groq` *Call the Groq API to interact with fast opensource models on Groq*

Description

Call the Groq API to interact with fast opensource models on Groq

Usage

```
groq(
  .llm,
  .model = "llama-3.2-90b-text-preview",
  .max_tokens = 1024,
  .temperature = NULL,
  .top_p = NULL,
  .frequency_penalty = NULL,
  .presence_penalty = NULL,
  .api_url = "https://api.groq.com/",
  .timeout = 60,
  .verbose = FALSE,
  .wait = TRUE,
  .min_tokens_reset = 0L
)
```

Arguments

`.llm`	An existing LLMMessage object or an initial text prompt.
`.model`	The model identifier (default: "llama-3.2-90b-text-preview").
`.max_tokens`	The maximum number of tokens to generate (default: 1024).
`.temperature`	Control for randomness in response generation (optional).
`.top_p`	Nucleus sampling parameter (optional).
`.frequency_penalty`	Controls repetition frequency (optional).
`.presence_penalty`	Controls how much to penalize repeating content (optional)
`.api_url`	Base URL for the API (default: "https://api.anthropic.com/v1/messages").
`.timeout`	Request timeout in seconds (default: 60).
`.verbose`	Should additional information be shown after the API call

.wait Should we wait for rate limits if necessary?

.min_tokens_reset
How many tokens should be remaining to wait until we wait for token reset?

Value

Returns an updated LLMMessage object.

`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` *Retrieve Last Reply from an Assistant*

Description

This function extracts the last reply made by the assistant from a given LLMMessage object. It is particularly useful for fetching the most recent response from the assistant to display or log it separately.

Usage

```
last_reply(.llm = NULL, .json = FALSE)
```

Arguments

.llm An LLMMessage object containing the history of messages exchanged. This must be a valid LLMMessage object; otherwise, the function will stop with an error.

.json Should structured json data from the last reply be returned as R list (default: FALSE)

Value

Returns the content of the last reply made by the assistant. If the assistant has not replied yet, or if there are no assistant messages in the history, NULL is returned.

Note

This function only returns the content of the last assistant message and does not include media or other types of content that might be attached to the message.

See Also

LLMMessage for details on the message object structure.

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$has_image()`
- `LLMMessage$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)
```

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. 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")
```

Arguments:

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

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

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.

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).
`.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 whose output should be captured and attached (optional).

Value

Returns an updated or new LLMMessage object.

`ollama` *Send LLMMessage to ollama API*

Description

Send LLMMessage to ollama API

Usage

```
ollama(
  .llm,
  .model = "llama3",
  .stream = FALSE,
  .seed = NULL,
  .json = FALSE,
  .temperature = NULL,
  .num_ctx = 2048,
  .ollama_server = "http://localhost:11434",
  .timeout = 120
)
```

Arguments

`.llm`	An existing LLMMessage object or an initial text prompt.
`.model`	The model identifier (default: "llama3").
`.stream`	Should the answer be streamed to console as it comes (optional)
`.seed`	Which seed should be used for random numbers (optional).
`.json`	Should output be structured as JSON (default: FALSE).
`.temperature`	Control for randomness in response generation (optional).
`.num_ctx`	The size of the context window in tokens (optional)
`.ollama_server`	The URL of the ollama server to be used
`.timeout`	When should our connection time out

Value

Returns an updated LLMMessage object.

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 data frame.

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.

parse_duration_to_seconds *An internal function to parse the duration strings that OpenAI APIs return for ratelimit resets*

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

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,
  parse_response_fn = NULL
)
```

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.
parse_response_fn	A function to parse the assistant's reply.

Value

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

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 print. If not provided, the rate limit info for all APIs in the environment will be returned

Value

A tibble containing the rate limit information.

`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

---

`wait_rate_limit` *Wait for ratelimit restore times to ellapse if necessary*

Description

This function implements a standardized wait for rate limit resets

Usage

```
wait_rate_limit(.api_name, .min_tokens_reset)
```

Arguments

`.api_name`	The name of the API for which rate limit we want to wait
`.min_tokens_reset`	A token boundary at which we wish to reset (Typically should be larger than the size of the message)

Index