Package: LLMR (via r-universe)

November 22, 2024

Title Interface for Large Language Model APIs in R

Version 0.1.1

Description A unified interface to interact with various Large Language Model (LLM) APIs such as 'OpenAI' (see <https://platform.openai.com/docs/quickstart> for details), 'Anthropic' (see <https://docs.anthropic.com/en/api/getting-started> for details), 'Groq' (see <https://console.groq.com/docs/api-reference> for details), and 'Together AI' (see <https://docs.together.ai/docs/quickstart> for details). Allows users to configure API parameters, send messages, and retrieve responses seamlessly within R.

License MIT + file LICENSE

Encoding UTF-8

Imports httr2, purrr, rlang

Suggests testthat (>= 3.0.0), roxygen2 (>= 7.1.2), httptest2

RoxygenNote 7.2.3

Config/testthat/edition 3

NeedsCompilation no

Author Ali Sanaei [aut, cre]

Maintainer Ali Sanaei <sanaei@uchicago.edu>

Date/Publication 2024-10-21 11:51:30 UTC

Config/pak/sysreqs libssl-dev

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

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

RemoteRef HEAD

RemoteSha a56be7931757a0aa7213c9e78e0a8ca381141ba4

Contents

`call_llm` *Call LLM API*

Description

Sends a message to the specified LLM API and retrieves the response.

Usage

```
call_llm(config, messages, verbose = FALSE, json = FALSE)
```

Arguments

`config`	An ‘llm_config‘ object created by ‘llm_config()‘.
`messages`	A list of message objects to send to the API.
`verbose`	Logical. If ‘TRUE‘, prints the full API response.
`json`	Logical. If ‘TRUE‘, returns the raw JSON response as an attribute.

Value

The generated text response with additional attributes based on parameters.

See Also

`llm_config`

Examples

```
## Not run:
  # ----- Groq Example -----
  groq_config <- llm_config(
    provider = "groq",
    model = "llama-3.1-8b-instant",
    api_key = Sys.getenv("GROQ_KEY"),
    temperature = 0.7,
    max_tokens = 500
  )

  # Define the message with a system prompt
  message <- list(
   list(role = "system", content = "You ONLY fill in the blank. Do NOT answer in full sentences."),
    list(role = "user", content = "What's the capital of France? ----")
  )

  # Call the LLM
  response <- call_llm(groq_config, message)
```

```
# Display the response
cat("Groq Response:", response, "\n")

# Extract and print the full API response
full.response <- attr(response, which = 'full_response')
print(full.response)


# -----  OpenAI Example with More Parameters -----
# Create a configuration with more parameters
comprehensive_openai_config <- llm_config(
  provider = "openai",
  model = "gpt-4o-mini",
  api_key = Sys.getenv("OPENAI_KEY"),
  temperature = 1,          # Controls the randomness of the output
  max_tokens = 750,            # Maximum number of tokens to generate
  top_p = 1,                 # Nucleus sampling parameter
  frequency_penalty = 0.5,     # Penalizes new tokens based on their frequency
  presence_penalty = 0.3        # Penalizes new tokens based on their presence
)

# Define a more complex message
comprehensive_message <- list(
  list(role = "system", content = "You are an expert data scientist."),
  list(role = "user", content = "When will you ever use OLS?")
)

# Call the LLM with all parameters
comprehensive_response <- call_llm(
  config = comprehensive_openai_config,
  messages = comprehensive_message,
  json = TRUE        # Retrieve the raw JSON response as an attribute
)

# Display the generated text response
cat("Comprehensive OpenAI Response:", comprehensive_response, "\n")

# Access and print the raw JSON response
raw_json_response <- attr(comprehensive_response, "raw_json")
print(raw_json_response)

## End(Not run)
```

`llm_config` *Create LLM Configuration*

Description

Creates a configuration object for interacting with a specified LLM API provider.

Usage

```
llm_config(provider, model, api_key, ...)
```

Arguments

provider	A string specifying the API provider. Supported providers include: "openai" for OpenAI, "anthropic" for Anthropic, "groq" for Groq, "together" for Together AI; This configuration is extendable; to add support for additional providers, define a new S3 method for ‘call_llm‘ corresponding to the provider.
model	The model name to use. This depends on the provider. For example: OpenAI: "gpt-4o-mini", Anthropic: "claude-3-opus-20240229", Groq: "llama-3.1-8b-instant", Together AI: "meta-llama/Meta-Llama-3.1-8B-Instruct-Turbo".
api_key	Your API key for the provider.
...	Additional model-specific parameters (e.g., ‘temperature‘, ‘max_tokens‘).

Value

An object of class ‘llm_config‘ containing API and model parameters.

See Also

call_llm

Examples

```
## Not run:
  # Obtain API keys from:
  # OpenAI: https://platform.openai.com/api-keys
  # Groq: https://console.groq.com/keys
  # Anthropic: https://console.anthropic.com/settings/keys
  # Together AI: https://api.together.ai/settings/api-keys

  # ----- OpenAI Example -----
  openai_config <- llm_config(
    provider = "openai",
    model = "gpt-4o-mini",
    api_key = OPENAI_KEY,
    temperature = 0.7,
    max_tokens = 500
  )

  # ----- Anthropic Example -----
  anthropic_config <- llm_config(
    provider = "anthropic",
    model = "claude-3-opus-20240229",
    api_key = ANTHROPIC_API_KEY,
    max_tokens = 500
  )

  # ----- Groq Example -----
```

```
  groq_config <- llm_config(
    provider = "groq",
    model = "llama-3.1-8b-instant",
    api_key = GROQ_API_KEY,
    temperature = 0.3,
    max_tokens = 1000
  )

  # ----- Together AI Example -----
  together_config <- llm_config(
    provider = "together",
    model = "meta-llama/Meta-Llama-3.1-8B-Instruct-Turbo",
    api_key = TOGETHER_KEY,
    temperature = 0.5,
    max_tokens = 1000
  )

 # This configuration is extendable by defining new `call_llm` methods for additional providers.

## End(Not run)
```

Index