# Package 'chatLLM'

May 8, 2026

**Type** Package

**Title** A Flexible Interface for 'LLM' API Interactions

**Version** 0.1.4

**Maintainer** Kwadwo Daddy Nyame Owusu Boakye <kwadwo.owusuboakye@outlook.com>

**Description** Provides a flexible interface for interacting with Large Language Model ('LLM') providers including 'OpenAI', 'Azure OpenAI', 'Azure AI Foundry', 'Groq', 'Anthropic', 'DeepSeek', 'DashScope', 'Gemini', 'Grok', 'GitHub Models', and AWS Bedrock. Supports both synchronous and asynchronous
chat-completion APIs, with
features such as retry logic, dynamic model selection, customizable parameters, and multi-message conversation handling. Designed to streamline integration with state-of-the-art LLM services across multiple platforms.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Imports** httr (>= 1.4.0), jsonlite (>= 1.7.2), stats

**Suggests** aws.signature, future, promises, later, testthat, roxygen2

**URL** https://github.com/knowusuboaky/chatLLM,
https://knowusuboaky.github.io/chatLLM/

**BugReports** https://github.com/knowusuboaky/chatLLM/issues

**NeedsCompilation** no

**Author** Kwadwo Daddy Nyame Owusu Boakye [aut, cre]

**Repository** CRAN

**Date/Publication** 2026-02-15 11:30:02 UTC

## Contents

`call_llm` *Unified chat - completion interface*

## Description

A unified wrapper for several "OpenAI - compatible" chat - completion APIs (OpenAI, Groq, Anthropic, DeepSeek, Alibaba DashScope, GitHub Models, Grok, Gemini) plus Azure OpenAI, Azure AI Foundry, and AWS Bedrock via the Converse API. Accepts either a single ‘prompt‘ **or** a full ‘messages‘ list, adds the correct authentication headers, retries on transient failures, and returns the assistant’s text response. You can toggle informational console output with ‘verbose = TRUE/FALSE‘. If the chosen ‘model‘ is no longer available, the function stops early and suggests running ‘list_models("<provider>")‘.

## Arguments

| | |
|---|---|
| `prompt` | Character. Single user prompt (optional if ‘messages‘). |
| `messages` | List. Full chat history; see *Messages*. |
| `provider` | Character. One of ‘"openai"‘, ‘"groq"‘, ‘"anthropic"‘, ‘"deepseek"‘, ‘"dashscope"‘,‘"grok"‘, ‘"gemini"‘, ‘"github"‘, ‘"azure_openai"‘, ‘"azure_foundry"‘, or ‘"bedrock"‘. |
| `model` | Character. Model ID. If ‘NULL‘, uses the provider default. |
| `temperature` | Numeric. Sampling temperature (0 - 2). Default ‘0.7‘. |
| `max_tokens` | Integer. Max tokens to generate. Default ‘1000‘. |
| `api_key` | Character. Override API key; if ‘NULL‘, uses the environment variable for that provider. |
| `n_tries` | Integer. Retry attempts on failure. Default ‘3‘. |
| `backoff` | Numeric. Seconds between retries. Default ‘2‘. |
| `verbose` | Logical. Whether to display informational messages (‘TRUE‘) or suppress them (‘FALSE‘). Default ‘TRUE‘. |
| `endpoint_url` | Character. Custom endpoint; if ‘NULL‘, a sensible provider - specific default is used. |
| `github_api_version` | Character. Header ‘X - GitHub - Api - Version‘. Default ‘"2022 - 11 - 28"‘. |
| `anthropic_api_version` | Character. Header ‘anthropic - version‘. Default ‘"2023 - 06 - 01"‘. |
| `azure_api_version` | Character. Query param used by Azure OpenAI. Default comes from ‘AZURE_OPENAI_API_VERSION (fallback ‘"2024-02-15-preview"‘). |
| `azure_endpoint` | Character. Azure OpenAI resource endpoint, e.g. ‘https://my-resource.openai.azure.com‘. |
| `azure_foundry_api_version` | Character. Query param used by Azure AI Foundry chat/model endpoints. Default comes from ‘AZURE_FOUNDRY_API_VERSION‘ (fallback ‘"2024-05-01-preview"‘). |

azure_foundry_endpoint
: Character. Azure AI Foundry endpoint base, e.g. ‘https://my-foundry.models.ai.azure.com‘.

azure_foundry_token
: Character. Optional bearer token for Azure AI Foundry (‘Authorization: Bearer ...‘). If set, this is used instead of ‘AZURE_FOUNDRY_API_KEY‘.

aws_region
: Character. AWS region for ‘provider = "bedrock"‘. Defaults to ‘AWS_REGION‘, then ‘AWS_DEFAULT_REGION‘, then ‘"us-east-1"‘.

aws_access_key_id
: Character. Optional AWS access key ID override.

aws_secret_access_key
: Character. Optional AWS secret access key override.

aws_session_token
: Character. Optional AWS session token override.

...
: Extra JSON - body fields (e.g. ‘top_p‘, ‘stop‘, ‘presence_penalty‘).

.post_func
: Internal. HTTP POST function (default ‘httr::POST‘).

## Details

Core chat - completion wrapper for multiple providers

## Value

Character scalar: assistant reply text.

## Messages

* ‘prompt‘ - character scalar treated as a single *user* message. * ‘messages‘ - list of lists; each element must contain ‘role‘ and ‘content‘. If both arguments are supplied, the ‘prompt‘ is appended as an extra user message.

## Examples

```
## Not run:

## 1. Listing available models
# List all providers at once
all_mods <- list_models("all")
str(all_mods)

# List OpenAI-only, Groq-only, Anthropic-only
openai_mods   <- list_models("openai")
groq_mods     <- list_models("groq")
anthropic_mods<- list_models("anthropic", anthropic_api_version = "2023-06-01")

## 2. Single-prompt interface

# 2a. Basic usage
Sys.setenv(OPENAI_API_KEY = "sk-...")
res_basic <- call_llm(
```

```
  prompt   = "Hello, how are you?",
  provider = "openai"
)
cat(res_basic)

# 2b. Adjust sampling and penalties
res_sampling <- call_llm(
  prompt      = "Write a haiku about winter",
  provider    = "openai",
  temperature = 1.2,
  top_p       = 0.5,
  presence_penalty  = 0.6,
  frequency_penalty = 0.4
)
cat(res_sampling)

# 2c. Control length and retries
res_len <- call_llm(
  prompt      = "List 5 uses for R",
  provider    = "openai",
  max_tokens  = 50,
  n_tries     = 5,
  backoff     = 0.5
)
cat(res_len)

# 2d. Using stop sequences
res_stop <- call_llm(
  prompt   = "Count from 1 to 10:",
  provider = "openai",
  stop     = c("6")
)
cat(res_stop)

# 2e. Override API key for one call
res_override <- call_llm(
  prompt   = "Override test",
  provider = "openai",
  api_key  = "sk-override",
  max_tokens = 20
)
cat(res_override)

# 2f. Factory interface for repeated prompts
GitHubLLM <- call_llm(
  provider   = "github",
  max_tokens = 60,
  verbose    = FALSE
)
# direct invocation
story1 <- GitHubLLM("Tell me a short story")
cat(story1)
```

```
## 3. Multi-message conversation

# 3a. Simple system + user
convo1 <- list(
  list(role = "system",    content = "You are a helpful assistant."),
  list(role = "user",      content = "Explain recursion.")
)
res1 <- call_llm(
  messages   = convo1,
  provider   = "openai",
  max_tokens = 100
)
cat(res1)

# 3b. Continue an existing chat by appending a prompt
prev <- list(
  list(role = "system", content = "You are concise."),
  list(role = "user",   content = "Summarize the plot of Hamlet.")
)
res2 <- call_llm(
  messages = prev,
  prompt   = "Now give me three bullet points."
)
cat(res2)

# 3c. Use stop sequence in multi-message
convo2 <- list(
  list(role = "system", content = "You list items."),
  list(role = "user",   content = "Name three colors.")
)
res3 <- call_llm(
  messages = convo2,
  stop     = c(".")
)
cat(res3)

# 3d. Multi-message via factory interface
ScopedLLM <- call_llm(provider = "openai", temperature = 0.3)
chat_ctx <- list(
  list(role = "system", content = "You are a math tutor.")
)
ans1 <- ScopedLLM(messages = chat_ctx, prompt = "Solve 2+2.")
cat(ans1)
ans2 <- ScopedLLM("What about 10*10?")
cat(ans2)

## End(Not run)
```

---

`list_models` *List Available Models for Supported Providers*

---

## Description

Retrieve the catalog of available model IDs for one or all supported chat - completion providers. Useful for discovering active models and avoiding typos or deprecated defaults.

Supported providers:

- `"openai"` - OpenAI Chat Completions API
- `"groq"` - Groq OpenAI - compatible endpoint
- `"anthropic"` - Anthropic Claude API
- `"deepseek"` - DeepSeek chat API
- `"dashscope"` - Alibaba DashScope compatible API
- `"github"` - GitHub Models OpenAI - compatible API
- `"azure_openai"` - Azure OpenAI deployments/models
- `"azure_foundry"` - Azure AI Foundry chat/models endpoints
- `"bedrock"` - AWS Bedrock (Converse API)
- `"all"` - Fetch catalogs for all of the above

## Arguments

| | |
|---|---|
| `provider` | Character. One of `"github"`, `"openai"`, `"groq"`, `"anthropic"`, `"deepseek"`, `"dashscope"`, `"azure_openai"`, `"azure_foundry"`, `"bedrock"` or `"all"`. Case - insensitive. |
| `...` | Additional arguments passed to the per - provider helper (e.g. `limit` for Anthropic, or `api_version` for GitHub). |
| `github_api_version` | Character. Header value for `X - GitHub - Api - Version` (GitHub Models). Default `"2022 - 11 - 28"`. |
| `anthropic_api_version` | Character. Header value for `anthropic - version` (Anthropic). Default `"2023 - 06 - 01"`. |
| `azure_api_version` | Character. Query version for Azure OpenAI listing. Default `"2024-02-15-preview"`. |
| `azure_foundry_api_version` | Character. Query version for Azure AI Foundry model listing. Default `"2024-05-01-preview"`. |

## Value

If `provider != "all"`, a character vector of model IDs for that single provider. If `provider == "all"`, a named list of character vectors, one per provider.

## See Also

`call_llm`

## Examples

```
## Not run:
Sys.setenv(OPENAI_API_KEY = "sk-...")
openai_models <- list_models("openai")
head(openai_models)

Sys.setenv(ANTHROPIC_API_KEY = "sk-...")
anthro_models <- list_models("anthropic", anthropic_api_version = "2023-06-01")

Sys.setenv(GH_MODELS_TOKEN = "ghp-...")
github_models <- list_models("github", github_api_version = "2022-11-28")

## End(Not run)
```

# Index