

Package: quackingllama (via r-universe)

September 28, 2026

Title Process text with Ollama, retrieve structured results, cache them locally in DuckDB

Version 0.1.0.9004

Description Process text with Ollama, store results in DuckDB.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports base64enc, cld3, cli, dplyr, duckdb, fs, httpuv, httr2, jsonlite, purrr, rlang, stringr, tibble, yyjsonr

Depends R (>= 3.5)

LazyData true

URL <https://giocomai.github.io/quackingllama/>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake make libicu-dev libprotobuf-dev libuv1-dev libssl-dev protobuf-compiler libprotoc-dev xz-utils zlib1g-dev

Repository <https://giocomai.r-universe.dev>

Date/Publication 2026-09-28 12:17:02 UTC

RemoteUrl <https://github.com/giocomai/quackingllama>

RemoteRef HEAD

RemoteSha 32e32dcdb4b1c766ba710983420ead1195517252

Contents

ql_create_schema_enum	2
ql_disable_db	3
ql_enable_db	4
ql_generate	4
ql_get_api_keys	5
ql_get_db_options	6
ql_get_models	7

ql_get_options	7
ql_hash	8
ql_prompt	9
ql_read_images	10
ql_request	11
ql_set_api_keys	12
ql_set_db_options	12
ql_set_options	13
ql_translate	14
Index	17

ql_create_schema_enum	<i>Create a schema with one or more questions, and a single set of answers</i>
-----------------------	--

Description

Create a schema with one or more questions, and a single set of answers

Usage

ql_create_schema_enum(questions, answers)

Arguments

- questions A character vector, of length 1 or more, with questions to be asked, usually about a text given as prompt.
- answers A character vector: the response to each question should be one selected among the given options.

Value

A list object, that can be passed to the format argument of [ql_prompt](#)

Examples

```
library("quackingllama")
schema_1 <- ql_create_schema_enum(
  questions = c(
    "Does this story include humans",
    "Does this story include animals",
    "Is this story about a woman",
    "Does this story have a happy ending"
  ),
  answers = c("Yes", "No", "Unclear")
)
```

```

schema_1

## Not run:
responses_df <- ql_prompt(
  prompt = "this it the story of a duck that escapes",
  format = schema_1,
) |>
  ql_generate() |>
  dplyr::pull("response") |>
  yyjsonr::read_json_str() |>
  tibble::as_tibble()

responses_df

responses_df <- ql_prompt(
  prompt = "this it the story of a duck that escapes from a lady",
  format = schema_1
) |>
  ql_generate() |>
  dplyr::pull("response") |>
  yyjsonr::read_json_str() |>
  tibble::as_tibble()

responses_df

## End(Not run)

```

ql_disable_db	<i>Disable caching for the current session</i>
---------------	--

Description

Disable caching for the current session

Usage

```
ql_disable_db()
```

Value

Nothing, used for its side effects.

See Also

Other database: [ql_enable_db\(\)](#), [ql_set_db_options\(\)](#)

Examples

```
ql_disable_db()
```

ql_enable_db	<i>Enable storing data in a database for the current session</i>
--------------	--

Description

Enable storing data in a database for the current session

Usage

```
ql_enable_db(db_type = "DuckDB")
```

Arguments

db_type Defaults to DuckDB.

Value

Nothing, used for its side effects.

See Also

Other database: [ql_disable_db\(\)](#), [ql_set_db_options\(\)](#)

Examples

```
ql_enable_db()
```

ql_generate	<i>Generate a response and return the result in a data frame</i>
-------------	--

Description

Generate a response and return the result in a data frame

Usage

```
ql_generate(
  prompt_df,
  only_cached = FALSE,
  only_done = TRUE,
  host = NULL,
  messages = NULL,
  keep_alive = NULL,
  timeout = NULL,
  error = c("fail", "warn")
)
```

Arguments

prompt_df	A data frame with all inputs passed to the LLM, typically created with <code>ql_prompt()</code> .
only_cached	Defaults to FALSE. If TRUE, only cached responses are returned.
only_done	Defaults to TRUE. If TRUE, if a cached response was not "done" (i.e. not completed), it ignores the cache and processes it again.
host	The address where the Ollama API can be reached, e.g. <code>http://localhost:11434</code> for locally deployed Ollama.
messages	Defaults to NULL. If given, in line with official Ollama documentation: "the messages of the chat, this can be used to keep a chat memory".
keep_alive	Defaults to "5m". Controls how long the model will stay loaded into memory following the request.
timeout	If not set with <code>ql_set_options()</code> , defaults to 300 seconds (5 minutes).
error	Defines how errors should be handled, defaults to "fail", i.e. if an error emerges while querying the LLM, the function stops. If set to "warn", it sets the response to <code>NA_character_</code> and stores it in database. This can be useful e.g. for proceed if the prompts include a request that routinely time outs without giving a response. This does not imply that the model would never give a responses, e.g. re-running the same query with longer time out may work.

Value

A data frame, including a response column, as well as other information returned by the model.

Examples

```
## Not run:
ql_prompt("a haiku") |>
  ql_generate()

## End(Not run)
```

ql_get_api_keys	<i>Retrieve previously set Ollama API keys typically used for cloud models</i>
-----------------	--

Description

Retrieve previously set Ollama API keys typically used for cloud models

Usage

```
ql_get_api_keys(api_keys = NULL)
```

Arguments

api_keys	Valid Ollama API keys, typically used for cloud models.
----------	---

Value

API keys if previously set with ql_set_api_keys(), an empty string if not set, or the same keys provided as input to this function.

Examples

```
ql_set_api_keys("<your_api_keys_here>")
ql_get_api_keys()
```

ql_get_db_options	<i>Retrieve</i>
-------------------	-----------------

Description

Retrieve

Usage

```
ql_get_db_options(options = c("db", "db_type", "db_folder", "db_filename"))
```

Arguments

options Available options that

Value

A list with the selected options.

Examples

```
ql_get_db_options()

## Retrieve only selected option
ql_get_db_options("db_type")
```

ql_get_models	<i>Get available models</i>
---------------	-----------------------------

Description

Get available models

Usage

```
ql_get_models(host = "http://localhost:11434")
```

Arguments

host	Defaults to "http://localhost:11434", where locally deployed Ollama usually responds.
------	---

Value

A data frame (a tibble) with details on all locally available models.

Examples

```
## Not run:  
ql_get_models()  
  
## End(Not run)
```

ql_get_options	<i>Get options</i>
----------------	--------------------

Description

Get options

Usage

```
ql_get_options(  
  options = c("system", "model", "think", "host", "temperature", "seed", "keep_alive",  
             "timeout"),  
  system = NULL,  
  model = NULL,  
  think = NULL,  
  host = NULL,  
  temperature = NULL,  
  seed = NULL,  
  keep_alive = NULL,  
  timeout = NULL  
)
```

Arguments

`options` A character vector used to filter which options should effectively be returned. Defaults to all available.

Value

A list with all available options (or those selected with `options`)

Examples

```
ql_set_options(
  model = "llama3.2",
  host = "http://localhost:11434",
  system = "You are a helpful assistant.",
  temperature = 0,
  seed = 42,
  keep_alive = "5m"
)

ql_get_options()
```

<code>ql_hash</code>	<i>Hash all inputs relevant to the call to the LLM and create a hash to be used for caching.</i>
----------------------	--

Description

Mostly used internally.

Usage

```
ql_hash(prompt_df)
```

Arguments

`prompt_df` A data frame with all inputs passed to the LLM, typically created with [ql_prompt\(\)](#).

Value

A tibble, such as those returned by [ql_prompt\(\)](#), but always including a hash column.

Examples

```
ql_prompt("a haiku", hash = FALSE) |> ql_hash()
```

ql_prompt	<i>Generate a data frame with all relevant inputs for the LLM.</i>
-----------	--

Description

Typically passed to `ql_generate()`.

Usage

```
ql_prompt(
  prompt,
  system = NULL,
  format = NULL,
  model = NULL,
  think = NULL,
  images = NULL,
  temperature = NULL,
  seed = NULL,
  host = NULL,
  hash = TRUE
)
```

Arguments

prompt	A prompt for the LLM.
system	System message to pass to the model. See official documentation for details. For example: "You are a helpful assistant."
format	A json schema used to format the response. See README for examples. For more context about structured outputs, see the the dedicated section of the documentation .
model	The name of the model, e.g. llama3.2 or phi3.5:3.8b. Run <code>ollama list</code> from the command line to see a list of locally available models.
think	If TRUE, Ollama enables thinking mode for models which support it. For more information, see Ollama's announcement of 'thinking' capabilities .
temperature	Numeric value comprised between 0 and 1 passed to the model. When set to 0 and with the same seed, the response to the same prompt is always exactly the same. When closer to one, the response is more variable and creative. Use 0 for consistent responses. Setting this to 0.7 is a common choice for creative or interactive tasks.
seed	An integer. When temperature is set to 0 and the seed is constant, the model consistently returns the same response to the same prompt.
host	The address where the Ollama API can be reached, e.g. <code>http://localhost:11434</code> for locally deployed Ollama.
hash	Defaults to TRUE. If TRUE, adds a column with the hash of all other components of the prompt. Used internally for caching. Can be added separately with <code>ql_hash()</code> .

Details

For more details and context about each parameter, see <https://github.com/ollama/ollama/blob/main/docs/api.md>.

Value

A tibble with all main components of a query, to be passed to `ql_generate()`.

Examples

```
ql_prompt("a haiku")
```

ql_read_images	<i>Read image in order to pass it to multimodal models</i>
----------------	--

Description

Read image in order to pass it to multimodal models

Usage

```
ql_read_images(path)
```

Arguments

path Path to image file.

Value

A list object of character vectors of base 64 encoded images.

Examples

```
if (interactive()) {
  library("quackingllama")

  img_path <- fs::file_temp(ext = "png")

  download.file(
    url = "https://ollama.com/public/ollama.png",
    destfile = img_path
  )

  resp_df <- ql_prompt(
    prompt = "what is this?",
    images = img_path,
    model = "llama3.2-vision"
  ) |>
  ql_generate()
```

```

    resp_df
    resp_df$response
  }

```

ql_request

*Create http2 request for both generate and chat endpoints***Description**

Create http2 request for both generate and chat endpoints

Usage

```

ql_request(
  prompt_df,
  endpoint = "generate",
  host = NULL,
  messages = NULL,
  keep_alive = NULL,
  timeout = NULL
)

```

Arguments

endpoint	Defaults to "generate". Must be either "generate" or "chat".
host	The address where the Ollama API can be reached, e.g. <code>http://localhost:11434</code> for locally deployed Ollama.
messages	Defaults to NULL. If given, in line with official Ollama documentation: "the messages of the chat, this can be used to keep a chat memory".
keep_alive	Defaults to "5m". Controls how long the model will stay loaded into memory following the request.
timeout	If not set with ql_set_options() , defaults to 300 seconds (5 minutes).

Value

A http2 request object.

Examples

```

ql_prompt(prompt = "a haiku")

ql_prompt(prompt = "a haiku") |>
  ql_request() |>
  http2::req_dry_run()

```

ql_set_api_keys	<i>Set Ollama API keys typically used for cloud models</i>
-----------------	--

Description

Set Ollama API keys typically used for cloud models

Usage

```
ql_set_api_keys(api_keys)
```

Arguments

api_keys	Valid Ollama API keys, typically used for cloud models.
----------	---

Value

Invisibly returns API keys, mainly used for its side effects.

Examples

```
ql_set_api_keys("<your_api_keys_here>")
```

ql_set_db_options	<i>Set options for the local database and enables caching</i>
-------------------	---

Description

Set options for the local database and enables caching

Usage

```
ql_set_db_options(db_filename = NULL, db_type = "DuckDB", db_folder = ".")
```

Arguments

db_filename	Defaults NULL. Internally, defaults to a combination of quacklinglama, followed by the name of the model used. Name given to the local database file. Useful for differentiating among different approaches or projects when storing multiple database files in the same folder.
db_type	Defaults to DuckDB.
db_folder	Defaults to ., i.e., to the current working directory.

Value

Nothing, used for its side effects.

See Also

Other database: [ql_disable_db\(\)](#), [ql_enable_db\(\)](#)

Examples

```
ql_set_db_options(db_filename = "testing_ground")
```

ql_set_options	<i>Set basic options for the current session.</i>
----------------	---

Description

Set basic options for the current session.

Usage

```
ql_set_options(  
  system = NULL,  
  model = NULL,  
  think = NULL,  
  host = NULL,  
  temperature = NULL,  
  seed = NULL,  
  keep_alive = NULL,  
  timeout = NULL  
)
```

Arguments

system	System message to pass to the model. See official documentation for details. For example: "You are a helpful assistant."
model	The name of the model, e.g. llama3.2 or phi3.5:3.8b. Run <code>ollama list</code> from the command line to see a list of locally available models.
think	If TRUE, Ollama enables thinking mode for models which support it. For more information, see Ollama's announcement of 'thinking' capabilities .
host	The address where the Ollama API can be reached, e.g. <code>http://localhost:11434</code> for locally deployed Ollama.
temperature	Numeric value comprised between 0 and 1 passed to the model. When set to 0 and with the same seed, the response to the same prompt is always exactly the same. When closer to one, the response is more variable and creative. Use 0 for consistent responses. Setting this to 0.7 is a common choice for creative or interactive tasks.
seed	An integer. When temperature is set to 0 and the seed is constant, the model consistently returns the same response to the same prompt.

keep_alive	Defaults to "5m". Controls how long the model will stay loaded into memory following the request.
timeout	Time in seconds before the request times out. Defaults to 300 (corresponding to 5 minutes).

Value

Nothing, used for its side effects. Options can be retrieved with `ql_get_db_options()`

Examples

```
ql_set_options(
    model = "llama3.2",
    host = "http://localhost:11434",
    system = "You are a helpful assistant.",
    temperature = 0,
    seed = 42
)

ql_get_options()
```

ql_translate	<i>Translate strings relying on dedicated models</i>
--------------	--

Description

Translate strings relying on dedicated models

Usage

```
ql_translate(
    text,
    source_language = NULL,
    source_language_code = NULL,
    target_language = NULL,
    target_language_code = NULL,
    translation_model = "translategemma:4b",
    temperature = NULL,
    seed = NULL,
    host = NULL,
    hash = TRUE,
    only_cached = FALSE,
    keep_alive = NULL,
    timeout = NULL,
    error = c("fail", "warn")
)
```

Arguments

text	Text to be translated.
source_language	Defaults to NULL. If not given, detected automatically. If source_language_code given, source_language is ignored. Source language as a full word, e.g. "English". For available languages with the default translategemma model, see the list of available languages in the documentation on Ollama's website .
source_language_code	Defaults to NULL. If not given, detected automatically. If source_language_code given, source_language is ignored. Source language as a language code, e.g. "en", or "en-GB". For available languages with the default translategemma model, see the list of available languages in the documentation on Ollama's website .
target_language	Defaults to NULL. If target_language_code given, target_language is ignored. Target language as a full word, e.g. "English". For available languages with the default translategemma model, see the list of available languages in the documentation on Ollama's website .
target_language_code	Defaults to en. If not given, detected from target_language. Either target_language or target_language_code must be given. Target language as a language code, e.g. "en", or "en-GB". For available languages with the default translategemma model, see the list of available languages in the documentation on Ollama's website .
translation_model	Defaults to translategemma:4b (12b and 27 are also available). May work also with other models not specifically targeting translation.
temperature	Numeric value comprised between 0 and 1 passed to the model. When set to 0 and with the same seed, the response to the same prompt is always exactly the same. When closer to one, the response is more variable and creative. Use 0 for consistent responses. Setting this to 0.7 is a common choice for creative or interactive tasks.
seed	An integer. When temperature is set to 0 and the seed is constant, the model consistently returns the same response to the same prompt.
host	The address where the Ollama API can be reached, e.g. http://localhost:11434 for locally deployed Ollama.
hash	Defaults to TRUE. If TRUE, adds a column with the hash of all other components of the prompt. Used internally for caching. Can be added separately with ql_hash() .
only_cached	Defaults to FALSE. If TRUE, only cached responses are returned.
keep_alive	Defaults to "5m". Controls how long the model will stay loaded into memory following the request.
timeout	If not set with ql_set_options() , defaults to 300 seconds (5 minutes).
error	Defines how errors should be handled, defaults to "fail", i.e. if an error emerges while querying the LLM, the function stops. If set to "warn", it sets the response

to NA_character_ and stores it in database. This can be useful e.g. for proceed if the prompts include a request that routinely time outs without giving a response. This does not imply that the model would never give a responses, e.g. re-running the same query with longer time out may work.

Value

A data frame such as those returned by `ql_generate()` with the translation of the given input text in the response column.

Examples

```
## Not run:
translation <- ql_translate(
  text = "A new collection of open translation models built on Gemma 3, helping people communicate across 55 languages"
  target_language = "french"
)
translation$response

## End(Not run)
```

Index

* database

- ql_disable_db, [3](#)
- ql_enable_db, [4](#)
- ql_set_db_options, [12](#)

- ql_create_schema_enum, [2](#)
- ql_disable_db, [3](#)
- ql_disable_db(), [4](#), [13](#)
- ql_enable_db, [4](#)
- ql_enable_db(), [3](#), [13](#)
- ql_generate, [4](#)
- ql_generate(), [9](#), [10](#), [16](#)
- ql_get_api_keys, [5](#)
- ql_get_db_options, [6](#)
- ql_get_models, [7](#)
- ql_get_options, [7](#)
- ql_hash, [8](#)
- ql_hash(), [9](#), [15](#)
- ql_prompt, [2](#), [9](#)
- ql_prompt(), [5](#), [8](#)
- ql_read_images, [10](#)
- ql_request, [11](#)
- ql_set_api_keys, [12](#)
- ql_set_db_options, [12](#)
- ql_set_db_options(), [3](#), [4](#)
- ql_set_options, [13](#)
- ql_set_options(), [5](#), [11](#), [15](#)
- ql_translate, [14](#)