Package ‘rollama’

May 1, 2024

Title Communicate with 'Ollama'

Version 0.1.0

Description Wraps the 'Ollama' <`https://ollama.com`> API, which can be used to
communicate with generative large language models locally.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.1

Imports callr, cli, dplyr, httr2, jsonlite, methods, prettyunits,
purrr, rlang, tibble

Suggests base64enc, covr, knitr, rmarkdown, spelling, testthat (>=
3.0.0)

Depends R (>= 4.1.0)

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

URL `https://jbgruber.github.io/rollama/`,
`https://github.com/JBGruber/rollama`

NeedsCompilation no

Author Johannes B. Gruber [aut, cre] (<`https://orcid.org/0000-0001-9177-1772`>),
Maximilian Weber [aut, ctb] (<`https://orcid.org/0000-0002-1174-449X`>)

Maintainer Johannes B. Gruber <`JohannesB.Gruber@gmail.com`>

Repository CRAN

Date/Publication 2024-05-01 07:20:02 UTC

R topics documented:

`chat_history` *Handle conversations*

Description

Shows and deletes (`new_chat`) the local prompt and response history to start a new conversation.

Usage

```
chat_history()

new_chat()
```

Value

chat_history: tibble with chat history

new_chat: Does not return a value

`check_model_installed` *Check if one or several models are installed on the server*

Description

Check if one or several models are installed on the server

Usage

```
check_model_installed(model, auto_pull = FALSE, server = NULL)
```

Arguments

`model`	names of one or several models as character vector.
`auto_pull`	if FALSE, the default, asks before downloading models.
`server`	URL to an Ollama server (not the API). Defaults to "http://localhost:11434".

Value

invisible TRUE/FALSE

`create_model` *Create a model from a Modelfile*

Description

Create a model from a Modelfile

Usage

```
create_model(model, modelfile, server = NULL)
```

Arguments

`model`	name of the model to create
`modelfile`	either a path to a model file to be read or the contents of the model file as a character vector.
`server`	URL to an Ollama server (not the API). Defaults to "http://localhost:11434".

Details

Custom models are the way to save your system message and model parameters in a dedicated shareable way. If you use `show_model()`, you can look at the configuration of a model in the column modelfile. To get more information and a list of valid parameters, check out `https://github.com/ollama/ollama/blob/main/docs/modelfile.md`. Most options are also available through the `query` and `chat` functions, yet are not persistent over sessions.

Value

Nothing. Called to create a model on the Ollama server.

Examples

```
modelfile <- system.file("extdata", "modelfile.txt", package = "rollama")
## Not run: create_model("mario", modelfile)
modelfile <- "FROM llama3\nSYSTEM You are mario from Super Mario Bros."
## Not run: create_model("mario", modelfile)
```

embed_text *Generate Embeddings*

Description

Generate Embeddings

Usage

```
embed_text(
  text,
  model = NULL,
  server = NULL,
  model_params = NULL,
  verbose = getOption("rollama_verbose", default = interactive())
)
```

Arguments

`text`	text vector to generate embeddings for.
`model`	which model to use. See `https://ollama.com/library` for options. Default is "llama3". Set option(rollama_model = "modelname") to change default for the current session. See pull_model for more details.
`server`	URL to an Ollama server (not the API). Defaults to "http://localhost:11434".
`model_params`	a named list of additional model parameters listed in the documentation for the Modelfile.
`verbose`	Whether to print status messages to the Console (`TRUE`/`FALSE`). The default is to have status messages in interactive sessions. Can be changed with `options(rollama_verbose = FALSE)`.

Value

a tibble with embeddings.

Examples

```
## Not run:
embed_text(c("Here is an article about llamas...",
             "R is a language and environment for statistical computing and graphics."))

## End(Not run)
```

`list_models` *List models that are available locally.*

Description

List models that are available locally.

Usage

```
list_models(server = NULL)
```

Arguments

`server`	URL to an Ollama server (not the API). Defaults to "http://localhost:11434".

Value

a tibble of installed models

`ping_ollama` *Ping server to see if Ollama is reachable*

Description

Ping server to see if Ollama is reachable

Usage

```
ping_ollama(server = NULL, silent = FALSE)
```

Arguments

`server`	URL to an Ollama server (not the API). Defaults to "http://localhost:11434".
`silent`	suppress warnings and status (only return `TRUE`/`FALSE`).

Value

TRUE if server is running

`pull_model` *Pull, show and delete models*

Description

Pull, show and delete models

Usage

```
pull_model(model = NULL, server = NULL, insecure = FALSE)

show_model(model = NULL, server = NULL)

delete_model(model, server = NULL)

copy_model(model, destination = paste0(model, "-copy"), server = NULL)
```

Arguments

`model`	name of the model. Defaults to "llama3" when `NULL` (except in `delete_model`).
`server`	URL to an Ollama server (not the API). Defaults to "http://localhost:11434".
`insecure`	allow insecure connections to the library. Only use this if you are pulling from your own library during development. description
`destination`	name of the copied model.

Details

- `pull_model()`: downloads model
- `show_model()`: displays information about a local model
- `copy_model()`: creates a model with another name from an existing model
- `delete_model()`: deletes local model

Model names: Model names follow a model:tag format, where model can have an optional namespace such as example/model. Some examples are orca-mini:3b-q4_1 and llama3:70b. The tag is optional and, if not provided, will default to latest. The tag is used to identify a specific version.

Value

(invisible) a tibble with information about the model (except in `delete_model`)

Examples

```
## Not run:
model_info <- pull_model("mixtral")
# after you pull, you can get the same information with:
model_info <- show_model("mixtral")

## End(Not run)
```

`query` *Chat with a LLM through Ollama*

Description

Chat with a LLM through Ollama

Usage

```
query(
  q,
  model = NULL,
  screen = TRUE,
  server = NULL,
  images = NULL,
  model_params = NULL,
  format = NULL,
  template = NULL
)

chat(
  q,
  model = NULL,
  screen = TRUE,
  server = NULL,
  images = NULL,
  model_params = NULL,
  template = NULL
)
```

Arguments

`q`	the question as a character string or a conversation object.
`model`	which model(s) to use. See `https://ollama.com/library` for options. Default is "llama3". Set option(rollama_model = "modelname") to change default for the current session. See pull_model for more details.
`screen`	Logical. Should the answer be printed to the screen.
`server`	URL to an Ollama server (not the API). Defaults to "http://localhost:11434".
`images`	path(s) to images (for multimodal models such as llava).
`model_params`	a named list of additional model parameters listed in the documentation for the Modelfile such as temperature. Use a seed and set the temperature to zero to get reproducible results (see examples).
`format`	the format to return a response in. Currently the only accepted value is `"json"`.
`template`	the prompt template to use (overrides what is defined in the Modelfile).

Details

query sends a single question to the API, without knowledge about previous questions (only the config message is relevant). chat treats new messages as part of the same conversation until new_chat is called.

Value

an httr2 response.

Examples

```
## Not run:
# ask a single question
query("why is the sky blue?")

# hold a conversation
chat("why is the sky blue?")
chat("and how do you know that?")

# save the response to an object and extract the answer
resp <- query(q = "why is the sky blue?")
answer <- resp$message$content

# ask question about images (to a multimodal model)
images <- c("https://avatars.githubusercontent.com/u/23524101?v=4", # remote
            "/path/to/your/image.jpg") # or local images supported
query(q = "describe these images",
      model = "llava",
      images = images)

# set custom options for the model at runtime (rather than in create_model())
query("why is the sky blue?",
      model_params = list(
        num_keep = 5,
        seed = 42,
        num_predict = 100,
        top_k = 20,
        top_p = 0.9,
        tfs_z = 0.5,
        typical_p = 0.7,
        repeat_last_n = 33,
        temperature = 0.8,
        repeat_penalty = 1.2,
        presence_penalty = 1.5,
        frequency_penalty = 1.0,
        mirostat = 1,
        mirostat_tau = 0.8,
        mirostat_eta = 0.6,
        penalize_newline = TRUE,
        stop = c("\n", "user:"),
        numa = FALSE,
        num_ctx = 1024,
        num_batch = 2,
        num_gqa = 1,
        num_gpu = 1,
        main_gpu = 0,
        low_vram = FALSE,
        f16_kv = TRUE,
        vocab_only = FALSE,
        use_mmap = TRUE,
        use_mlock = FALSE,
        embedding_only = FALSE,
        rope_frequency_base = 1.1,
        rope_frequency_scale = 0.8,
        num_thread = 8
      ))

# use a seed and zero temperature to get reproducible results
query("why is the sky blue?", model_params = list(seed = 42, temperature = 0)

# this might be interesting if you want to turn off the GPU and load the
# model into the system memory (slower, but most people have more RAM than
# VRAM, which might be interesting for larger models)
query("why is the sky blue?",
       model_params = list(num_gpu = 0))

# You can use a custom prompt to override what prompt the model receives
query("why is the sky blue?",
      template = "Just say I'm a llama!")

# Asking the same question to multiple models is also supported
query("why is the sky blue?", model = c("llama3", "orca-mini"))

## End(Not run)
```

`rollama-options` *rollama Options*

Description

The behaviour of `rollama` can be controlled through `options()`. Specifically, the options below can be set.

Details

rollama_server This controls the default server where Ollama is expected to run. It assumes that you are running Ollama locally in a Docker container.

`"http://localhost:11434"`

rollama_model The default model is llama3, which is a good overall option with reasonable performance and size for most tasks. You can change the model in each function call or globally with this option.

`"llama3"`

default: **rollama_verbose** Whether the package tells users what is going on, e.g., showing a spinner while the models are thinking or showing the download speed while pulling models. Since this adds some complexity to the code, you might want to disable it when you get errors (it won't fix the error, but you get a better error trace).

`TRUE`

default: **rollama_config** The default configuration or system message. If NULL, the system message defined in the used model is employed.

None

Examples

default: `options(rollama_config = "You make answers understandable to a 5 year old")`

Index