Package: ollamar (via r-universe)

October 11, 2024

Title 'Ollama' Language Models

Version 1.2.1.9000

Description An interface to easily run local language models with 'Ollama' <https://ollama.com> server and API endpoints (see <https://github.com/ollama/ollama/blob/main/docs/api.md> for details). It lets you run open-source large language models locally on your machine.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports base64enc, crayon, glue, httr2, jsonlite, tibble

BugReports https://github.com/hauselin/ollama-r/issues

URL https://hauselin.github.io/ollama-r/, https://github.com/hauselin/ollama-r

VignetteBuilder knitr

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

RemoteUrl https://github.com/hauselin/ollama-r

RemoteRef HEAD

RemoteSha e45b5c58c3d8207e197e3065fbb1dba55b2504af

Contents

`append_message` *Append message to a list*

Description

Appends a message (add to end of a list) to a list of messages. The role and content will be converted to a list and appended to the input list.

Usage

```
append_message(content, role = "user", x = NULL, ...)
```

Arguments

content	The content of the message.
role	The role of the message. Can be "user", "system", "assistant". Default is "user".
x	A list of messages. Default is NULL.
...	Additional arguments such as images.

Value

A list of messages with the new message appended.

Examples

```
append_message("user", "Hello")
append_message("system", "Always respond nicely")
```

---

chat *Generate a chat completion with message history*

Description

Generate a chat completion with message history

Usage

```
chat(
  model,
  messages,
  tools = list(),
  stream = FALSE,
  keep_alive = "5m",
  output = c("resp", "jsonlist", "raw", "df", "text", "req"),
  endpoint = "/api/chat",
  host = NULL,
  ...
)
```

Arguments

model	A character string of the model name such as "llama3".
messages	A list with list of messages for the model (see examples below).
tools	Tools for the model to use if supported. Requires stream = FALSE. Default is an empty list.
stream	Enable response streaming. Default is FALSE.
keep_alive	The duration to keep the connection alive. Default is "5m".

output	The output format. Default is "resp". Other options are "jsonlist", "raw", "df", "text", "req" (httr2_request object).
endpoint	The endpoint to chat with the model. Default is "/api/chat".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.
...	Additional options to pass to the model.

Value

A response in the format specified in the output parameter.

References

API documentation

Examples

```
# one message
messages <- list(
    list(role = "user", content = "How are you doing?")
)
chat("llama3", messages) # returns response by default
chat("llama3", messages, output = "text") # returns text/vector
chat("llama3", messages, temperature = 2.8) # additional options
chat("llama3", messages, stream = TRUE) # stream response
chat("llama3", messages, output = "df", stream = TRUE) # stream and return dataframe

# multiple messages
messages <- list(
    list(role = "user", content = "Hello!"),
    list(role = "assistant", content = "Hi! How are you?"),
    list(role = "user", content = "Who is the prime minister of the uk?"),
    list(role = "assistant", content = "Rishi Sunak"),
    list(role = "user", content = "List all the previous messages.")
)
chat("llama3", messages, stream = TRUE)

# image
image_path <- file.path(system.file("extdata", package = "ollamar"), "image1.png")
messages <- list(
   list(role = "user", content = "What is in the image?", images = image_path)
)
chat("benzie/llava-phi-3", messages, output = 'text')
```

check_options *Check if a vector of options are valid*

Description

Check if a vector of options are valid

Usage

```
check_options(opts = NULL)
```

Arguments

opts	A vector of options to check.

Value

Returns a list with two elements: valid_options and invalid_options.

Examples

```
check_options(c("mirostat", "invalid_option"))
check_options(c("mirostat", "num_predict"))
```

check_option_valid *Check if an option is valid*

Description

Check if an option is valid

Usage

```
check_option_valid(opt)
```

Arguments

opt	An option (character) to check.

Value

Returns TRUE if the option is valid, FALSE otherwise.

Examples

```
check_option_valid("mirostat")
check_option_valid("invalid_option")
```

copy *Copy a model*

Description

Creates a model with another name from an existing model.

Usage

```
copy(source, destination, endpoint = "/api/copy", host = NULL)
```

Arguments

source	The name of the model to copy.
destination	The name for the new model.
endpoint	The endpoint to copy the model. Default is "/api/copy".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A httr2 response object.

References

API documentation

Examples

```
copy("llama3", "llama3_copy")
delete("llama3_copy")  # delete the model was just got copied
```

create *Create a model from a Modelfile*

Description

It is recommended to set `modelfile` to the content of the Modelfile rather than just set path.

Usage

```
create(
  name,
  modelfile = NULL,
  stream = FALSE,
  path = NULL,
  endpoint = "/api/create",
  host = NULL
)
```

Arguments

name	Name of the model to create.
modelfile	Contents of the Modelfile as character string. Default is NULL.
stream	Enable response streaming. Default is FALSE.
path	The path to the Modelfile. Default is NULL.
endpoint	The endpoint to create the model. Default is "/api/create".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A response in the format specified in the output parameter.

References

API documentation

Examples

```
create("mario", "FROM llama3\nSYSTEM You are mario from Super Mario Bros.")
generate("mario", "who are you?", output = "text")  # model should say it's Mario
delete("mario")  # delete the model created above
```

---

create_message *Create a message*

Description

Create a message

Usage

```
create_message(content, role = "user", ...)
```

Arguments

content	The content of the message.
role	The role of the message. Can be "user", "system", "assistant". Default is "user".
...	Additional arguments such as images.

Value

A list of messages.

Examples

```
create_message("Hello", "user")
create_message("Always respond nicely", "system")
create_message("I am here to help", "assistant")
```

---

create_messages *Create a list of messages*

Description

Create messages for `chat()` function.

Usage

```
create_messages(...)
```

Arguments

...	A list of messages, each of list class.

Value

A list of messages, each of list class.

Examples

```
messages <- create_messages(
    create_message("be nice", "system"),
    create_message("tell me a 3-word joke")
)

messages <- create_messages(
    list(role = "system", content = "be nice"),
    list(role = "user", content = "tell me a 3-word joke")
)
```

create_request *Create a httr2 request object*

Description

Creates a httr2 request object with base URL, headers and endpoint. Used by other functions in the package and not intended to be used directly.

Usage

```
create_request(endpoint, host = NULL)
```

Arguments

endpoint	The endpoint to create the request
host	The base URL to use. Default is NULL, which uses http://127.0.0.1:11434

Value

A httr2 request object.

Examples

```
create_request("/api/tags")
create_request("/api/chat")
create_request("/api/embeddings")
```

delete *Delete a model and its data*

Description

Delete a model from your local machine that you downloaded using the pull() function. To see which models are available, use the list_models() function.

Usage

```
delete(name, endpoint = "/api/delete", host = NULL)
```

Arguments

name	A character string of the model name such as "llama3".
endpoint	The endpoint to delete the model. Default is "/api/delete".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A httr2 response object.

References

API documentation

Examples

```
## Not run:
delete("llama3")

## End(Not run)
```

`delete_message` *Delete a message in a specified position from a list*

Description

Delete a message using positive or negative positions/indices. Negative positions/indices can be used to refer to elements/messages from the end of the sequence.

Usage

```
delete_message(x, position = -1)
```

Arguments

`x`	A list of messages.
`position`	The position of the message to delete.

Value

A list of messages with the message at the specified position removed.

Examples

```
messages <- list(
    list(role = "system", content = "Be friendly"),
    list(role = "user", content = "How are you?")
)
delete_message(messages, 1) # delete first message
delete_message(messages, -2) # same as above (delete first message)
delete_message(messages, 2) # delete second message
delete_message(messages, -1) # same as above (delete second message)
```

`embed` *Generate embedding for inputs*

Description

Supercedes the `embeddings()` function.

Usage

```
embed(
  model,
  input,
  truncate = TRUE,
  normalize = TRUE,
  keep_alive = "5m",
  endpoint = "/api/embed",
  host = NULL,
  ...
)
```

Arguments

`model`	A character string of the model name such as "llama3".
`input`	A vector of characters that you want to get the embeddings for.
`truncate`	Truncates the end of each input to fit within context length. Returns error if FALSE and context length is exceeded. Defaults to TRUE.
`normalize`	Normalize the vector to length 1. Default is TRUE.
`keep_alive`	The time to keep the connection alive. Default is "5m" (5 minutes).
`endpoint`	The endpoint to get the vector embedding. Default is "/api/embeddings".
`host`	The base URL to use. Default is NULL, which uses Ollama's default base URL.
`...`	Additional options to pass to the model.

Value

A numeric matrix of the embedding. Each column is the embedding for one input.

References

API documentation

Examples

```
embed("nomic-embed-text:latest", "The quick brown fox jumps over the lazy dog.")
# pass multiple inputs
embed("nomic-embed-text:latest", c("Good bye", "Bye", "See you."))
# pass model options to the model
embed("nomic-embed-text:latest", "Hello!", temperature = 0.1, num_predict = 3)
```

`embeddings` *Generate embeddings for a single prompt - deprecated in favor of* `embed()`

Description

This function will be deprecated over time and has been superceded by `embed()`. See `embed()` for more details.

Usage

```
embeddings(
  model,
  prompt,
  normalize = TRUE,
  keep_alive = "5m",
  endpoint = "/api/embeddings",
  host = NULL,
  ...
)
```

Arguments

`model`	A character string of the model name such as "llama3".
`prompt`	A character string of the prompt that you want to get the vector embedding for.
`normalize`	Normalize the vector to length 1. Default is TRUE.
`keep_alive`	The time to keep the connection alive. Default is "5m" (5 minutes).
`endpoint`	The endpoint to get the vector embedding. Default is "/api/embeddings".
`host`	The base URL to use. Default is NULL, which uses Ollama's default base URL.
`...`	Additional options to pass to the model.

Value

A numeric vector of the embedding.

References

API documentation

Examples

```
embeddings("nomic-embed-text:latest", "The quick brown fox jumps over the lazy dog.")
# pass model options to the model
embeddings("nomic-embed-text:latest", "Hello!", temperature = 0.1, num_predict = 3)
```

---

encode_images_in_messages *Encode images in messages to base64 format*

Description

Encode images in messages to base64 format

Usage

```
encode_images_in_messages(messages)
```

Arguments

messages	A list of messages, each of list class. Generally used in the `chat()` function.

Value

A list of messages with images encoded in base64 format.

Examples

```
image <- file.path(system.file("extdata", package = "ollamar"), "image1.png")
message <- create_message(content = "what is in the image?", images = image)
message_updated <- encode_images_in_messages(message)
```

---

generate *Generate a response for a given prompt*

Description

Generate a response for a given prompt

Usage

```
generate(
  model,
  prompt,
  suffix = "",
  images = "",
  system = "",
  template = "",
  context = list(),
  stream = FALSE,
  raw = FALSE,
  keep_alive = "5m",
  output = c("resp", "jsonlist", "raw", "df", "text", "req"),
  endpoint = "/api/generate",
  host = NULL,
  ...
)
```

Arguments

model	A character string of the model name such as "llama3".
prompt	A character string of the prompt like "The sky is..."
suffix	A character string after the model response. Default is "".
images	A path to an image file to include in the prompt. Default is "".
system	A character string of the system prompt (overrides what is defined in the Modelfile). Default is "".
template	A character string of the prompt template (overrides what is defined in the Modelfile). Default is "".
context	A list of context from a previous response to include previous conversation in the prompt. Default is an empty list.
stream	Enable response streaming. Default is FALSE.
raw	If TRUE, no formatting will be applied to the prompt. You may choose to use the raw parameter if you are specifying a full templated prompt in your request to the API. Default is FALSE.
keep_alive	The time to keep the connection alive. Default is "5m" (5 minutes).
output	A character vector of the output format. Default is "resp". Options are "resp", "jsonlist", "raw", "df", "text", "req" (httr2_request object).
endpoint	The endpoint to generate the completion. Default is "/api/generate".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.
...	Additional options to pass to the model.

Value

A response in the format specified in the output parameter.

References

API documentation

Examples

```
# text prompt
generate("llama3", "The sky is...", stream = FALSE, output = "df")
# stream and increase temperature
generate("llama3", "The sky is...", stream = TRUE, output = "text", temperature = 2.0)

# image prompt
# something like "image1.png"
image_path <- file.path(system.file("extdata", package = "ollamar"), "image1.png")
# use vision or multimodal model such as https://ollama.com/benzie/llava-phi-3
generate("benzie/llava-phi-3:latest", "What is in the image?", images = image_path, output = "text")
```

---

image_encode_base64 *Read image file and encode it to base64*

Description

Read image file and encode it to base64

Usage

```
image_encode_base64(image_path)
```

Arguments

image_path	The path to the image file.

Value

A base64 encoded string.

Examples

```
image_path <- file.path(system.file("extdata", package = "ollamar"), "image1.png")
substr(image_encode_base64(image_path), 1, 5) # truncate output
```

insert_message *Insert message into a list at a specified position*

Description

Inserts a message at a specified position in a list of messages. The role and content are converted to a list and inserted into the input list at the given position.

Usage

```
insert_message(content, role = "user", x = NULL, position = -1, ...)
```

Arguments

content	The content of the message.
role	The role of the message. Can be "user", "system", "assistant". Default is "user".
x	A list of messages. Default is NULL.
position	The position at which to insert the new message. Default is -1 (end of list).
...	Additional arguments such as images.

Value

A list of messages with the new message inserted at the specified position.

Examples

```
messages <- list(
    list(role = "system", content = "Be friendly"),
    list(role = "user", content = "How are you?")
)
insert_message("INSERT MESSAGE AT THE END", "user", messages)
insert_message("INSERT MESSAGE AT THE BEGINNING", "user", messages, 2)
```

list_models *List models that are available locally*

Description

List models that are available locally

Usage

```
list_models(
  output = c("df", "resp", "jsonlist", "raw", "text"),
  endpoint = "/api/tags",
  host = NULL
)
```

Arguments

output	The output format. Default is "df". Other options are "resp", "jsonlist", "raw", "text".
endpoint	The endpoint to get the models. Default is "/api/tags".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A response in the format specified in the output parameter.

References

API documentation

Examples

```
list_models() # returns dataframe
list_models("df") # returns dataframe
list_models("resp") # httr2 response object
list_models("jsonlist")
list_models("raw")
```

---

model_avail *Check if model is available locally*

Description

Check if model is available locally

Usage

```
model_avail(model)
```

Arguments

model	A character string of the model name such as "llama3".

Value

A logical value indicating if the model exists.

Examples

```
model_avail("codegemma:7b")
model_avail("abc")
model_avail("llama3")
```

model_options *Model options*

Description

Model options

Usage

```
model_options
```

Format

An object of class `list` of length 13.

ohelp *Chat with a model in real-time in R console*

Description

Chat with a model in real-time in R console

Usage

```
ohelp(model = "codegemma:7b", ...)
```

Arguments

`model`	A character string of the model name such as "llama3". Defaults to "codegemma:7b" which is a decent coding model as of 2024-07-27.
`...`	Additional options. No options are currently available at this time.

Value

Does not return anything. It prints the conversation in the console.

Examples

```
ohelp(first_prompt = "quit")
# regular usage: ohelp()
```

package_config *Package configuration*

Description

Package configuration

Usage

```
package_config
```

Format

An object of class `list` of length 3.

prepend_message *Prepend message to a list*

Description

Prepends a message (add to beginning of a list) to a list of messages. The role and content will be converted to a list and prepended to the input list.

Usage

```
prepend_message(content, role = "user", x = NULL, ...)
```

Arguments

`content`	The content of the message.
`role`	The role of the message. Can be "user", "system", "assistant".
`x`	A list of messages. Default is NULL.
`...`	Additional arguments such as images.

Value

A list of messages with the new message prepended.

Examples

```
prepend_message("user", "Hello")
prepend_message("system", "Always respond nicely")
```

ps *List models that are currently loaded into memory*

Description

List models that are currently loaded into memory

Usage

```
ps(
  output = c("df", "resp", "jsonlist", "raw", "text"),
  endpoint = "/api/ps",
  host = NULL
)
```

Arguments

output	The output format. Default is "df". Supported formats are "df", "resp", "jsonlist", "raw", and "text".
endpoint	The endpoint to list the running models. Default is "/api/ps".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A response in the format specified in the output parameter.

References

API documentation

Examples

```
ps("text")
```

pull *Pull/download a model from the Ollama library*

Description

See https://ollama.com/library for a list of available models. Use the list_models() function to get the list of models already downloaded/installed on your machine. Cancelled pulls are resumed from where they left off, and multiple calls will share the same download progress.

Usage

```
pull(
  name,
  stream = FALSE,
  insecure = FALSE,
  endpoint = "/api/pull",
  host = NULL
)
```

Arguments

name	A character string of the model name to download/pull, such as "llama3".
stream	Enable response streaming. Default is FALSE.
insecure	Allow insecure connections Only use this if you are pulling from your own library during development. Default is FALSE.
endpoint	The endpoint to pull the model. Default is "/api/pull".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A httr2 response object.

References

API documentation

Examples

```
pull("llama3")
pull("all-minilm", stream = FALSE)
```

---

push *Push or upload a model to a model library*

Description

Push or upload a model to an Ollama model library. Requires registering for ollama.ai and adding a public key first.

Usage

```
push(
  name,
  insecure = FALSE,
  stream = FALSE,
  output = c("resp", "jsonlist", "raw", "text", "df"),
  endpoint = "/api/push",
  host = NULL
)
```

Arguments

name	A character string of the model name to upload, in the form of <namespace>/<model>:<tag>
insecure	Allow insecure connections. Only use this if you are pushing to your own library during development. Default is FALSE.
stream	Enable response streaming. Default is FALSE.
output	The output format. Default is "resp". Other options are "jsonlist", "raw", "text", and "df".
endpoint	The endpoint to push the model. Default is "/api/push".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A httr2 response object.

References

API documentation

Examples

```
push("mattw/pygmalion:latest")
```

---

resp_process *Process httr2 response object*

Description

Process httr2 response object

Usage

```
resp_process(resp, output = c("df", "jsonlist", "raw", "resp", "text"))
```

Arguments

resp	A httr2 response object.
output	The output format. Default is "df". Other options are "jsonlist", "raw", "resp" (httr2 response object), "text"

Value

A data frame, json list, raw or httr2 response object.

Examples

```
resp <- list_models("resp")
resp_process(resp, "df") # parse response to dataframe/tibble
resp_process(resp, "jsonlist") # parse response to list
resp_process(resp, "raw") # parse response to raw string
resp_process(resp, "resp") # return input response object
resp_process(resp, "text") # return text/character vector
```

---

search_options *Search for options based on a query*

Description

Search for options based on a query

Usage

```
search_options(query)
```

Arguments

query	A query (character) to search for in the options.

Value

Returns a list of matching options.

Examples

```
search_options("learning rate")
search_options("tokens")
search_options("invalid query")
```

show *Show model information*

Description

Model information includes details, modelfile, template, parameters, license, system prompt.

Usage

```
show(
  name,
  verbose = FALSE,
  output = c("jsonlist", "resp", "raw"),
  endpoint = "/api/show",
  host = NULL
)
```

Arguments

name	Name of the model to show
verbose	Returns full data for verbose response fields. Default is FALSE.
output	The output format. Default is "jsonlist". Other options are "resp", "raw".
endpoint	The endpoint to show the model. Default is "/api/show".
host	The base URL to use. Default is NULL, which uses Ollama's default base URL.

Value

A response in the format specified in the output parameter.

References

API documentation

Examples

```
# show("llama3") # returns jsonlist
show("llama3", output = "resp") # returns response object
```

test_connection *Test connection to Ollama server*

Description

`test_connection()` tests whether the Ollama server is running or not.

Usage

```
test_connection(url = "http://localhost:11434")
```

Arguments

`url`	The URL of the Ollama server. Default is http://localhost:11434

Value

A httr2 response object.

Examples

```
test_connection()
test_connection("http://localhost:11434") # default url
test_connection("http://127.0.0.1:11434")
```

validate_message *Validate a message*

Description

Validate a message to ensure it has the required fields and the correct data types for the `chat()` function.

Usage

```
validate_message(message)
```

Arguments

`message`	A list with a single message of list class.

Value

TRUE if message is valid, otherwise an error is thrown.

Examples

```
validate_message(create_message("Hello"))
validate_message(list(role = "user", content = "Hello"))
```

validate_messages *Validate a list of messages*

Description

Validate a list of messages to ensure they have the required fields and the correct data types for the `chat()` function.

Usage

```
validate_messages(messages)
```

Arguments

`messages`	A list of messages, each of list class.

Value

TRUE if all messages are valid, otherwise warning messages are printed and FALSE is returned.

Examples

```
validate_messages(create_messages(
   create_message("Be friendly", "system"),
   create_message("Hello")
))
```

validate_options *Validate additional options or parameters provided to the API call*

Description

Validate additional options or parameters provided to the API call

Usage

```
validate_options(...)
```

Arguments

`...`	Additional options or parameters provided to the API call

Value

TRUE if all additional options are valid, FALSE otherwise

Examples

```
validate_options(mirostat = 1, mirostat_eta = 0.2, num_ctx = 1024)
validate_options(mirostat = 1, mirostat_eta = 0.2, invalid_opt = 1024)
```

Index