Package 'TheOpenAIR'

April 27, 2023

Type Package

Title Integrate 'OpenAI' Large Language Models into Your 'R' Workflows

Version 0.1.0

Maintainer Ulrich Matter <umatter@protonmail.com>

Description Utilizing the 'OpenAI' API as the back end (<https://platform.openai.com/docs/api-reference>), 'TheOpenAIR' offers 'R' wrapper functions for the 'ChatGPT' endpoint and several high-level functions that enable the integration of 'ChatGPT' capabilities in diverse data-related tasks, such as data cleansing and automated analytics script generation.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Suggests readr, jsonlite, tidyr, dplyr, lubridate, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

Config/reticulate list(packages = list(list(package = ``tiktoken'')))

Imports reticulate, magrittr, tibble, methods, stringr, stringi, data.table, httr, R.utils, xml2, utils, cli, rstudioapi

URL http://openair-lib.org/

Depends R (>= 2.10)

NeedsCompilation no

Author Ulrich Matter [aut, cre, cph], Jonathan Chassot [aut]

Repository CRAN

Date/Publication 2023-04-27 12:32:30 UTC

R topics documented:

`add_roxygen` *Add Roxygen2 documentation to an R function*

Description

This function adds Roxygen2 documentation to an R function.

Usage

```
add_roxygen(file)
```

Arguments

`file`	A character string indicating the path to the file containing the R function.

Value

If the path provided is a character string, this function returns the documented function as a character string. If the input is a file path, this function returns the path of the file to which documentation was added to the file.

Author(s)

Ulrich Matter umatter@protonmail.com

add_roxygen_prompt *Input: message content template for add_roxygen()*

Description

This message content serves as user input for the add_roxygen function.

Usage

```
add_roxygen_prompt
```

Format

'add_roxygen_prompt' A data frame with 2 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

add_to_chatlog *Add data to a chat log*

Description

This function adds data to the in-memory chat log, which is in the global environment. The data can be any R object, such as a vector, list, or data frame. The function supports adding data to the chat log by providing either a data.frame or a chatlog object.

Usage

```
add_to_chatlog(msgs, chatlog_id = NULL)
```

Arguments

msgs	A chatlog object or a data.frame containing the messages to be added to the chat log.
chatlog_id	The id of the chatlog object. Required when the provided msgs is a data.frame.

Value

The updated chatlog object with the newly added messages.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
## Not run:
# Add a data frame of messages to an existing chat log
chatlog_df <- data.frame(
user = c("user1", "user2"),
message = c("Hello!", "Hi!")
)
updated_chatlog <- add_to_chatlog(chatlog_df, "existing_chatlog_id")

# Add messages from one chat log to another
chatlog1 <- create_chatlog("chatlog1")
chatlog2 <- create_chatlog("chatlog2")
chatlog1 <- add_message(chatlog1, "user1", "Hello!")
chatlog2 <- add_message(chatlog2, "user2", "Hi!")
merged_chatlog <- add_to_chatlog(chatlog1, chatlog2@chatlog_id)

## End(Not run)
```

---

chat *Start or continue a chat conversation*

Description

This function starts or continues a chat conversation by adding the user's message to the conversation. If the conversation does not exist, a new one will be initiated. The response can be displayed in the console, returned as a character vector, or returned as a full response object from the ChatGPT API.

Usage

```
chat(
  message,
  chatlog_id = ".__CURRENTCHAT__",
  output = "message_to_console",
  ...
)
```

Arguments

message	A character string representing the message to be added to the chat conversation.
chatlog_id	A character string representing the ID of the chat conversation to start or continue. Default is ".__CURRENTCHAT__".
output	A character string indicating the output format of the response. Default is "message_to_console". Valid options are "message_to_console", "message", or "response_object".
...	Additional arguments to be passed to the 'chat_completion' function.

Value

Depending on the value of the 'output' argument, this function returns one of the following: * "message_to_console": a message containing the response text is printed to the console (default). * "message": the response text as a character vector. * "response_object": the full response object from the ChatGPT API.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
## Not run:
# Start a new chat and print the response to the console
chat("What's the weather like today?")

# Continue the chat, but return the response as a character vector
response_text <- chat("What about tomorrow's weather?", output = "message")

## End(Not run)
```

---

chatlog-class *chatlog class*

Description

chatlog class

Slots

`messages` The message data of the object

`chatlog_id` The chatlog's ID

---

chat_completion *Generate Text Using the OpenAI API's Chat Endpoint*

Description

This function generates natural language text in a conversational style using the OpenAI API's chat endpoint. It takes a series of chat messages as input, either as a data.frame or a chatlog object, and generates a text completion based on the conversation history and the specified model parameters.

Usage

```
chat_completion(
  msgs,
  model = "gpt-3.5-turbo",
  temperature = NULL,
  max_tokens = NULL,
  n = NULL,
  stop = NULL,
  presence_penalty = NULL,
  frequency_penalty = NULL,
  best_of = NULL,
  logit_bias = NULL,
  stream = FALSE,
  top_p = NULL,
  user = NULL
)
```

Arguments

msgs	A data.frame containing the chat history to generate text from or a chatlog object.
model	A character string specifying the ID of the model to use. The default value is "gpt-3.5-turbo".
temperature	An optional numeric scalar specifying the sampling temperature to use.
max_tokens	An optional integer scalar specifying the maximum number of tokens to generate in the text.
n	An optional integer scalar specifying the number of text completions to generate.
stop	An optional character string or character vector specifying one or more stop sequences to use when generating the text.
presence_penalty	An optional numeric scalar specifying the presence penalty to use when generating the text. The default value is NULL.
frequency_penalty	An optional numeric scalar specifying the frequency penalty to use when generating the text. The default value is NULL.
best_of	An optional integer scalar specifying the number of completions to generate and return the best one. The default value is NULL.
logit_bias	An optional named numeric vector specifying the logit bias to use for each token in the generated text.
stream	An optional logical scalar specifying whether to use the streaming API. The default value is FALSE.
top_p	An optional numeric scalar specifying the top p sampling ratio. The default value is NULL.
user	A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.

Value

A character vector containing the generated text(s).

Author(s)

Ulrich Matter umatter@protonmail.com

See Also

https://platform.openai.com/docs/ for more information on the OpenAI API.

Examples

```
## Not run:
openai_api_key("your_api_key_here")
msgs_df <- data.frame(role=c("system",
"user",
"assistant",
"user"),
content=c("You are a helpful assistant",
"Who won the world series in 2020?",
"The Los Angeles Dodgers won the World Series in 2020.",
"Where was it played?"))
chat_completion(msgs_df)

## End(Not run)
```

---

clean_output *Clean Output From Code Responses*

Description

This function extracts the content from a given text string that is enclosed between the '“‘' markers. It can be used to extract any kind of code or text content.

Usage

```
clean_output(text)
```

Arguments

text	A character string containing the code or text content with '“‘' markers.

Value

A character string containing the extracted code or text content.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
code_text <- "```
example_code <- function(x) {
  return(x * 2)
}
```"
clean_output(code_text)
```

---

## clear_chatlog *Clear a chat log*

---

### Description

This function clears a chat log, leaving only the initial (system) message.

### Usage

```
clear_chatlog(chatlog_id = ".__CURRENTCHAT__")
```

Arguments

chatlog_id	character string indicating the name of the chat log to clear. Default is ".__CURRENTCHAT__".

Value

This function does not return anything.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
## Not run:
# Clear the current chat log
clear_chatlog()

## End(Not run)
```
```

contains_roxygen *Check if a text file or character string contains Roxygen2 documentation*

Description

This function parses a text file or a character string and returns TRUE if it contains Roxygen2 documentation. It uses a regex pattern to identify possible Roxygen2 documentation lines.

Usage

```
contains_roxygen(input)
```

Arguments

input	A character string specifying the path to the text file, or a character string containing the text to be checked for Roxygen2 documentation.

Value

A logical value (TRUE or FALSE) indicating whether the input contains Roxygen2 documentation.

contains_r_func *Check if a text file or character string contains an R function definition*

Description

This function parses a text file or a character string and returns TRUE if it contains a valid R function definition. It uses a regex pattern to identify possible R function definitions and then attempts to parse the matched lines as R code.

Usage

```
contains_r_func(input)
```

Arguments

input	A character string specifying the path to the text file, or a character string containing the text to be checked for an R function definition.

Value

A logical value (TRUE or FALSE) indicating whether the input contains a valid R function definition.

Examples

```
# Create a temporary file with an R function definition
temp_file <- tempfile(fileext = ".R")
writeLines("example_function <- function(x) {\n  return(x * 2)\n}", temp_file)

# Check if the temporary file contains an R function definition
result <- contains_r_func(temp_file)
print(result) # Should print TRUE

# Check if a character string contains an R function definition
result <- contains_r_func("example_function <- function(x) { return(x * 2) }")
print(result) # Should print TRUE

# Remove the temporary file
file.remove(temp_file)
```

`count_tokens` *Count the number of tokens in a text string*

Description

This function takes a file path, URL or character string as input and returns the number of tokens in the text. Tokens are defined as words and/or special characters.

Usage

```
count_tokens(text)
```

Arguments

`text`	A file path, URL or character string representing the text to be tokenized.

Value

An integer representing the number of tokens in the text.

Examples

```
## Not run:
# Example 1: File path
test_file_path <- tempfile()
writeLines("This is a test.", test_file_path)
expect_equal(count_tokens(test_file_path), 5)
file.remove(test_file_path)

# Example 2: URL
url <- "https://www.gutenberg.org/files/2701/2701-0.txt"
count_tokens(url)
```

```
# Example 3: Character string
text <- "This is a test string."
count_tokens(text)

## End(Not run)
```

`created` *Extract Created DateTime from OpenAI API response*

Description

This function extracts the date-time string of when the response was created from the parsed HTTP response of an API call to the OpenAI chat completions endpoint.

Usage

```
created(response)
```

Arguments

`response`	a list object representing the HTTP response

Value

a Date object representing the date-time string of when the response was created

Author(s)

Ulrich Matter umatter@protonmail.com

`df_to_csv` *Convert a data.frame to a CSV-formatted character string*

Description

The ‘df_to_csv‘ function takes a data.frame as input and returns a character string representing the content of the original data.frame formatted as a CSV file. The resulting CSV-formatted string can be written to a file or further processed as needed.

Usage

```
df_to_csv(df)
```

Arguments

`df`	A data.frame to be converted to a CSV-formatted character string.

Value

A character string representing the data values in the input data.frame formatted as a CSV file.

Examples

```
# Create a data.frame
example_data <- data.frame(
  Name = c("Alice", "Bob", "Carol"),
  Age = c(30, 25, 28),
  Height = c(168, 175, 162),
  stringsAsFactors = FALSE
)

# Convert the data.frame to a CSV-formatted character string
csv_string <- df_to_csv(example_data)
cat(csv_string)
```

---

edit_code *Edit code based on user's input*

Description

The 'edit_code' function prompts the user to provide a plain English description of how the code in a given file should be modified. The function then generates the modified code based on the user's input and writes it back to the specified file.

Usage

```
edit_code(filename, chatlog_id = ".__CURRENTCODEFILE__")
```

Arguments

filename	A character string representing the name of the file containing the code to be edited.
chatlog_id	An optional character string representing the chatlog ID. Defaults to ".__CURRENTCODEFILE__". This ID is used to maintain the conversation history with the AI.

Value

Returns the name of the file containing the modified code.

Examples

```
## Not run:
# Edit code in an existing file based on user input
modified_code_file <- edit_code("example_code.R")

# Check the content of the modified code file
cat(readLines(modified_code_file))

## End(Not run)
```

`extract_blocks` *Extract blocks of a specified type from a list of blocks*

Description

Extract blocks of a specified type from a list of blocks

Usage

```
extract_blocks(block_list, block_type)
```

Arguments

`block_list`	A list of blocks
`block_type`	The type of blocks to be extracted

Value

A list of blocks of the specified type

Author(s)

Jonathan Chassot

Examples

```
## Not run:
# Example 1: Extract 'code' blocks
blocks <- list(
  list(type = "text", content = "Hello world!"),
  list(type = "code", content = "print('Hello world!')")
)
extract_blocks(blocks, "code")
# Output:
# [[1]]
# $type
# [1] "code"
#
# $content
```

```
# [1] "print('Hello world!')"

# Example 2: Extract 'text' blocks
blocks <- list(
  list(type = "text", content = "Hello world!"),
  list(type = "code", content = "print('Hello world!')")
)
extract_blocks(blocks, "text")
# Output:
# [[1]]
# $type
# [1] "text"
#
# $content
# [1] "Hello world!"

## End(Not run)
```

`extract_blocks_content` *Extract the content parts of blocks*

Description

This function takes a list of blocks and returns a list of their content parts.

Usage

```
extract_blocks_content(block_list)
```

Arguments

`block_list`	A list of blocks to extract the content from

Value

A list of content parts

`extract_entities` *Extract Entities from a Text*

Description

This function takes a character string or a path to a text file and returns a tibble describing the entities found in the text. The type of entities to be searched for and extracted can be defined by the user.

Usage

```
extract_entities(
  text,
  entity_types = c("locations", "persons", "organizations"),
  batch_size = NULL
)
```

Arguments

text	A character string containing the text to be processed, or a path to a text file
entity_types	A character vector containing names of entity types to be extracted. Defaults to c("locations", "persons", "organizations").
batch_size	An integer indicating the size of each batch, if the text input is supposed to be processed in batches. Set this to NULL to process all at once.

Value

A tibble

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
 ## Not run:
extract_entities("Hello, how are you?")
extract_entities("path/to/text/file.txt",  batch_size = 100)

## End(Not run)
```

---

extract_entities_prompt

Input: message content template for extract_entities()

Description

This message content serves as user input for the extract_entities function.

Usage

```
extract_entities_prompt
```

Format

‘extract_entities_prompt‘ A data frame with 2 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

`extract_roxygen2` *Extract roxygen2 documentation lines from a function definition*

Description

This function takes a character vector containing the lines of a function definition and returns a character string containing only the lines belonging to the roxygen2 documentation (lines starting with "#'").

Usage

```
extract_roxygen2(func_def)
```

Arguments

`func_def`	A character vector containing the lines of a function definition

Value

A character string containing the roxygen2 documentation lines

Examples

```
func_def <- c(
  "#' Extracts object from a response list",
  "#'",
  "#' @export",
  "object <- function(response) {",
  "  if (!is.list(response)) {",
  "    stop('Invalid response format. Expected list object.')",
  "  }",
  "}")

roxygen2_docu <- extract_roxygen2(func_def)
print(roxygen2_docu)
```

extract_r_code *Extract R code and comments from a given input string.*

Description

This function takes an input string, detects R code and comments, and returns a character vector containing the R code and comments. The input string is split into lines based on newline characters, and each line is examined for R code and comment patterns. Only the lines that match either of these patterns are returned.

Usage

```
extract_r_code(input_string)
```

Arguments

input_string	A character string containing R code and comments, mixed with other text. The string may contain multiple lines separated by newline characters.

Value

A character vector containing R code and comment lines extracted from the input string. Each element in the vector corresponds to one line of code or a comment.

Examples

```
example_string <-
"This is a text string with R code and comments.\n
# A comment\n
x <- 5\
ny = 10\n
z <- x + y\n
Another line of text."
extract_r_code(example_string)
```

get_chatlog *Get the current chatlog*

Description

This function extracts the current chatlog of a given chat from the chat environment.

Usage

```
get_chatlog(x = ".__CURRENTCHAT__")
```

Arguments

x	either a chatlog object r, or a character string representing the id of a chatlog (the default is ".__CURRENTCHAT__", the current log of the chat()-function)

Value

a chatlog object

Author(s)

Ulrich Matter umatter@protonmail.com

---

id *Extract ID from OpenAI API response*

Description

This function extracts the ID from the parsed HTTP response of an API call to the OpenAI chat completions endpoint.

Usage

```
id(response)
```

Arguments

response	a list object representing the HTTP response

Value

a character string representing the ID from the response

Author(s)

Ulrich Matter umatter@protonmail.com

`initialize_messages` *Initialize a new messages object for OpenAI API chat completions*

Description

This function initializes a new messages object as a template for messages sent to the OpenAI API https://api.openai.com/v1/chat/completions endpoint. By default, the template contains a single message from the system to the user with the initial content "You are a helpful assistant.", but you can customize the content by specifying a different value for the "initial_system_content" parameter.

Usage

```
initialize_messages(
  initial_role = "system",
  initial_content = "You are a helpful assistant."
)
```

Arguments

`initial_role`	A character string representing the role issueing the initial content (per default: "system")
`initial_content`	A character string representing the initial message from the system

Value

A data frame containing a single message from the system to the user, with columns for the message role and content.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
messages <- initialize_messages()
messages_custom <- initialize_messages("Hello! How can I assist you today?")
```

install_tiktoken *Install the tiktoken Python package*

Description

This function installs the tiktoken Python package using the specified installation method and Conda environment (if any).

Usage

```
install_tiktoken(method = "auto", conda = "auto")
```

Arguments

method	The installation method to use. Can be one of "auto" (default), "conda", "virtualenv", "pip", or "windows".
conda	The name or path of the Conda environment to use for the installation, or "auto" (default) to let reticulate automatically manage the environment.

is_chatlog *Determine if an object is a chatlog*

Description

This function checks if an object is of class "chatlog".

Usage

```
is_chatlog(object)
```

Arguments

object	An R object to check

Value

TRUE if the object is of class "chatlog", FALSE otherwise

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
is_chatlog("Hello, World!")
# [1] FALSE

chat <- new("chatlog")
is_chatlog(chat)
# [1] TRUE
```

`is_json` *Check if the provided string is in valid JSON format.*

Description

Check if the provided string is in valid JSON format.

Usage

```
is_json(input_string)
```

Arguments

`input_string`	A character string to be checked for JSON format.

Value

A logical value. If the input string is in valid JSON format, returns TRUE, otherwise returns FALSE.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
is_json('{"name": "John", "age": 30}')
# TRUE

is_json('{"name": "John", age: 30}')
# FALSE

is_json('')
# FALSE
```

is_python *Check if a Character String Contains Valid Python Code*

Description

This function takes a character string as input and checks if it contains valid Python code. It returns TRUE if the string contains valid Python code, and FALSE otherwise.

Usage

```
is_python(code)
```

Arguments

code	A character string containing code to be checked for Python code validity.

Details

This function presuposes that python is installed on the system.

Value

A logical value: TRUE if the input character string contains valid Python code, and FALSE otherwise.

Examples

```
## Not run:
# Check if the string contains valid Python code
is_python("print('Hello, World!')")

# Check if the string contains invalid Python code
is_python("prit('Hello, World!')")

## End(Not run)
```

is_r *Check if a character string contains valid R code*

Description

This function takes a character string as input and attempts to parse it as R code using the 'parse' function. If the parsing is successful, the function returns TRUE, indicating that the input string contains valid R code. If parsing fails, the function returns FALSE.

Usage

```
is_r(code)
```

Arguments

code	A character string containing the R code to be checked.

Value

A logical value indicating whether the input code string contains valid R code or not.

Examples

```
# Valid R code
valid_code <- "x <- 5; y <- 10; z <- x + y"
is_r(valid_code)

# Invalid R code
invalid_code <- "x <- 5 + 'a'"
is_r(invalid_code)
```

---

is_xml *Check if the provided string is in valid XML format.*

Description

Check if the provided string is in valid XML format.

Usage

```
is_xml(input_string)
```

Arguments

input_string	A character string to be checked for XML format.

Value

A logical value. If the input string is in valid XML format, returns TRUE, otherwise returns FALSE.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
is_xml('<?xml version="1.0"?><root><element>value</element></root>')
# TRUE

is_xml('<root><element>value</element></root>')
# FALSE

is_xml('')
# FALSE
```

`java_to_r` *Convert Java code to R code*

Description

This function takes Java code as input and returns the corresponding R code. It uses GPT-4 powered chat to perform the code conversion.

Usage

```
java_to_r(java)
```

Arguments

`java`	A character string containing the Java code to be converted to R, or a filename with the Java code.

Details

This function is not guaranteed to provide perfect conversions and might produce invalid R code in some cases. Users are encouraged to verify the output code carefully.

Value

If the input is a character string, the resulting R code will be printed and returned as a character string. If the input is a filename, the resulting R code will be saved as a file with the same name as the input file, but with the extension '.R', and the filename will be returned.

Examples

```
## Not run:
# Convert a simple Java code snippet to R
java_code <- "public class HelloWorld {
               public static void main(String[] args) {
                 System.out.println(\"Hello, world!\");
               }
             }"
r_code <- java_to_r(java_code)
cat(r_code)
```

```
# Convert Java code from a file and save the result to an R file
input_file <- "path/to/java_file.java"
output_file <- java_to_r(input_file)

## End(Not run)
```

`java_to_r_prompt` *Input: message content template for java_to_r()*

Description

This message content serves as user input for the java_to_r() function.

Usage

```
java_to_r_prompt
```

Format

'java_to_r_prompt' A data frame with 2 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

`messages` *Extract messages from a response object or a chatlog object*

Description

This function takes a response object or a chatlog object as input and returns the messages. If the input is a response object, the function extracts and returns the messages from the choices. If the input is a chatlog object, the function returns the messages directly.

Usage

```
messages(x)
```

Arguments

`x`	A list representing a response object or a chatlog object

Value

A data.frame containing the messages

Examples

```
## Not run:
# Using a response object
response <- list(choices = list(message = "This is a message."))
messages_from_response <- messages(response)
print(messages_from_response)

# Using a chatlog object
chatlog_id <- chat("Hello, how are you?")
chatlog <- get_chatlog(chatlog_id)
messages_from_chatlog <- messages(chatlog)
print(messages_from_chatlog)

## End(Not run)
```

---

messages_content *Extract Messages Content from OpenAI API Response*

Description

This function extracts the messages content from the HTTP response of an API call to the OpenAI chat completions endpoint.

Usage

```
messages_content(response)
```

Arguments

response	a list object representing the HTTP response

Value

a character string representing the choices/messages from the response

Author(s)

Ulrich Matter umatter@protonmail.com

model *Extract Model from OpenAI API response*

Description

This function extracts the model from the parsed HTTP response of an API call to the OpenAI chat completions endpoint.

Usage

```
model(response)
```

Arguments

response	a list object representing the HTTP response

Value

a character string representing the model from the response

Author(s)

Ulrich Matter umatter@protonmail.com

nested_to_pipe *Convert nested R code to pipe syntax*

Description

This function takes an R script containing traditional (nested syntax) R code and converts it to magrittr-style syntax, using the pipe (It also validates the input and output code to ensure proper R syntax.

Usage

```
nested_to_pipe(r, n_tokens_limit = 3000, ...)
```

Arguments

r	A file path or character string containing the R code to be converted.
n_tokens_limit	The maximum number of tokens allowed in the input text (default: 3000).
...	Additional arguments passed to the chat_completion function.

Value

If r is a character string, the function returns the converted R code as a character string. If r is a file path, the function writes the converted code to a new file with the same name and a "-pipe.R" suffix, and returns the path to the output file.

Examples

```
## Not run:
# Converting a character string
input <- "result <- mean(sqrt(abs(rnorm(10, 0, 1))), na.rm = TRUE)"
output <- nested_to_pipe(input)
cat(output)

# Converting a file
# Create a temporary input file
input_file <- tempfile(fileext = ".R")
write("result <- mean(sqrt(abs(rnorm(10, 0, 1))), na.rm = TRUE)", input_file)

# Convert the file using nested_to_pipe
output_file <- nested_to_pipe(input_file)

# Check the converted file content
cat(readLines(output_file))

## End(Not run)
```

---

nested_to_pipe_prompt *Input: message content template for nested_to_pipe()*

Description

This message content serves as user input for the nested_to_pipe() function.

Usage

```
nested_to_pipe_prompt
```

Format

'nested_to_pipe_prompt' A data frame with 4 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

num_tokens *Get number of tokens in a string using OpenAI's tiktoken library*

Description

This function uses the num_tokens_from_string function provided in OpenAI's tiktoken python library to get the number of tokens in a string.

Usage

```
num_tokens(text, encoding = "cl100k_base")
```

Arguments

text	a character string to count the number of tokens from
encoding	a character string that specifies how text is converted into tokens. The default is "cl100k_base" (for ChatGPT models; see https://github.com/openai/openai-cookbook/blob/main/examples/How_to_count_tokens_with_tiktoken.ipynb for details)

Value

an integer indicating the number of tokens in the input string

References

https://github.com/openai/openai-cookbook/blob/main/examples/How_to_count_tokens_with_tiktoken.ipynb

num_tokens_file *Compute total number of tokens in a text file*

Description

The function batch-wise computes the total number of tokens in a text file. The function returns a numeric value indicating the total number of tokens in the file. The function can be used on very large text files.

Usage

```
num_tokens_file(filename, batch_size = 1000, encoding = "cl100k_base")
```

Arguments

filename	character string indicating the name of the text file to read in
batch_size	integer indicating the number of lines to read in per batch (default is 1000)
encoding	character string indicating the encoding to use (default is "cl100k_base")

Value

a numeric value indicating the total number of tokens in the text file

---

`object` *Extract object information from OpenAI API response*

Description

This function extracts the object data from the parsed HTTP response of an API call to the OpenAI chat completions endpoint (provides information about the endpoint).

Usage

```
object(response)
```

Arguments

`response`	a list object representing the HTTP response

Value

a data frame with the usage statistics of the API call (how many tokens used)

Author(s)

Ulrich Matter umatter@protonmail.com

`openai_api_key` *Set OpenAI API Key as an Environment Variable*

Description

This function sets the OpenAI API key as an environment variable in the current R session. It takes the API key as an input and stores it as an environment variable, allowing other functions to access the key when needed.

Usage

```
openai_api_key(api_key)
```

Arguments

`api_key`	A character string containing the OpenAI API key.

Value

Nothing is returned; the function is called for its side effects.

Author(s)

Ulrich Matter umatter@protonmail.com

See Also

https://platform.openai.com/docs/ for more information on the OpenAI API.

Examples

```
## Not run:
# Set the OpenAI API key for the current R session
openai_api_key("your_api_key_here")

## End(Not run)
```

---

parse_response *Parse a Response From the API*

Description

This function takes an input string containing both text and code blocks, and returns a list of blocks with their respective type, content, and language (if applicable). Each block is either of type 'text' or 'code'.

Usage

```
parse_response(input_string)
```

Arguments

input_string	The response from the API

Value

A list of blocks with type, content, and language (for code blocks)

Author(s)

Jonathan Chassot

Examples

```
## Not run:
parse_response("Hello world!")
# [[1]]
# [[1]]$type
# [1] "text"
#
# [[1]]$content
```

```
# [1] "Hello world!"
#
parse_response("```python\nprint('Hello world!')\n```")
# [[1]]
# [[1]]$type
# [1] "code"
#
# [[1]]$content
# [1] "print('Hello world!')"
#
# [[1]]$language
# [1] "python"
#
parse_response("Hello world!\n\n```python\nprint('Hello world!')\n```")
# [[1]]
# [[1]]$type
# [1] "text"
#
# [[1]]$content
# [1] "Hello world!"
#
# [[2]]
# [[2]]$type
# [1] "code"
#
# [[2]]$content
# [1] "print('Hello world!')"
#
# [[2]]$language
# [1] "python"
#

## End(Not run)
```

`python_to_r` *Convert Python code to R code*

Description

This function takes a Python code file or character string as input and attempts to convert the code to R using the OpenAI API. The function provides feedback on the total tokens used during the conversion and warns if the output might not be valid R code.

Usage

```
python_to_r(py)
```

Arguments

`py`	A file path or character string containing the Python code to be converted.

Value

If the input is a character string, the function returns the converted R code as a character string. If the input is a file, the function writes the converted R code to a new file with the same name and a ".R" extension, and returns the output file path.

Examples

```
## Not run:
# Convert a Python code string to R code
python_code <- "x = 5"
r_code <- python_to_r(python_code)
print(r_code)

# Convert a Python code file to an R code file
python_file <- "path/to/your/python_file.py"
r_file <- python_to_r(python_file)
cat(readLines(r_file), sep = "\n")

## End(Not run)
```

---

python_to_r_prompt *Input: message content template for python_to_r()*

Description

This message content serves as user input for the python_to_r() function.

Usage

```
python_to_r_prompt
```

Format

'python_to_r_prompt' A data frame with 2 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

read_text *Convert Text to Tidy-Text Format*

Description

This function takes a character string or a path to a text file as input and converts it to tidy-text format. The resulting tibble contains one row for each line of the input text, along with the file name, and line number.

Usage

```
read_text(text)
```

Arguments

text	A character string containing the text to be converted, or a path to a text file.

Value

A tibble containing the converted text in tidy-text format, with columns for the text, file name, line number, and batch index (if applicable).

Examples

```
read_text("Hello, how are you?")
read_text("path/to/text/file.txt")
```

read_text_batches *Load data from text files in batches*

Description

This function reads in data from text files in batches using the read_lines_chunked function from the readr package.

Usage

```
read_text_batches(text, batch_size = 3500)
```

Arguments

text	character string, either containing a path to a text file to read in or containing the text.
batch_size	integer indicating the number of lines to read in per batch (default is 3500)

Value

a numeric value indicating the total number of tokens in the text file

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
## Not run:
data_path <- system.file("text", "lorem.txt", package = "OpenAIR")
text_data <- read_text_batches(data_path)

## End(Not run)
```

---

refactor *Refactor R Code with AI Assistance*

Description

This function refactors R code with the assistance of an AI chatbot.

Usage

```
refactor(file, ...)
```

Arguments

file	character, file name of a file containing R code to be refactored or a character string containing R code to be refactored
...	optional arguments to be passed to the chat_completion function

Value

The refactored code either to the console (if the input ‘file‘ was a character string) or written to a file (if the input was a file name).

Examples

```
## Not run:
# Create a sample R function file
cat("my_sum <- function(a, b) {", "return(a + b)", "}", file = "sample_function.R")

# Refactor the R function and return the output
refactored_function <- refactor(file = "sample_function.R")

# Refactor the R function and write the output to the same file
refactor(file = "sample_function.R")

## End(Not run)
```

refactor_prompt *Input: message content template for refactor()*

Description

This message content serves as user input for the refactor() function.

Usage

```
refactor_prompt
```

Format

'nested_to_pipe_prompt' A data frame with 2 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

references_to_bibtex *Convert references in plain text to BibTeX format*

Description

This function takes a character string or a file path to plain text references and converts them into BibTeX format. The function reads the input text, processes it, and returns a character string containing the references in BibTeX format. If a file path is provided, the function also writes the BibTeX entries to a .bib file in the same directory.

Usage

```
references_to_bibtex(references)
```

Arguments

references	A character string or a file path to a file containing the plain text references to convert.

Value

A character string containing the references in BibTeX format or, if a file path is provided, the function writes the BibTeX entries to a .bib file in the same directory and returns the file path of the newly created .bib file.

Author(s)

Ulrich Matter umatter@protonmail.com

See Also

https://ctan.org/pkg/bibtex for more information on BibTeX format

Examples

```
## Not run:
# Convert plain text references to BibTeX format
references <- "Doe, J., & Smith, J. (2020). The title of the paper.
Journal of Scientific Computing, 12, 45-67."
references_to_bibtex(references)

## End(Not run)
```

---

references_to_bibtex_prompt

Input: message content template for references_to_bibtex()

Description

This message content serves as user input for the references_to_bibtex function.

Usage

```
references_to_bibtex_prompt
```

Format

'references_to_bibtex_prompt' A data frame with 4 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

regenerate *Regenerate the last response in an ongoing chat*

Description

This function removes the last response in a chatlog, generates a new response based on the updated chatlog, and updates the chatlog accordingly. The output can be displayed as a message in the console, returned as a message, or returned as a response object.

Usage

```
regenerate(chatlog_id = ".__CURRENTCHAT__", output = "message_to_console")
```

Arguments

chatlog_id	A character string specifying the ID of the chatlog (default: '.__CURRENTCHAT__')
output	A character string specifying the output format. Options are 'message_to_console', 'message', or 'response_object' (default: 'message_to_console')

Value

If output is 'message_to_console', the function returns NULL and prints the message to the console. If output is 'message', the function returns a character string containing the message. If output is 'response_object', the function returns the full response object.

Examples

```
## Not run:
# Start a new chat and save the chatlog ID
chatlog_id <- chat("Hello, how are you?")

# Regenerate the last response in the chat and display it in the console
regenerate(chatlog_id)

# Regenerate the last response in the chat and return it as a message
message <- regenerate(chatlog_id, output = "message")
print(message)

# Regenerate the last response in the chat and return it as a response object
response_object <- regenerate(chatlog_id, output = "response_object")
print(response_object)

## End(Not run)
```

replace_file_extension *Replace File Extension*

Description

This function replaces the file extension of a given filename with a specified new extension. It validates the input to ensure the filename and the new extension are single character strings. Then, it replaces the old extension with the new one and returns the modified filename.

Usage

```
replace_file_extension(filename, new_extension)
```

Arguments

filename	The input filename as a character string.
new_extension	The new file extension to replace the old one (including the dot, e.g., ".bib").

Value

A character string representing the filename with the replaced file extension.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
## Not run:
# Replace the file extension of a text file with a BibTeX extension
new_filename <- replace_file_extension("example_document.txt", ".bib")
print(new_filename) # "example_document.bib"

## End(Not run)
```

r_to_python *Convert R Code to Python Code*

Description

This function takes an R code file as input and uses a language model to convert the R code to Python code. The converted Python code is then either returned as a character string or written to a file, depending on the input.

Usage

```
r_to_python(r)
```

Arguments

r	The R code file to be converted to Python code. This should be a file path in the form of a character string.

Value

If the input filename is a "character string", the converted Python code will be returned as a character string. Otherwise, a new Python file will be created with the same name as the input file but with a ".py" extension, and the function will return the file path of the newly created Python file.

Examples

```
## Not run:
# Convert R code to Python code and display the result as a character string
r_to_python("example.R")

# Convert R code to Python code and save it to a file
r_to_python("example.R", output_file = "example.py")

## End(Not run)
```

---

r_to_python_prompt *Input: message content template for r_to_python()*

Description

This message content serves as user input for the r_to_python() function.

Usage

```
r_to_python_prompt
```

Format

‘r_to_python_prompt‘ A data frame with 2 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

set_chatlog *Set up a new chatlog*

Description

This function sets up a new chatlog object for a conversation.

Usage

```
set_chatlog(
  initial_role = "system",
  initial_content = "You are a helpful assistant.",
  chatlog_id = NULL
)
```

Arguments

initial_role	A character string representing the role issueing the initial content (per default: "system")
initial_content	A character string representing the initial message from the system
chatlog_id	A character string representing the ID of this conversation. Per default, this will be set automatically.

Value

A new chatlog object

Examples

```
chat <- set_chatlog("Welcome to our chat!")
is_chatlog(chat)
chat
```

split_text *Split Text into Chunks*

Description

This function splits a text string into a vector of strings with a specified number of tokens each.

Usage

```
split_text(text, N)
```

Arguments

text	A character vector containing the text to be split.
N	An integer specifying the number of tokens per chunk.

Value

A character vector containing the chunks of text with N tokens each.

Examples

```
large_text <- "This is an example of a large text string
that will be split into chunks of N tokens each by our custom R function."
num_tokens_per_chunk <- 5
split_text(large_text, num_tokens_per_chunk)
```

---

start_chat *Start a new chat session*

Description

This function starts a new chat session by initializing a messages object with an initial system message, creating a new log environment to store the messages, and adding the messages object to the log environment.

Usage

```
start_chat(
  initial_role = "system",
  initial_content = "You are a helpful assistant.",
  show = FALSE,
  chatlog_id = NULL
)
```

Arguments

initial_role	A character string representing the role issueing the initial content (per default: "system")
initial_content	A character string representing the initial message from the system
show	Logical, if TRUE, the current chat log is displayed via View(). Default is FALSE.
chatlog_id	A character string representing the ID of this conversation. Per default, this will be set automatically.

Value

A character string indicating the name of the log environment created for the chat session.

Examples

```
# Start a new chat session with the default system message
chatlog_id <- start_chat()

# Start a new chat session with a custom system message
chatlog_id <- start_chat("How can I assist you today?")
```

`trump` *Essay on How Donald Trump Became President of the United States*

Description

This dataset contains an essay describing how Donald Trump became the 45th President of the United States in 2016. The essay is written in the style of Mark Twain, a famous American writer known for his satirical and humorous style.

Usage

```
trump
```

Format

'trump' A character string.

Source

This essay was written by a language model trained by OpenAI.

`usage` *Extract usage data from OpenAI API response*

Description

This function extracts the usage data from the parsed HTTP response of an API call to the OpenAI chat completions endpoint.

Usage

```
usage(response)
```

Arguments

`response`	a list object representing the HTTP response

Value

a data frame with the usage statistics of the API call (how many tokens used)

write_code *Write code based on user's input*

Description

The 'write_code' function prompts the user to provide a plain English description of a program or function and the programming language it should be written in. The function then generates the code based on the user's input and writes it to a specified file.

Usage

```
write_code(filename, chatlog_id = ".__CURRENTCODEFILE__")
```

Arguments

filename	A character string representing the name of the file where the generated code will be saved.
chatlog_id	An optional character string representing the chatlog ID. Defaults to ".__CURRENTCODEFILE__". This ID is used to maintain the conversation history with the chatbot API.

Value

Returns the name of the file containing the generated code.

Author(s)

Ulrich Matter

Examples

```
## Not run:
# Generate code based on user input and save it to a file
generated_code_file <- write_code("example_code.R")

# Check the content of the generated code file
cat(readLines(generated_code_file))

# Clean up
unlink(generated_code_file)

## End(Not run)
```

write_test *Write test for an R function*

Description

This function reads an R function from a file and generates a test file with documentation.

Usage

```
write_test(file)
```

Arguments

file	The file path of the R function.

Value

If the input is a character string, the function returns the generated output without creating a test file. Otherwise, it creates a test file and returns the file name.

Examples

```
## Not run:
# Write test for an R function
write_test("path/to/file.R")

## End(Not run)
```

write_test_prompt *Input: message content template for write_test()*

Description

This message content serves as user input for the write_test function.

Usage

```
write_test_prompt
```

Format

‘write_test_prompt‘ A data frame with 2 rows and 2 columns representing the messages object in API calls.

Source

Contributed by umatter

`%c%` *Send a message to ChatGPT and assign the response to a variable*

Description

This function sends a message to the ChatGPT API using the 'chat()' function and returns the response in the specified output format.

Usage

```
message %c% output
```

Arguments

`message`	A character string containing the message to be sent to the ChatGPT API.
`output`	A character string specifying the output format. Valid options are "message_to_console", "message", or "response_object". Default is "message_to_console".

Value

Depending on the value of the 'output' argument, this function returns one of the following: * "message_to_console": a message containing the response text to be printed to the console. * "message": the response text as a character vector. * "response_object": the full response object from the ChatGPT API.

Author(s)

Ulrich Matter umatter@protonmail.com

Examples

```
## Not run:
# Send a message and assign the response to a variable
response_var <- "Hello, ChatGPT!" %c% "message"

# Print the response
print(response_var)

# Send a message and return the full response object
response_obj <- "Hello, ChatGPT!" %c% "response_object"

# Print the response
print(response_obj)

## End(Not run)
```

Index