Package 'AIscreenR'

May 6, 2026

Title AI Screening Tools in R for Systematic Reviewing

Version 0.3.2

Description
Provides functions to conduct title and abstract screening in systematic reviews using large language models, such as the Generative Pre-trained Transformer (GPT) models from 'OpenAI' <https://developers.openai.com/>. These functions can enhance the quality of title and abstract screenings while reducing the total screening time significantly. In addition, the package includes tools for quality assessment of title and abstract screenings, as described in Vembye, Christensen, Mølgaard, and Schytt (2025) <DOI:10.1037/met0000769>.

Depends R (>= 4.1.0)

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Imports dplyr, tibble, httr2, stringr, furrr, tidyr, tictoc, askpass, curl, purrr, lifecycle, jsonlite, htmltools, tidyselect, rlang

Suggests future, knitr, rmarkdown, usethis, testthat (>= 3.0.0), withr, readtext, quarto

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first tabscreen_gpt, approximate_price_gpt, rate_limits, api_key_functions

URL https://mikkelvembye.github.io/AIscreenR/, https://github.com/MikkelVembye/AIscreenR

BugReports https://github.com/MikkelVembye/AIscreenR/issues

LazyData true

VignetteBuilder quarto

Config/Needs/website quarto

NeedsCompilation no

Author Mikkel H. Vembye [aut, cre] (ORCID:
<https://orcid.org/0000-0001-9071-0724>),
Thomas Olsen [aut]

Maintainer Mikkel H. Vembye <mikkel.vembye@gmail.com>

Repository CRAN

Date/Publication 2026-04-20 20:50:17 UTC

Contents

`approximate_price_gpt` *Approximate price estimation for title and abstract screening using OpenAI's GPT API models*

Description

[Experimental]

This function supports the approximation of the price of title and abstract screenings when using OpenAI's GPT API models. The function only provide approximately accurate price estimates. When detailed descriptions are used, this will increase the completion tokens with an unknown amount.

Usage

```
approximate_price_gpt(
  data,
  prompt,
  studyid,
  title,
  abstract,
  model = "gpt-4o-mini",
  reps = 1,
  top_p = 1,
  token_word_ratio = 1.6,
  reasoning_effort = "medium",
  verbosity = "low"
)
```

Arguments

`data`	Dataset containing the titles and abstracts.
`prompt`	Prompt(s) to be added before the title and abstract.
`studyid`	Unique Study ID. If missing, this is generated automatically.
`title`	Name of the variable containing the title information.
`abstract`	Name of variable containing the abstract information.
`model`	Character string with the name of the completion model. Can take multiple models, including gpt-4 models. Default = `"gpt-4o-mini"`. Find available model at https://developers.openai.com/api/docs/models/model-endpoint-compatibility.
`reps`	Numerical value indicating the number of times the same question should be sent to the GPT server. This can be useful to test consistency between answers. Default is `1` but when using gpt-3.5-turbo or gpt-4o-mini models, we recommend setting this value to `10`.

top_p
: 'An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered. We generally recommend altering this or temperature but not both.' (OPEN-AI). Default is 1. Find documentation at https://developers.openai.com/api/reference/resources/chat#chat/create-top_p.

token_word_ratio
: The multiplier used to approximate the number of tokens per word. Default is `1.6` which we empirically have found to be the average number of tokens per word.

reasoning_effort
: Character string indicating the level of reasoning effort required for the task. Default is `"medium"`. Can take the values `"low"`, `"medium"`, and `"high"`. Only applicable for gpt-5 models.

verbosity
: Character string indicating the level of verbosity in the model's responses. Default is `"low"`. Can take the values `"low"`, `"medium"`, and `"high"`. Only applicable for gpt-5 models.

Value

An object of class `"gpt_price"`. The object is a list containing the following components:

price
: numerical value indicating the total approximate price (in USD) of the screening across all gpt-models expected to be used for the screening.

price_data
: dataset with prices across all gpt models expected to be used for screening.

Examples

```
prompt <- "This is a prompt"

app_price <- approximate_price_gpt(
  data = filges2015_dat[1:2,],
  prompt = prompt,
  studyid = studyid,
  title = title,
  abstract = abstract,
  model = c("gpt-4o-mini", "gpt-4"),
  reps = c(10, 1)
)

app_price
app_price$price_dollar
app_price$price_data
```

`create_fine_tune_data` *Function to generate dataset to be used for fine-tuning models*

Description

This function creates the initial data that can be used to fine tune models from OpenAI.

Usage

```
create_fine_tune_data(data, prompt, studyid, title, abstract)
```

Arguments

`data`	Dataset containing the titles and abstracts.
`prompt`	Prompt(s) to be added before the title and abstract.
`studyid`	Unique Study ID. If missing, this is generated automatically.
`title`	Name of the variable containing the title information.
`abstract`	Name of variable containing the abstract information.

Value

A dataset of class `'fine_tune_data'`.

Note

The dataset contains at least the following variables:

studyid	`integer/character/factor`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
question	`character`	indicating the final question sent to OpenAI's GPT API models for training.

See Also

`save_fine_tune_data()`

Examples

```
# Extract 5 irrelevant and relevant records, respectively.
dat <- filges2015_dat[c(1:5, 261:265),]

prompt <- "Is this study about functional family therapy?"

dat <-
  create_fine_tune_data(
    data = dat,
```

```
    prompt = prompt,
    studyid = studyid,
    title = title,
    abstract = abstract
   )

dat
```

`disagreements` *Disagreement sample data*

Description

Example rows where human screening decisions differ from GPT decisions. Each row is a (study × prompt) screening outcome.

Usage

```
disagreements
```

Format

A tibble/data.frame with one row per screened (studyid, promptid) and 17 columns:

author	character	Study authors
human_code	numeric	Human screening decision (1 include, 0 exclude)
studyid	integer	Unique study identifier
title	character	Study title
abstract	character	Study abstract
promptid	integer	Prompt identifier
prompt	character	Original short screening prompt text
model	character	Model used for the run
question	character	Full constructed question sent to model
top_p	numeric	Nucleus sampling parameter
incl_p	numeric	Estimated probability of inclusion (if repetitions)
final_decision_gpt	character	GPT final label: Include / Exclude / Check
final_decision_gpt_num	numeric	Numeric GPT decision (1 include/check, 0 exclude)
longest_answer	character	Longest rationale text returned
reps	integer	Number of repetitions attempted
n_mis_answers	integer	Count of missing answers across reps
submodel	character	Specific model variant (if applicable)

filges2015_dat *RIS file data from Functional Family Therapy (FFT) systematic review*

Description

Bibliometric toy data from a systematic review regarding Functional Family Therapy (FFT) for Young People in Treatment for Non-opioid Drug Use (Filges et al., 2015). The data includes all 90 included and 180 excluded randomly sampled references from the literature search of the systematic review.

Usage

```
filges2015_dat
```

Format

A `tibble` with 270 rows/studies and 6 variables/columns

author	`character`	indicating the authors of the reference
eppi_id	`character`	indicating a unique eppi-ID for each study
studyid	`numeric`	indicating a unique study-ID for each study
title	`character`	with the title of the study
abstract	`character`	with the study abstract
human_code	`numeric`	indicating the human screening decision. 1 = included, 0 = excluded.

References

Filges, T., Andersen, D, & Jørgensen, A-M. K (2015). Functional Family Therapy (FFT) for Young People in Treatment for Non-opioid Drug Use: A Systematic Review *Campbell Systematic Reviews*, doi:10.4073/csr.2015.14

get_api_key *Get API key from R environment variable.*

Description

Get API key from R environment variable.

Usage

```
get_api_key(env_var = "OPENAI_API_KEY")
```

Arguments

env_var	Character string indicating the name of the temporary R environment variable with the API key and the used AI model. Currently, the argument only takes env_var = "OPENAI_API_KEY". See set_api_key() to set/create this variable.

Details

get_api_key() can be used after executing set_api_key() or by adding the api key permanently to your R environment by using usethis::edit_r_environ(). Then write OPENAI_API_KEY=[insert your api key here] and close the .Renviron window and restart R. For backward compatibility, it will also check CHATGPT_KEY if OPENAI_API_KEY is not set.

Value

The specified API key (NOTE: Avoid exposing this in the console).

Note

Find your personal API key via the OpenAI quickstart guide at https://developers.openai.com/api/docs/quickstart#generate-an-api-key.

See Also

set_api_key.

Examples

```
## Not run:
get_api_key()

## End(Not run)
```

---

get_api_key_groq *Get GROQ API key from R environment variable.*

Description

Get GROQ API key from R environment variable.

Usage

```
get_api_key_groq(env_var = "GROQ_API_KEY")
```

Arguments

env_var	Character string indicating the name of the temporary R environment variable with the API key and the used AI model. Currently, the argument only takes env_var = "GROQ_API_KEY". See set_api_key() to set/create this variable.

Details

`get_api_key_groq()` can be used after executing `set_api_key()` or by adding the api key permanently to your R environment by using `usethis::edit_r_environ()`. Then write `GROQ_API_KEY=[insert your api key h` and close the `.Renviron` window and restart R.

Value

The specified API key (NOTE: Avoid exposing this in the console).

Note

Find your personal API key at https://console.groq.com/keys.

See Also

`set_api_key`.

Examples

```
## Not run:
get_api_key_groq()

## End(Not run)
```

`groq_model_prizes` *Groq model prices (last updated March 18, 2026)*

Description

Data set containing input and output prizes for all GROQ's API models.

Usage

```
groq_model_prizes
```

Format

A `data.frame` containing 4 rows/models and 3 variables/columns

model	`character`	indicating the specific GPT model
price_in_per_token	`character`	indicating the input prize per token
price_out_per_token	`character`	indicating the output prize per token

References

GROQ. *Pricing*. https://groq.com/pricing

is_chatgpt *Test if the object is a 'chatgpt' object*

Description

[Deprecated]

This function returns TRUE for chatgpt objects, and FALSE for all other objects.

Usage

```
is_chatgpt(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the chatgpt class.

is_chatgpt_tbl *Test if the object is a 'chatgpt_tbl' object*

Description

[Deprecated]

This function returns TRUE for chatgpt_tbl objects, and FALSE for all other objects.

Usage

```
is_chatgpt_tbl(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the chatgpt_tbl class.

is_gpt *Test if the object is a 'gpt' object*

Description

This function returns TRUE for gpt objects, and FALSE for all other objects.

Usage

```
is_gpt(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the gpt class.

is_gpt_agg_tbl *Test if the object is a 'gpt_agg_tbl' object*

Description

This function returns TRUE for gpt_agg_tbl objects, and FALSE for all other objects.

Usage

```
is_gpt_agg_tbl(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the gpt_agg_tbl class.

is_gpt_tbl *Test if the object is a 'gpt_tbl' object*

Description

This function returns TRUE for gpt_tbl objects, and FALSE for all other objects.

Usage

```
is_gpt_tbl(x)
```

Arguments

x	An object

Value

TRUE if the object inherits from the gpt_tbl class.

model_prizes *Model prize data (last updated March 18, 2026)*

Description

Dataset mainly containing input and output prizes for all OpenAI's GPT API models.

Usage

```
model_prizes
```

Format

A data.frame containing 36 rows/models and 3 variables/columns

model	character	indicating the specific GPT model
price_in_per_token	character	indicating the input prize per token
price_out_per_token	character	indicating the output prize per token

References

OpenAI. *Pricing*. https://developers.openai.com/api/docs/pricing

print.chatgpt *Print methods for 'chatgpt' objects*

Description

Print methods for 'chatgpt' objects

Usage

```
## S3 method for class 'chatgpt'
print(x, ...)
```

Arguments

x	an object of class 'chatgpt'.
...	other print arguments.

Value

Information about how to find answer data sets and pricing information.

Examples

```
## Not run:
print(x)

## End(Not run)
```

print.gpt *Print methods for 'gpt' objects*

Description

Print methods for 'gpt' objects

Usage

```
## S3 method for class 'gpt'
print(x, ...)
```

Arguments

x	an object of class 'gpt'.
...	other print arguments.

Value

Information about how to find answer data sets and pricing information.

Examples

```
## Not run:
print(x)

## End(Not run)
```

`print.gpt_price` *Print methods for* `'gpt_price'` *objects*

Description

Print methods for `'gpt_price'` objects

Usage

```
## S3 method for class 'gpt_price'
print(x, ...)
```

Arguments

`x`	an object of class `"gpt_price"`.
`...`	other print arguments.

Value

The total price of the screening across all gpt-models expected to be used for the screening.

Examples

```
## Not run:
print(x)

## End(Not run)
```

print.groq *Print method for 'groq' objects*

Description

Print method for 'groq' objects

Usage

```
## S3 method for class 'groq'
print(x, ...)
```

Arguments

x	A groq object from tabscreen_groq().
...	Additional arguments passed to print.

Value

Information about how to find answer data sets and pricing information.

Examples

```
## Not run:
print(x)

## End(Not run)
```

rate_limits_per_minute *Find updated rate limits for API models*

Description

[Stable]

rate_limits_per_minute reports the rate limits for a given API model. The function returns the available requests per minute (RPM) as well as tokens per minute (TPM). Find general information at https://developers.openai.com/api/docs/models/model-endpoint-compatibility.

Usage

```
rate_limits_per_minute(
  model = "gpt-4o-mini",
  AI_tool = "OpenAI",
  api_key = NULL
)
```

Arguments

model	Character string with the name of the completion model. Default is "gpt-4o-mini". Can take multiple values. For OpenAI models, find available models at https://developers.openai.com/api/docs/models/model-endpoint-compatibility. For Groq models, find available models at https://console.groq.com/docs/models.
AI_tool	Character string specifying the AI tool from which the API is issued. Currently supports "OpenAI" (default) and "Groq".
api_key	Character string with the API key. For OpenAI, use get_api_key(). For Groq, use get_api_key_groq().

Value

A tibble including variables with information about the model used, the number of requests and tokens per minute.

Examples

```
## Not run:
set_api_key()

rate_limits_per_minute(
  model = "gpt-4o-mini",
  AI_tool = "OpenAI",
  api_key = get_api_key()
)

# Groq example
rate_limits_per_minute(
  model = "llama3-70b-8192",
  AI_tool = "Groq",
  api_key = get_api_key_groq()
)

## End(Not run)
```

---

read_ris_to_dataframe *Read an RIS file into a data frame*

Description

Parses an RIS file into a data.frame, preserving the order of tags as they first appear in the file. Repeated tags within a record are collapsed into a single semicolon-separated string.

Usage

```
read_ris_to_dataframe(file_path)
```

Arguments

file_path	character. Path to the RIS file to read.

Value

A data.frame with one row per record and one column per encountered RIS tag, using descriptive column names. Columns are ordered by first appearance of the tag in the file. Repeated tag values are collapsed with "; ".

Examples

```
## Not run:
df <- read_ris_to_dataframe("data-raw/raw data/apa_psycinfo_test_data.ris")

## End(Not run)
```

---

report *Generate a report for screening disagreements between human and AI decisions*

Description

[Stable]

This function generates a report for screening disagreements between human and GPT decisions. It extracts information from the provided data. The function then compiles this information into a report using Quarto. The report can be saved in formats as HTML, PDF or Word. The generated report includes sections for each study, displaying the study ID, title, abstract, model, run date information and the decision generated by the GPT API model. The report also includes a section for a comment on the GPT decision. The function also provides options to customize the document title, subtitle, and output directory.

Usage

```
report(
  data,
  studyid,
  title,
  abstract,
  gpt_answer,
  human_code,
  final_decision_gpt_num,
  file,
  format = "html",
  open = TRUE,
  document_title,
  document_subtitle = "",
  directory = getwd()
)
```

Arguments

data	Data frame containing the screening data of disagreements between human decisions and GPT decisions.
studyid	Column name for the study ID.
title	Column name for the title.
abstract	Column name for the abstract.
gpt_answer	Column name for the AI's answer.
human_code	Column name for the human screening decision (numeric 0/1).
final_decision_gpt_num	Column name for the final numeric GPT decision (0/1).
file	Name of the output file. You can also provide a full path.
format	Format of the output file. Valid formats are 'html', 'pdf', 'docx'.
open	Logical indicating whether to open the report after generation. Default is TRUE.
document_title	Title of the document.
document_subtitle	Subtitle of the document. Default is an empty string.
directory	Directory where the output file will be saved. Default is the current working directory.

Value

An object of class `'report'`. The object is a list containing the following components:

file_out	string indicating the path to the generated report file.
...	some additional attributed values/components, including an attributed list with the arguments used in the function.

Examples

```
## Not run:
# Generate a report from the disagreements data
report(
  data = disagreements,
  studyid = studyid,
  title = title,
  abstract = abstract,
  gpt_answer = longest_answer,
  human_code = human_code,
  final_decision_gpt_num = final_decision_gpt_num,
  file = "Screening_Disagreements_Report",
  format = "html",
  document_title = "Study Report - Disagreement Explanations",
  open = TRUE
)

## End(Not run)
```

`sample_references` *Random sample references*

Description

`sample_references`samples n rows from the dataset with titles and abstracts either with or without replacement. This function is supposed to support the construct of a test dataset, as suggested by Vembye et al. (2025).

Usage

```
sample_references(
  data,
  n,
  with_replacement = FALSE,
  prob_vec = rep(1/n, nrow(data))
)
```

Arguments

`data`	Dataset containing the titles and abstracts wanted to be screened.
`n`	A non-negative integer giving the number of rows to choose.
`with_replacement`	Logical indicating if sampling should be done with of without replacement. Default is `FALSE`.
`prob_vec`	'A vector of probability weights for obtaining the elements of the vector being sampled.' Default is a vector of 1/n.

Value

A dataset with n rows.

References

Vembye, M. H., Christensen, J., Mølgaard, A. B., & Schytt, F. L. W. (2025). Generative Pretrained Transformer Models Can Function as Highly Reliable Second Screeners of Titles and Abstracts in Systematic Reviews: A Proof of Concept and Common Guidelines. *Psychological Methods*. doi:10.1037/met0000769

Examples

```
excl_test_dat <- filges2015_dat[1:200,] |> sample_references(100)
```

save_dataframe_to_ris *Write a data frame to a RIS file*

Description

Writes a data.frame to a RIS file, one record per row. If the data frame was created by `read_ris_to_dataframe()`, the original RIS tag order and tags are preserved where possible. Otherwise, a standard RIS format is used.

Usage

```
save_dataframe_to_ris(df, file_path)
```

Arguments

`df`	data.frame. The data to write.
`file_path`	character. Path to the output RIS file.

Details

If a field value contains semicolons, it is split and written as multiple tag lines. The `TY` (source type) field is written first for each record, followed by all other fields. Records are terminated with `ER -` .

Value

A character string indicating the file path where the RIS file was saved.

Examples

```
## Not run:
df <- read_ris_to_dataframe("data-raw/raw data/apa_psycinfo_test_data.ris")
save_dataframe_to_ris(df, "path/to/output.ris")

## End(Not run)
```

save_fine_tune_data *Function to write/save fine tune dataset in required jsonl format*

Description

This function creates `jsonl` training data that can be used to fine tune models from OpenAI. To generate a fine tuned model, this written data can be uploaded via https://developers.openai.com/api/docs/guides/supervised-fine-tuning.

Usage

```
save_fine_tune_data(
  data,
  role_and_subject,
  file,
  true_answer,
  roles = c("system", "user", "assistant")
)
```

Arguments

data	The dataset with questions strings that should be used for training. The data must be of class 'fine_tune_data', containing two variables named question and true_answer.
role_and_subject	Descriptions of the role of the GPT model and the subject under review, respectively.
file	A character string naming the file to write to. If not specified the written file name and format will be "fine_tune_data.jsonl".
true_answer	Optional name of the variable containing the true answers/decisions used for training. Only relevant, if the the dataset contains a variable with the name true_answer.
roles	String variable defining the various role the model should take. Default is roles = c("system", "user", "assistant").

Value

A jsonl dataset to the set working directory.

See Also

create_fine_tune_data()

Examples

```
# Extract 5 irrelevant and relevant records, respectively.
library(dplyr)

dat <- filges2015_dat[c(1:5, 261:265),]

prompt <- "Is this study about functional family therapy?"

ft_dat <-
  create_fine_tune_data(
    data = dat,
    prompt = prompt,
    studyid = studyid,
    title = title,
    abstract = abstract
```

```
    ) |>
    mutate(true_answer = if_else(human_code == 1, "Include", "Exclude"))

role_subject <- paste0(
  "Act as a systematic reviewer that is screening study titles and ",
  "abstracts for your systematic reviews regarding the the effects ",
  "of family-based interventions on drug abuse reduction for young ",
  "people in treatment for non-opioid drug use."
)

# Saving data in jsonl format (required format by OpenAI)
fil <- tempfile("fine_tune_data", fileext = ".jsonl")

save_fine_tune_data(
  data = ft_dat,
  role_and_subject = role_subject,
  file = fil
)
```

`screen_analyzer` *Analyze performance between the human and AI screening.*

Description

[Stable]

When both the human and AI title and abstract screening has been done, this function allows you to calculate performance measures of the screening, including the overall accuracy, specificity/recall, and sensitivity of the screening, as well as inter-rater reliability kappa statistics (Gartlehner et al., 2019; McHugh, 2012; Syriani et al., 2024).

Usage

```
screen_analyzer(x, human_decision = human_code, key_result = TRUE)
```

Arguments

`x`	An object of either class `'gpt'` or `'chatgpt'` or a dataset of either class `'gpt_tbl'`, `'chatgpt_tbl'`, or `'gpt_agg_tbl'`
`human_decision`	Indicate the variable in the data that contains the human_decision. This variable must be numeric, containing 1 (for included references) and 0 (for excluded references) only.
`key_result`	Logical indicating if only the raw agreement, recall, and specificity measures should be returned. Default is `TRUE`.

Value

A `tibble` with screening performance measures. The `tibble` includes the following variables:

promptid	`integer`	indicating the prompt ID.
model	`character`	indicating the specific gpt-model used.
reps	`integer`	indicating the number of times the same question was sent to GPT server.
top_p	`numeric`	indicating the applied top_p.
n_screened	`integer`	indicating the number of screened references.
n_missing	`numeric`	indicating the number of missing responses.
n_refs	`integer`	indicating the total number of references expected to be screened for the given condition
human_in_gpt_ex	`numeric`	indicating the number of references included by humans and excluded by gpt.
human_ex_gpt_in	`numeric`	indicating the number of references excluded by humans and included by gpt.
human_in_gpt_in	`numeric`	indicating the number of references included by humans and included by gpt.
human_ex_gpt_ex	`numeric`	indicating the number of references excluded by humans and excluded by gpt.
accuracy	`numeric`	indicating the overall percent disagreement between human and gpt (Gartlehner et al., 20
p_agreement	`numeric`	indicating the overall percent agreement between human and gpt.
precision	`numeric`	"measures the ability to include only articles that should be included" (Syriani et al., 202
recall	`numeric`	"measures the ability to include all articles that should be included" (Syriani et al., 2023)
npv	`numeric`	Negative predictive value (NPV) "measures the ability to exclude only articles that shoul
specificity	`numeric`	"measures the ability to exclude all articles that should be excluded" (Syriani et al., 2023
bacc	`numeric`	"capture the accuracy of deciding both inclusion and exclusion classes" (Syriani et al., 20
F2	`numeric`	F-measure that "consider the cost of getting false negatives twice as costly as getting fals
mcc	`numeric`	indicating percent agreement for excluded references (Gartlehner et al., 2019).
irr	`numeric`	indicating the inter-rater reliability as described in McHugh (2012).
se_irr	`numeric`	indicating standard error for the inter-rater reliability.
cl_irr	`numeric`	indicating lower confidence interval for the inter-rater reliability.
cu_irr	`numeric`	indicating upper confidence interval for the inter-rater reliability.
level_of_agreement	`character`	interpretation of the inter-rater reliability as suggested by McHugh (2012).

References

Gartlehner, G., Wagner, G., Lux, L., Affengruber, L., Dobrescu, A., Kaminski-Hartenthaler, A., & Viswanathan, M. (2019). Assessing the accuracy of machine-assisted abstract screening with DistillerAI: a user study. *Systematic Reviews*, 8:277, 1-10. doi:10.1186/s13643019122 13

McHugh, M. L. (2012). Interrater reliability: The kappa statistic. *Biochemia Medica*, 22(3), 276-282. https://pubmed.ncbi.nlm.nih.gov/23092060/

Syriani, E., David, I., & Kumar, G. (2023). Assessing the Ability of ChatGPT to Screen Articles for Systematic Reviews. *ArXiv Preprint ArXiv:2307.06464*.

Examples

```
## Not run:

library(future)

set_api_key()
```

```
prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

plan(multisession)

res <- tabscreen_gpt(
  data = filges2015_dat[1:2,],
  prompt = prompt,
  studyid = studyid,
  title = title,
  abstract = abstract
  )

plan(sequential)

res |> screen_analyzer()


## End(Not run)
```

`screen_errors` *Generic function to re-screen failed title and abstract requests.*

Description

This is a generic function to re-screen failed title and abstract requests. It reuses the arguments captured during the original screening and only re-submits the rows stored in `object$error_data` to the appropriate backend.

Usage

```
screen_errors(
  object,
  api_key = NULL,
  max_tries = NULL,
  max_seconds = NULL,
  is_transient = NULL,
  backoff = NULL,
  after = NULL,
  studyid = NULL,
  title = NULL,
  abstract = NULL,
  ...
)
```

Arguments

`object`	An object of either class `'gpt'` or `'groq'`, as returned by `tabscreen_gpt()` or `tabscreen_groq()`. Objects of class `'ollama'` are not supported.

`api_key` Optional API key. If omitted, the selected backend uses its own default (e.g., `get_api_key()` for GPT and `get_api_key_groq()` for Groq).

`max_tries, max_seconds`
'Cap the maximum number of attempts with `max_tries` or the total elapsed time from the first request with `max_seconds`. If neither option is supplied (the default), `httr2::req_perform()` will not retry' (Wickham, 2023). If missing, the values from the original screening (stored in `attr(object, "arg_list")`) will be reused.

`is_transient` 'A predicate function that takes a single argument (the response) and returns `TRUE` or `FALSE` specifying whether or not the response represents a transient error' (Wickham, 2023). If missing, the `is_transient` function from the original screening will be used.

`backoff` 'A function that takes a single argument (the number of failed attempts so far) and returns the number of seconds to wait' (Wickham, 2023). If missing, the `backoff` value from the original screening will be used.

`after` 'A function that takes a single argument (the response) and returns either a number of seconds to wait or `NULL`, which indicates that a precise wait time is not available and that the `backoff` strategy should be used instead' (Wickham, 2023). If missing, the `after` value from the original screening will be used.

`studyid` Optional column (unquoted) for study id. Defaults to 'studyid'.

`title` Optional column (unquoted) for title. If omitted, inferred (e.g., title, ti, t1).

`abstract` Optional column (unquoted) for abstract. If omitted, inferred (e.g., abstract, ab, abs).

`...` Further arguments forwarded to the underlying backend function (`tabscreen_gpt()` or `tabscreen_groq()`). If arguments were supplied in the original screening and should differ for re-screening, pass them again here.

Details

The backend is derived from `class(object)` and mapped to either `tabscreen_gpt()` or `tabscreen_groq()`. Only rows in `object$error_data` are re-submitted. To avoid name collisions during unnesting in the backend, columns that will be regenerated (currently `decision_binary`, `decision_description`, `error_message`, `res`) are dropped from `error_data` before the call. The original arguments from the first screening are taken from `attr(object, "arg_list")` and are combined with any non-`NULL` overrides provided here.

Value

An object of class `'gpt'` or `'groq'` similar to the object returned by the original screening function, with:

- `answer_data` updated to include newly successful rows,
- `error_data` updated to include only remaining failures. Other fields (e.g., `price_data`, `price_dollar`, and `arg_list`) are preserved or updated by the backend.

See Also

`tabscreen_gpt()`, `tabscreen_groq()`

Examples

```
## Not run:

# Example with openai
set_api_key()
prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

obj_with_error <-
  tabscreen_gpt(
    data = filges2015_dat[1:2,],
    prompt = prompt,
    studyid = studyid,
    title = title,
    abstract = abstract,
    model = "gpt-4o-mini"
    )

obj_rescreened <-
  obj_with_error |>
  screen_errors()


# Example with groq
prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

obj_with_error <-
  tabscreen_groq(
    data = filges2015_dat[1:2,],
    prompt = prompt,
    studyid = studyid,
    title = title,
    abstract = abstract,
    model = "llama-3.3-70b-versatile"
  )

obj_rescreened <-
  obj_with_error |>
  screen_errors()

## End(Not run)
```

---

screen_errors.chatgpt *Re-screen failed requests.*

Description

[Superseded]

This function supports re-screening of all failed title and abstract requests screened with `tabscreen_gpt.original()`. This function has been deprecated because OpenAI has deprecated the function_call and functions argument that was used in `tabscreen_gpt.original()`.

Usage

```
screen_errors.chatgpt(
  object,
  ...,
  api_key = get_api_key(),
  max_tries = 4,
  max_seconds,
  is_transient,
  backoff,
  after
)
```

Arguments

object	An object of class 'chatgpt'.
...	Further argument to pass to the request body. See https://developers.openai.com/api/reference/resources/chat. If used in the original screening (e.g., with tabscreen_gpt.original()), the argument(s) must be specified again here.
api_key	Numerical value with your personal API key.
max_tries, max_seconds	'Cap the maximum number of attempts with max_tries or the total elapsed time from the first request with max_seconds. If neither option is supplied (the default), httr2::req_perform() will not retry' (Wickham, 2023). Default max_tries is 4. If missing, the value of max_seconds from the original screening (e.g., conducted with tabscreen_gpt.original()) will be used.
is_transient	'A predicate function that takes a single argument (the response) and returns TRUE or FALSE specifying whether or not the response represents a transient error' (Wickham, 2023). If missing, the is_transient function from the original screening (e.g., conducted with tabscreen_gpt.original()) will be used.
backoff	'A function that takes a single argument (the number of failed attempts so far) and returns the number of seconds to wait' (Wickham, 2023). If missing, the backoffvalue from the original screening (e.g., conducted with tabscreen_gpt.original()) will be used.
after	'A function that takes a single argument (the response) and returns either a number of seconds to wait or NULL, which indicates that a precise wait time is not available that the backoff strategy should be used instead' (Wickham, 2023). If missing, the after value from the original screening (e.g., conducted with tabscreen_gpt.original()) will be used.

Value

Object of class 'chatgpt' similar to the object returned by tabscreen_gpt.original(). See documentation value for tabscreen_gpt.original().

References

Wickham H (2023). *httr2: Perform HTTP Requests and Process the Responses*. https://httr2.r-lib.org, https://github.com/r-lib/httr2.

See Also

tabscreen_gpt.original()

Examples

```
## Not run:

set_api_key()

prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

obj_with_error <-
  tabscreen_gpt(
    data = filges2015_dat[1:2,],
    prompt = prompt,
    studyid = studyid,
    title = title,
    abstract = abstract,
    model = c("gpt-3.5-turbo-0613", "gpt-3.5-turbo-16k-0613"),
    max_tries = 1,
    reps = 10
    )

obj_rescreened <-
  obj_with_error |>
  screen_error()

# Alternatively re-set max_tries if errors still appear
obj_rescreened <-
  obj_with_error |>
  screen_error(max_tries = 16)

## End(Not run)
```

---

screen_errors.gpt *Re-screen failed requests.*

Description

[Superseded]

This function supports re-screening of all failed title and abstract requests screened with tabscreen_gpt()/tabscreen_gpt.

Usage

```
screen_errors.gpt(
  object,
  api_key = get_api_key(),
  max_tries = 16,
  max_seconds,
  is_transient,
  backoff,
  after,
  ...
)
```

Arguments

object	An object of class 'gpt'.
api_key	Numerical value with your personal API key. Default setting draws on the get_api_key() to retrieve the API key from the R environment, so that the key is not compromised. The API key can be added to the R environment via set_api_key() or by using usethis::edit_r_environ(). In the .Renviron file, write CHATGPT_KEY=INSERT_YOUR_KEY_HERE. After entering the API key, close and save the .Renviron file and restart RStudio (ctrl + shift + F10). Alternatively, one can use httr2::secret_make_key(), httr2::secret_encrypt(), and httr2::secret_decrypt() to scramble and decrypt the API key.
max_tries, max_seconds	'Cap the maximum number of attempts with max_tries or the total elapsed time from the first request with max_seconds. If neither option is supplied (the default), httr2::req_perform() will not retry' (Wickham, 2023). Default max_tries is 16. If missing, the value of max_seconds from the original screening conducted with tabscreen_gpt() will be used.
is_transient	'A predicate function that takes a single argument (the response) and returns TRUE or FALSE specifying whether or not the response represents a transient error' (Wickham, 2023). If missing, the is_transient function from the original screening conducted with tabscreen_gpt() will be used.
backoff	'A function that takes a single argument (the number of failed attempts so far) and returns the number of seconds to wait' (Wickham, 2023). If missing, the backoffvalue from the original screening conducted with tabscreen_gpt() will be used.
after	'A function that takes a single argument (the response) and returns either a number of seconds to wait or NULL, which indicates that a precise wait time is not available that the backoff strategy should be used instead' (Wickham, 2023). If missing, the after value from the original screening conducted with tabscreen_gpt() will be used.
...	Further argument to pass to the request body. See https://developers.openai.com/api/reference/resources/chat. If used in the original screening in tabscreen_gpt(), the argument(s) must be specified again here.

Value

An object of class 'gpt' similar to the object returned by tabscreen_gpt(). See documentation for tabscreen_gpt().

References

Wickham H (2023). *httr2: Perform HTTP Requests and Process the Responses*. https://httr2.r-lib.org, https://github.com/r-lib/httr2.

See Also

tabscreen_gpt(), tabscreen_gpt.tools()

Examples

```
## Not run:
prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

obj_with_error <-
  tabscreen_gpt(
    data = filges2015_dat[1:10,],
    prompt = prompt,
    studyid = studyid,
    title = title,
    abstract = abstract,
    model = "gpt-4o"
    )

obj_rescreened <-
  obj_with_error |>
  screen_errors()


## End(Not run)
```

---

set_api_key *Creating a temporary R environment API key variable*

Description

This function automatically sets/creates an interim R environment variable with the API key to call a given AI model (e.g. ChatGPT). Thereby users avoid exposing their API keys. If the API key is set in the console, it will/can be revealed via the .Rhistory. Find more information about this issue at https://httr2.r-lib.org/articles/wrapping-apis.html.

Usage

```
set_api_key(key, env_var = "OPENAI_API_KEY")
```

Arguments

key	Character string with an (ideally encrypted) API key. See how to encrypt key here: https://httr2.r-lib.org/articles/wrapping-apis.html#basics. If not provided, it returns a password box in which the true API key can be secretly entered.
env_var	Character string indicating the name of the temporary R environment variable with the API key and the used AI model. Default is env_var = "OPENAI_API_KEY".

Details

When set_api_key() has successfully been executed, get_api_key() automatically retrieves the API key from the R environment and the users do not need to specify the API when running functions from the package that call the API. The API key can be permanently set by using usethis::edit_r_environ(). Then write OPENAI_API_KEY=[insert your api key here] and close the .Renviron window and restart R.

Value

A temporary environment variable with the name from env_var. If key is missing, it returns a password box in which the true API key can be entered.

Note

Find your personal API key via the OpenAI quickstart guide at https://developers.openai.com/api/docs/quickstart#generate-an-api-key.

See Also

get_api_key

Examples

```
## Not run:
set_api_key()

## End(Not run)
```

---

tabscreen_gpt.original

Title and abstract screening with GPT API models using function calls via the original function call arguments

---

Description

[Deprecated]

This function has been deprecated (but can still be used) because OpenAI has deprecated the function_call and and functions argument which is used in this function. Instead use the `tabscreen_gpt.tools()` that handles the function calling via the tools and tool_choice arguments.

This function supports the conduct of title and abstract screening with GPT API models in R. This function only works with GPT-4, more specifically gpt-4-0613. To draw on other models, use `tabscreen_gpt.tools()`. The function allows to run title and abstract screening across multiple prompts and with repeated questions to check for consistency across answers. This function draws on the newly developed function calling to better steer the output of the responses. This function was used in Vembye, Christensen, Mølgaard, and Schytt. (2025).

Usage

```
tabscreen_gpt.original(
  data,
  prompt,
  studyid,
  title,
  abstract,
  ...,
  model = "gpt-4",
  role = "user",
  functions = incl_function_simple,
  function_call_name = list(name = "inclusion_decision_simple"),
  top_p = 1,
  time_info = TRUE,
  token_info = TRUE,
  api_key = get_api_key(),
  max_tries = 16,
  max_seconds = NULL,
  is_transient = gpt_is_transient,
  backoff = NULL,
  after = NULL,
  rpm = 10000,
  reps = 1,
  seed_par = NULL,
  progress = TRUE,
  messages = TRUE,
  incl_cutoff_upper = 0.5,
  incl_cutoff_lower = incl_cutoff_upper - 0.1,
  force = FALSE
)
```

Arguments

data	Dataset containing the titles and abstracts.
prompt	Prompt(s) to be added before the title and abstract.

studyid — Unique Study ID. If missing, this is generated automatically.

title — Name of the variable containing the title information.

abstract — Name of variable containing the abstract information.

... — Further argument to pass to the request body. See https://developers.openai.com/api/reference/resources/chat.

model — Character string with the name of the completion model. Can take multiple models, including gpt-4 models. Default = "gpt-4" (i.e., gpt-4-0613). This model has been shown to outperform the gpt-3.5-turbo models in terms of its ability to detect relevant studies (Vembye et al., Under preparation). Find available model at https://developers.openai.com/api/docs/models/model-endpoint-compatibility.

role — Character string indicate the role of the user. Default is "user".

functions — Function to steer output. Default is incl_function_simple. To get detailed responses use the hidden function call incl_function from the package. Also see 'Examples below. Find further documentation for function calling at https://developers.openai.com/api/reference/resources/chat#chat-create-tools.

function_call_name — Functions to call. Default is list(name = "inclusion_decision_simple"). To get detailed responses use list(name = "inclusion_decision"). Also see 'Examples below.

top_p — 'An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered. We generally recommend altering this or temperature but not both.' (OPEN-AI). Default is 1. Find documentation at https://developers.openai.com/api/reference/resources/chat#chat/create-top_p.

time_info — Logical indicating whether the run time of each request/question should be included in the data. Default = TRUE.

token_info — Logical indicating whether the number of prompt and completion tokens per request should be included in the output data. Default = TRUE. When TRUE, the output object will include price information of the conducted screening.

api_key — Numerical value with your personal API key. Find setup guidance at https://developers.openai.com/api/docs/quickstart#generate-an-api-key. Use httr2::secret_make_key(), httr2::secret_encrypt(), and httr2::secret_decrypt() to scramble and decrypt the api key and use set_api_key() to securely automate the use of the api key by setting the api key as a locale environment variable.

max_tries, max_seconds — 'Cap the maximum number of attempts with max_tries or the total elapsed time from the first request with max_seconds. If neither option is supplied (the default), httr2::req_perform() will not retry' (Wickham, 2023).

is_transient — 'A predicate function that takes a single argument (the response) and returns TRUE or FALSE specifying whether or not the response represents a transient error' (Wickham, 2023).

backoff — 'A function that takes a single argument (the number of failed attempts so far) and returns the number of seconds to wait' (Wickham, 2023).

`after` 'A function that takes a single argument (the response) and returns either a number of seconds to wait or `NULL`, which indicates that a precise wait time is not available that the `backoff` strategy should be used instead' (Wickham, 2023).

`rpm` Numerical value indicating the number of requests per minute (rpm) available for the specified api key. Find more information at https://developers.openai.com/api/docs/models/model-endpoint-compatibility. Alternatively, use `rate_limits_per_minute()`.

`reps` Numerical value indicating the number of times the same question should be sent to OpenAI's GPT API models. This can be useful to test consistency between answers. Default is `1` but when using 3.5 models, we recommend setting this value to `10`.

`seed_par` Numerical value for a seed to ensure that proper, parallel-safe random numbers are produced.

`progress` Logical indicating whether a progress line should be shown when running the title and abstract screening in parallel. Default is `TRUE`.

`messages` Logical indicating whether to print messages embedded in the function. Default is `TRUE`.

`incl_cutoff_upper` Numerical value indicating the probability threshold for which a studies should be included. Default is 0.5, which indicates that titles and abstracts that OpenAI's GPT API model has included more than 50 percent of the times should be included.

`incl_cutoff_lower` Numerical value indicating the probability threshold above which studies should be check by a human. Default is 0.4, which means that if you ask OpenAI's GPT API model the same questions 10 times and it includes the title and abstract 4 times, we suggest that the study should be check by a human.

`force` Logical argument indicating whether to force the function to use more than 10 iterations for gpt-3.5 models and more than 1 iteration for gpt-4 models. This argument is developed to avoid the conduct of wrong and extreme sized screening. Default is `FALSE`.

Value

An object of class `"chatgpt"`. The object is a list containing the following components:

`answer_data_sum` dataset with the summarized, probabilistic inclusion decision for each title and abstract across multiple repeated questions.

`answer_data_all` dataset with all individual answers.

`price` numerical value indicating the total price (in USD) of the screening.

`price_data` dataset with prices across all gpt models used for screening.

Note

The `answer_data_sum` data contains the following mandatory variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific gpt-model used.
question	`character`	indicating the final question sent to OpenAI's GPT API models.
top_p	`numeric`	indicating the applied top_p.
incl_p	`numeric`	indicating the probability of inclusion calculated across multiple repeated responses
final_decision_gpt	`character`	indicating the final decision reached by gpt - either 'Include', 'Exclude', or 'Check'
final_decision_gpt_num	`integer`	indicating the final numeric decision reached by gpt - either 1 or 0.
longest_answer	`character`	indicating the longest gpt response obtained across multiple repeated responses on t
reps	`integer`	indicating the number of times the same question has been sent to OpenAI's GPT A
n_mis_answers	`integer`	indicating the number of missing responses.

The `answer_data_all` data contains the following mandatory variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific gpt-model used.
iterations	`numeric`	indicating the number of times the same question has been sent to OpenAI's GPT API r
question	`character`	indicating the final question sent to OpenAI's GPT API models.
top_p	`numeric`	indicating the applied top_p.
decision_gpt	`character`	indicating the raw gpt decision - either `"1"`, `"0"`, `"1.1"` for inclusion, exclusion, or un
detailed_description	`character`	indicating detailed description of the given decision made by OpenAI's GPT API mode
decision_binary	`integer`	indicating the binary gpt decision, that is 1 for inclusion and 0 for exclusion. 1.1 decisio
prompt_tokens	`integer`	indicating the number of prompt tokens sent to the server for the given request.
completion_tokens	`integer`	indicating the number of completion tokens sent to the server for the given request.
run_time	`numeric`	indicating the time it took to obtain a response from the server for the given request.
n	`integer`	indicating request ID.

If any requests failed to reach the server, the `chatgpt` object contains an error data set (`error_data`) having the same variables as `answer_data_all` but with failed request references only.

The `price_data` data contains the following variables:

model	`character`	gpt model.
input_price_dollar	`integer`	price for all prompt/input tokens for the correspondent gpt-model.
output_price_dollar	`integer`	price for all completion/output tokens for the correspondent gpt-model.
price_total_dollar	`integer`	total price for all tokens for the correspondent gpt-model.

Find current token pricing at https://developers.openai.com/api/docs/pricing.

References

Vembye, M. H., Christensen, J., Mølgaard, A. B., & Schytt, F. L. W. (2025) *GPT API Models Can Function as Highly Reliable Second Screeners of Titles and Abstracts in Systematic Reviews: A Proof of Concept and Common Guidelines* https://psycnet.apa.org/record/2026-37236-001

Wickham H (2023). *httr2: Perform HTTP Requests and Process the Responses*. https://httr2.r-lib.org, https://github.com/r-lib/httr2.

Examples

```
## Not run:

set_api_key()

prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

tabscreen_gpt.original(
  data = filges2015_dat[1:2,],
  prompt = prompt,
  studyid = studyid,
  title = title,
  abstract = abstract,
  max_tries = 2
  )

 # Get detailed descriptions of the gpt decisions by using the
 # embedded function calling functions from the package. See example below.
 tabscreen_gpt.original(
   data = filges2015_dat[1:2,],
   prompt = prompt,
   studyid = studyid,
   title = title,
   abstract = abstract,
   functions = incl_function,
   function_call_name = list(name = "inclusion_decision"),
   max_tries = 2
 )

## End(Not run)
```

`tabscreen_gpt.tools` *Title and abstract screening with GPT API models using function calls via the tools argument*

Description

[Stable]

This function supports title and abstract screening using GPT API models in R. Specifically, it allows users to draw on all OpenAI GPT API completion models, including fine-tuned versions. The function enables title and abstract screening across multiple prompts, with repeated questions to assess consistency across responses. All of this can be performed in parallel. The function utilizes function calling, which is invoked via the tools argument in the request body. This is the main difference between `tabscreen_gpt.tools()` and `tabscreen_gpt.original()`. Function calls ensure more reliable and consistent responses to users' requests. See Vembye, Christensen, Mølgaard, and Schytt. (2025) for guidance on how adequately to conduct title and abstract screening with GPT models.

Usage

```
tabscreen_gpt.tools(data, prompt, studyid, title, abstract,
  api_url = "https://api.openai.com/v1/chat/completions", model = "gpt-4o-mini",
  role = "user", tools = NULL, tool_choice = NULL, top_p = 1,
  time_info = TRUE, token_info = TRUE, api_key = get_api_key(), max_tries = 16,
  max_seconds = NULL, is_transient = gpt_is_transient, backoff = NULL,
  after = NULL, rpm = 10000, reps = 1, seed_par = NULL, progress = TRUE,
  decision_description = FALSE, messages = TRUE, incl_cutoff_upper = NULL,
  incl_cutoff_lower = NULL, force = FALSE, custom_model = FALSE,
  fine_tuned = deprecated(), reasoning_effort = "medium", verbosity = "low",
  overinclusive = TRUE, ...)

tabscreen_gpt(data, prompt, studyid, title, abstract,
  api_url = "https://api.openai.com/v1/chat/completions", model = "gpt-4o-mini",
  role = "user", tools = NULL, tool_choice = NULL, top_p = 1,
  time_info = TRUE, token_info = TRUE, api_key = get_api_key(), max_tries = 16,
  max_seconds = NULL, is_transient = gpt_is_transient, backoff = NULL,
  after = NULL, rpm = 10000, reps = 1, seed_par = NULL, progress = TRUE,
  decision_description = FALSE, messages = TRUE, incl_cutoff_upper = NULL,
  incl_cutoff_lower = NULL, force = FALSE, custom_model = FALSE,
  fine_tuned = deprecated(), reasoning_effort = "medium", verbosity = "low",
  overinclusive = TRUE, ...)
```

Arguments

`data`	Dataset containing the titles and abstracts.
`prompt`	Prompt(s) to be added before the title and abstract.
`studyid`	Unique Study ID. If missing, this is generated automatically.
`title`	Name of the variable containing the title information.
`abstract`	Name of variable containing the abstract information.
`api_url`	Character string with the endpoint URL for OpenAI's API. Default is `"https://api.openai.com/v1/ch`
`model`	Character string with the name of the completion model. Can take multiple models. Default is the latest `"gpt-4o-mini"`. Find available model at `https://developers.openai.com/api/docs/models/model-endpoint-compatibility`.

`role` Character string indicating the role of the user. Default is `"user"`.

`tools` This argument allows this user to apply customized functions. See https://developers.openai.com/api/reference/resources/chat#chat-create-tools. Default is `NULL`. If not specified the default function calls from `AIscreenR` are used.

`tool_choice` If a customized function is provided this argument 'controls which (if any) tool is called by the model' (OpenAI). Default is `NULL`. If set to `NULL` when using a customized function, the default is `"auto"`. See https://developers.openai.com/api/reference/resources/chat#chat-create-tool_choice.

`top_p` 'An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered. We generally recommend altering this or temperature but not both.' (OpenAI). Default is 1. Find documentation at https://developers.openai.com/api/reference/resources/chat#chat/create-top_p.

`time_info` Logical indicating whether the run time of each request/question should be included in the data. Default is `TRUE`.

`token_info` Logical indicating whether token information should be included in the output data. Default is `TRUE`. When `TRUE`, the output object will include price information of the conducted screening.

`api_key` Numerical value with your personal API key. Default setting draws on the `get_api_key()` to retrieve the API key from the R environment, so that the key is not compromised. The API key can be added to the R environment via `set_api_key()` or by using `usethis::edit_r_environ()`. In the `.Renviron` file, write `CHATGPT_KEY=INSERT_YOUR_KEY_HERE`. After entering the API key, close and save the `.Renviron` file and restart `RStudio` (ctrl + shift + F10). Alternatively, one can use `httr2::secret_make_key()`, `httr2::secret_encrypt()`, and `httr2::secret_decrypt()` to scramble and decrypt the API key.

`max_tries, max_seconds` 'Cap the maximum number of attempts with `max_tries` or the total elapsed time from the first request with `max_seconds`. If neither option is supplied (the default), `httr2::req_perform()` will not retry' (Wickham, 2023). The default of `max_tries` is 16.

`is_transient` 'A predicate function that takes a single argument (the response) and returns `TRUE` or `FALSE` specifying whether or not the response represents a transient error' (Wickham, 2023). This function runs automatically in the AIscreenR but can be customized by the user if necessary.

`backoff` 'A function that takes a single argument (the number of failed attempts so far) and returns the number of seconds to wait' (Wickham, 2023).

`after` 'A function that takes a single argument (the response) and returns either a number of seconds to wait or `NULL`, which indicates that a precise wait time is not available that the `backoff` strategy should be used instead' (Wickham, 2023).

`rpm` Numerical value indicating the number of requests per minute (rpm) available for the specified model. Find more information at https://developers.openai.com/api/docs/models/model-endpoint-compatibility. Alternatively, use `rate_limits_per_minute()`.

reps
: Numerical value indicating the number of times the same question should be send to the server. This can be useful to test consistency between answers, and/or can be used to make inclusion judgments based on how many times a study has been included across a the given number of screenings. Default is 1 but when using gpt-3.5-turbo models or gpt-4o-mini, we recommend setting this value to 10 to catch model uncertainty.

seed_par
: Numerical value for a seed to ensure that proper, parallel-safe random numbers are produced.

progress
: Logical indicating whether a progress line should be shown when running the title and abstract screening in parallel. Default is TRUE.

decision_description
: Logical indicating whether a detailed description should follow the decision made by GPT. Default is FALSE. When conducting large-scale screening, we generally recommend not using this feature as it will substantially increase the cost of the screening. We generally recommend using it when encountering disagreements between GPT and human decisions.

messages
: Logical indicating whether to print messages embedded in the function. Default is TRUE.

incl_cutoff_upper
: Numerical value indicating the probability threshold for which a studies should be included. ONLY relevant when the same questions is requested multiple times (i.e., when any reps > 1). Default is 0.1, indicating that titles and abstracts should only be included if GPT has included the study more than 10 percent of the times (e.g., 1 out of 10 screenings). This has been shown by Vembye et al. (2025) to work well with cheaper models.

incl_cutoff_lower
: Numerical value indicating the probability threshold above which studies should be checked by a human. ONLY relevant when the same questions is requested multiple times (i.e., when any reps > 1) and incl_cutoff_upper > 0.1. Records with inclusion probabilities between incl_cutoff_lower and incl_cutoff_upper will be flagged for human checking. Default is NULL, which means that no studies will be flagged for human checking.

force
: Logical argument indicating whether to force the function to use more than 10 iterations for gpt-3.5 models and more than 1 iteration for gpt-4 models other than gpt-4o-mini. This argument is developed to avoid the conduct of wrong and extreme sized screening. Default is FALSE.

custom_model
: Logical indicating whether a fine-tuned or custom model is used. Default is FALSE.

fine_tuned
: **[Deprecated]** Use custom_model instead.

reasoning_effort
: Character string indicating the level of reasoning effort required for the task. Default is "low". Can take the values "low", "medium", and "high". See https://developers.openai.com/api/docs/guides/reasoning for more information.

verbosity
: Character string indicating the level of verbosity in the model's responses. Default is "low". Can take the values "low", "medium", and "high". See https:

//developers.openai.com/api/reference/resources/chat for more information.

`overinclusive` Logical indicating whether uncertain decisions (`"1.1"`) should be allowed in the default function calling setup. Default is `TRUE`, which means that the default function calling setup will allow for uncertain decisions. If `FALSE`, the default function calling setup will not allow for uncertain decisions and will only return binary decisions (i.e., "1" or "0"). This argument only affects the default function calling setup.

`...` Further argument to pass to the request body. See https://developers.openai.com/api/reference/resources/chat.

Value

An object of class `'gpt'`. The object is a list containing the following datasets and components:

`answer_data` dataset of class `'gpt_tbl'` with all individual answers.

`price_dollar` numerical value indicating the total price (in USD) of the screening.

`price_data` dataset with prices across all gpt models used for screening.

`run_date` string indicating the date when the screening was ran. In some frameworks, time details are considered important to report (see e.g., Thomas et al., 2024).

`...` some additional attributed values/components, including an attributed list with the arguments used in the function. These are used in `screen_errors()` to re-screen transient errors.

If the same question is requested multiple times, the object will also contain the following dataset with results aggregated across the iterated requests/questions.

`answer_data_aggregated`
dataset of class `'gpt_agg_tbl'` with the summarized, probabilistic inclusion decision for each title and abstract across multiple repeated questions.

Note

The `answer_data` data contains the following *mandatory* variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific gpt-model used.
iterations	`numeric`	indicating the number of times the same question has been sent to OpenAI's GPT API r
question	`character`	indicating the final question sent to OpenAI's GPT API models.
top_p	`numeric`	indicating the applied top_p.
decision_gpt	`character`	indicating the raw gpt decision - either `"1"`, `"0"`, `"1.1"` for inclusion, exclusion, or un
detailed_description	`character`	indicating detailed description of the given decision made by OpenAI's GPT API mode
decision_binary	`integer`	indicating the binary gpt decision, that is 1 for inclusion and 0 for exclusion. 1.1 decisio
prompt_tokens	`integer`	indicating the number of prompt tokens sent to the server for the given request.

completion_tokens	`integer`	indicating the number of completion tokens sent to the server for the given request.
submodel	`character`	indicating the exact (sub)model used for screening.
run_time	`numeric`	indicating the time it took to obtain a response from the server for the given request.
run_date	`character`	indicating the date the given response was received.
n	`integer`	indicating iteration ID. Is only different from 1, when `reps` > 1.

If any requests failed, the `gpt` object contains an error dataset (`error_data`) containing the same variables as `answer_data` but with failed request references only.

When the same question is requested multiple times, the `answer_data_aggregated` data contains the following *mandatory* variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific gpt-model used.
question	`character`	indicating the final question sent to OpenAI's GPT API models.
top_p	`numeric`	indicating the applied top_p.
incl_p	`numeric`	indicating the probability of inclusion calculated across multiple repeated responses
final_decision_gpt	`character`	indicating the final decision reached by gpt - either 'Include', 'Exclude', or 'Check'
final_decision_gpt_num	`integer`	indicating the final numeric decision reached by gpt - either 1 or 0.
longest_answer	`character`	indicating the longest gpt response obtained across multiple repeated responses on t
reps	`integer`	indicating the number of times the same question has been sent to OpenAI's GPT A
n_mis_answers	`integer`	indicating the number of missing responses.
submodel	`character`	indicating the exact (sub)model used for screening.

The `price_data` data contains the following variables:

prompt	`character`	if multiple prompts are used this variable indicates the given prompt-id.
model	`character`	the specific gpt model used.
iterations	`integer`	indicating the number of times the same question was requested.
input_price_dollar	`integer`	price for all prompt/input tokens for the correspondent gpt-model.
output_price_dollar	`integer`	price for all completion/output tokens for the correspondent gpt-model.
total_price_dollar	`integer`	total price for all tokens for the correspondent gpt-model.

Find current token pricing at `https://developers.openai.com/api/docs/pricing` or model_prizes.

References

Vembye, M. H., Christensen, J., Mølgaard, A. B., & Schytt, F. L. W. (2025). Generative Pretrained Transformer Models Can Function as Highly Reliable Second Screeners of Titles and Abstracts in Systematic Reviews: A Proof of Concept and Common Guidelines. *Psychological Methods*. doi:10.1037/met0000769

Thomas, J. et al. (2024). Responsible AI in Evidence SynthEsis (RAISE): guidance and recommendations. https://osf.io/cn7x4

Wickham H (2023). *httr2: Perform HTTP Requests and Process the Responses*. https://httr2.r-lib.org, https://github.com/r-lib/httr2.

Examples

```
## Not run:

library(future)

set_api_key()

prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

plan(multisession)

tabscreen_gpt(
  data = filges2015_dat[1:2,],
  prompt = prompt,
  studyid = studyid,
  title = title,
  abstract = abstract
  )

plan(sequential)

 # Get detailed descriptions of the gpt decisions.

 plan(multisession)

 tabscreen_gpt(
   data = filges2015_dat[1:2,],
   prompt = prompt,
   studyid = studyid,
   title = title,
   abstract = abstract,
   decision_description = TRUE
 )

plan(sequential)


## End(Not run)
```

`tabscreen_groq` *Title and abstract screening with GROQ API models using function calls via the tools argument*

Description

This function supports the conduct of title and abstract screening with Groq API models in R. Specifically, it allows the user to draw on Groq-hosted models. The function allows to run title and abstract screening across multiple prompts and with repeated questions to check for consistency across answers. All of which can be done in parallel. The function draws on function calling which is called via the tools argument in the request body. Function calls ensure more reliable and consistent responses to ones requests. See Vembye, Christensen, Mølgaard, and Schytt. (2025) for guidance on how adequately to conduct title and abstract screening with GPT models.

Usage

```
tabscreen_groq(data, prompt, studyid, title, abstract,
api_url = "https://api.groq.com/openai/v1/chat/completions",
 ..., model = "llama-3.1-8b-instant", role = "user",
tools = NULL, tool_choice = NULL, top_p = 1,
time_info = TRUE, token_info = TRUE, api_key = get_api_key_groq(),
max_tries = 16, max_seconds = NULL, is_transient = .groq_is_transient,
backoff = NULL, after = NULL, rpm = 10000, reps = 1, seed_par = NULL,
progress = TRUE, decision_description = FALSE, overinclusive = TRUE,
messages = TRUE, incl_cutoff_upper = NULL, incl_cutoff_lower = NULL,
force = FALSE)
```

Arguments

data	Dataset containing the titles and abstracts.
prompt	Prompt(s) to be added before the title and abstract.
studyid	Unique Study ID. If missing, this is generated automatically.
title	Name of the variable containing the title information.
abstract	Name of variable containing the abstract information.
api_url	Character string with the endpoint URL for Groq's API. Default is "https://api.groq.com/openai/v1/
...	Further argument to pass to the request body.
model	Character string with the name of the completion model. Can take multiple Groq models. Default = "llama3-70b-8192". Find available models at https://console.groq.com/docs/models.
role	Character string indicate the role of the user. Default is "user".
tools	List of function definitions for tool calling. Default behavior is set based on decision_description parameter. For detailed responses, the function uses tools that include detailed description capabilities.
tool_choice	Specification for which tool to use. Default behavior is set based on decision_description parameter. For simple responses uses "inclusion_decision_simple", for detailed responses uses "inclusion_decision".
top_p	'An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered. We generally recommend altering this or temperature but not both.' (Groq). Default is 1.

`time_info` Logical indicating whether the run time of each request/question should be included in the data. Default = TRUE.

`token_info` Logical indicating whether the number of prompt and completion tokens per request should be included in the output data. Default = TRUE. When TRUE, the output object will include price information of the conducted screening.

`api_key` Numerical value with your personal API key. Find at https://console.groq.com/keys. Set with `Sys.setenv(GROQ_API_KEY = "your-api-key")` or use `get_api_key_groq()`.

`max_tries, max_seconds`
'Cap the maximum number of attempts with `max_tries` or the total elapsed time from the first request with `max_seconds`. If neither option is supplied (the default), `httr2::req_perform()` will not retry'. (Wickham, 2023). The default of `max_tries` is 16.

`is_transient` 'A predicate function that takes a single argument (the response) and returns TRUE or FALSE specifying whether or not the response represents a transient error' (Wickham, 2023). This function runs automatically in the AIscreenR but can be customized by the user if necessary.

`backoff` 'A function that takes a single argument (the number of failed attempts so far) and returns the number of seconds to wait' (Wickham, 2023).

`after` 'A function that takes a single argument (the response) and returns either a number of seconds to wait or NULL, which indicates that a precise wait time is not available that the `backoff` strategy should be used instead' (Wickham, 2023).

`rpm` Numerical value indicating the number of requests per minute (rpm) available for the specified model.

`reps` Numerical value indicating the number of times the same question should be sent to Groq's API models. This can be useful to test consistency between answers. Default is 1.

`seed_par` Numerical value for a seed to ensure that proper, parallel-safe random numbers are produced.

`progress` Logical indicating whether a progress line should be shown when running the title and abstract screening in parallel. Default is TRUE.

`decision_description`
Logical indicating whether to include detailed descriptions of decisions. Default is FALSE. When conducting large-scale screening, we generally recommend not using this feature as it will substantially increase the cost of the screening. We generally recommend using it when encountering disagreements between GPT and human decisions.

`overinclusive` Logical indicating whether uncertain decisions (`"1.1"`) should be allowed in the default function calling setup. Default is TRUE, which means that the default function calling setup will allow for uncertain decisions. If FALSE, the default function calling setup will not allow for uncertain decisions and will only return binary decisions (i.e., "1" or "0"). This argument only affects the default function calling setup.

`messages` Logical indicating whether to print messages embedded in the function. Default is TRUE.

`incl_cutoff_upper`
: Numerical value indicating the probability threshold for which a studie should be included. ONLY relevant when the same questions is requested multiple times (i.e., when any reps > 1). Default is 0.5, which indicates that titles and abstracts that Groq's API model has included more than 50 percent of the times should be included.

`incl_cutoff_lower`
: Numerical value indicating the probability threshold above which studies should be check by a human. ONLY relevant when the same questions is requested multiple times (i.e., when any reps > 1). Default is 0.4, which means that if you ask Groq's API model the same questions 10 times and it includes the title and abstract 4 times, we suggest that the study should be check by a human.

`force`
: Logical argument indicating whether to force the function to use more than 10 iterations. This argument is developed to avoid the conduct of wrong and extreme sized screening. Default is `FALSE`.

Value

An object of class `"gpt"`. The object is a list containing the following components:

`answer_data_aggregated`
: dataset with the summarized, probabilistic inclusion decision for each title and abstract across multiple repeated questions (only when reps > 1).

`answer_data`
: dataset with all individual answers.

`price_dollar`
: numerical value indicating the total price (in USD) of the screening.

`price_data`
: dataset with prices across all models used for screening.

`error_data`
: dataset with failed requests (only included if errors occurred).

`run_date`
: date when the screening was conducted.

Note

The `answer_data_aggregated` data (only present when reps > 1) contains the following mandatory variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific model used.
question	`character`	indicating the final question sent to Groq's API models.
top_p	`numeric`	indicating the applied top_p.
incl_p	`numeric`	indicating the probability of inclusion calculated across multiple repeated responses
final_decision_gpt	`character`	indicating the final decision reached by model - either 'Include', 'Exclude', or 'Che
final_decision_gpt_num	`integer`	indicating the final numeric decision reached by model - either 1 or 0.
longest_answer	`character`	indicating the longest response obtained across multiple repeated responses on the s
reps	`integer`	indicating the number of times the same question has been sent to Groq's API mode
n_mis_answers	`integer`	indicating the number of missing responses.

The `answer_data` data contains the following mandatory variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific model used.
iterations	`numeric`	indicating the number of times the same question has been sent to Groq's API models.
question	`character`	indicating the final question sent to Groq's API models.
top_p	`numeric`	indicating the applied top_p.
decision_gpt	`character`	indicating the raw decision - either `"1"`, `"0"`, `"1.1"` for inclusion, exclusion, or uncert
detailed_description	`character`	indicating detailed description of the given decision made by Groq's API models. Only
decision_binary	`integer`	indicating the binary decision, that is 1 for inclusion and 0 for exclusion. 1.1 decision a
prompt_tokens	`integer`	indicating the number of prompt tokens sent to the server for the given request.
completion_tokens	`integer`	indicating the number of completion tokens sent to the server for the given request.
run_time	`numeric`	indicating the time it took to obtain a response from the server for the given request.
n	`integer`	indicating request ID.

If any requests failed to reach the server, the object contains an error data set (`error_data`) having the same variables as `answer_data` but with failed request references only.

The `price_data` data contains the following variables:

model	`character`	model name.
input_price_dollar	`integer`	price for all prompt/input tokens for the correspondent model.
output_price_dollar	`integer`	price for all completion/output tokens for the correspondent model.
price_total_dollar	`integer`	total price for all tokens for the correspondent model.

Find current token pricing at https://groq.com/pricing.

References

Vembye, M. H., Christensen, J., Mølgaard, A. B., & Schytt, F. L. W. (2025). Generative Pretrained Transformer Models Can Function as Highly Reliable Second Screeners of Titles and Abstracts in Systematic Reviews: A Proof of Concept and Common Guidelines. *Psychological Methods*. doi:10.1037/met0000769

Thomas, J. et al. (2024). Responsible AI in Evidence SynthEsis (RAISE): guidance and recommendations. https://osf.io/cn7x4

Wickham H (2023). *httr2: Perform HTTP Requests and Process the Responses*. https://httr2.r-lib.org, https://github.com/r-lib/httr2.

Examples

```
## Not run:

set_api_key_groq()

prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

plan(multisession)

tabscreen_groq(
  data = filges2015_dat[1:2,],
  prompt = prompt,
  studyid = studyid,
  title = title,
  abstract = abstract,
  model = "llama3-70b-8192",
  max_tries = 2
  )
plan(sequential)

 # Get detailed descriptions of the decisions by using the
 # decision_description option.
plan(multisession)

 tabscreen_groq(
   data = filges2015_dat[1:2,],
   prompt = prompt,
   studyid = studyid,
   title = title,
   abstract = abstract,
   model = "llama3-70b-8192",
   decision_description = TRUE,
   max_tries = 2
 )
plan(sequential)

## End(Not run)
```

`tabscreen_ollama` *Title and abstract screening with OLLAMA API models using function calls via the tools argument*

Description

This function supports the conduct of title and abstract screening with OLLAMA API models in R. Specifically, it allows the user to draw on locally hosted ollama models (e.g., Llama 3 / 3.1 variants, Mixtral/Mistral, Gemma, DeepSeek and Qwen). For more information on how to install and use OLLAMA, see https://docs.ollama.com/. Be aware that this function requires that you have OLLAMA installed and running on your local machine. The function allows to run title and abstract

screening across multiple prompts and with repeated questions to check for consistency across answers. All of which can be done in parallel. The function draws on the newly developed function calling which is called via the tools argument in the request body. Function calls ensure more reliable and consistent responses to ones requests. See Vembye, Christensen, Mølgaard, and Schytt. (2025) for guidance on how adequately to conduct title and abstract screening with OLLAMA models.

Usage

```
tabscreen_ollama(data, prompt, studyid, title, abstract,
api_url = "http://127.0.0.1:11434/api/chat", ..., model, role = "user",
tools = NULL, tool_choice = NULL, top_p = 1, time_info = TRUE,
max_tries = 16, max_seconds = NULL, backoff = NULL, after = NULL,
reps = 1, seed_par = NULL, progress = TRUE, decision_description = FALSE,
overinclusive = TRUE, messages = TRUE, incl_cutoff_upper = NULL,
incl_cutoff_lower = NULL, force = FALSE)
```

Arguments

data	Dataset containing the titles and abstracts.
prompt	Prompt(s) to be added before the title and abstract.
studyid	Unique Study ID. If missing, this is generated automatically.
title	Name of the variable containing the title information.
abstract	Name of variable containing the abstract information.
api_url	Character string with the endpoint URL for OLLAMA's API. Default is "http://127.0.0.1:11434/api
...	Further argument to pass to the request body.
model	Character string with the name of the OLLAMA model. Can take multiple OLLAMA models. Default = "llama3.2:latest". Find available models at https://ollama.com/library.
role	Character string indicate the role of the user. Default is "user".
tools	List of function definitions for tool calling. Default behavior is set based on decision_description parameter. For detailed responses, the function uses tools that include detailed description capabilities.
tool_choice	Specification for which tool to use. Default behavior is set based on decision_description parameter. For simple responses uses "inclusion_decision_simple", for detailed responses uses "inclusion_decision".
top_p	An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with top_p probability mass. So 0.1 means only the tokens comprising the top 10% probability mass are considered. We generally recommend altering this or temperature but not both. Default is 1.
time_info	Logical indicating whether the run time of each request/question should be included in the data. Default = TRUE.
max_tries, max_seconds	Cap the maximum number of attempts with max_tries or the total elapsed time from the first request with max_seconds. Default for max_tries is 16. If max_tries is not supplied, httr2::req_perform() will not retry.

backoff A function that takes a single argument (the number of failed attempts so far) and returns the number of seconds to wait.

after A function that takes a single argument (the response) and returns either a number of seconds to wait or NULL, which indicates that a precise wait time is not available that the backoff strategy should be used instead.

reps Numerical value indicating the number of times the same question should be sent to OLLAMA models. This can be useful to test consistency between answers. Default is 1.

seed_par Numerical value for a seed to ensure that proper, parallel-safe random numbers are produced.

progress Logical indicating whether a progress line should be shown when running the title and abstract screening in parallel. Default is TRUE.

decision_description
Logical indicating whether to include detailed descriptions of decisions. Default is FALSE. When conducting large-scale screening, we generally recommend not using this feature as it will substantially increase the time of the screening.

overinclusive Logical indicating whether uncertain decisions ("1.1") should be allowed in the default function calling setup. Default is TRUE, which means that the default function calling setup will allow for uncertain decisions. If FALSE, the default function calling setup will not allow for uncertain decisions and will only return binary decisions (i.e., "1" or "0"). This argument only affects the default function calling setup.

messages Logical indicating whether to print messages embedded in the function. Default is TRUE.

incl_cutoff_upper
Numerical value indicating the probability threshold for which a studies should be included. Default is 0.5, which indicates that titles and abstracts that the OLLAMA model has included more than 50 percent of the times should be included.

incl_cutoff_lower
Numerical value indicating the probability threshold above which studies should be check by a human. Default is 0.4, which means that if you ask the OLLAMA model the same questions 10 times and it includes the title and abstract 4 times, we suggest that the study should be check by a human.

force Logical argument indicating whether to force the function to use more than 10 iterations. This argument is developed to avoid the conduct of wrong and extreme sized screening. Default is FALSE.

Value

An object of class "gpt". The object is a list containing the following components:

answer_data_aggregated
dataset with the summarized, probabilistic inclusion decision for each title and abstract across multiple repeated questions (only when reps > 1).

answer_data dataset with all individual answers.

error_data dataset with failed requests (only included if errors occurred).

run_date date when the screening was conducted.

Note

The `answer_data_aggregated` data (only present when reps > 1) contains the following mandatory variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific model used.
question	`character`	indicating the final question sent to OLLAMA models.
top_p	`numeric`	indicating the applied top_p.
incl_p	`numeric`	indicating the probability of inclusion calculated across multiple repeated responses
final_decision_gpt	`character`	indicating the final decision reached by model - either 'Include', 'Exclude', or 'Che
final_decision_gpt_num	`integer`	indicating the final numeric decision reached by model - either 1 or 0.
longest_answer	`character`	indicating the longest response obtained across multiple repeated responses on the s
reps	`integer`	indicating the number of times the same question has been sent to OLLAMA model
n_mis_answers	`integer`	indicating the number of missing responses.

The `answer_data` data contains the following mandatory variables:

studyid	`integer`	indicating the study ID of the reference.
title	`character`	indicating the title of the reference.
abstract	`character`	indicating the abstract of the reference.
promptid	`integer`	indicating the prompt ID.
prompt	`character`	indicating the prompt.
model	`character`	indicating the specific model used.
iterations	`numeric`	indicating the number of times the same question has been sent to OLLAMA models.
question	`character`	indicating the final question sent to OLLAMA models.
top_p	`numeric`	indicating the applied top_p.
decision_gpt	`character`	indicating the raw decision - either `"1"`, `"0"`, `"1.1"` for inclusion, exclusion, or uncert
detailed_description	`character`	indicating detailed description of the given decision made by OLLAMA models. Only i
decision_binary	`integer`	indicating the binary decision, that is 1 for inclusion and 0 for exclusion. 1.1 decision a
run_time	`numeric`	indicating the time it took to obtain a response from the server for the given request.
n	`integer`	indicating request ID.

If any requests failed to reach the server, the object contains an error data set (`error_data`) having the same variables as `answer_data` but with failed request references only.

References

Vembye, M. H., Christensen, J., Mølgaard, A. B., & Schytt, F. L. W. (2025). Generative Pretrained Transformer Models Can Function as Highly Reliable Second Screeners of Titles and Abstracts in Systematic Reviews: A Proof of Concept and Common Guidelines. *Psychological Methods*. doi:10.1037/met0000769

Thomas, J. et al. (2024). Responsible AI in Evidence SynthEsis (RAISE): guidance and recommendations. https://osf.io/cn7x4

Wickham H (2023). *httr2: Perform HTTP Requests and Process the Responses*. https://httr2.r-lib.org, https://github.com/r-lib/httr2.

Examples

```
## Not run:


prompt <- "Is this study about a Functional Family Therapy (FFT) intervention?"

plan(multisession)

tabscreen_ollama(
  data = filges2015_dat[1:2,],
  prompt = prompt,
  studyid = studyid,
  title = title,
  abstract = abstract,
  model = "llama3.2:latest",
  max_tries = 2
  )
plan(sequential)

 # Get detailed descriptions of the decisions by using the
 # decision_description option.
plan(multisession)

 tabscreen_ollama(
   data = filges2015_dat[1:2,],
   prompt = prompt,
   studyid = studyid,
   title = title,
   abstract = abstract,
   model = "llama3.2:latest",
   decision_description = TRUE,
   max_tries = 2
 )
plan(sequential)

## End(Not run)
```

Index