Package ‘askgpt’

Type Package

Title Asking GPT About R Stuff

Version 0.1.2

Description A chat package connecting to API endpoints by ‘OpenAI’ (<https://platform.openai.com/>) to answer questions (about R).

Depends R (>= 4.1.0)

Imports cli, callr, dplyr, glue, methods, rlang, httr2, rappdirs, jsonlite

Suggests covr, knitr, miniUI, rmarkdown, rstudioapi, shiny, shinyCSSloaders, spelling, testthat (>= 3.0.0), withr

URL https://github.com/JBGruber/askgpt

BugReports https://github.com/JBGruber/askgpt/issues

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.2.3

VignetteBuilder knitr

Config/testthat/edition 3

Language en-GB

LazyData true

NeedsCompilation no

Author Johannes Gruber [aut, cre]

Maintainer Johannes Gruber <johannesb.gruber@gmail.com>

Repository CRAN

Date/Publication 2023-06-26 11:30:02 UTC
annotate_code

Annotate R code with inline comments

Description
Annotate R code with inline comments

Usage
annotate_code(code, ...)

Arguments

- **code**: A character vector of R code. If missing the code currently selected in RStudio is documented (If RStudio is used).

- **...**: passed on to askgpt.

Value
A character vector.
askgpt  

Ask openai’s GPT models a question

Description

Ask openai’s GPT models a question

Usage

askgpt(prompt, chat = TRUE, progress = TRUE, return_answer = FALSE, ...)

Arguments

prompt  
What you want to ask

chat  
whether to use the chat API (i.e., the same model as ChatGPT) or the completions API.

progress  
Show a progress spinner while the request to the API has not been fulfilled.

return_answer  
Should the answer be returned as an object instead of printing it to the screen?

...  
additional options forwarded to chat_api or completions_api respectively.

Value

either an httr2 response from one of the APIs or a character vector (if return_answer).

Examples

```r
## Not run:
askgpt("What is an R function?"
askgpt("What is wrong with my last command?"
askgpt("Can you help me with the function aes() from ggplot2?"

## End(Not run)
```

chat_api  

Request answer from openai’s chat API

Description

Request answer from openai’s chat API
completions_api

Usage

```r
chat_api(
  prompt,
  model = NULL,
  config = NULL,
  max_tokens = NULL,
  api_key = NULL,
  ...
)
```

Arguments

- **prompt**: character string of the prompt to be completed.
- **model**: character string of the model to be used (defaults to "text-davinci-003").
- **config**: a configuration prompt to tell the model how it should behave.
- **max_tokens**: The maximum number of tokens to generate in the completion. 2048L is the maximum the models accept.
- **api_key**: set the API key. If NULL, looks for the env OPENAI_API_KEY.
- **...**: additional parameters to be passed to the API (see [the API documentation](https://platform.openai.com/docs/api-reference/completions))

Value

- A tibble with available models
- a htr2 response object

Examples

```r
## Not run:
chat_api("Hi, how are you?", config = "answer as a friendly chat bot")
## End(Not run)
```

completions_api Request answer from openai's completions API

Description

Mostly used under the hood for askgpt.
completions_api

Usage

```r
completions_api(
  prompt,
  model = NULL,
  temperature = NULL,
  max_tokens = NULL,
  api_key = NULL,
  ...)
```

Arguments

- `prompt` character string of the prompt to be completed.
- `model` character string of the model to be used (defaults to "text-davinci-003").
- `temperature` numeric value between 0 and 1 to control the randomness of the output (defaults to 0.2; lower values like 0.2 will make answers more focused and deterministic).
- `max_tokens` The maximum number of tokens to generate in the completion. 2048L is the maximum the models accept.
- `api_key` set the API key. If NULL, looks for the env OPENAI_API_KEY.
- `...` additional parameters to be passed to the API (see [the API documentation](https://platform.openai.com/docs/api-reference/completions)

Details

Only a few parameters are implemented by name. Most can be sent through the ... For example, you could use the n parameter just like this completions_api("The quick brown fox", n = 2).

A couple of defaults are used by the package:

- the model used by default is "text-davinci-003"
- the default temperature is 0.2
- the default for max_tokens is 2048L

You can configure how askgpt makes requests by setting options that start with askgpt_. For example, to use a different model use options(askgpt_model = "text-curie-001"). It does not matter if the API parameter ist listed in the function or not. All are used.

Value

a htr2 response object

Examples

```r
## Not run:
completions_api("The quick brown fox")

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

**Description**

Document R Code

**Usage**

```r
document_code(code, ...)
```

**Arguments**

- `code`: A character vector of R code. If missing the code currently selected in RStudio is documented (If RStudio is used).
- `...`: passed on to `askgpt`.

**Value**

A character vector.

**Examples**

```r
## Not run:
document_code()
## End(Not run)
```

---

### estimate_token

**Description**

Estimate token count

**Usage**

```r
estimate_token(x, mult = 1.6)
```

**Arguments**

- `x`: character vector
- `mult`: the multiplier used
**explain_code**

**Details**

This function estimates how many tokens the API will make of the input words. For the models 1 word is more than one token. The default multiplier value resulted from testing the API. See <https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them> for more information.

**Value**

A integer vector of token counts

**Examples**

```r
estimate_token("this is a test")
```

---

**explain_code**  
**Explain R code**

**Description**

Explain R code

**Usage**

```r
explain_code(code, ...)
```

**Arguments**

- `code` A character vector of R code. If missing the code currently selected in RStudio is documented (If RStudio is used).
- `...` passed on to askgpt.

**Value**

A character vector.
list_models  

**List Models**

**Description**

List the models available in the API. You can refer to the [Models documentation](https://platform.openai.com/docs/models) to understand what models are available and the differences between them.

**Usage**

```r
list_models(api_key = NULL)
```

**Arguments**

- `api_key` set the API key. If NULL, looks for the env OPENAI_API_KEY.

**Value**

A tibble with available models

**Examples**

```r
## Not run:
completions_api("The quick brown fox")
## End(Not run)
```

login  

**Log in to OpenAI**

**Description**

Log in to OpenAI

**Usage**

```r
login(api_key, force_refresh = FALSE, cache_dir = NULL, no_cache = FALSE)
```

**Arguments**

- `api_key` API key to use for authentication. If not provided, the function look for a cached key or guide the user to obtain one.
- `force_refresh` Log in again even if an API key is already cached.
- `cache_dir` dir location to save keys on disk. The default is to use `rappdirs::user_cache_dir("askgpt")`.
- `no_cache` Don’t cache the API key, only load it into the environment.
log_init

**Value**

a character vector with an API key

---

log_init | Initiate error logging

**Description**

Initiate error logging

**Usage**

log_init(...)

**Arguments**

... forwarded to global_entrace

**Details**

Just an alias for rlang::global_entrace() with a more fitting name (for the purpose here).

**Value**

No return value, called to enable rlang error logging

---

new_conversation | Start a new conversation

**Description**

Deletes the local prompt and response history to start a new conversation.

**Usage**

new_conversation()

**Value**

Does not return a value
### parse_response
*Parse response from API functions*

**Description**
Parse response from API functions

**Usage**
```
parse_response(response)
```

**Arguments**
- `response`: a response object from `chat_api` or `completions_api`

**Value**
- a character vector

---

### prompt_history
*Return the prompt/response history*

**Description**
Return the prompt/response history

**Usage**
```
prompt_history(n = Inf)
```

**Arguments**
- `n`: number of prompts/responses to return.

**Value**
a character vector
response_history

Return the prompt/response history

Description
Return the prompt/response history

Usage
response_history(n = Inf)

Arguments
n  number of prompts/responses to return.

Value
a character vector

test_function
Test R code

Description
Test R code

Usage
test_function(code, ...)

Arguments
code  A character vector of R code. If missing the code currently selected in RStudio is documented (If RStudio is used).
...
    passed on to askgpt.

Value
A character vector.
### token_limits

**Max tokens limits of the different models**

<table>
<thead>
<tr>
<th>token_limits</th>
<th>Max tokens limits of the different models</th>
</tr>
</thead>
</table>

#### Description

OpenAI’s token limits for different models.

#### Usage

```r
token_limits
```

#### Format

An object of class `data.frame` with 6 rows and 2 columns.

#### Source

[https://platform.openai.com/docs/models/overview](https://platform.openai.com/docs/models/overview)

### tutorialise_addin

**Turn R code into a tutorial**

<table>
<thead>
<tr>
<th>tutorialise_addin</th>
<th>Turn R code into a tutorial</th>
</tr>
</thead>
</table>

#### Description

`tutorialise_addin()` opens an [RStudio gadget](https://shiny.rstudio.com/articles/gadgets.html) and [addin](http://rstudio.github.io/rstudioaddins/) that turns selected code into an R Markdown/Quarto Tutorial.

#### Usage

```r
tutorialise_addin()
```

#### Value

No return value, opens a new file in RStudio
Index

* datasets
  token_limits, 12
annotate_code, 2
askgpt, 2, 3, 4–7, 11
chat_api, 3, 3, 10
completions_api, 3, 4, 10
document_code, 6
estimate_token, 6
explain_code, 7
global_entrace, 9
list_models, 8
log_init, 9
login, 8
new_conversation, 9
parse_response, 10
prompt_history, 10
response_history, 11
test_function, 11
token_limits, 12
tutorialise_addin, 12