Package 'GenAI'

January 20, 2025

Type Package

Title Generative Artificial Intelligence

Version 0.2.0

Maintainer Li Yuan <lyuan@gd.edu.kg>

Description Utilizing Generative Artificial Intelligence models like 'GPT-4' and 'Gemini Pro' as coding and writing assistants for 'R' users. Through these models, 'GenAI' offers a variety of functions, encompassing text generation, code optimization, natural language processing, chat, and image interpretation. The goal is to aid 'R' users in streamlining laborious coding and language processing tasks.

License CC BY 4.0

URL https://genai.gd.edu.kg/

BugReports https://github.com/GitData-GA/GenAI/issues

Encoding UTF-8

RoxygenNote 7.3.0

Depends magrittr

Imports base64enc, httr, jsonlite, tools, R6, listenv, magick, ggplotify

NeedsCompilation no

Author Li Yuan [aut, cre] (<https://orcid.org/0009-0008-1075-9922>)

Repository CRAN

Date/Publication 2024-02-15 19:00:02 UTC

Contents

`available.models` *Get Supported Generative AI Models*

Description

This function sends a request to GenAI database API to retrieve information about available generative AI models.

Usage

```
available.models()
```

Details

The function utilizes the GenAI database API to fetch the latest information about available Generative AI models. The retrieved data includes details about different models offered by various service providers.

Value

If successful, the function returns a list containing generative AI service providers and their corresponding models. If the function encounters an error, it will halt execution and provide an error message.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

all.models = available.models() %>% print()
```

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

---

chat *Chat Generation with Text as the Input*

Description

This function establishes a connection to a generative AI model through a generative AI object. It generates a chat response based on the provided prompt and stores it in the chat history along with the generative AI object.

Usage

```
chat(genai.object, prompt, verbose = FALSE, config = list())
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.
`prompt`	A character string representing the query for chat generation.
`verbose`	Optional. Default to `FALSE`. A boolean value determining whether or not to print out the details of the chat request.
`config`	Optional. Default to `list()`. A list of configuration parameters for chat generation.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

In addition, this function modifies the chat history along with the generative AI object directly, meaning the chat history is mutable. You can print out the chat history using the function `chat.history.print` or simply use `verbose = TRUE` in this function. If you want to edit a message, use the function `chat.edit`. To reset the chat history along with the generative AI object, use the function `chat.history.reset`.

For **Google Generative AI** models, available configurations are as follows. For more detail, please refer to `https://ai.google.dev/api/rest/v1/HarmCategory`, `https://ai.google.dev/api/rest/v1/SafetySettin` and `https://ai.google.dev/api/rest/v1/GenerationConfig`.

- `harm.category.dangerous.content`

  Optional. An integer, from 1 to 5 inclusive, representing the threshold for dangerous content, with a higher value representing a lower probability of being blocked.

- `harm.category.harassment`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for harasment content, with a higher value representing a lower probability of being blocked.
- `harm.category.hate.speech`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for hate speech and content, with a higher value representing a lower probability of being blocked.
- `harm.category.sexually.explicit`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for sexually explicit content, with a higher value representing a lower probability of being blocked.
- `stop.sequences`
  Optional. A list of character sequences (up to 5) that will stop output generation. If specified, the API will stop at the first appearance of a stop sequence. The stop sequence will not be included as part of the response.
- `max.output.tokens`
  Optional. An integer, value varies by model, representing maximum number of tokens to include in a candidate.
- `temperature`
  Optional. A number, from 0.0 to 1.0 inclusive, controlling the randomness of the output.
- `top.p`
  Optional. A number, value varies by model, representing maximum cumulative probability of tokens to consider when sampling.
- `top.k`
  Optional. A number, value varies by model, representing maximum number of tokens to consider when sampling.

For **Moonshot AI** models, available configurations are as follows. For more detail, please refer to `https://platform.moonshot.cn/api.html#chat-completion`.

- `max.tokens`
  Optional. An integer. The maximum number of tokens that will be generated when the chat completes. If the chat is not finished by the maximum number of tokens generated, the finish reason will be "length", otherwise it will be "stop".
- `temperature`
  Optional. A number. What sampling temperature to use, between 0 and 1. Higher values (e.g. 0.7) will make the output more random, while lower values (e.g. 0.2) will make it more focused and deterministic.
- `top.p`
  Optional. A number. Another sampling temperature.

For **OpenAI** models, available configurations are as follows. For more detail, please refer to `https://platform.openai.com/docs/api-reference/chat/create`.

- `frequency.penalty`
  Optional. A number from -2.0 to 2.0 inclusive. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

- `logit.bias`

  Optional. A map. Modify the likelihood of specified tokens appearing in the completion. Accepts a JSON object that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.

- `logprobs`

  Optional. A boolean value. Whether to return log probabilities of the output tokens or not. If true, returns the log probabilities of each output token returned in the content of message

- `top.logprobs`

  Optional. An integer between 0 and 5 specifying the number of most likely tokens to return at each token position, each with an associated log probability. `logprobs` must be set to `TRUE` if this parameter is used.

- `max.tokens`

  Optional. An integer. The maximum number of tokens that can be generated in the chat completion. The total length of input tokens and generated tokens is limited by the model's context length.

- `presence.penalty`

  Optional. A Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.

- `response.format`

  Optional. An object specifying the format that the model must output. Compatible with GPT-4 Turbo and all GPT-3.5 Turbo models newer than `gpt-3.5-turbo-1106`.

- `seed`

  Optional. An integer. If specified, our system will make a best effort to sample deterministically, such that repeated requests with the same seed and parameters should return the same result.

- `stop`

  Optional. A character string or list contains up to 4 sequences where the API will stop generating further tokens.

- `temperature`

  Optional. A number. What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

- `top.p`

  Optional. A number. 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

- `tools`

  Optional. A list of tools the model may call. Currently, only functions are supported as a tool. Use this to provide a list of functions the model may generate JSON inputs for.

- `tool.choice`

  Optional. A character string or object. Controls which (if any) function is called by the model. `none` means the model will not call a function and instead generates a message. `auto` means the model can pick between generating a message or calling a function.

- `user`

  Optional. A character string. A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.

Value

If successful, the most recent chat response will be returned. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  chat(prompt = "Write a story about Mars in 50 words.") %>%
  cat()

# Method 2: use the reference operator "$"
cat(genai.model$chat(prompt = "Write a story about Jupiter in 50 words."))

# Method 3: use the function chat() directly
cat(chat(genai.object = genai.model,
         prompt = "Summarize the chat."))

## End(Not run)
```

---

chat.edit *Chat Edit with New Text as the Input*

Description

This function establishes a connection to a generative AI model through a generative AI object. It generates a chat response based on the new prompt and stores it in the chat history along with the generative AI object.

Usage

```
chat.edit(
  genai.object,
  prompt,
  message.to.edit,
  verbose = FALSE,
  config = list()
)
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.
`prompt`	A character string representing the query for chat generation.
`message.to.edit`	An integer representing the index of the message to be edited.
`verbose`	Optional. Default to `FALSE`. A boolean value determining whether or not to print out the details of the chat request.
`config`	Optional. Default to `list()`. A list of configuration parameters for chat generation.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

In addition, this function modifies the chat history along with the generative AI object directly, meaning the chat history is mutable. You can print out the chat history using the function `chat.history.print` or simply use `verbose = TRUE` in this function. To reset the chat history along with the chat history along with the generative AI object, use the function `chat.history.reset`.

For **Google Generative AI** models, available configurations are as follows. For more detail, please refer to `https://ai.google.dev/api/rest/v1/HarmCategory`, `https://ai.google.dev/api/rest/v1/SafetySettin` and `https://ai.google.dev/api/rest/v1/GenerationConfig`.

- `harm.category.dangerous.content`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for dangerous content, with a higher value representing a lower probability of being blocked.
- `harm.category.harassment`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for harasment content, with a higher value representing a lower probability of being blocked.
- `harm.category.hate.speech`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for hate speech and content, with a higher value representing a lower probability of being blocked.
- `harm.category.sexually.explicit`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for sexually explicit content, with a higher value representing a lower probability of being blocked.

- `stop.sequences`

  Optional. A list of character sequences (up to 5) that will stop output generation. If specified, the API will stop at the first appearance of a stop sequence. The stop sequence will not be included as part of the response.

- `max.output.tokens`

  Optional. An integer, value varies by model, representing maximum number of tokens to include in a candidate.

- `temperature`

  Optional. A number, from 0.0 to 1.0 inclusive, controlling the randomness of the output.

- `top.p`

  Optional. A number, value varies by model, representing maximum cumulative probability of tokens to consider when sampling.

- `top.k`

  Optional. A number, value varies by model, representing maximum number of tokens to consider when sampling.

For **Moonshot AI** models, available configurations are as follows. For more detail, please refer to `https://platform.moonshot.cn/api.html#chat-completion`.

- `max.tokens`

  Optional. An integer. The maximum number of tokens that will be generated when the chat completes. If the chat is not finished by the maximum number of tokens generated, the finish reason will be "length", otherwise it will be "stop".

- `temperature`

  Optional. A number. What sampling temperature to use, between 0 and 1. Higher values (e.g. 0.7) will make the output more random, while lower values (e.g. 0.2) will make it more focused and deterministic.

- `top.p`

  Optional. A number. Another sampling temperature.

For **OpenAI** models, available configurations are as follows. For more detail, please refer to `https://platform.openai.com/docs/api-reference/chat/create`.

- `frequency.penalty`

  Optional. A number from -2.0 to 2.0 inclusive. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

- `logit.bias`

  Optional. A map. Modify the likelihood of specified tokens appearing in the completion. Accepts a JSON object that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.

- `logprobs`

  Optional. A boolean value. Whether to return log probabilities of the output tokens or not. If true, returns the log probabilities of each output token returned in the content of message

- `top.logprobs`

  Optional. An integer between 0 and 5 specifying the number of most likely tokens to return at each token position, each with an associated log probability. `logprobs` must be set to `TRUE` if this parameter is used.

- `max.tokens`

  Optional. An integer. The maximum number of tokens that can be generated in the chat completion. The total length of input tokens and generated tokens is limited by the model's context length.

- `presence.penalty`

  Optional. A Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.

- `response.format`

  Optional. An object specifying the format that the model must output. Compatible with GPT-4 Turbo and all GPT-3.5 Turbo models newer than `gpt-3.5-turbo-1106`.

- `seed`

  Optional. An integer. If specified, our system will make a best effort to sample deterministically, such that repeated requests with the same seed and parameters should return the same result.

- `stop`

  Optional. A character string or list contains up to 4 sequences where the API will stop generating further tokens.

- `temperature`

  Optional. A number. What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

- `top.p`

  Optional. A number. 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

- `tools`

  Optional. A list of tools the model may call. Currently, only functions are supported as a tool. Use this to provide a list of functions the model may generate JSON inputs for.

- `tool.choice`

  Optional. A character string or object. Controls which (if any) function is called by the model. `none` means the model will not call a function and instead generates a message. `auto` means the model can pick between generating a message or calling a function.

- `user`

  Optional. A character string. A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.

Value

If successful, the most recent chat response will be returned. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  chat.edit(prompt = "What is XGBoost?",
            message.to.edit = 5,
            verbose = TRUE,
            config = parameters) %>%
  cat()

# Method 2: use the reference operator "$"
cat(genai.model$chat.edit(prompt = "What is CatBoost?",
                          message.to.edit = 3))

# Method 3: use the function chat.edit() directly
cat(chat.edit(genai.object = genai.model,
              prompt = "What is LightGBM?",
              message.to.edit = 1))

## End(Not run)
```

---

chat.history.convert *Chat History Convert*

Description

This function converts the chat history along with a generative AI object to a valid format for another generative AI object.

Usage

```
chat.history.convert(from.genai.object, to.genai.object)
```

Arguments

`from.genai.object`
: A source generative AI object containing necessary and correct information.

`to.genai.object`
: A target generative AI object containing necessary and correct information.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`. Moreover, you can print out the chat history using the function `chat.history.print` or simply use `verbose = TRUE` during the chat.

Value

If successful, the converted chat history list will be returned.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there are two GenAI objects named 'genai.model' and 'another.genai.model'
# supporting this function, please refer to the "Live Demo in Colab" above for
# real examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
converted.history = genai.model %>%
  chat.history.convert(to.genai.object = another.genai.model)

# Method 2: use the reference operator "$"
converted.history = genai.model$chat.history.convert(to.genai.object = another.genai.model)

# Method 3: use the function chat.history.convert() directly
converted.history = chat.history.convert(from.genai.object = genai.model,
                                         to.genai.object = another.genai.model)

## End(Not run)
```

`chat.history.export` *Chat History Export*

Description

This function exports the chat history along with a generative AI object as a list.

Usage

```
chat.history.export(genai.object)
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

Value

If successful, the chat history list will be returned.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
exported.history = genai.model %>%
  chat.history.export()

# Method 2: use the reference operator "$"
exported.history = genai.model$chat.history.export()

# Method 3: use the function chat.history.export() directly
exported.history = chat.history.export(genai.object = genai.model)

## End(Not run)
```

chat.history.import *Chat History Import*

Description

This function imports a chat history in list format to a generative AI object.

Usage

```
chat.history.import(genai.object, new.chat.history)
```

Arguments

genai.object	A generative AI object containing necessary and correct information.
new.chat.history	A list containing a chat history in correct format.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function available.models.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function and a valid chat history list named 'new.history', please
# refer to the "Live Demo in Colab" above for real examples. The
# following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  chat.history.import(new.chat.history = new.history)

# Method 2: use the reference operator "$"
genai.model$chat.history.import(new.chat.history = new.history)

# Method 3: use the function chat.history.import() directly
chat.history.import(genai.object = genai.model,
                    new.chat.history = new.history)
```

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

`chat.history.print` *Chat History Print*

Description

This function prints out the chat history along with a generative AI object.

Usage

```
chat.history.print(genai.object, from = 1, to = NULL)
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.
`from`	Optional. Default to 1. An integer representing the first message in the chat history that needs to be printed.
`to`	Optional. Default to `NULL`, prints until the last message in the chat history. An integer representing the last message in the chat history that needs to be printed.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  chat.history.print()

# Method 2: use the reference operator "$"
genai.model$chat.history.print(from = 3)
```

```
# Method 3: use the function chat.history.print() directly
chat.history.print(genai.object = genai.model,
                   from = 3,
                   to = 5)

## End(Not run)
```

`chat.history.reset` *Chat History Reset*

Description

This function resets the chat history along with a generative AI object.

Usage

```
chat.history.reset(genai.object)
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  chat.history.reset()

# Method 2: use the reference operator "$"
genai.model$chat.history.reset()
```

```
# Method 3: use the function chat.history.reset() directly
chat.history.reset(genai.object = genai.model)

## End(Not run)
```

`chat.history.save` *Chat History Save*

Description

This function saves a chat history along with a generative AI object as a JSON file.

Usage

```
chat.history.save(genai.object, file.name)
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.
`file.name`	A character string representing the name of the JSON file for the chat history.

Details

Providing accurate and valid information for each argument is crucial for successful chat generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

Value

If successful, the chat history will be saved as a JSON file in your current or specified directory.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  chat.history.save(file.name = "saved_history")
```

```
# Method 2: use the reference operator "$"
genai.model$chat.history.save(file.name = "saved_history")

# Method 3: use the function chat.history.save() directly
chat.history.save(genai.object = genai.model,
                  file.name = "saved_history")

## End(Not run)
```

`genai.google` *Google Generative AI Object Creation*

Description

This function establishes a connection to a Google generative AI model by providing essential parameters.

Usage

```
genai.google(api, model, version, proxy = FALSE)
```

Arguments

`api`	A character string representing the API key required for accessing the model.
`model`	A character string representing the specific model.
`version`	A character string representing the version of the chosen model.
`proxy`	Optional. Default to `FALSE`. A boolean value indicating whether to use a proxy for accessing the API URL. If your local internet cannot access the API, set this parameter to `TRUE`.

Details

Providing accurate and valid information for each argument is crucial for successful text generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

Please refer to `https://ai.google.dev/tutorials/setup` for the API key.

The API proxy service is designed to address the needs of users who hold a valid API key but find themselves outside their home countries or regions due to reasons such as travel, work, or study in locations that may not be covered by certain Generative AI service providers.

Please be aware that although GenAI and its affiliated organization - GitData - do not gather user information through this service, the server providers for GenAI API proxy service and the Generative AI service providers may engage in such data collection. Furthermore, the proxy service cannot guarantee a consistent connection speed. Users are strongly encouraged to utilize this service with caution and at their own discretion.

Value

If successful, the function returns a Google generative AI object. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

GenAI - Generative Artificial Intelligence API Proxy Service

Live Demo in Colab

Examples

```
## Not run:
# Please change YOUR_GOOGLE_API to your own API key of Google Generative AI
Sys.setenv(GOOGLE_API = "YOUR_GOOGLE_API")

all.models = available.models() %>% print()

# Create a Google Generative AI object
google = genai.google(api = Sys.getenv("GOOGLE_API"),
                      model = all.models$google$model[1],
                      version = all.models$google$version[1],
                      proxy = FALSE)

## End(Not run)
```

---

genai.moonshot *Moonshot AI Object Creation*

Description

This function establishes a connection to a Moonshot AI model by providing essential parameters.

Usage

```
genai.moonshot(api, model, version, proxy = FALSE)
```

Arguments

api	A character string representing the API key required for accessing the model.
model	A character string representing the specific model.
version	A character string representing the version of the chosen model.
proxy	Optional. Default to FALSE. A boolean value indicating whether to use a proxy for accessing the API URL. If your local internet cannot access the API, set this parameter to TRUE.

Details

Providing accurate and valid information for each argument is crucial for successful text generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

Please refer to `https://platform.moonshot.cn/console/api-keys` for the API key.

The API proxy service is designed to address the needs of users who hold a valid API key but find themselves outside their home countries or regions due to reasons such as travel, work, or study in locations that may not be covered by certain Generative AI service providers.

Please be aware that although GenAI and its affiliated organization - GitData - do not gather user information through this service, the server providers for GenAI API proxy service and the Generative AI service providers may engage in such data collection. Furthermore, the proxy service cannot guarantee a consistent connection speed. Users are strongly encouraged to utilize this service with caution and at their own discretion.

Value

If successful, the function returns an moonshot object. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

GenAI - Generative Artificial Intelligence API Proxy Service

Live Demo in Colab

Examples

```
## Not run:
# Please change YOUR_MOONSHOT_API to your own API key of Moonshot AI
Sys.setenv(MOONSHOT_API = "YOUR_MOONSHOT_API")

all.models = available.models() %>% print()

# Create an moonshot object
moonshot = genai.moonshot(api = Sys.getenv("MOONSHOT_API"),
                          model = all.models$moonshot$model[1],
                          version = all.models$moonshot$version[1],
                          proxy = FALSE)

## End(Not run)
```

genai.openai *OpenAI Object Creation*

Description

This function establishes a connection to an OpenAI model by providing essential parameters.

Usage

```
genai.openai(api, model, version, proxy = FALSE, organization.id = NULL)
```

Arguments

api	A character string representing the API key required for accessing the model.
model	A character string representing the specific model.
version	A character string representing the version of the chosen model.
proxy	Optional. Default to FALSE. A boolean value indicating whether to use a proxy for accessing the API URL. If your local internet cannot access the API, set this parameter to TRUE.
organization.id	Optional. Default to NULL. A character string representing the organization ID.

Details

Providing accurate and valid information for each argument is crucial for successful text generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function available.models.

Please refer to https://platform.openai.com/api-keys for the API key. Moreover, please refer to https://platform.openai.com/account/organization for the optional organization ID.

The API proxy service is designed to address the needs of users who hold a valid API key but find themselves outside their home countries or regions due to reasons such as travel, work, or study in locations that may not be covered by certain Generative AI service providers.

Please be aware that although GenAI and its affiliated organization - GitData - do not gather user information through this service, the server providers for GenAI API proxy service and the Generative AI service providers may engage in such data collection. Furthermore, the proxy service cannot guarantee a consistent connection speed. Users are strongly encouraged to utilize this service with caution and at their own discretion.

Value

If successful, the function returns an OpenAI object. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

GenAI - Generative Artificial Intelligence API Proxy Service

Live Demo in Colab

Examples

```
## Not run:
# Please change YOUR_OPENAI_API to your own API key of OpenAI
Sys.setenv(OPENAI_API = "YOUR_OPENAI_API")

# Oprional. Please change YOUR_OPENAI_ORG to your own organization ID for OpenAI
Sys.setenv(OPENAI_ORG = "YOUR_OPENAI_ORG")

all.models = available.models() %>% print()

# Create an OpenAI object
openai = genai.openai(api = Sys.getenv("OPENAI_API"),
                      model = all.models$openai$model[1],
                      version = all.models$openai$version[1],
                      proxy = FALSE,
                      organization.id = Sys.getenv("OPENAI_ORG"))

## End(Not run)
```

---

img *Image Generation with Text as the Input*

Description

This function establishes a connection to a generative AI model through a generative AI object. It generates an image response based on the provided prompt.

Usage

```
img(genai.object, prompt, verbose = FALSE, config = list())
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.
`prompt`	A character string representing the query for image generation.
`verbose`	Optional. Default to `FALSE`. A boolean value determining whether or not to print out the details of the image request.
`config`	Optional. Default to `list()`. A list of configuration parameters for image generation.

Details

Providing accurate and valid information for each argument is crucial for successful image generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

This function is only available when using OpenAI's models.

For **OpenAI** models, available configurations are as follows. For more detail, please refer to `https://platform.openai.com/docs/api-reference/images/create`.

- `quality`
  Optional. A character string. The quality of the image that will be generated. `hd` creates images with finer details and greater consistency across the image.
- `size`
  Optional. A character string. The size of the generated images. Must be one of `256x256`, `512x512`, or `1024x1024` for `dall-e-2`. Must be one of `1024x1024`, `1792x1024`, or `1024x1792` for `dall-e-3` models.
- `style`
  Optional. The style of the generated images. Must be one of `vivid` or `natural`. Vivid causes the model to lean towards generating hyper-real and dramatic images. Natural causes the model to produce more natural, less hyper-real looking images.
- `user`
  Optional. A character string. A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.

Value

If successful, a image in `ggplot` format will be returned. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
generated.image = genai.model %>%
  img(prompt = "A very cute panda eating banboo.")
generated.image

# Method 2: use the reference operator "$"
```

```
generated.image = genai.model$img(prompt = "A very cute sea otter on a rock.")
generated.image

# Method 3: use the function img() directly
generated.image = img(genai.object = genai.model,
                      prompt = "A very cute bear.")
generated.image

## End(Not run)
```

`txt` *Text Generation with Text as the Input*

Description

This function establishes a connection to a generative AI model through a generative AI object. It generates a text response based on the provided prompt.

Usage

```
txt(genai.object, prompt, verbose = FALSE, config = list())
```

Arguments

`genai.object`	A generative AI object containing necessary and correct information.
`prompt`	A character string representing the query for text generation.
`verbose`	Optional. Default to `FALSE`. A boolean value determining whether or not to print out the details of the text request.
`config`	Optional. Default to `list()`. A list of configuration parameters for text generation.

Details

Providing accurate and valid information for each argument is crucial for successful text generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function `available.models`.

For **Google Generative AI** models, available configurations are as follows. For more detail, please refer to `https://ai.google.dev/api/rest/v1/HarmCategory`, `https://ai.google.dev/api/rest/v1/SafetySettin` and `https://ai.google.dev/api/rest/v1/GenerationConfig`.

- `harm.category.dangerous.content`

  Optional. An integer, from 1 to 5 inclusive, representing the threshold for dangerous content, with a higher value representing a lower probability of being blocked.

- `harm.category.harassment`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for harasment content, with a higher value representing a lower probability of being blocked.
- `harm.category.hate.speech`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for hate speech and content, with a higher value representing a lower probability of being blocked.
- `harm.category.sexually.explicit`
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for sexually explicit content, with a higher value representing a lower probability of being blocked.
- `stop.sequences`
  Optional. A list of character sequences (up to 5) that will stop output generation. If specified, the API will stop at the first appearance of a stop sequence. The stop sequence will not be included as part of the response.
- `max.output.tokens`
  Optional. An integer, value varies by model, representing maximum number of tokens to include in a candidate.
- `temperature`
  Optional. A number, from 0.0 to 1.0 inclusive, controlling the randomness of the output.
- `top.p`
  Optional. A number, value varies by model, representing maximum cumulative probability of tokens to consider when sampling.
- `top.k`
  Optional. A number, value varies by model, representing maximum number of tokens to consider when sampling.

For **Moonshot AI** models, available configurations are as follows. For more detail, please refer to `https://platform.moonshot.cn/api.html#chat-completion`.

- `max.tokens`
  Optional. An integer. The maximum number of tokens that will be generated when the chat completes. If the chat is not finished by the maximum number of tokens generated, the finish reason will be "length", otherwise it will be "stop".
- `temperature`
  Optional. A number. What sampling temperature to use, between 0 and 1. Higher values (e.g. 0.7) will make the output more random, while lower values (e.g. 0.2) will make it more focused and deterministic.
- `top.p`
  Optional. A number. Another sampling temperature.

For **OpenAI** models, available configurations are as follows. For more detail, please refer to `https://platform.openai.com/docs/api-reference/chat/create`.

- `frequency.penalty`
  Optional. A number from -2.0 to 2.0 inclusive. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.

- `logit.bias`

  Optional. A map. Modify the likelihood of specified tokens appearing in the completion. Accepts a JSON object that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.

- `logprobs`

  Optional. A boolean value. Whether to return log probabilities of the output tokens or not. If true, returns the log probabilities of each output token returned in the content of message

- `top.logprobs`

  Optional. An integer between 0 and 5 specifying the number of most likely tokens to return at each token position, each with an associated log probability. `logprobs` must be set to `TRUE` if this parameter is used.

- `max.tokens`

  Optional. An integer. The maximum number of tokens that can be generated in the chat completion. The total length of input tokens and generated tokens is limited by the model's context length.

- `presence.penalty`

  Optional. A Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.

- `response.format`

  Optional. An object specifying the format that the model must output. Compatible with GPT-4 Turbo and all GPT-3.5 Turbo models newer than `gpt-3.5-turbo-1106`.

- `seed`

  Optional. An integer. If specified, our system will make a best effort to sample deterministically, such that repeated requests with the same seed and parameters should return the same result.

- `stop`

  Optional. A character string or list contains up to 4 sequences where the API will stop generating further tokens.

- `temperature`

  Optional. A number. What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

- `top.p`

  Optional. A number. 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

- `tools`

  Optional. A list of tools the model may call. Currently, only functions are supported as a tool. Use this to provide a list of functions the model may generate JSON inputs for.

- `tool.choice`
  Optional. A character string or object. Controls which (if any) function is called by the model. `none` means the model will not call a function and instead generates a message. `auto` means the model can pick between generating a message or calling a function.
- `user`
  Optional. A character string. A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.

Value

If successful, a text response will be returned. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, please refer to the "Live Demo in Colab" above for real
# examples. The following examples are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  txt(prompt = "Write a story about Mars in 50 words.") %>%
  cat()

# Method 2: use the reference operator "$"
cat(genai.model$txt(prompt = "Write a story about Jupiter in 50 words."))

# Method 3: use the function txt() directly
# Set verbose to TRUE to see the detail
cat(txt(genai.object = genai.model,
        prompt = "Write a story about Earth in 50 words."))

## End(Not run)
```

`txt.image` *Text Generation with Text and Image as the Input*

Description

This function establishes a connection to a generative AI model through a generative AI object. It generates a text response based on the provided prompt.

Usage

```
txt.image(genai.object, prompt, image.path, verbose = FALSE, config = list())
```

Arguments

genai.object	A generative AI object containing necessary and correct information.
prompt	A character string representing the query for text generation.
image.path	A character string representing the path to the image. It should be a link starting with https/http or a local directory path to an image.
verbose	Optional. Default to FALSE. A boolean value determining whether or not to print out the details of the text request.
config	Optional. Default to list(). A list of configuration parameters for text generation.

Details

Providing accurate and valid information for each argument is crucial for successful text generation by the generative AI model. If any parameter is incorrect, the function responds with an error message based on the API feedback. To view all supported generative AI models, use the function available.models.

For **Google Generative AI** models, available configurations are as follows. For more detail, please refer to https://ai.google.dev/api/rest/v1/HarmCategory, https://ai.google.dev/api/rest/v1/SafetySettin and https://ai.google.dev/api/rest/v1/GenerationConfig.

- harm.category.dangerous.content
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for dangerous content, with a higher value representing a lower probability of being blocked.
- harm.category.harassment
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for harasment content, with a higher value representing a lower probability of being blocked.
- harm.category.hate.speech
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for hate speech and content, with a higher value representing a lower probability of being blocked.
- harm.category.sexually.explicit
  Optional. An integer, from 1 to 5 inclusive, representing the threshold for sexually explicit content, with a higher value representing a lower probability of being blocked.
- stop.sequences
  Optional. A list of character sequences (up to 5) that will stop output generation. If specified, the API will stop at the first appearance of a stop sequence. The stop sequence will not be included as part of the response.
- max.output.tokens
  Optional. An integer, value varies by model, representing maximum number of tokens to include in a candidate.
- temperature
  Optional. A number, from 0.0 to 1.0 inclusive, controlling the randomness of the output.

- `top.p`
  Optional. A number, value varies by model, representing maximum cumulative probability of tokens to consider when sampling.
- `top.k`
  Optional. A number, value varies by model, representing maximum number of tokens to consider when sampling.

For **OpenAI** models, available configurations are as follows. For more detail, please refer to `https://platform.openai.com/docs/api-reference/chat/create`.

- `frequency.penalty`
  Optional. A number from -2.0 to 2.0 inclusive. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim.
- `logit.bias`
  Optional. A map. Modify the likelihood of specified tokens appearing in the completion. Accepts a JSON object that maps tokens (specified by their token ID in the tokenizer) to an associated bias value from -100 to 100. Mathematically, the bias is added to the logits generated by the model prior to sampling. The exact effect will vary per model, but values between -1 and 1 should decrease or increase likelihood of selection; values like -100 or 100 should result in a ban or exclusive selection of the relevant token.
- `logprobs`
  Optional. A boolean value. Whether to return log probabilities of the output tokens or not. If true, returns the log probabilities of each output token returned in the content of message
- `top.logprobs`
  Optional. An integer between 0 and 5 specifying the number of most likely tokens to return at each token position, each with an associated log probability. `logprobs` must be set to `TRUE` if this parameter is used.
- `max.tokens`
  Optional. An integer. The maximum number of tokens that can be generated in the chat completion. The total length of input tokens and generated tokens is limited by the model's context length.
- `presence.penalty`
  Optional. A Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics.
- `response.format`
  Optional. An object specifying the format that the model must output. Compatible with GPT-4 Turbo and all GPT-3.5 Turbo models newer than `gpt-3.5-turbo-1106`.
- `seed`
  Optional. An integer. If specified, our system will make a best effort to sample deterministically, such that repeated requests with the same seed and parameters should return the same result.
- `stop`
  Optional. A character string or list contains up to 4 sequences where the API will stop generating further tokens.

- `temperature`

  Optional. A number. What sampling temperature to use, between 0 and 2. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic.

- `top.p`

  Optional. A number. 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

- `tools`

  Optional. A list of tools the model may call. Currently, only functions are supported as a tool. Use this to provide a list of functions the model may generate JSON inputs for.

- `tool.choice`

  Optional. A character string or object. Controls which (if any) function is called by the model. `none` means the model will not call a function and instead generates a message. `auto` means the model can pick between generating a message or calling a function.

- `user`

  Optional. A character string. A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse.

Value

If successful, a text response will be returned. If the API response indicates an error, the function halts execution and provides an error message.

See Also

GenAI - R Package "GenAI" Documentation

Live Demo in Colab

Examples

```
## Not run:
# Assuming there is a GenAI object named 'genai.model' supporting this
# function, an image in your current directory named 'example.png', and
# an online image 'https://example.com/example.png/', please refer to
# the "Live Demo in Colab" above for real examples. The following examples
# are just some basic guidelines.

# Method 1 (recommended): use the pipe operator "%>%"
genai.model %>%
  txt.image(prompt = "Please describe the following image.",
            image.path = "https://example.com/example.png/") %>%
  cat()

# Method 2: use the reference operator "$"
cat(genai.model$txt.image(prompt = "Please describe the following image.",
                          image.path = "https://example.com/example.png/"))
```

```
# Method 3: use the function txt.image() directly
cat(txt.image(genai.object = genai.model,
              prompt = "Please describe the following image.",
              image.path = "example.png"))

## End(Not run)
```

Index