Package ‘MacBehaviour’

February 20, 2024

Type Package

Title Behavioural Studies of Large Language Models

Version 1.1.3

Maintainer Shixuan Li <`shixuanli@cuhk.edu.hk`>

Description We provide an efficient way to design and conduct psycholinguistic experiments for testing the performance of large language models. It simplifies the process of setting up experiments, and data collection via large language models' API, streamlining workflow for researchers in the field of machine behavior. For methodology details, see Duan, X., Li, S., & Cai, Z. G. (2023) <doi:10.31234/osf.io/ywtfd>.

License LGPL-3

Encoding UTF-8

Depends R (>= 3.5.0), openxlsx, httr, dplyr, rjson

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, roxygen2

NeedsCompilation no

Author Xufeng Duan [aut, cph],
Shixuan Li [aut, cre],
Zhenguang Cai [aut]

Repository CRAN

Date/Publication 2024-02-20 20:20:12 UTC

R topics documented:

experimentDesign *Step3: Generate the experimental design matrix.*

Description

Defines the experiment setup based on the stimuli loaded.

Usage

```
experimentDesign(data, Step = 1, random = FALSE)
```

Arguments

data	A data frame that has been processed through the 'loadData' function, containing the experimental items and their attributes.
Step	An integer indicating how many sessions (the whole set of trials) should be run. Default is 1, meaning no repetition.
random	A logical indicating whether the trials should be randomized. Default is FALSE, meaning trials will occur in the order provided.

Value

A data.frame with the designed structure for the experiment, including any repetitions and randomizations as specified. Each row corresponds to a single trial or instance in the experiment.

Examples

```
df <- data.frame(
Run = c(1,2),
Item = c(1,2),
Condition = c(1,2),
TargetPrompt = c("1","2")
)

ExperimentItem=loadData(df$Run,df$Item,df$Condition,promptList = df$TargetPrompt)

Design=experimentDesign(ExperimentItem,Step=1,random = TRUE)
```

`loadData` *Step2: Load and format data*

Description

Prepares the stimuli data for the experiment.

Usage

```
loadData(runList, itemIDList, conditionList, promptList, header = TRUE)
```

Arguments

`runList`	A numeric vector of data representing the 'Run' column in the experiment.
`itemIDList`	A numeric vector of data representing the 'itemID' column in the experiment.
`conditionList`	A numeric/character vector of data representing the 'Condition' column in the experiment.
`promptList`	A character vector of the main prompt (usually experiment items).
`header`	A logical value indicating if the output data.frame should include column headers (default is TRUE).

Value

A data frame with the processed columns 'Run', 'Trial', 'Item', 'Condition', and 'Prompt', ready for use in experiments.

Examples

```
df <- data.frame(
Run = c(1,2),
Item = c(1,2),
Condition = c(1,2),
TargetPrompt = c("1","2")
)
ExperimentItem=loadData(df$Run,df$Item,df$Condition,promptList = df$TargetPrompt)
```

`magicTokenizer` *magicTokenizer*

Description

This function provides the number of tokens in a specified text, acting as a wrapper for an internal tokenizer function.

Usage

```
magicTokenizer(text)
```

Arguments

`text`	A character string: the text for which the number of tokens is required.

Value

Returns the number of tokens in the provided text.

`preCheck` *Step4: Pre-check for token usage in experiment design.*

Description

Configures experimental parameters before execution.

Usage

```
preCheck(
  data,
  checkToken = FALSE,
  systemPrompt = "",
  max_tokens = 500,
  temperature = 1,
  top_p = 1,
  n = 1,
  modality = "base",
  imgDetail = "auto"
)
```

Arguments

`data`	A data.frame that has been structured by the 'experimentDesign' function, containing the experimental setup.
`checkToken`	Whether to perform token count check, select TRUE to submit your experiment to our server's tokenizer for token count check, the default selection is FALSE (i.e., no token check will be performed, but you need to manually check if the number of tokens exceeds the model limit to avoid errors in the experiment).
`systemPrompt`	The system prompt text used in the chatGPT model interaction. If left empty, a space character is assumed.
`max_tokens`	The maximum number of tokens allowed for the model's response, default is 500.
`temperature`	The temperature setting for the chatGPT model, controlling randomness. Default is 0.7.
`top_p`	The top_p setting for the chatGPT model, controlling the diversity of responses. Default is 0.9.
`n`	The number of model responses per prompt, default is 1. Relevant only if 'oneTrialMode' is TRUE.
`modality`	The default mode of GPT is "base," with "img" as an optional choice.
`imgDetail`	The image quality of the img modality is set to auto by default, with low/high as selectable options.

Value

A list containing the original data and the parameters for the chatGPT model interaction, confirming that the setup has passed the token checks or indicating issues if found.

Examples

```
df <- data.frame(
Run = c(1,2),
Item = c(1,2),
Condition = c(1,2),
TargetPrompt = c("1","2")
)

ExperimentItem=loadData(df$Run,df$Item,df$Condition,promptList = df$TargetPrompt)

Design=experimentDesign(ExperimentItem,Step=1,random = TRUE)

gptConfig=preCheck(Design, systemPrompt="You are a participant in a psycholinguistic experiment",
             max_tokens=10,temperature=0.7,top_p=1,n=1,modality='base',imgDetail="low")
```

`runExperiment` *Run an Experiment Based on the Configuration*

Description

Executes the experiment and saves the results to an Excel file.

Usage

```
runExperiment(gptConfig, savePath = "./output.xlsx")
```

Arguments

`gptConfig`	A list containing the configuration for the language model, including the system prompt, model specifications, and token settings.
`savePath`	The file path where the experiment results will be saved in Excel format. Defaults to './output.xlsx' in the current working directory.

Value

This function does not return a value but saves the experiment results to the specified Excel file. Upon completion, "Done." will be printed to the console.

Examples

```
## Not run:

runExperiment(Experiment_config,"./output.xlsx")

#The first argument Experiment_config is generated by preCheck() function.

Experiment_config <- preCheck(data)

## End(Not run)
```

`setKey` *Step1: Set model's API key and url.*

Description

This function allows users to set and verify an API key for data collection. You can change the default api_url for open-source models' API.

todo

Usage

```
setKey(api_key, api_url = "https://api.openai.com/v1/chat/completions", model)
```

Arguments

`api_key`	A character string: the user's OpenAI/huggingface/other API key.Please fill 'NA' for self-deployed models.
`api_url`	A character string: the user's OpenAI/huggingface/other url .default is OpenAI.
`model`	A character string: specify the model version.

Value

Prints a message to the console indicating whether the API key setup was successful. If the setup fails, the function stops with an error message.

Examples

```
## Not run:
set_key(api_key="YOUR_API_KEY", api_url="api.openai.com/v1/chat/completions",model="gpt-3.5-turbo")

## End(Not run)
```

Index