Package ‘PaLMr’

October 20, 2023

Type Package
Title Interface for 'Google Pathways Language Model (PaLM)'
Version 0.1.4
Maintainer Li Yuan <lyuan@gd.edu.kg>
Description 'Google's 'PaLM' <https://developers.generativeai.google/> as a coding and writing assistant designed for 'R' and 'RStudio.' With a range of functions, including natural language processing and coding optimization, to assist R developers in simplifying tedious coding tasks and content searching.
License CC BY 4.0
URL https://palmr.r.ly.gd.edu.kg/
BugReports https://github.com/lygitdata/PaLMr/issues
Encoding UTF-8
RoxygenNote 7.2.3
Imports httr, jsonlite
NeedsCompilation no
Author Li Yuan [aut, cre]
Repository CRAN
Date/Publication 2023-10-20 10:30:02 UTC

R topics documented:

explainCodePALM ............................................. 2
fixGrammarPALM .......................................... 6
getReferencePALM .......................................... 9
getTextPALM ................................................ 13
optimizeCodePALM ......................................... 17
setupPALM .................................................. 21

Index 24
explainCodePALM

Explain code based on a query using the Google PaLM model.

Description

This function sends a query with a code snippet to the Google PaLM model and generates a detailed explanation of the code. It supports various programming languages and allows you to customize the explanation. The explanation includes a step-by-step breakdown of how the code works.

Usage

```r
explainCodePALM(
  palmParameter,  # A character vector containing the API key, model version, and model type, as provided by Google. The API key should be a 39-character string. Model version and type are specified by Google. See function `setupPALM()` for details.
inquery,         # A character string representing the code snippet for explanation. The length of the code snippet should be between 1 and 8196 characters, inclusive.
language = "R",  # A character string specifying the programming language used in the code (default: "R").
temperature = 0.7, # A numeric value between 0.0 and 1.0, inclusive (default: 0.7). Controls the randomness of the generated explanation. A higher value (e.g., 0.9) results in more creative responses, while a lower value (e.g., 0.3) produces more straightforward explanations.
maxOutputTokens = 1024, # An integer value between 1 and 1024, inclusive (default: 1024). Specifies the maximum number of tokens to include in the generated explanation.
  topP = 0.95,
  topK = 40,
  htUnspecified = "meda",
  htDerogatory = "meda",
  htToxicity = "meda",
  htViolence = "meda",
  htSexual = "meda",
  htMedical = "meda",
  htDangerous = "meda"
)
```

Arguments

- **palmParameter**: A character vector containing the API key, model version, and model type, as provided by Google. The API key should be a 39-character string. Model version and type are specified by Google. See function `setupPALM()` for details.
- **inquery**: A character string representing the code snippet for explanation. The length of the code snippet should be between 1 and 8196 characters, inclusive.
- **language**: A character string specifying the programming language used in the code (default: "R").
- **temperature**: A numeric value between 0.0 and 1.0, inclusive (default: 0.7). Controls the randomness of the generated explanation. A higher value (e.g., 0.9) results in more creative responses, while a lower value (e.g., 0.3) produces more straightforward explanations.
- **maxOutputTokens**: An integer value between 1 and 1024, inclusive (default: 1024). Specifies the maximum number of tokens to include in the generated explanation.
**topP**
A numeric value between 0.0 and 1.0, inclusive (default: 0.95). Defines the maximum cumulative probability of tokens considered when sampling. It controls the diversity of the explanation generated.

**topK**
An integer value between 1 and 1,000,000, inclusive (default: 40). Sets the maximum number of tokens to consider when sampling.

**htUnspecified**
Safety setting threshold for unspecified harm. The default threshold is "meda." Refer to HarmCategory - Google PaLMr. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htDerogatory**
Safety setting threshold for derogatory harm. The default threshold is "meda." Refer to HarmCategory - Google PaLMr. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htToxicity**
Safety setting threshold for toxicity harm. The default threshold is "meda." Refer to HarmCategory - Google PaLMr. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htViolence**
Safety setting threshold for violence harm. The default threshold is "meda." Refer to HarmCategory - Google PaLMr. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htSexual**
Safety setting threshold for sexual harm. The default threshold is "meda." Refer to HarmCategory - Google PaLMr. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htMedical**
Safety setting threshold for medical harm. The default threshold is "meda." Refer to HarmCategory - Google PaLMr. Valid options include:
explainCodePALM

- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htDangerous Safety setting threshold for dangerous harm. The default threshold is "meda."
Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

Details

This function interacts with the Google PaLM model by sending a code query to explain code. It
allows you to customize the generated code explanations by specifying the programming language,
and additional parameters like temperature, token limits, and safety settings.

If the function is successful, it returns a detailed explanation of the provided code as a character
string. If an error occurs during the API request, it will stop execution and provide an error message.

The ‘palmParameter’ argument should be a character vector with the API key, model version, and
model type provided by Google. You can obtain this information by following the instructions
provided by Google for using the PaLM API.

The safety settings control the content’s safety level based on different harm categories. Harm
thresholds are specified as per Google’s guidelines and can be customized to control the content
generated.

For more information on safety settings, harm categories, and harm thresholds, refer to the official

Value

A character string containing the detailed explanation of the provided code snippet based on the
query and parameters.

See Also

Safety Setting - Google PaLM
HarmCategory - Google PaLM

Examples

```r
## Not run:
# Set up the PaLM parameters
# Replace your_api_key_here with the API key you get from Google
palmParameter <- c("your_api_key_here", "v1beta3", "text-bison-001")

# Explain code based on a query
```
# Example 1

```r
inquery <- "
foo <- function(n) {
  if (n == 0) {
    return(1)
  } else {
    return(n * foo(n - 1))
  }
}
"

print(explainCodePALM(palmParameter, inquery, language = "R"))
```

# Example output:

## The function `foo` takes a number `n` as input and returns the value of `n`!
## The function uses a recursive approach, which means that it calls itself to
## calculate the factorial of smaller numbers. The base case is when `n == 0`,
## in which case the function simply returns `1`. Otherwise, the function multiplies
## `n` by the factorial of `n - 1`.
## ## Here is a step-by-step breakdown of how the function works:
## ## 1. When `n == 0`, the function returns `1`.
## ## 2. When `n > 0`, the function first multiplies `n` by the factorial of `n - 1`.
## ## 3. The function then calls itself to calculate the factorial of `n - 1`.
## ## 4. This process continues until the base case is reached, at which point the
## ## function returns `1`.
## 
## The following table shows how the function would calculate the factorial of 5:
## 
## | n | `foo(n)` |
## |---|---|
## | 0 | 1 |
## | 1 | 1 |
## | 2 | 2 |
## | 3 | 6 |
## | 4 | 24 |
## | 5 | 120 |

# Example 2

```r
inquery <- "
llm_data %>%
ggplot(aes(x=Training_Data,y=Params, label=Model)) +
geom_label() +
labs(
x = 'Training Data (billion tokens)',
y = 'Parameters (billions)') +
theme_bw()"

print(explainCodePALM(palmParameter, inquery, language = "Python"))
```

# Example output:

## The code above is using the `ggplot2` library to create a scatter plot of
Fix grammar and rewrite text using the Google PaLM model based on a query.

Description

This function sends a query with grammatical issues to the Google PaLM model and generates corrected text as a response. It allows customization of the generated text using various parameters.

Usage

```r
fixGrammarPALM(
  palmParameter, 
  inquery, 
  temperature = 0.7, 
  maxOutputTokens = 1024, 
  topP = 0.95, 
  topK = 40, 
  htUnspecified = "meda", 
  htDerogatory = "meda", 
  htToxicity = "meda", 
  htViolence = "meda", 
  htSexual = "meda", 
  htMedical = "meda", 
  htDangerous = "meda"
)
```
Arguments

palmParameter A character vector containing the API key, model version, and model type, as provided by Google. The API key should be a 39-character string. Model version and type are specified by Google. See function 'setupPALM()' for details.

inquiry A character string representing the text with grammatical issues that you want to rewrite. The length of the text should be between 1 and 8196 characters, inclusive.

temperature A numeric value between 0.0 and 1.0, inclusive (default: 0.7). Controls the randomness of the generated text. A higher value (e.g., 0.9) results in more creative responses, while a lower value (e.g., 0.3) produces more straightforward text.

maxOutputTokens An integer value between 1 and 1024, inclusive (default: 1024). Specifies the maximum number of tokens to include in the generated text.

topP A numeric value between 0.0 and 1.0, inclusive (default: 0.95). Defines the maximum cumulative probability of tokens considered when sampling. It controls the diversity of the text generated.

topK An integer value between 1 and 1,000,000, inclusive (default: 40). Sets the maximum number of tokens to consider when sampling.

htUnspecified Safety setting threshold for unspecified harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htDerogatory Safety setting threshold for derogatory harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htToxicity Safety setting threshold for toxicity harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htViolence Safety setting threshold for violence harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htSexual**  Safety setting threshold for sexual harm. The default threshold is "meda." Refer to [HarmCategory - Google PaLMr](#). Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htMedical**  Safety setting threshold for medical harm. The default threshold is "meda." Refer to [HarmCategory - Google PaLMr](#). Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htDangerous**  Safety setting threshold for dangerous harm. The default threshold is "meda." Refer to [HarmCategory - Google PaLMr](#). Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**Details**

This function interacts with the Google PaLM model by sending a query with grammatical issues using the specified parameters. It allows you to customize the generated text by adjusting the ‘temperature’, ‘maxOutputTokens’, ‘topP’, ‘topK’, and safety settings.

If the function is successful, it returns a character string containing the rewritten text with corrected grammar. If an error occurs during the API request, it will stop execution and provide an error message.

The ‘palmParameter’ argument should be a character vector with the API key, model version, and model type provided by Google. You can obtain this information by following the instructions provided by Google for using the PaLM API.

The safety settings control the content’s safety level based on different harm categories. Harm thresholds are specified as per Google’s guidelines and can be customized to control the content generated.

For more information on safety settings, harm categories, and harm thresholds, refer to the official Google PaLM API documentation: [Safety Setting - Google PaLMr](#)
**Value**

A character string containing the rewritten text with corrected grammar, generated by the Google PaLM API based on the provided query and parameters.

**See Also**

Safety Setting - Google PaLMr  
HarmCategory - Google PaLMr

**Examples**

```r
## Not run:  
# Set up the PaLM parameters  
# Replace your_api_key_here with the API key you get from Google  
palmParameter <- c("your_api_key_here", "v1beta3", "text-bison-001")  

# Fix grammar and rewrite text based on a query  

# Example 1  
inquiry <- "Yesterday, I will buy a book for my younger sister as his birthday gift. They were very happen when seeing this gift earlier today."
print(fixGrammarPALM(palmParameter, inquiry, temperature = 0.7))  

# Example output:  
# Yesterday, I bought a book for my younger sister as her birthday gift.  
# She was very happy when she saw it earlier today.

# Example 2  
inquiry <- "Dora begun to understan how the Teacher come up with the rules. By the end of the year, Dora though Mrs. Davis was the best Teacher she evere had!"
print(fixGrammarPALM(palmParameter, inquiry, temperature = 1))  

# Example output:  
# Dora began to understand how the teacher came up with the rules. By the end of the year, Dora thought Mrs. Davis was the best teacher she had ever had!  

## End(Not run)
```

**getReferencePALM**  
Get references based on a query using the Google PaLM model.

**Description**

This function sends a query to the Google PaLM model and generates a list of references based on the query. It allows customization of the generated references and supports various citation styles and source types.
Usage

getReferencePALM(
  palmParameter,
  inquery,
  sourceType = "articles",
  numSource = 5,
  citationStyle = "APA",
  sourceDate = "most recent",
  temperature = 0.7,
  maxOutputTokens = 1024,
  topP = 0.95,
  topK = 40,
  htUnspecified = "meda",
  htDerogatory = "meda",
  htToxicity = "meda",
  htViolence = "meda",
  htSexual = "meda",
  htMedical = "meda",
  htDangerous = "meda"
)

Arguments

palmParameter A character vector containing the API key, model version, and model type, as provided by Google. The API key should be a 39-character string. Model version and type are specified by Google. See function 'setupPALM()' for details.

inquery A character string representing the query for finding references. The length of the query should be between 1 and 8196 characters, inclusive.

sourceType A character string either "articles" or "websites" specifying the type of sources to search for (default: "articles").

numSource An integer value specifying the number of sources to retrieve (default: 5).

citationStyle A character string specifying the citation style for the references (default: "APA").

sourceDate A character string specifying the date range for the sources (default: "most recent").

temperature A numeric value between 0.0 and 1.0, inclusive (default: 0.7). Controls the randomness of the generated references. A higher value (e.g., 0.9) results in more creative responses, while a lower value (e.g., 0.3) produces more straightforward references.

maxOutputTokens An integer value between 1 and 1024, inclusive (default: 1024). Specifies the maximum number of tokens to include in the generated references.

topP A numeric value between 0.0 and 1.0, inclusive (default: 0.95). Defines the maximum cumulative probability of tokens considered when sampling. It controls the diversity of the references generated.

topK An integer value between 1 and 1,000,000, inclusive (default: 40). Sets the maximum number of tokens to consider when sampling.
Safety setting threshold for unspecified harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

Safety setting threshold for derogatory harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

Safety setting threshold for toxicity harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

Safety setting threshold for violence harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

Safety setting threshold for sexual harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

Safety setting threshold for medical harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)
getReferencePALM

htDANGEROUS  Safety setting threshold for dangerous harm. The default threshold is "meda."
Refer to HarmCategory - Google PaLMr. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

Details
This function interacts with the Google PaLM model by sending a query to find references. It
allows you to customize the generated references by specifying the number of sources, citation
style, source type, date range, and safety settings.
If the function is successful, it returns a character vector containing the generated references. If an
error occurs during the API request, it will stop execution and provide an error message.
The 'palmParameter' argument should be a character vector with the API key, model version, and
model type provided by Google. You can obtain this information by following the instructions
provided by Google for using the PaLM API.
The safety settings control the content’s safety level based on different harm categories. Harm
thresholds are specified as per Google’s guidelines and can be customized to control the content
generated.
For more information on safety settings, harm categories, and harm thresholds, refer to the official
Google PaLM API documentation: Safety Setting - Google PaLMr

Value
A character string containing the generated references based on the provided query and parameters.

See Also
Safety Setting - Google PaLMr
HarmCategory - Google PaLMr

Examples
## Not run:
# Set up the PaLM parameters
# Replace your_api_key_here with the API key you get from Google
palmParameter <- c("your_api_key_here", "v1beta3", "text-bison-001")

# Get references based on a query

# Example 1
inquiry <- "recurrent neural network"
print(getReferencePALM(palmParameter, inquiry, temperature = 0.7,
sourceType = "articles", numSource = 3,
citationStyle = "APA", sourceDate = "most recent"))
# Example output:

# Example 2

inquiry <- "H5N1"
print(getReferencePALM(palmParameter, inquiry, sourceType = "websites", numSource = 8, citationStyle = "MLA", sourceDate = "2019"))

# Example output:

## End(Not run)
Description

This function sends a query to the Google PaLM model and generates text as a response. It allows customization of the generated text using various parameters.

Usage

```
getTextPALM(
  palmParameter,  
  inquery, 
  temperature = 0.7, 
  maxOutputTokens = 1024, 
  topP = 0.95, 
  topK = 40, 
  htUnspecified = "meda", 
  htDerogatory = "meda", 
  htToxicity = "meda", 
  htViolence = "meda", 
  htSexual = "meda", 
  htMedical = "meda", 
  htDangerous = "meda"
)
```

Arguments

- **palmParameter**: A character vector containing the API key, model version, and model type, as provided by Google. The API key should be a 39-character string. Model version and type are specified by Google. See function `setupPALM()` for detail.

- **inquery**: A character string representing the query or prompt for text generation. The length of the query should be between 1 and 8196 characters, inclusive.

- **temperature**: A numeric value between 0.0 and 1.0, inclusive (default: 0.7). Controls the randomness of the generated text. A higher value (e.g., 0.9) results in more creative responses, while a lower value (e.g., 0.3) produces more straightforward text.

- **maxOutputTokens**: An integer value between 1 and 1024, inclusive (default: 1024). Specifies the maximum number of tokens to include in the generated text.

- **topP**: A numeric value between 0.0 and 1.0, inclusive (default: 0.95). Defines the maximum cumulative probability of tokens considered when sampling. It controls the diversity of the text generated.

- **topK**: An integer value between 1 and 1,000,000, inclusive (default: 40). Sets the maximum number of tokens to consider when sampling.

- **htUnspecified**: Safety setting threshold for unspecified harm. The default threshold is "meda." Refer to HarmCategory - Google PaLMr. Valid options include:
  - "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
  - "lowa" (BLOCK_LOW_AND_ABOVE)
  - "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htDerogatory Safety setting threshold for derogatory harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htToxicity Safety setting threshold for toxicity harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htViolence Safety setting threshold for violence harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htSexual Safety setting threshold for sexual harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htMedical Safety setting threshold for medical harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htDangerous Safety setting threshold for dangerous harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)
Details

This function interacts with the Google PaLM model by sending a query using the specified parameters. It allows you to customize the generated text by adjusting the 'temperature', 'maxOutputTokens', 'topP', 'topK', and safety settings.

If the function is successful, it returns a character string containing the generated text. If an error occurs during the API request, it will stop execution and provide an error message.

The 'palmParameter' argument should be a character vector with the API key, model version, and model type provided by Google. You can obtain this information by following the instructions provided by Google for using the PaLM API.

The safety settings control the content’s safety level based on different harm categories. Harm thresholds are specified as per Google’s guidelines and can be customized to control the content generated.

For more information on safety settings, harm categories, and harm thresholds, refer to the official Google PaLM API documentation: Safety Setting - Google PaLMr

Value

A character string generated by the Google PaLM API based on the provided query and parameters.

See Also

Safety Setting - Google PaLMr
HarmCategory - Google PaLMr

Examples

## Not run:
# Set up the PaLM parameters
# Replace your_api_key_here with the API key you get from Google
palmParameter <- c("your_api_key_here", "v1beta3", "text-bison-001")

# Generate text based on a query

# Example 1
inquery <- "Write a short fiction about Mars."
print(getTextPALM(palmParameter, inquery, temperature = 0.7,

maxOutputTokens = 100, topP = 0.95, topK = 40))

# The output character string may look like this if successful:
# "The year is 2042. Humanity has finally reached Mars, and the first human
# colony is being established. The colonists are a diverse group of people
# from all over the world, and they are all eager to start a new life on the
# Red Planet.
#
# The colony is located in a small valley near the equator. The climate is
# mild, and the soil is fertile. The colonists have brought with them everything
# they need to survive, including food, water, and shelter. They have also
# brought with them their hopes and dreams for the future."
## Example 2

```r
inquery <- "What is linear regression?"
print(getTextPALM(palmParameter, inquery))
```

# The output character string may look like this if successful:
# "Linear regression is a statistical method that is used to predict the
# value of a dependent variable based on the values of one or more
# independent variables. The dependent variable is the variable that you are
# trying to predict, and the independent variables are the variables that
# you are using to make the prediction."

## End(Not run)

---

### optimizeCodePALM

Optimize code based on a query using the Google PaLM model.

#### Description

This function sends a query with a code snippet to the Google PaLM model and generates an optimized version of the code. You can specify the programming language and the aspect you want to optimize (e.g., "runtime" or "memory"). The optimized code is provided along with the original code for comparison.

#### Usage

```r
optimizeCodePALM(
  palmParameter,
  inquery,
  language = "R",
  aspect = "general",
  temperature = 0.7,
  maxOutputTokens = 1024,
  topP = 0.95,
  topK = 40,
  htUnspecified = "meda",
  htDerogatory = "meda",
  htToxicity = "meda",
  htViolence = "meda",
  htSexual = "meda",
  htMedical = "meda",
  htDangerous = "meda"
)
```

#### Arguments

- `palmParameter` A character vector containing the API key, model version, and model type, as provided by Google. The API key should be a 39-character string. Model version and type are specified by Google. See function `setupPALM()` for details.
inquiry A character string representing the code snippet to be optimized. The length of the code snippet should be between 1 and 8196 characters, inclusive.

language A character string specifying the programming language used in the code (default: "R").

aspect A character string specifying the aspect you want to optimize, such as "runtime", "memory", "runtime&memory", and "general" (default: "general").

temperature A numeric value between 0.0 and 1.0, inclusive (default: 0.7). Controls the randomness of the generated optimization. A higher value (e.g., 0.9) results in more creative optimizations, while a lower value (e.g., 0.3) produces more straightforward optimizations.

maxOutputTokens An integer value between 1 and 1024, inclusive (default: 1024). Specifies the maximum number of tokens to include in the generated optimization.

topP A numeric value between 0.0 and 1.0, inclusive (default: 0.95). Defines the maximum cumulative probability of tokens considered when sampling. It controls the diversity of the optimization generated.

topK An integer value between 1 and 1,000,000, inclusive (default: 40). Sets the maximum number of tokens to consider when sampling.

htUnspecified Safety setting threshold for unspecified harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htDerogatory Safety setting threshold for derogatory harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htToxicity Safety setting threshold for toxicity harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

htViolence Safety setting threshold for violence harm. The default threshold is "meda." Refer to HarmCategory - Google PaLM. Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htSexual**  
Safety setting threshold for sexual harm. The default threshold is "meda." Refer to [HarmCategory - Google PaLM](#). Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htMedical**  
Safety setting threshold for medical harm. The default threshold is "meda." Refer to [HarmCategory - Google PaLM](#). Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**htDangerous**  
Safety setting threshold for dangerous harm. The default threshold is "meda." Refer to [HarmCategory - Google PaLM](#). Valid options include:
- "unsp" (HARM_BLOCK_THRESHOLD_UNSPECIFIED)
- "lowa" (BLOCK_LOW_AND_ABOVE)
- "meda" (BLOCK_MEDIUM_AND_ABOVE)
- "high" (BLOCK_ONLY_HIGH)
- "none" (BLOCK_NONE)

**Details**

This function interacts with the Google PaLM model by sending a code query for code optimization. It allows you to customize the generated code optimizations by specifying the programming language, optimization aspect, and additional parameters like temperature, token limits, and safety settings.

If the function is successful, it returns an optimized version of the provided code as a character string. If an error occurs during the API request, it will stop execution and provide an error message.

The 'palmParameter' argument should be a character vector with the API key, model version, and model type provided by Google. You can obtain this information by following the instructions provided by Google for using the PaLM API.

The safety settings control the content’s safety level based on different harm categories. Harm thresholds are specified as per Google’s guidelines and can be customized to control the content generated.

For more information on safety settings, harm categories, and harm thresholds, refer to the official Google PaLM API documentation: [Safety Setting - Google PaLM](#)

**Value**

A character string containing the optimized version of the provided code snippet based on the query and parameters.
See Also

Safety Setting - Google PaLMr
HarmCategory - Google PaLMr

Examples

```r
## Not run:
# Set up the PaLM parameters
# Replace your_api_key_here with the API key you get from Google
palmParameter <- c("your_api_key_here", "v1beta3", "text-bison-001")

# Optimize code based on a query

# Example 1
inquiry <- "
  foo <- function(n) {
    if (n <= 0) {
      return(0)
    } else if (n == 1) {
      return(1)
    } else {
      return(foo(n - 1) + foo(n - 2))
    }
  }
"

print(optimizeCodePALM(palmParameter, inquiry, temperature = 0.7,
  language = "R", aspect = "runtime"))

# Example output
## ``````````````````
## foo <- function(n) {
##   if (n <= 0) {
##     return(0)
##   } else if (n == 1) {
##     return(1)
##   } else {
##     return(foo(n - 1) + foo(n - 2))
##   }
## }
##`````````````````
## # Optimized version
## ```
## foo <- function(n) {
##   if (n <= 0) {
##     return(0)
##   } else if (n == 1) {
##     return(1)
##   } else {
##     a <- foo(n - 1)
##     b <- foo(n - 2)
##     return(a + b)
##   }
## ```
```
Example 2

```c++
#include <iostream>
#include <vector>

unsigned long long factorial_recursive(int n) {
    if (n <= 1) {
        return 1;
    } else {
        return n * factorial_recursive(n - 1);
    }
}

int main() {
    int n = 10;
    unsigned long long result = factorial_recursive(n);
    std::cout << 'Factorial of ' << n << ' is ' << result << std::endl;
    return 0;
}
```

Example output

```c++
#include <iostream>
#include <vector>

unsigned long long factorial_iterative(int n) {
    unsigned long long product = 1;
    for (int i = 2; i <= n; i++) {
        product *= i;
    }
    return product;
}

int main() {
    int n = 10;
    unsigned long long result = factorial_iterative(n);
    std::cout << "Factorial of " << n << " is " << result << std::endl;
    return 0;
}
```

End (Not run)
Description

This function establishes a connection to the Google PaLM model by specifying the API key, model version, and model type.

Usage

setupPALM(apiKey, modelVersion, modelType)

Arguments

apiKey  A character string representing the API key for accessing the Google PaLM model. The API key should be 39 characters long and must be of the "character" class.
modelVersion  The model version to use ("v1beta2" or "v1beta3"). Must be one of the available model versions.
modelType  The model type ("text-bison-001"). This is the only model type available for now.

Details

This function performs the necessary setup to connect to the Google PaLM model. It validates the provided API key and checks the correctness of the model version and type. If the input is valid, it constructs the API request and sends it to the PaLM API endpoint.

If an error occurs during the API request, such as an invalid API key or input parameters, an error message is displayed. If the API request is successful, the function prints the model details to the console and returns a character vector with the API key, model version, and model type.

Obtaining Google PaLM API key may not be available in some regions. Please refer to Available Regions - Google PaLM and Get An API Key - Google PaLM.

Value

If successful, the function returns a character vector containing the API key, model version, and model type. If the API response indicates an error, the function stops execution and provides an error message.

See Also

Available Regions - Google PaLM
Get An API Key - Google PaLM

Examples

## Not run:
# Replace your_api_key_here with the API key you get from Google
apiKey <- "your_api_key_here"
modelVersion <- "v1beta3"
modelType <- "text-bison-001"
result <- setupPALM(apiKey, modelVersion, modelType)
# The 'result' character vector may look like this if successful:
# [1] "your_api_key_here" "v1beta3" "text-bison-001"

## End(Not run)
Index

explainCodePALM, 2
fixGrammarPALM, 6
getReferencePALM, 9
g getTextPALM, 13
optimizeCodePALM, 17
setupPALM, 21