Package 'batchLLM'

October 14, 2024

Type Package

Title Batch Process LLM Text Completions Using a Data Frame

Version 0.2.0

Maintainer Dylan Pieper <dylanpieper@gmail.com>

Description Batch process large language model (LLM) text completions using data frame rows, with support for OpenAI's 'GPT' (<https://chat.openai.com>), Anthropic's 'Claude' (<https://claude.ai>), and Google's 'Gemini' (<https://gemini.google.com>). Includes features such as local storage, metadata logging, API rate limiting delays, and a 'shiny' app addin.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/dylanpieper/batchLLM

BugReports https://github.com/dylanpieper/batchLLM/issues

Imports openai, gemini.R, rlang, stats, digest, dplyr, shiny, shinyWidgets, shinydashboard, DT, httr, jsonlite, spsComps, shinyjs, readr, readxl

Depends R (>= 2.10)

LazyData true

NeedsCompilation no

Author Dylan Pieper [aut, cre, cph]

Repository CRAN

Date/Publication 2024-10-14 09:30:06 UTC

Contents

---

batchLLM *Batch Process LLM Text Completions Using a Data Frame*

Description

Batch process large language model (LLM) text completions by looping across the rows of a data frame column. The package currently supports OpenAI's GPT, Anthropic's Claude, and Google's Gemini models, with built-in delays for API rate limiting. The package provides advanced text processing features, including automatic logging of batches and metadata to local files, side-by-side comparison of outputs from different LLMs, and integration of a user-friendly Shiny App Addin. Use cases include natural language processing tasks such as sentiment analysis, thematic analysis, classification, labeling or tagging, and language translation.

Usage

```
batchLLM(
  df,
  df_name = NULL,
  col,
  prompt,
  LLM = "openai",
  model = "gpt-4o-mini",
  temperature = 0.5,
  max_tokens = 500,
  batch_delay = "random",
  batch_size = 10,
  case_convert = NULL,
  sanitize = FALSE,
  attempts = 1,
  log_name = "batchLLM-log",
  hash_algo = "crc32c",
  ...
)
```

Arguments

df	A data frame that contains the input data.
df_name	An optional string specifying the name of the data frame to log. This is particularly useful in Shiny applications or when the data frame is passed programmatically rather than explicitly. Default is NULL.
col	The name of the column in the data frame to process.
prompt	A system prompt for the LLM model.
LLM	A string for the name of the LLM with the options: "openai", "anthropic", and "google". Default is "openai".

model	A string for the name of the model from the LLM. Default is "gpt-4o-mini".
temperature	A temperature for the LLM model. Default is .5.
max_tokens	A maximum number of tokens to generate before stopping. Default is 500.
batch_delay	A string for the batch delay with the options: "random", "min", and "sec". Numeric examples include "1min" and "30sec". Default is "random" which is an average of 10.86 seconds (n = 1,000 simulations).
batch_size	The number of rows to process in each batch. Default is 10.
case_convert	A string for the case conversion of the output with the options: "upper", "lower", or NULL (no change). Default is NULL.
sanitize	Extract the LLM text completion from the model's response by returning only content in <result> XML tags. Additionally, remove all punctuation. This feature prevents unwanted text (e.g., preamble) or punctuation from being included in the model's output. Default is FALSE.
attempts	The maximum number of loop retry attempts. Default is 1.
log_name	A string for the name of the log without the .rds file extension. Default is "batchLLM-log".
hash_algo	A string for a hashing algorithm from the 'digest' package. Default is crc32c.
...	Additional arguments to pass on to the LLM API function.

Value

Returns the input data frame with an additional column containing the text completion output. The function also writes the output and metadata to the log file after each batch in a nested list format.

Examples

```
## Not run:
library(batchLLM)

# Set API keys
Sys.setenv(OPENAI_API_KEY = "your_openai_api_key")
Sys.setenv(ANTHROPIC_API_KEY = "your_anthropic_api_key")
Sys.setenv(GEMINI_API_KEY = "your_gemini_api_key")

# Define LLM configurations
llm_configs <- list(
  list(LLM = "openai", model = "gpt-4o-mini"),
  list(LLM = "anthropic", model = "claude-3-haiku-20240307"),
  list(LLM = "google", model = "1.5-flash")
)

# Apply batchLLM function to each configuration
beliefs <- lapply(llm_configs, function(config) {
  batchLLM(
    df = beliefs,
    col = statement,
    prompt = "classify as a fact or misinformation in one word",
    LLM = config$LLM,
```

```
    model = config$model,
    batch_size = 10,
    batch_delay = "1min",
    case_convert = "lower"
  )
})[[length(llm_configs)]]

# Print the updated data frame
print(beliefs)

## End(Not run)
```

`batchLLM_shiny` *Interact with batchLLM via a Shiny Gadget*

Description

This function provides a user interface using Shiny to interact with the `batchLLM` package. It allows users to configure and execute batch processing through an interactive dashboard.

Usage

```
batchLLM_shiny()
```

Value

No return value. Launches a Shiny Gadget that allows users to interact with the `batchLLM` package.

`beliefs` *Beliefs Dataset*

Description

The beliefs dataset consists of 20 statements representing opposing views on various scientific, environmental, and societal topics.

Usage

```
beliefs
```

Format

A data frame with 20 rows and 1 variable:

statement A character string with a statement representing a belief.

Examples

```
head(beliefs)
```

claudeR *Interact with Anthropic's Claude API*

Description

This function provides an interface to interact with Claude AI models via Anthropic's API, allowing for flexible text generation based on user inputs. This function was adapted from the claudeR repository by yrvelez on GitHub (MIT License).

Usage

```
claudeR(
  prompt,
  model = "claude-3-5-sonnet-20240620",
  max_tokens = 500,
  stop_sequences = NULL,
  temperature = 0.7,
  top_k = -1,
  top_p = -1,
  api_key = NULL,
  system_prompt = NULL
)
```

Arguments

prompt	A string vector for Claude-2, or a list for Claude-3 specifying the input for the model.
model	The model to use for the request. Default is the latest Claude-3 model.
max_tokens	A maximum number of tokens to generate before stopping.
stop_sequences	Optional. A list of strings upon which to stop generating.
temperature	Optional. Amount of randomness injected into the response.
top_k	Optional. Only sample from the top K options for each subsequent token.
top_p	Optional. Does nucleus sampling.
api_key	Your API key for authentication.
system_prompt	Optional. An optional system role specification.

Value

The resulting completion up to and excluding the stop sequences.

Examples

```
## Not run:
library(batchLLM)

# Set API in the env or use api_key parameter in the claudeR call
Sys.setenv(ANTHROPIC_API_KEY = "your_anthropic_api_key")

# Using Claude-2
response <- claudeR(
  prompt = "What is the capital of France?",
  model = "claude-2.1",
  max_tokens = 50
)
cat(response)

# Using Claude-3
response <- claudeR(
  prompt = list(
    list(role = "user", content = "What is the capital of France?")
  ),
  model = "claude-3-5-sonnet-20240620",
  max_tokens = 50,
  temperature = 0.8
)
cat(response)

# Using a system prompt
response <- claudeR(
  prompt = list(
    list(role = "user", content = "Summarize the history of France in one paragraph.")
  ),
  system_prompt = "You are a concise summarization assistant.",
  max_tokens = 500
)
cat(response)

## End(Not run)
```

---

get_batches *Get Batches*

Description

Get batches of generated output in a single data frame from the `.rds` log file.

Usage

```
get_batches(df_name = NULL, log_name = "batchLLM-log")
```

Arguments

df_name	A string to match the name of a processed data frame.
log_name	A string specifying the name of the log without the .rds file extension. Default is "batchLLM-log".

Value

A data frame containing the generated output.

Examples

```
## Not run:
library(batchLLM)

# Assuming you have a log file with data for "beliefs_40a3012b" (see batchLLM example)
batches <- get_batches("beliefs_40a3012b")
head(batches)

# Using a custom log file name
custom_batches <- get_batches("beliefs_40a3012b", log_name = "custom-log.rds")
head(custom_batches)

## End(Not run)
```

---

scrape_metadata *Scrape Metadata*

Description

Scrape metadata from the .rds log file.

Usage

```
scrape_metadata(df_name = NULL, log_name = "batchLLM-log")
```

Arguments

df_name	Optional. A string to match the name of a processed data frame.
log_name	A string specifying the name of the log file without the extension. Default is "batchLLM-log".

Value

A data frame containing metadata.

Examples

```
library(batchLLM)

# Scrape metadata for all data frames in the default log file
all_metadata <- scrape_metadata()
head(all_metadata)

# Scrape metadata for a specific data frame
specific_metadata <- scrape_metadata("beliefs_40a3012b")
head(specific_metadata)

# Use a custom log file name
custom_metadata <- scrape_metadata(log_name = "custom-log")
head(custom_metadata)
```

Index