Package 'edgemodelr'

September 22, 2025

Type Package

Title Local Language Model Inference

Version 0.1.0

Author Pawan Rama Mali [aut, cre]

Maintainer Pawan Rama Mali <prm@outlook.in>

Description Enables R users to run large language models locally using 'GGUF' model files and the 'llama.cpp' inference engine. Provides a complete R interface for loading models, generating text completions, and streaming responses in real-time. Supports local inference without requiring cloud APIs or internet connectivity, ensuring complete data privacy and control. References: 'Gerganov' et al. (2023) <https://github.com/ggml-org/llama.cpp>.

License MIT + file LICENSE

URL https://github.com/PawanRamaMali/edgemodelr

BugReports https://github.com/PawanRamaMali/edgemodelr/issues

Encoding UTF-8

Depends R (>= 4.0)

LinkingTo Rcpp

Imports Rcpp (>= 1.0.0), utils, tools

Suggests testthat (>= 3.0.0), knitr, rmarkdown

SystemRequirements C++17, GNU make or equivalent for building

Note This package includes a self-contained 'llama.cpp' implementation resulting in a larger installation size (~56MB) to provide complete functionality without external dependencies.

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-09-22 12:00:08 UTC

Contents

`build_chat_prompt` *Build chat prompt from conversation history*

Description

Build chat prompt from conversation history

Usage

```
build_chat_prompt(history)
```

Arguments

`history`	List of conversation turns with role and content

Value

Formatted prompt string

edge_chat_stream *Interactive chat session with streaming responses*

Description

Interactive chat session with streaming responses

Usage

```
edge_chat_stream(ctx, system_prompt = NULL, max_history = 10, n_predict = 200L,
                 temperature = 0.8, verbose = TRUE)
```

Arguments

ctx	Model context from edge_load_model()
system_prompt	Optional system prompt to set context
max_history	Maximum conversation turns to keep in context (default: 10)
n_predict	Maximum tokens per response (default: 200)
temperature	Sampling temperature (default: 0.8)
verbose	Whether to print responses to console (default: TRUE)

Value

NULL (runs interactively)

Examples

```
setup <- edge_quick_setup("TinyLlama-1.1B")
ctx <- setup$context

if (!is.null(ctx)) {
  # Start interactive chat with streaming
  # edge_chat_stream(ctx,
  #   system_prompt = "You are a helpful R programming assistant.")

  edge_free_model(ctx)
}
```

edge_clean_cache *Clean up cache directory and manage storage*

Description

Remove outdated model files from the cache directory to comply with CRAN policies about actively managing cached content and keeping sizes small.

Usage

```
edge_clean_cache(
  cache_dir = NULL,
  max_age_days = 30,
  max_size_mb = 500,
  interactive = TRUE,
  verbose = TRUE
)
```

Arguments

cache_dir	Cache directory path (default: user cache directory)
max_age_days	Maximum age of files to keep in days (default: 30)
max_size_mb	Maximum total cache size in MB (default: 500)
interactive	Whether to ask for user confirmation before deletion
verbose	Whether to print detailed messages about cleaning progress

Value

Invisible list of deleted files

Examples

```
# Clean cache files older than 30 days
edge_clean_cache()

# Clean cache with custom settings
edge_clean_cache(max_age_days = 7, max_size_mb = 100)
```

edge_completion *Generate text completion using loaded model*

Description

Generate text completion using loaded model

Usage

```
edge_completion(ctx, prompt, n_predict = 128L, temperature = 0.8, top_p = 0.95)
```

Arguments

ctx	Model context from edge_load_model()
prompt	Input text prompt
n_predict	Maximum tokens to generate (default: 128)
temperature	Sampling temperature (default: 0.8)
top_p	Top-p sampling parameter (default: 0.95)

Value

Generated text as character string

Examples

```
model_path <- "model.gguf"
if (file.exists(model_path)) {
  ctx <- edge_load_model(model_path)
  result <- edge_completion(ctx, "The capital of France is", n_predict = 50)
  cat(result)
  edge_free_model(ctx)
}
```

edge_download_model *Download a GGUF model from Hugging Face*

Description

Download a GGUF model from Hugging Face

Usage

```
edge_download_model(model_id, filename, cache_dir = NULL,
                    force_download = FALSE, verbose = TRUE)
```

Arguments

model_id	Hugging Face model identifier (e.g., "TheBloke/TinyLlama-1.1B-Chat-v1.0-GGUF")
filename	Specific GGUF file to download
cache_dir	Directory to store downloaded models (default: "~/.cache/edgemodelr")
force_download	Force re-download even if file exists
verbose	Whether to print download progress messages

Value

Path to the downloaded model file

Examples

```
# Download TinyLlama model
model_path <- edge_download_model(
  model_id = "TheBloke/TinyLlama-1.1B-Chat-v1.0-GGUF",
  filename = "tinyllama-1.1b-chat-v1.0.q4_k_m.gguf"
)

# Use the downloaded model (example only - requires actual model)
if (FALSE && file.exists(model_path)) {
  ctx <- edge_load_model(model_path)
  response <- edge_completion(ctx, "Hello, how are you?")
  edge_free_model(ctx)
}
```

---

edge_free_model *Free model context and release memory*

Description

Free model context and release memory

Usage

```
edge_free_model(ctx)
```

Arguments

ctx	Model context from edge_load_model()

Value

NULL (invisibly)

Examples

```
model_path <- "model.gguf"
if (file.exists(model_path)) {
  ctx <- edge_load_model(model_path)
  # ... use model ...
  edge_free_model(ctx)  # Clean up
}
```

`edge_list_models` *List popular pre-configured models*

Description

List popular pre-configured models

Usage

```
edge_list_models()
```

Value

Data frame with model information

`edge_load_model` *Load a local GGUF model for inference*

Description

Load a local GGUF model for inference

Usage

```
edge_load_model(model_path, n_ctx = 2048L, n_gpu_layers = 0L)
```

Arguments

`model_path`	Path to a .gguf model file
`n_ctx`	Maximum context length (default: 2048)
`n_gpu_layers`	Number of layers to offload to GPU (default: 0, CPU-only)

Value

External pointer to the loaded model context

Examples

```
# Load a TinyLlama model
model_path <- "~/models/TinyLlama-1.1B-Chat.Q4_K_M.gguf"
if (file.exists(model_path)) {
  ctx <- edge_load_model(model_path, n_ctx = 2048)

  # Generate completion
  result <- edge_completion(ctx, "Explain R data.frame:", n_predict = 100)
  cat(result)

  # Free model when done
  edge_free_model(ctx)
}
```

`edge_quick_setup` *Quick setup for a popular model*

Description

Quick setup for a popular model

Usage

```
edge_quick_setup(model_name, cache_dir = NULL, verbose = TRUE)
```

Arguments

`model_name`	Name of the model from edge_list_models()
`cache_dir`	Directory to store downloaded models
`verbose`	Whether to print setup progress messages

Value

List with model path and context (if llama.cpp is available)

Examples

```
# Quick setup with TinyLlama
setup <- edge_quick_setup("TinyLlama-1.1B")
ctx <- setup$context

if (!is.null(ctx)) {
  response <- edge_completion(ctx, "Hello!")
  cat("Response:", response, "\n")
  edge_free_model(ctx)
}
```

edge_set_verbose *Control llama.cpp logging verbosity*

Description

Enable or disable verbose output from the underlying llama.cpp library. By default, all output except errors is suppressed to comply with CRAN policies.

Usage

```
edge_set_verbose(enabled = FALSE)
```

Arguments

enabled	Logical. If TRUE, enables verbose llama.cpp output. If FALSE (default), suppresses all output except errors.

Value

Invisible NULL

Examples

```
# Enable verbose output (not recommended for normal use)
edge_set_verbose(TRUE)

# Disable verbose output (default, recommended)
edge_set_verbose(FALSE)
```

edge_stream_completion *Stream text completion with real-time token generation*

Description

Stream text completion with real-time token generation

Usage

```
edge_stream_completion(ctx, prompt, callback, n_predict = 128L, temperature = 0.8,
                       top_p = 0.95)
```

Arguments

ctx	Model context from edge_load_model()
prompt	Input text prompt
callback	Function called for each generated token. Receives list with token info.
n_predict	Maximum tokens to generate (default: 128)
temperature	Sampling temperature (default: 0.8)
top_p	Top-p sampling parameter (default: 0.95)

Value

List with full response and generation statistics

Examples

```
model_path <- "model.gguf"
if (file.exists(model_path)) {
  ctx <- edge_load_model(model_path)

  # Basic streaming with token display
  result <- edge_stream_completion(ctx, "Hello, how are you?",
    callback = function(data) {
      if (!data$is_final) {
        cat(data$token)
        flush.console()
      } else {
        cat("\n[Done: ", data$total_tokens, " tokens]\n")
      }
      return(TRUE)  # Continue generation
    })

  edge_free_model(ctx)
}
```

---

is_valid_model *Check if model context is valid*

Description

Check if model context is valid

Usage

```
is_valid_model(ctx)
```

Arguments

ctx	Model context to check

Value

Logical indicating if context is valid

Index