

Package ‘edgemodelr’

January 22, 2026

Type Package

Title Local Large Language Model Inference Engine

Version 0.1.4

Author Pawan Rama Mali [aut, cre, cph],
Georgi Gerganov [aut, cph] (Author of llama.cpp and GGML library),
The ggml authors [cph] (llama.cpp and GGML contributors),
Jeffrey Quesnelle [ctb, cph] (YaRN RoPE implementation),
Bowen Peng [ctb, cph] (YaRN RoPE implementation),
pi6am [ctb] (DRY sampler from Koboldcpp),
Ivan Yurchenko [ctb] (Z-algorithm implementation),
Dirk Eddelbuettel [ctb] (Connection handling fix)

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. Based on the 'llama.cpp' project by Georgi Gerganov (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, curl

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

Note Package includes self-contained 'llama.cpp' implementation (~56MB) for complete functionality without external dependencies.

Config/testthat/edition 3
RoxygenNote 7.3.3
NeedsCompilation yes
Repository CRAN
Date/Publication 2026-01-22 08:10:08 UTC

Contents

build_chat_prompt	2
edge_benchmark	3
edge_chat_stream	4
edge_clean_cache	5
edge_completion	6
edge_download_model	6
edge_download_url	7
edge_find_gguf_models	8
edge_find_ollama_models	10
edge_free_model	11
edge_list_models	11
edge_load_model	12
edge_load_ollama_model	12
edge_quick_setup	13
edge_set_verbose	14
edge_small_model_config	15
edge_stream_completion	16
is_valid_model	17
test_ollama_model_compatibility	17

Index	19
--------------	-----------

build_chat_prompt	<i>Build chat prompt from conversation history</i>
-------------------	--

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_benchmark	<i>Performance benchmarking for model inference</i>
----------------	---

Description

Test inference speed and throughput with the current model to measure the effectiveness of optimizations.

Usage

```
edge_benchmark(  
  ctx,  
  prompt = "The quick brown fox",  
  n_predict = 50,  
  iterations = 3  
)
```

Arguments

- ctx Model context from edge_load_model()
- prompt Test prompt to use for benchmarking (default: standard test)
- n_predict Number of tokens to generate for the test
- iterations Number of test iterations to average results

Value

List with performance metrics

Examples

```
setup <- edge_quick_setup("TinyLlama-1.1B")  
if (!is.null(setup$context)) {  
  ctx <- setup$context  
  perf <- edge_benchmark(ctx)  
  print(perf)  
  edge_free_model(ctx)  
}
```

edge_chat_stream	<i>Interactive chat session with streaming responses</i>
------------------	--

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	<i>Clean up cache directory and manage storage</i>
------------------	--

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 status messages (default: TRUE)

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	<i>Generate text completion using loaded model</i>
-----------------	--

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	<i>Download a GGUF model from Hugging Face</i>
---------------------	--

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_download_url	<i>Download a model from a direct URL</i>
-------------------	---

Description

Downloads a GGUF model file from any URL. Supports resume and validates GGUF format. This function is useful for downloading models from GPT4All CDN or other direct sources that don't require authentication.

Usage

```
edge_download_url(url, filename, cache_dir = NULL,
                  force_download = FALSE, verbose = TRUE)
```

Arguments

url	Direct download URL for the model
filename	Local filename to save as
cache_dir	Directory to store downloaded models (default: user cache directory)
force_download	Force re-download even if file exists
verbose	Whether to print progress messages

Value

Path to the downloaded model file

Examples

```
# Download from GPT4All CDN
model_path <- edge_download_url(
  url = "https://gpt4all.io/models/gguf/mistral-7b-instruct-v0.1.Q4_0.gguf",
  filename = "mistral-7b.gguf"
)

# Use the downloaded model
if (file.exists(model_path)) {
  ctx <- edge_load_model(model_path)
  response <- edge_completion(ctx, "Hello!")
  edge_free_model(ctx)
}
```

edge_find_gguf_models *Find and prepare GGUF models for use with edgemodelr*

Description

This function finds compatible GGUF model files from various sources including Ollama installations, custom directories, or any folder containing GGUF files. It tests each model for compatibility with edgemodelr and creates organized copies or links for easy access.

Usage

```
edge_find_gguf_models(
  source_dirs = NULL,
  target_dir = NULL,
  create_links = TRUE,
  model_pattern = NULL,
  test_compatibility = TRUE,
  min_size_mb = 50,
  verbose = TRUE
)
```

Arguments

source_dirs	Vector of directories to search for GGUF files. If NULL, automatically searches common locations including Ollama installation.
target_dir	Directory where to create links/copies of compatible models. If NULL, creates a "local_models" directory in the current working directory.
create_links	Logical. If TRUE (default), creates symbolic links to save disk space. If FALSE, copies the files (uses more disk space but more compatible).

model_pattern	Optional pattern to filter model files by name.
test_compatibility	Logical. If TRUE (default), tests each GGUF file for compatibility with edgemodelr before including it.
min_size_mb	Minimum file size in MB to consider (default: 50MB). Helps filter out config files and focus on actual models.
verbose	Logical. Whether to print detailed progress information.

Details

This function performs the following steps:

1. Searches specified directories (or auto-detects common locations)
2. Identifies GGUF format files above the minimum size threshold
3. Optionally tests each file for compatibility with edgemodelr
4. Creates organized symbolic links or copies in the target directory
5. Returns detailed information about working models

The function automatically searches these locations if no source_dirs specified:

- Ollama models directory (~/.ollama/models or %USERPROFILE%/.ollama/models)
- Current working directory
- ~/models directory (if exists)
- Common model storage locations

Value

List containing information about compatible models, including paths and metadata

Examples

```
# Basic usage - auto-detect and test all GGUF models
models_info <- edge_find_gguf_models()
if (!is.null(models_info) && length(models_info$models) > 0) {
  # Load the first compatible model
  ctx <- edge_load_model(models_info$models[[1]]$path)
  result <- edge_completion(ctx, "Hello", n_predict = 20)
  edge_free_model(ctx)
}

# Search specific directories
models_info <- edge_find_gguf_models(source_dirs = c("~/Downloads", "~/models"))

# Skip compatibility testing (faster but less reliable)
models_info <- edge_find_gguf_models(test_compatibility = FALSE)

# Copy files instead of creating links
models_info <- edge_find_gguf_models(create_links = FALSE)
```

```
# Filter for specific models
models_info <- edge_find_gguf_models(model_pattern = "llama")
```

```
edge_find_ollama_models
```

Find and load Ollama models

Description

Utility functions to discover and work with locally stored Ollama models. Ollama stores models as SHA-256 named blobs which are GGUF files that can be used directly with edgemodelr.

Usage

```
edge_find_ollama_models(
  ollama_dir = NULL,
  test_compatibility = FALSE,
  max_size_gb = 10
)
```

Arguments

`ollama_dir` Optional path to Ollama models directory. If NULL, will auto-detect.

`test_compatibility` If TRUE, test if each model can be loaded successfully

`max_size_gb` Maximum model size in GB to consider (default: 10)

Value

List with `ollama_path` and discovered models information

Examples

```
# Find Ollama models
ollama_info <- edge_find_ollama_models()

if (!is.null(ollama_info) && length(ollama_info$models) > 0) {
  # Use first compatible model
  model_path <- ollama_info$models[[1]]$path
  ctx <- edge_load_model(model_path)
  result <- edge_completion(ctx, "Hello", n_predict = 10)
  edge_free_model(ctx)
}
```

edge_free_model	<i>Free model context and release memory</i>
-----------------	--

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	<i>List popular pre-configured models</i>
------------------	---

Description

List popular pre-configured models

Usage

```
edge_list_models()
```

Value

Data frame with model information

edge_load_model	<i>Load a local GGUF model for inference</i>
-----------------	--

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_load_ollama_model	<i>Load an Ollama model by partial SHA-256 hash</i>
------------------------	---

Description

Find and load an Ollama model using a partial SHA-256 hash instead of the full path. This is more convenient than typing out the full blob path.

Usage

```
edge_load_ollama_model(partial_hash, n_ctx = 2048L, n_gpu_layers = 0L)
```

Arguments

partial_hash	First few characters of the SHA-256 hash
n_ctx	Maximum context length (default: 2048)
n_gpu_layers	Number of layers to offload to GPU (default: 0)

Value

Model context if successful, throws error if not found or incompatible

Examples

```
# Load model using first 8 characters of SHA hash
# ctx <- edge_load_ollama_model("b112e727")
# result <- edge_completion(ctx, "Hello", n_predict = 10)
# edge_free_model(ctx)
```

edge_quick_setup	<i>Quick setup for a popular model</i>
------------------	--

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	<i>Control llama.cpp logging verbosity</i>
------------------	--

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_small_model_config`*Get optimized configuration for small language models*

Description

Returns recommended parameters for loading and using small models (1B-3B parameters) to maximize inference speed on resource-constrained devices.

Usage

```
edge_small_model_config(  
  model_size_mb = NULL,  
  available_ram_gb = NULL,  
  target = "laptop"  
)
```

Arguments

<code>model_size_mb</code>	Model file size in MB (if known). If NULL, uses conservative defaults.
<code>available_ram_gb</code>	Available system RAM in GB. If NULL, uses conservative defaults.
<code>target</code>	Device target: "mobile", "laptop", "desktop", or "server" (default: "laptop")

Value

List with optimized parameters for `edge_load_model()` and `edge_completion()`

Examples

```
# Get optimized config for a 700MB model on a laptop  
config <- edge_small_model_config(model_size_mb = 700, available_ram_gb = 8)  
  
# Use the config to load a model  
  
model_path <- "path/to/tinyllama.gguf"  
if (file.exists(model_path)) {  
  ctx <- edge_load_model(  
    model_path,  
    n_ctx = config$n_ctx,  
    n_gpu_layers = config$n_gpu_layers  
  )  
  
  result <- edge_completion(  
    ctx,  
    prompt = "Hello",  
    n_predict = config$recommended_n_predict,  
    temperature = config$recommended_temperature  
  )  
}
```

```

    )
    edge_free_model(ctx)
}

```

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	<i>Check if model context is valid</i>
----------------	--

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

test_ollama_model_compatibility	<i>Test if an Ollama model blob can be used with edgemodelr</i>
---------------------------------	---

Description

This function tries to load an Ollama GGUF blob with edgemodelr using a minimal configuration and then runs a very short completion. It is intended to quickly detect common incompatibilities (unsupported architectures, invalid or unsupported GGUF files, or models that cannot run inference) before you attempt to use the model in a longer session.

Usage

```
test_ollama_model_compatibility(model_path, verbose = FALSE)
```

Arguments

model_path Path to the Ollama blob file (a GGUF file, typically named by its SHA-256 hash inside the Ollama models/blobs directory).

verbose If TRUE, print human-readable diagnostics for models that fail the compatibility checks.

Details

A model is considered compatible if:

- `edge_load_model()` succeeds with a small context size (`n_ctx = 256`) and CPU-only execution (`n_gpu_layers = 0`),
- the resulting model context passes `is_valid_model()`,
- and a minimal call to `edge_completion()` (1 token) returns without error.

When `verbose = TRUE`, this function classifies common failure modes: unsupported model architecture, invalid GGUF file, unsupported GGUF version, or a generic error (first 80 characters reported with truncation indicator).

Value

Logical: `TRUE` if the model loads and can run a short completion successfully, `FALSE` otherwise.

Examples

```
# Test an individual Ollama blob
# is_ok <- test_ollama_model_compatibility("/path/to/blob", verbose = TRUE)
#
# This function is also used internally by edge_find_ollama_models()
# when test_compatibility = TRUE.
```

Index

`build_chat_prompt`, [2](#)

`edge_benchmark`, [3](#)
`edge_chat_stream`, [4](#)
`edge_clean_cache`, [5](#)
`edge_completion`, [6](#)
`edge_download_model`, [6](#)
`edge_download_url`, [7](#)
`edge_find_gguf_models`, [8](#)
`edge_find_ollama_models`, [10](#)
`edge_free_model`, [11](#)
`edge_list_models`, [11](#)
`edge_load_model`, [12](#)
`edge_load_ollama_model`, [12](#)
`edge_quick_setup`, [13](#)
`edge_set_verbose`, [14](#)
`edge_small_model_config`, [15](#)
`edge_stream_completion`, [16](#)

`is_valid_model`, [17](#)

`test_ollama_model_compatibility`, [17](#)