Package: Rbebelm (via r-universe)

July 21, 2026

Title Native 'BebeLM', 'EmbeddingGemma', and 'ColBERT' CPU Backends for R

Version 0.3.6-0.1.0

Description Provides focused R interfaces to the pure-Rust 'BebeLM' CPU inference library <https://github.com/maximecb/bebelm> and a native pure-Rust implementation of the retrieval-trained 'EmbeddingGemma' encoder <https://ai.google.dev/gemma/docs/embeddinggemma>, and a native profile for LiquidAI's retrieval-trained LFM2.5-'ColBERT' GGUF encoder. The package exposes GGUF model loading, tokenizer access, semantic text embeddings, late-interaction token vectors and MaxSim scoring, streaming generation events, persistent conversation agents, 'BebeLM' tool-call parsing, R tool dispatch, and Rust-thread async generation jobs. Loaded agents share model weights in process through Rust reference-counted handles, and GGUF files are mapped read-only so repeated loads of the same file can share physical pages through the operating-system page cache. Native libraries are built as runtime-selected CPU backends so portable R binaries can use scalar code by default and optimized SIMD backends when available.

License GPL (>= 3)

URL https://github.com/sounkou-bioinfo/Rbebelm, https://sounkou-bioinfo.github.io/Rbebelm

BugReports https://github.com/sounkou-bioinfo/Rbebelm/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

SystemRequirements Rust toolchain with cargo (>= 1.85), GNU make

Imports S7, yyjsonr

Suggests tinytest, roxygen2, rmarkdown, knitr, pkgdown

Config/roxygen2/version 8.0.0

Config/pak/sysreqs make libclang-dev

Repository https://rgenomicsetl.r-universe.dev

Date/Publication 2026-07-20 19:41:18 UTC

RemoteUrl https://github.com/RGenomicsETL/Rbebelm

RemoteRef HEAD

RemoteSha 690b70436c2c9583b3c87f0d35e9b99e8caa5670

Contents

bebel_agent *Create a persistent BebeLM agent*

Description

A BebelAgent owns an independent token transcript and decode cache while sharing the loaded model weights through Rust Arc<Model>.

Usage

```
bebel_agent(
  model,
  greedy = FALSE,
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL
)
```

Arguments

model	A BebelModel object.
greedy	Use deterministic greedy decoding.
max_gen, max_context, max_think	Optional generation limits.
temperature, top_k, repeat_penalty	Optional sampling settings.

Value

A BebelAgent object.

bebel_agent_configure *Configure a BebeLM agent*

Description

Configure a BebeLM agent

Usage

```
bebel_agent_configure(
  agent,
  greedy = NULL,
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL
)
```

Arguments

agent	A BebelAgent object.
greedy	Use deterministic greedy decoding.
max_gen, max_context, max_think	Optional generation limits.
temperature, top_k, repeat_penalty	Optional sampling settings.

Value

Updated agent info.

---

bebel_agent_generate *Generate a raw continuation from a BebeLM agent transcript*

Description

Generate a raw continuation from a BebeLM agent transcript

Usage

```
bebel_agent_generate(agent, on_event = NULL, check_interrupt = TRUE)
```

Arguments

agent	A BebelAgent object.
on_event	Event handler function, named list of event-specific handlers, or NULL. Event types are bebel_event_types(). Delta events contain delta, id, and index; final events contain accumulated content or text.
check_interrupt	Check for R interrupts during synchronous agent generation.

Value

A classed generation result.

bebel_agent_generate_async *Start a background raw agent generation job*

Description

The job runs on a cloned agent snapshot. The original agent's transcript and decode cache are not mutated, while the model weights are shared.

Usage

```
bebel_agent_generate_async(agent)
```

Arguments

agent	A BebelAgent object.

Value

A BebelAsyncJob.

bebel_agent_info *Inspect a BebeLM agent*

Description

Inspect a BebeLM agent

Usage

```
bebel_agent_info(agent)
```

Arguments

agent	A BebelAgent object.

Value

Named list of state and configuration.

bebel_agent_run *Run a BebeLM agent with R tool dispatch*

Description

This is an Agent-first orchestration loop. It observes `tool_call_end` events, parses tool calls, invokes matching R tools with private `context`, appends tool results to the agent transcript, and continues generation.

Usage

```
bebel_agent_run(
  agent,
  tools = list(),
  context = new.env(parent = emptyenv()),
  hooks = list(),
  parse_tool_call = bebel_parse_tool_calls,
  max_steps = 4,
  on_event = NULL,
  check_interrupt = TRUE
)
```

Arguments

`agent`	A `BebelAgent` object.
`tools`	A list of `bebel_tool()` objects or named functions.
`context`	Private run context passed to tools and hooks but not appended to the model transcript.
`hooks`	Optional named list of hooks: `turn_start`, `event`, `tool_request`, `tool_result`, `tool_error`, `turn_end`.
`parse_tool_call`	Function converting tool-call content to either one `list(name, arguments, raw)` or a list of such calls.
`max_steps`	Maximum assistant/tool iterations.
`on_event`	Optional event handler function or named handler list for model events.
`check_interrupt`	Check for Ctrl-C during generation.

Value

A `bebelAgentRun` list with turns, tool calls, and final agent info.

bebel_append *Append raw text to a BebeLM agent transcript*

Description

Append raw text to a BebeLM agent transcript

Usage

```
bebel_append(agent, text)
```

Arguments

agent	A BebelAgent object.
text	Raw text to append.

Value

Invisibly returns agent.

bebel_append_system *Append an upstream BebeLM system turn to an agent transcript*

Description

Delegates ChatML system-turn rendering to upstream BebeLM. When tools are supplied, their schemas are rendered in upstream's List of tools: [...] system-block preamble before message.

Usage

```
bebel_append_system(agent, message, tools = NULL)
```

Arguments

agent	A BebelAgent object.
message	System instruction text.
tools	Optional list of bebel_tool() objects or named functions to advertise.

Value

Invisibly returns agent.

bebel_append_tokens *Append token ids to a BebeLM agent transcript*

Description

Append token ids to a BebeLM agent transcript

Usage

```
bebel_append_tokens(agent, ids)
```

Arguments

agent	A BebelAgent object.
ids	Integer token ids.

Value

Invisibly returns agent.

bebel_append_tool_result *Append a ChatML tool result turn to a BebeLM agent transcript*

Description

Append a ChatML tool result turn to a BebeLM agent transcript

Usage

```
bebel_append_tool_result(agent, content)
```

Arguments

agent	A BebelAgent object.
content	Tool result content to append.

Value

Invisibly returns agent.

bebel_append_user *Append a ChatML user turn to a BebeLM agent transcript*

Description

Append a ChatML user turn to a BebeLM agent transcript

Usage

```
bebel_append_user(agent, message)
```

Arguments

agent	A BebelAgent object.
message	User message.

Value

Invisibly returns agent.

bebel_assistant_turn *Generate and close an assistant ChatML turn from a BebeLM agent*

Description

Generate and close an assistant ChatML turn from a BebeLM agent

Usage

```
bebel_assistant_turn(agent, on_event = NULL, check_interrupt = TRUE)
```

Arguments

agent	A BebelAgent object.
on_event	Event handler function, named list of event-specific handlers, or NULL. Event types are bebel_event_types(). Delta events contain delta, id, and index; final events contain accumulated content or text.
check_interrupt	Check for R interrupts during synchronous agent generation.

Value

A classed generation result.

bebel_assistant_turn_async

Start a background assistant-turn job

Description

Start a background assistant-turn job

Usage

```
bebel_assistant_turn_async(agent)
```

Arguments

agent	A BebelAgent object.

Value

A BebelAsyncJob.

bebel_assistant_turn_tool_stop

Open an assistant turn and stop when a tool call closes

Description

This low-level variant mirrors upstream BebeLM's tool driver stop semantics: generation stops with stop == "tool_call" after <|tool_call_end|> so the caller can execute the requested tool and append one tool-result turn.

Usage

```
bebel_assistant_turn_tool_stop(agent, on_event = NULL, check_interrupt = TRUE)
```

Arguments

agent	A BebelAgent object.
on_event	Event handler function, named list of event-specific handlers, or NULL. Event types are bebel_event_types(). Delta events contain delta, id, and index; final events contain accumulated content or text.
check_interrupt	Check for R interrupts during synchronous agent generation.

Value

A bebelAssistantTurnResult list.

bebel_assistant_turn_tool_stop_async *Start a background assistant-turn job that stops on tool-call close*

Description

Start a background assistant-turn job that stops on tool-call close

Usage

```
bebel_assistant_turn_tool_stop_async(agent)
```

Arguments

agent	A BebelAgent object.

Value

A BebelAsyncJob.

bebel_async_cancel *Cancel a BebeLM async job*

Description

Requests cancellation from Rust. A cancelled job stops at the next generation checkpoint and raises an error when collected.

Usage

```
bebel_async_cancel(job)
```

Arguments

job	A BebelAsyncJob.

Value

TRUE when this call set the cancellation flag for the first time.

bebel_async_collect *Collect a BebeLM async job result*

Description

Collect a BebeLM async job result

Usage

```
bebel_async_collect(job, wait = TRUE)
```

Arguments

job	A BebelAsyncJob.
wait	If FALSE, return NULL when the job is still running.

Value

A classed generation result, or NULL.

bebel_async_events *Drain queued BebeLM async job events*

Description

Drain queued BebeLM async job events

Usage

```
bebel_async_events(job, max = NULL)
```

Arguments

job	A BebelAsyncJob.
max	Optional maximum number of queued events to drain.

Value

A list of generation event lists.

bebel_async_poll *Poll a BebeLM async job*

Description

Poll a BebeLM async job

Usage

```
bebel_async_poll(job)
```

Arguments

job	A BebelAsyncJob.

Value

"ready" or "pending".

bebel_async_wait *Wait for a BebeLM async job*

Description

Drains queued stream events on the R thread while polling the job, then collects the finished result.

Usage

```
bebel_async_wait(
  job,
  on_event = NULL,
  poll_interval = 0.005,
  cancel_on_interrupt = TRUE
)
```

Arguments

job	A BebelAsyncJob.
on_event	Event handler function, named list of event-specific handlers, or NULL.
poll_interval	Seconds to sleep between polls while the job is pending.
cancel_on_interrupt	Whether an interrupted wait should request Rust-side job cancellation.

Value

A classed generation result.

`bebel_benchmark_generation` *Benchmark async BebeLM generation throughput*

Description

Launches deterministic generation jobs in bounded async batches against one loaded model and records per-job timing, token counts, event counts, and aggregate throughput.

Usage

```
bebel_benchmark_generation(
  model,
  prompts,
  concurrency = min(length(prompts), 2L),
  repeats = 1L,
  greedy = TRUE,
  max_gen = 64L,
  max_context = NULL,
  max_think = 0L,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL,
  poll_interval = 0.001
)
```

Arguments

`model`	A `BebelModel` object.
`prompts`	Character vector of prompts.
`concurrency`	Maximum number of async jobs in flight.
`repeats`	Number of times to repeat the prompt set.
`greedy`	Use deterministic greedy decoding.
`max_gen`, `max_context`, `max_think`	Optional generation limits.
`temperature`, `top_k`, `repeat_penalty`	Optional sampling settings.
`poll_interval`	Seconds to sleep between async-job polls.

Value

A `bebelGenerationBenchmark` list.

bebel_chat *Generate a single ChatML assistant reply*

Description

Generate a single ChatML assistant reply

Usage

```
bebel_chat(
  model,
  message,
  greedy = FALSE,
  on_event = NULL,
  check_interrupt = TRUE,
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL,
  poll_interval = 0.005
)
```

Arguments

model	A BebelModel object.
message	User message.
greedy	Use deterministic greedy decoding.
on_event	Event handler function, named list of event-specific handlers, or NULL. Event types are bebel_event_types(). Delta events contain delta, id, and index; final events contain accumulated content or text.
check_interrupt	Cancel the underlying async job when the R wait is interrupted.
max_gen, max_context, max_think	Optional generation limits.
temperature, top_k, repeat_penalty	Optional sampling settings.
poll_interval	Seconds to sleep between async-job polls.

Value

A classed generation result.

bebel_chat_async *Start a background ChatML assistant reply job*

Description

Start a background ChatML assistant reply job

Usage

```
bebel_chat_async(
  model,
  message,
  greedy = FALSE,
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL
)
```

Arguments

`model`	A `BebelModel` object.
`message`	User message.
`greedy`	Use deterministic greedy decoding.
`max_gen`, `max_context`, `max_think`	Optional generation limits.
`temperature`, `top_k`, `repeat_penalty`	Optional sampling settings.

Value

A `BebelAsyncJob`.

bebel_clear *Clear a BebeLM agent transcript and caches*

Description

Clears the conversation state while keeping the loaded model weights and the agent's generation configuration.

Usage

```
bebel_clear(agent)
```

Arguments

agent	A BebelAgent object.

Value

Updated agent info.

---

bebel_detokenize *Decode BebeLM token ids*

Description

Decode BebeLM token ids

Usage

```
bebel_detokenize(model, ids)
```

Arguments

model	A BebelModel object.
ids	Integer token ids.

Value

Decoded text.

---

bebel_event_handler *Build a BebeLM generation event handler*

Description

bebel_event_handler() creates a single on_event handler function from handlers for individual event types. Current event types are returned by bebel_event_types().

Usage

```
bebel_event_handler(
  start = NULL,
  thinking_start = NULL,
  thinking_delta = NULL,
  thinking_end = NULL,
  text_start = NULL,
  text_delta = NULL,
  text_end = NULL,
  tool_list_start = NULL,
  tool_list_delta = NULL,
  tool_list_end = NULL,
  tool_call_start = NULL,
  tool_call_delta = NULL,
  tool_call_end = NULL,
  done = NULL,
  default = NULL
)
```

Arguments

`start`, `thinking_start`, `thinking_delta`, `thinking_end`, `text_start`, `text_delta`, `text_end`	Optional functions called for the corresponding stream event.
`tool_list_start`, `tool_list_delta`, `tool_list_end`	Optional handlers for BebeLM tool-list delimiter blocks.
`tool_call_start`, `tool_call_delta`, `tool_call_end`	Optional handlers for BebeLM tool-call delimiter blocks.
`done`	Function called for the final done event, or `NULL`.
`default`	Function called for events without a type-specific handler, or `NULL`.

Value

A function accepting one generation event list.

---

bebel_event_types *Return BebeLM stream event types.*

Description

Return BebeLM stream event types.

Usage

```
bebel_event_types()
```

`bebel_generate` *Generate a raw continuation from a prompt*

Description

Generate a raw continuation from a prompt

Usage

```
bebel_generate(
  model,
  prompt,
  greedy = FALSE,
  on_event = NULL,
  check_interrupt = TRUE,
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL,
  poll_interval = 0.005
)
```

Arguments

`model`	A `BebelModel` object.
`prompt`	Prompt text.
`greedy`	Use deterministic greedy decoding.
`on_event`	Event handler function, named list of event-specific handlers, or `NULL`. Event types are `bebel_event_types()`. Delta events contain `delta`, `id`, and `index`; final events contain accumulated `content` or `text`.
`check_interrupt`	Cancel the underlying async job when the R wait is interrupted.
`max_gen, max_context, max_think`	Optional generation limits.
`temperature, top_k, repeat_penalty`	Optional sampling settings.
`poll_interval`	Seconds to sleep between async-job polls.

Value

A classed list with generated text, token ids, stop reason, and timing statistics.

bebel_generate_async *Start a background raw generation job*

Description

Async jobs run BebeLM generation on Rust worker threads and reuse the loaded model weights. They are polled with `bebel_async_poll()` and collected with `bebel_async_collect()`.

Usage

```
bebel_generate_async(
  model,
  prompt,
  greedy = FALSE,
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL
)
```

Arguments

`model`	A `BebelModel` object.
`prompt`	Prompt text.
`greedy`	Use deterministic greedy decoding.
`max_gen`, `max_context`, `max_think`	Optional generation limits.
`temperature`, `top_k`, `repeat_penalty`	Optional sampling settings.

Value

A `BebelAsyncJob`.

bebel_history *Return a BebeLM agent token transcript*

Description

Return a BebeLM agent token transcript

Usage

```
bebel_history(agent)
```

Arguments

agent	A BebelAgent object.

Value

Integer token ids.

---

bebel_model_load *Load a BebeLM GGUF model*

Description

Load a BebeLM GGUF model

Usage

```
bebel_model_load(path, num_threads = NULL)
```

Arguments

path	Path to the GGUF weights file.
num_threads	Optional Rayon global thread-pool size. This can only be set once per R process.

Value

A BebelModel object.

---

bebel_parse_tool_call *Parse a single BebeLM tool call block*

Description

Parse a single BebeLM tool call block

Usage

```
bebel_parse_tool_call(content)
```

Arguments

content	Accumulated content between BebeLM tool-call delimiters.

Value

A list with name, arguments, and raw.

`bebel_parse_tool_calls` *Parse BebeLM tool calls*

Description

Delegates Pythonic BebeLM tool-call parsing (`[name(arg='value')]`) to upstream BebeLM. JSON call objects and legacy `name({...})` calls are parsed with imported package `yyjsonr`.

Usage

```
bebel_parse_tool_calls(content)
```

Arguments

`content`	Accumulated content between BebeLM tool-call delimiters.

Value

A list of calls, each with `name`, `arguments`, and `raw`.

`bebel_token_ids` *Return BebeLM tokenizer special token ids.*

Description

Return BebeLM tokenizer special token ids.

Usage

```
bebel_token_ids()
```

bebel_tokenize *Tokenize text with a BebeLM model tokenizer*

Description

Tokenize text with a BebeLM model tokenizer

Usage

```
bebel_tokenize(model, text, add_bos = TRUE)
```

Arguments

model	A BebelModel object.
text	Text to encode.
add_bos	Whether to prepend the BOS token.

Value

Integer token ids.

bebel_tool *Define a BebeLM R tool*

Description

Define a BebeLM R tool

Usage

```
bebel_tool(name, fun, description = NULL, schema = NULL)
```

Arguments

name	Tool name exposed to the tool dispatcher.
fun	Function to run. It is called as fun(args = ..., context = ..., call = ...) when it accepts those names, otherwise with progressively simpler fallbacks.
description	Optional human-readable description.
schema	Optional JSON-schema-like list or JSON string.

Value

A BebelToolSpec object.

bebel_tool_schema_json *Render a BebeLM tool schema*

Description

Converts an R `bebel_tool()` declaration into BebeLM's JSON tool schema string for the system `List of tools: [...]` preamble.

Usage

```
bebel_tool_schema_json(tool)
```

Arguments

`tool`	A `BebelToolSpec` object created by `bebel_tool()`.

Value

A character scalar containing the rendered tool schema.

bebel_transcript *Decode a BebeLM agent transcript*

Description

Decode a BebeLM agent transcript

Usage

```
bebel_transcript(agent)
```

Arguments

`agent`	A `BebelAgent` object.

Value

Transcript text.

`BebelAgent` *Persistent BebeLM conversation agent with transcript and decode caches.*

Description

Persistent BebeLM conversation agent with transcript and decode caches.

Usage

```
BebelAgent
```

`BebelAgentConfigureOptions` *Agent configuration update*

Description

Agent configuration update

Usage

```
BebelAgentConfigureOptions(
  greedy = NULL,
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL
)
```

Arguments

`greedy`	Optional logical decoding-mode update.
`max_gen`, `max_context`, `max_think`	Optional generation-limit updates.
`temperature`, `top_k`, `repeat_penalty`	Optional sampling-setting updates.

BebelAgentOptions *Agent construction options*

Description

Agent construction options

Usage

```
BebelAgentOptions(
  greedy = logical(0),
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL
)
```

Arguments

`greedy`	Use deterministic greedy decoding.
`max_gen`, `max_context`, `max_think`	Optional generation limits.
`temperature`, `top_k`, `repeat_penalty`	Optional sampling settings.

BebelAgentRef *BebeLM agent reference*

Description

BebeLM agent reference

Usage

```
BebelAgentRef(value = list())
```

Arguments

`value`	A one-element list containing a `BebelAgent`.

BebelAgentRunOptions *Agent run options*

Description

Agent run options

Usage

```
BebelAgentRunOptions(max_steps = integer(0), check_interrupt = logical(0))
```

Arguments

max_steps	Maximum assistant/tool iterations.
check_interrupt	Check for R user interrupts during synchronous generation.

BebelAsyncEventDrainOptions *Async event drain options*

Description

Async event drain options

Usage

```
BebelAsyncEventDrainOptions(max = NULL)
```

Arguments

max	Optional non-negative whole-number event limit.

BebelAsyncJob *Background BebeLM generation job.*

Description

Background BebeLM generation job.

Usage

```
BebelAsyncJob
```

BebelAsyncJobRef *BebeLM async job reference*

Description

BebeLM async job reference

Usage

```
BebelAsyncJobRef(value = list())
```

Arguments

value	A one-element list containing a BebelAsyncJob.

BebelAsyncWaitOptions *Async wait options*

Description

Async wait options

Usage

```
BebelAsyncWaitOptions(
  poll_interval = integer(0),
  cancel_on_interrupt = logical(0)
)
```

Arguments

poll_interval	Seconds to sleep between polls while a job is pending.
cancel_on_interrupt	Whether an interrupted wait should request Rust-side job cancellation.

BebelGenerationBenchmarkOptions

Generation benchmark options

Description

Generation benchmark options

Usage

```
BebelGenerationBenchmarkOptions(
  prompts = character(0),
  concurrency = integer(0),
  repeats = integer(0),
  poll_interval = integer(0)
)
```

Arguments

prompts	Character vector of prompts.
concurrency	Maximum number of async jobs in flight.
repeats	Number of times to repeat the prompt set.
poll_interval	Seconds to sleep between monitor polls.

BebelGenerationOptions

Generation options

Description

Generation options

Usage

```
BebelGenerationOptions(
  greedy = logical(0),
  check_interrupt = logical(0),
  max_gen = NULL,
  max_context = NULL,
  max_think = NULL,
  temperature = NULL,
  top_k = NULL,
  repeat_penalty = NULL
)
```

Arguments

`greedy`	Use deterministic greedy decoding.
`check_interrupt`	Check for R user interrupts during synchronous generation.
`max_gen`, `max_context`, `max_think`	Optional generation limits.
`temperature`, `top_k`, `repeat_penalty`	Optional sampling settings.

`BebelModel` *Loaded BebeLM GGUF generation model.*

Description

Loaded BebeLM GGUF generation model.

Usage

```
BebelModel
```

`BebelModelLoadOptions` *Model loading options*

Description

Model loading options

Usage

```
BebelModelLoadOptions(path = character(0), num_threads = NULL)
```

Arguments

`path`	Path to a GGUF weights file.
`num_threads`	Optional positive whole-number Rayon thread count.

BebelModelRef *BebeLM model reference*

Description

BebeLM model reference

Usage

```
BebelModelRef(value = list())
```

Arguments

value	A one-element list containing a BebelModel.

BebelScalarText *Scalar non-empty text*

Description

Scalar non-empty text

Usage

```
BebelScalarText(value = character(0))
```

Arguments

value	A non-empty character scalar.

BebelTokenizeOptions *Tokenizer options*

Description

Tokenizer options

Usage

```
BebelTokenizeOptions(add_bos = logical(0))
```

Arguments

add_bos	Whether to prepend the model's beginning-of-sequence token.

BebelToolRef *BebeLM tool reference*

Description

BebeLM tool reference

Usage

```
BebelToolRef(value = list())
```

Arguments

value	A one-element list containing a BebelToolSpec.

BebelToolSpec *R tool exposed to BebeLM*

Description

R tool exposed to BebeLM

Usage

```
BebelToolSpec(
  name = character(0),
  fun = NULL,
  description = NULL,
  schema = NULL
)
```

Arguments

name	Tool name exposed to the model and dispatcher.
fun	R function called for matching tool calls.
description	Optional tool description.
schema	Optional JSON-schema-like list or JSON string.

colbert_embedding_ids *Return model-input ids aligned to ColBERT token vectors*

Description

Return model-input ids aligned to ColBERT token vectors

Usage

```
colbert_embedding_ids(embeddings)
```

Arguments

embeddings	A ColbertEmbeddings object.

Value

Integer token ids, one per vector row.

colbert_embedding_vectors *Materialize ColBERT token vectors*

Description

Materialize ColBERT token vectors

Usage

```
colbert_embedding_vectors(embeddings)
```

Arguments

embeddings	A ColbertEmbeddings object.

Value

An n_tokens × 128 numeric matrix whose rows have unit L2 norm.

`colbert_embeddings_info` *Inspect ColBERT token vectors*

Description

Inspect ColBERT token vectors

Usage

```
colbert_embeddings_info(embeddings)
```

Arguments

`embeddings`	A `ColbertEmbeddings` object.

Value

A named list containing role, token count, and dimensions.

`colbert_encode_document` *Encode a retrieval document with ColBERT*

Description

Applies the profile's required `[D]` prefix, encodes up to 512 positions, projects them to L2-normalized token vectors, then removes only the profile's published punctuation skip-list from the scoring vectors.

Usage

```
colbert_encode_document(model, text)
```

Arguments

`model`	A `ColbertModel` object.
`text`	A non-empty document string.

Value

A document `ColbertEmbeddings` object.

colbert_encode_query *Encode a retrieval query with ColBERT*

Description

Applies the profile's required `[Q]` prefix, encodes exactly 32 query positions (including learned PAD expansion positions), projects them to L2-normalized 128-dimensional token vectors, and returns a `ColbertEmbeddings` handle.

Usage

```
colbert_encode_query(model, text)
```

Arguments

`model`	A `ColbertModel` object.
`text`	A non-empty query string.

Value

A query `ColbertEmbeddings` object.

colbert_maxsim *Score a query and document with ColBERT MaxSim*

Description

Computes `sum_i max_j(q_i dot d_j)`, where query and document token vectors were generated by `colbert_encode_query()` and `colbert_encode_document()`.

Usage

```
colbert_maxsim(query, document)
```

Arguments

`query`	Query `ColbertEmbeddings`.
`document`	Document `ColbertEmbeddings`.

Value

A numeric MaxSim relevance score.

colbert_model_info *Inspect a ColBERT model*

Description

Inspect a ColBERT model

Usage

```
colbert_model_info(model)
```

Arguments

model	A ColbertModel object.

Value

A named list describing its token-vector and MaxSim contract.

colbert_model_load *Load a native LFM2.5-ColBERT GGUF model*

Description

Loads LiquidAI's retrieval-trained late-interaction encoder. This profile is distinct from BebelModel: it uses bidirectional attention, learned token-level projections, separate query/document formatting, and ColBERT MaxSim rather than pooling states from a generation model.

Usage

```
colbert_model_load(path, num_threads = NULL)
```

Arguments

path	Path to an LFM2.5-ColBERT-350M GGUF file. The supported reference artifact is LFM2.5-ColBERT-350M-Q4_K_M.gguf from LiquidAI/LFM2.5-ColBERT-350M-GGUF.
num_threads	Optional Rayon global thread-pool size. This can only be set once per R process.

Value

A ColbertModel object.

colbert_rank *Rank documents with ColBERT MaxSim*

Description

This convenience helper performs exact in-memory scoring. It is suitable for a candidate set; production-scale retrieval needs an external late- interaction index built with the same profile and preprocessing contract.

Usage

```
colbert_rank(model, query, documents)
```

Arguments

model	A ColbertModel object.
query	A non-empty query string.
documents	A non-empty character vector of candidate documents.

Value

A decreasing named numeric vector of MaxSim scores with class colbertRanking.

ColbertEmbeddings *One query or document's L2-normalized ColBERT token vectors.*

Description

One query or document's L2-normalized ColBERT token vectors.

Usage

```
ColbertEmbeddings
```

ColbertEmbeddingsRef *ColBERT token-vector reference*

Description

ColBERT token-vector reference

Usage

```
ColbertEmbeddingsRef(value = list())
```

Arguments

value	A one-element list containing ColbertEmbeddings.

ColbertModel *Loaded LFM2.5-ColBERT GGUF model.*

Description

Loaded LFM2.5-ColBERT GGUF model.

Usage

```
ColbertModel
```

ColbertModelLoadOptions *ColBERT loading options*

Description

ColBERT loading options

Usage

```
ColbertModelLoadOptions(path = character(0), num_threads = NULL)
```

Arguments

path	Path to an LFM2.5-ColBERT GGUF weights file.
num_threads	Optional positive whole-number Rayon thread count.

ColbertModelRef *ColBERT model reference*

Description

ColBERT model reference

Usage

```
ColbertModelRef(value = list())
```

Arguments

value	A one-element list containing a ColbertModel.

embeddinggemma_embed *Generate EmbeddingGemma text embeddings*

Description

Runs the retrieval-trained bidirectional Gemma encoder, attention-mask mean pooling, and both learned dense projection layers. The requested task is deliberately mandatory: EmbeddingGemma was trained with different prompt contracts for queries, documents, semantic similarity, and other tasks.

Usage

```
embeddinggemma_embed(
  model,
  text,
  task,
  title = NULL,
  dimensions = 768L,
  normalize = TRUE,
  truncate = TRUE,
  check_interrupt = TRUE
)
```

Arguments

model	An EmbeddingGemmaModel object.
text	Character vector to embed.
task	One of "retrieval_query", "retrieval_document", "question_answering", "fact_verification", "classification", "clustering", "semantic_similarity", "code_retrieval", "summarization", or "raw". "raw" adds no task prompt and should only be used with already formatted model input.

`title`	Optional document-title scalar or vector. It is valid only for `task = "retrieval_document"`; `NULL` uses the model's `"none"` title.
`dimensions`	Output dimension: 768, 512, 256, or 128.
`normalize`	L2-normalize each output row. Keep this `TRUE` for cosine similarity and Matryoshka embeddings.
`truncate`	Truncate overlong inputs to the model's 2048-token context, preserving BOS and EOS. If `FALSE`, overlong inputs fail.
`check_interrupt`	Poll for R user interrupts during tokenization and between bounded packed inference batches.

Details

For retrieval, prefer `embeddinggemma_embed_query()` and `embeddinggemma_embed_document()` so query and document prompts cannot be confused. Matryoshka dimensions 512, 256, and 128 select the leading dimensions of the 768-vector and then re-normalize, as specified by the model card.

Value

An `embeddingGemmaEmbeddings` numeric matrix with one row per input. Its `embedding_info` attribute records prompts, token counts, truncation, dimensions, and normalization.

`embeddinggemma_embed_document` *Encode retrieval documents with EmbeddingGemma*

Description

Encode retrieval documents with EmbeddingGemma

Usage

```
embeddinggemma_embed_document(
  model,
  text,
  title = NULL,
  dimensions = 768L,
  normalize = TRUE,
  truncate = TRUE,
  check_interrupt = TRUE
)
```

Arguments

model	An EmbeddingGemmaModel object.
text	Character vector to embed.
title	Optional document-title scalar or vector. It is valid only for task = "retrieval_document"; NULL uses the model's "none" title.
dimensions	Output dimension: 768, 512, 256, or 128.
normalize	L2-normalize each output row. Keep this TRUE for cosine similarity and Matryoshka embeddings.
truncate	Truncate overlong inputs to the model's 2048-token context, preserving BOS and EOS. If FALSE, overlong inputs fail.
check_interrupt	Poll for R user interrupts during tokenization and between bounded packed inference batches.

Value

An embeddingGemmaEmbeddings numeric matrix.

---

embeddinggemma_embed_query *Encode retrieval queries with EmbeddingGemma*

Description

Encode retrieval queries with EmbeddingGemma

Usage

```
embeddinggemma_embed_query(
  model,
  text,
  dimensions = 768L,
  normalize = TRUE,
  truncate = TRUE,
  check_interrupt = TRUE
)
```

Arguments

model	An EmbeddingGemmaModel object.
text	Character vector to embed.
dimensions	Output dimension: 768, 512, 256, or 128.
normalize	L2-normalize each output row. Keep this TRUE for cosine similarity and Matryoshka embeddings.

truncate Truncate overlong inputs to the model's 2048-token context, preserving BOS and EOS. If FALSE, overlong inputs fail.

check_interrupt
Poll for R user interrupts during tokenization and between bounded packed inference batches.

Value

An embeddingGemmaEmbeddings numeric matrix.

---

embeddinggemma_model_info
Inspect an EmbeddingGemma model

Description

Inspect an EmbeddingGemma model

Usage

```
embeddinggemma_model_info(model)
```

Arguments

model An EmbeddingGemmaModel object.

Value

A named list describing the architecture and supported dimensions.

---

embeddinggemma_model_load
Load an EmbeddingGemma GGUF model

Description

Loads the dedicated, retrieval-trained gemma-embedding architecture through the package's pure-Rust CPU backend. The implementation does not link to 'llama.cpp', 'PyTorch', the 'ONNX Runtime', or the 'SentencePiece' C++ library. Model weights remain subject to the Gemma Terms of Use.

Usage

```
embeddinggemma_model_load(path, num_threads = NULL)
```

Arguments

path	Path to an EmbeddingGemma GGUF file. The supported reference artifact is embeddinggemma-300M-Q8_0.gguf from ggml-org/embeddinggemma-300M-GGUF.
num_threads	Optional Rayon global thread-pool size. This can only be set once per R process.

Value

An EmbeddingGemmaModel object.

---

embeddinggemma_tokenize

Tokenize EmbeddingGemma model input

Description

Applies the same mandatory task formatting as embeddinggemma_embed() and returns the exact BOS/content/EOS token sequence consumed by the encoder.

Usage

```
embeddinggemma_tokenize(model, text, task, title = NULL, truncate = TRUE)
```

Arguments

model	An EmbeddingGemmaModel object.
text	Character vector to embed.
task	One of "retrieval_query", "retrieval_document", "question_answering", "fact_verification", "classification", "clustering", "semantic_similarity", "code_retrieval", "summarization", or "raw". "raw" adds no task prompt and should only be used with already formatted model input.
title	Optional document-title scalar or vector. It is valid only for task = "retrieval_document"; NULL uses the model's "none" title.
truncate	Truncate overlong inputs to the model's 2048-token context, preserving BOS and EOS. If FALSE, overlong inputs fail.

Value

A named list containing integer ids, legible token pieces, the task-formatted text, and a truncation flag.

`EmbeddingGemmaLoadOptions` *EmbeddingGemma loading options*

Description

EmbeddingGemma loading options

Usage

```
EmbeddingGemmaLoadOptions(path = character(0), num_threads = NULL)
```

Arguments

`path`	Path to a `gemma-embedding` GGUF weights file.
`num_threads`	Optional positive whole-number Rayon thread count.

`EmbeddingGemmaModel` *Loaded EmbeddingGemma GGUF model.*

Description

Loaded EmbeddingGemma GGUF model.

Usage

```
EmbeddingGemmaModel
```

`EmbeddingGemmaModelRef` *EmbeddingGemma model reference*

Description

EmbeddingGemma model reference

Usage

```
EmbeddingGemmaModelRef(value = list())
```

Arguments

`value`	A one-element list containing an `EmbeddingGemmaModel`.

EmbeddingGemmaOptions *EmbeddingGemma encoding options*

Description

EmbeddingGemma encoding options

Usage

```
EmbeddingGemmaOptions(
  text = character(0),
  task = character(0),
  title = NULL,
  dimensions = integer(0),
  normalize = logical(0),
  truncate = logical(0),
  check_interrupt = logical(0)
)
```

Arguments

text	Character vector to encode.
task	Embedding task controlling the required model prompt.
title	Optional document title, used only for "retrieval_document".
dimensions	Matryoshka output size: 768, 512, 256, or 128.
normalize	Whether to L2-normalize each embedding.
truncate	Whether to truncate inputs exceeding the 2048-token context.
check_interrupt	Whether to poll R interrupts during tokenization and between bounded packed inference batches.

print.bebelGeneration *Print a BebeLM generation result*

Description

Print a BebeLM generation result

Usage

```
## S3 method for class 'bebelGeneration'
print(x, ...)
```

Arguments

x	A result returned by bebel_generate(), bebel_chat(), or bebel_async_collect().
...	Unused.

Value

Invisibly returns x.

---

rbebelm_backend_features *Return feature information reported by the loaded Rust backend.*

Description

Return feature information reported by the loaded Rust backend.

Usage

```
rbebelm_backend_features()
```

---

rbebelm_backend_info *Inspect Rbebelm backend dispatch state*

Description

Inspect Rbebelm backend dispatch state

Usage

```
rbebelm_backend_info()
```

Value

A named list describing installed, supported, requested, and selected backends, with class rbebelmBackendInfo.

rbebelm_cpuid_info *Inspect CPU SIMD support used by backend dispatch*

Description

Inspect CPU SIMD support used by backend dispatch

Usage

```
rbebelm_cpuid_info()
```

Value

A named list of logical CPU feature checks with class `rbebelmCpuidInfo`.

rbebelm_set_backend *Select the Rbebelm native backend*

Description

Must be called before loading a model or querying backend features.

Usage

```
rbebelm_set_backend(backend = "auto")
```

Arguments

backend	One of `"auto"`, `"scalar"`, `"avx2"`, `"avx512"`, `"neon"`, `"dotprod"`, or `"wasm_simd128"`.

Value

The requested backend name.

Index