

Package: Rlimite (via r-universe)

September 22, 2026

Title Native 'torch' Inference for Limite Mathematical Language Models

Version 0.1.0

Description Implements the Limite mathematical language-model architecture <<https://github.com/paradigma-inc/limite-violetto>> directly with 'torch'. Loads the published Base, Base Soup, and Violetto 'safetensors' checkpoints and tokenizers from the 'Hugging Face' Hub without a Python runtime or model-serving process. Provides cached autoregressive generation and the official single-turn prompt format used by the math-specialized Violetto checkpoint.

License GPL (>= 2)

URL <https://github.com/souunkou-bioinfo/Rlimite>,
<https://souunkou-bioinfo.github.io/Rlimite>

BugReports <https://github.com/souunkou-bioinfo/Rlimite/issues>

Depends R (>= 4.1)

Imports cli, hfhub (>= 0.1.2), jsonlite, safetensors (>= 0.2.1), tok (>= 0.2.3), torch (>= 0.17.0)

Suggests knitr, pkgload, rmarkdown, tinytest

Config/roxygen2/version 8.0.0

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/pak/sysreqs cmake make libuv1-dev libssl-dev libclang-dev

Repository <https://souunkou-bioinfo.r-universe.dev>

Date/Publication 2026-09-22 17:47:57 UTC

RemoteUrl <https://github.com/souunkou-bioinfo/Rlimite>

RemoteRef HEAD

RemoteSha dabee304050bc6bd1d6f2c9b02f8dca6e7e9576c

Contents

limite	2
limite_chat	2
limite_chat_prompt	3
limite_checkpoints	4
limite_config	4
limite_from_config	5
limite_from_pretrained	5
limite_generate	6
limite_state_dict	7
limite_tokenizer	7
Index	9

limite	<i>Construct a Limite language model</i>
--------	--

Description

Construct a Limite language model

Usage

```
limite(config = limite_config())
```

Arguments

config A configuration produced by `limite_config()` or loaded from a checkpoint.

Value

A `torch::nn_module` implementing `LimiteForCausalLM`.

limite_chat	<i>Generate a Limite response</i>
-------------	-----------------------------------

Description

Violetto is intended for direct, independent mathematical problems rather than general multi-turn assistant use.

Usage

```
limite_chat(model, messages, tokenizer = limite_tokenizer(), ...)
```

Arguments

<code>model</code>	A pretrained Limite model.
<code>messages</code>	Conversation messages accepted by <code>limite_chat_prompt()</code> .
<code>tokenizer</code>	A tokenizer returned by <code>limite_tokenizer()</code> .
<code>...</code>	Arguments passed to <code>limite_generate()</code> .

Value

A scalar response string with generated token IDs attached as the `token_ids` attribute.

<code>limite_chat_prompt</code>	<i>Format messages with the published Limite template</i>
---------------------------------	---

Description

Adds the Limite end-of-text and ChatML control tokens. A default system message is inserted when the first message is not a system message.

Usage

```
limite_chat_prompt(messages, add_generation_prompt = TRUE)
```

Arguments

<code>messages</code>	A list of messages. Each message contains scalar role and content fields. Supported roles are system, user, and assistant.
<code>add_generation_prompt</code>	Append the assistant prefix.

Value

A scalar prompt string.

limite_checkpoints	<i>Published Limite checkpoints</i>
--------------------	-------------------------------------

Description

Returns the model identifiers and immutable revisions covered by Rlimite's real-model tests. Violetto is the math-specialized single-turn solver; Base and Base Soup are published family checkpoints.

Usage

```
limite_checkpoints()
```

Value

A data frame with checkpoint labels, model identifiers, revisions, and profiles.

limite_config	<i>Limite model configuration</i>
---------------	-----------------------------------

Description

Creates the architecture contract used by the published 1B checkpoints. Values can be overridden to construct smaller models for experiments.

Usage

```
limite_config(...)
```

Arguments

... Named configuration overrides.

Value

A validated Limite configuration list.

limite_from_config	<i>Construct Limite from checkpoint configuration</i>
--------------------	---

Description

Construct Limite from checkpoint configuration

Usage

```
limite_from_config(identifier, revision = "main", local_files_only = FALSE)
```

Arguments

identifier	Model identifier or local path.
revision	Hugging Face revision.
local_files_only	If TRUE, do not download missing Hub files.

Value

An uninitialized Limite model with the requested architecture.

limite_from_pretrained	<i>Load a pretrained Limite model</i>
------------------------	---------------------------------------

Description

The model is created on the meta device and then attached to the checkpoint tensors, avoiding a second initialized copy of the 1B parameters.

Usage

```
limite_from_pretrained(
  identifier = "paradigma-inc/limite-1b-violetto",
  revision = "main",
  local_files_only = FALSE,
  device = NULL
)
```

Arguments

identifier	Model identifier or local path.
revision	Hugging Face revision.
local_files_only	If TRUE, do not download missing Hub files.
device	Optional torch device such as "cpu" or "cuda".

Value

A pretrained Limite model in evaluation mode.

limite_generate	<i>Generate tokens with a Limite model</i>
-----------------	--

Description

Public token IDs follow the zero-based Hugging Face convention. The model uses a key/value cache after the prompt prefill.

Usage

```
limite_generate(
    model,
    input_ids,
    max_new_tokens = 64L,
    temperature = 0.8,
    top_k = 50L,
    eos_token_id = model$config$eos_token_id,
    seed = NULL,
    include_prompt = FALSE,
    callback = NULL
)
```

Arguments

model	A model returned by limite_from_pretrained() or limite() .
input_ids	Zero-based token IDs for one prompt.
max_new_tokens	Maximum number of tokens to generate.
temperature	Sampling temperature. Use zero for greedy decoding.
top_k	Number of highest-scoring tokens retained for sampling, or NULL for the complete vocabulary.
eos_token_id	Zero-based end token. Defaults to the model config.
seed	Optional torch random seed.
include_prompt	Include the prompt IDs in the returned vector.
callback	Optional function called with each generated token ID.

Value

An integer vector of zero-based token IDs.

limite_state_dict	<i>Load a Limite checkpoint state dictionary</i>
-------------------	--

Description

Loads safetensors directly as R torch tensors. `identifier` may be a Hugging Face model identifier, a checkpoint directory, or one safetensors file.

Usage

```
limite_state_dict(identifier, revision = "main", local_files_only = FALSE)
```

Arguments

<code>identifier</code>	Model identifier or local path.
<code>revision</code>	Hugging Face revision.
<code>local_files_only</code>	If TRUE, do not download missing Hub files.

Value

A named list of torch tensors.

limite_tokenizer	<i>Load the Limite tokenizer</i>
------------------	----------------------------------

Description

Load the Limite tokenizer

Usage

```
limite_tokenizer(
  identifier = "paradigma-inc/limite-1b-violetto",
  revision = "main",
  local_files_only = FALSE
)
```

Arguments

<code>identifier</code>	Model identifier or local path.
<code>revision</code>	Hugging Face revision.
<code>local_files_only</code>	If TRUE, do not download missing Hub files.

8

limite_tokenizer

Value

A tok_tokenizer.

Index

`limite`, [2](#)
`limite()`, [6](#)
`limite_chat`, [2](#)
`limite_chat_prompt`, [3](#)
`limite_chat_prompt()`, [3](#)
`limite_checkpoints`, [4](#)
`limite_config`, [4](#)
`limite_config()`, [2](#)
`limite_from_config`, [5](#)
`limite_from_pretrained`, [5](#)
`limite_from_pretrained()`, [6](#)
`limite_generate`, [6](#)
`limite_generate()`, [3](#)
`limite_state_dict`, [7](#)
`limite_tokenizer`, [7](#)
`limite_tokenizer()`, [3](#)