Package 'easieRnmt'

July 16, 2026

Title Local Machine-Translation with 'EasyNMT' from 'R'

Version 0.0.5

Description 'R' wrapper around the 'Python' library 'EasyNMT', enabling easy-to-use, local, reproducible machine translation. Supports translation between multiple language pairs. The package takes care of preprocessing and language detection using 'fastText' from 'R'. 'Python' backend and installation of 'PyTorch' is handled by 'reticulate' and 'uv'.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 8.0.0

URL https://github.com/thieled/easieRnmt

BugReports https://github.com/thieled/easieRnmt/issues

Imports cli, data.table, emoji, fastText, pbapply, reticulate, stringi, stringr, textclean, tokenizers

Suggests testthat

NeedsCompilation no

Author Daniel Thiele [aut, cre] (ORCID: <https://orcid.org/0000-0003-0324-4943>)

Maintainer Daniel Thiele <daniel.thiele@fu-berlin.de>

Repository CRAN

Date/Publication 2026-07-16 14:10:21 UTC

Contents

check_backend 2
clean_text 2
detect_languages 4
initialize_easynmt 5
preprocess 6
replace_emoji_with_name 7
translate 8

Index **10**

check_backend *Check easynmt Backend Configuration*

Description

Displays diagnostic information about the easynmt Python backend setup.

Usage

```
check_backend()
```

Value

Invisibly returns a list with backend configuration details.

Examples

```
## Not run:
  easynmt::check_backend()

## End(Not run)
```

clean_text *Clean, Normalize, and Tokenize Text*

Description

Cleans and optionally tokenizes raw text data. Handles emoji replacement, removal of unsupported characters, normalization, sentence tokenization, and chunking into word-limited segments. Returns either a standardized data.table or a character vector.

Usage

```
clean_text(
  x,
  text_col = "text",
  id_col = NULL,
  lang_guess_col = NULL,
  replace_emojis = TRUE,
  replace_alphaless = TRUE,
  max_char = 5000,
  tokenize_sentences = TRUE,
  max_words = 50,
  clean_characters = TRUE,
  return_string = FALSE,
  verbose = TRUE
)
```

Arguments

x	Character vector or data.frame containing texts to clean.
text_col	Character, name of the text column if x is a data.frame. Ignored if x is a character vector. Default = "text".
id_col	Character, optional column name in the input data.frame to preserve as id. Default = NULL.
lang_guess_col	Character, optional column name in the input data.frame to preserve as lang_guess. Default = NULL.
replace_emojis	Logical, whether to replace emojis with placeholder names. Default = TRUE.
replace_alphaless	Logical, whether to replace strings that contain no alphabetic characters with empty strings. Default = TRUE.
max_char	Integer, maximum number of characters per text. Texts longer than this are truncated. Default = 5000.
tokenize_sentences	Logical, whether to split texts into sentences and chunks. Default = TRUE.
max_words	Integer, maximum number of words per sentence or chunk. Long sentences are split into chunks of this size. Default = 50.
clean_characters	Logical, whether to apply character normalization (regex replacements, squishing, punctuation handling). Default = TRUE.
return_string	Logical, if TRUE, return only the cleaned character vector (text_clean) instead of a full data.table. Default = FALSE.
verbose	Logical, whether to print progress messages. Default = TRUE.

Value

Either:

- A data.table with columns:
  - sen_id: Unique identifier for each sentence or chunk.
  - doc_idx: Row index of the original document in the input.
  - sen_idx: Sentence/chunk index within each document.
  - id: Optional user-provided identifier.
  - text_orig: Original text.
  - text_clean: Cleaned and normalized text.
  - lang_guess: Optional language guess column, if present.
- Or a character vector of cleaned texts if return_string = TRUE.

`detect_languages` *Detect Languages with fastText*

Description

Identifies the language of each text in a character vector or data.frame using fastText's pretrained language identification model. Calls `clean_text()` internally to normalize and optionally tokenize the text before prediction.

Usage

```
detect_languages(
  x,
  text_col = "text",
  id_col = NULL,
  lang_guess_col = NULL,
  model_path = NULL,
  prob_threshold = 0.25,
  und_label = "und",
  max_char = 5000,
  tokenize_sentences = FALSE,
  threads = parallel::detectCores(),
  verbose = TRUE,
  max_words = 50,
  ...
)
```

Arguments

`x`	Character vector or data.frame containing texts to clean.
`text_col`	Character, name of the text column if `x` is a data.frame. Ignored if `x` is a character vector. Default = "text".
`id_col`	Character, optional column name in the input data.frame to preserve as `id`. Default = NULL.
`lang_guess_col`	Character, optional column name in the input data.frame to preserve as `lang_guess`. Default = NULL.
`model_path`	Character, path to the fastText `lid.176.bin` model. Defaults to `~/.cache/easieRnmt/lid.176.bin`.
`prob_threshold`	Numeric, threshold below which the detected language is replaced by `und_label` (or by the user-provided `lang_guess`). Default = 0.25.
`und_label`	Character, label for undefined language if confidence is below `prob_threshold` and no `lang_guess` column is provided. Default = "und".
`max_char`	Integer, maximum number of characters per text. Texts longer than this are truncated. Default = 5000.
`tokenize_sentences`	Logical, whether to split texts into sentences and chunks. Default = TRUE.

threads	Integer, number of threads for fastText. Default = parallel::detectCores().
verbose	Logical, whether to print progress messages. Default = TRUE.
max_words	Integer, maximum number of words per sentence or chunk. Long sentences are split into chunks of this size. Default = 50.
...	Additional parameters passed on to clean_text().

Value

A data.table with standardized columns: sen_id, doc_idx, sen_idx, optional id, text_orig, text_clean, optional lang_guess, lang, and lang_prob.

---

initialize_easynmt *Initialize the easynmt Python environment*

Description

Declares all required Python packages via py_require, which provisions them using uv into an ephemeral virtual environment. PyTorch is installed with UV_TORCH_BACKEND=auto, which lets uv auto-detect NVIDIA (CUDA), AMD (ROCm), and Intel GPUs and pick the correct wheels — no manual driver querying needed. Pass gpu = FALSE to force CPU-only onnxruntime.

The function is called automatically at the start of each easynmt function and is a no-op after the first successful call within a session.

Usage

```
initialize_easynmt(gpu = check_gpu(), uv_cache_dir = NULL, models_dir = NULL)
```

Arguments

gpu	Logical. Whether to install onnxruntime-gpu (CUDA only). Defaults to auto-detection via check_gpu().
uv_cache_dir	Character (optional). Directory used by uv to install python libraries.
models_dir	Character (optional). Directory used to cache huggingface models.

Details

Non-torch packages are declared with py_require() so that reticulate / uv can resolve them together. PyTorch is handled by setting UV_TORCH_BACKEND=auto, which instructs uv to query for CUDA, ROCm, and Intel GPU drivers and select the matching PyTorch index automatically.

onnxruntime-gpu only supports CUDA, so the gpu parameter reflects NVIDIA presence only; AMD and Intel users get CPU onnxruntime while still getting GPU-accelerated PyTorch.

fasttext workaround: EasyNMT declares fasttext as a hard Requires-Dist dependency (not optional). The canonical fasttext package has no prebuilt wheels for Windows and fails to compile from C++ source on MSVC. The workaround uses UV_OVERRIDE with a direct-URL specifier

that points `fasttext` at the `fasttext-predict` wheel — a prediction-only fork that ships prebuilt binary wheels for all platforms and Python 3.9–3.13, and exposes the same `import fasttext` namespace. The override file contains `fasttext @ https://...` with the PyPI URL for the matching wheel, selected at runtime by platform and Python version. Language detection is handled in R, so the prediction-only scope of `fasttext-predict` is not a limitation.

Value

Invisibly returns `TRUE`.

`preprocess` *Preprocess Text for Translation*

Description

Wrapper around `detect_languages()` that detects languages, reprocesses low-confidence cases, and splits the data into homogeneous language groups for translation. Tokenization can be performed before or after language detection.

Usage

```
preprocess(
  x,
  text_col = "text",
  id_col = NULL,
  lang_guess_col = NULL,
  targ_lang,
  model_path = NULL,
  prob_threshold = 0.25,
  und_label = "und",
  max_char = 5000,
  chunk_size = NULL,
  tokenize_sentences = TRUE,
  tokenize_when = "after",
  threads = parallel::detectCores(),
  verbose = TRUE,
  max_words = max_words,
  ...
)
```

Arguments

`x`	Character vector or data.frame containing texts to clean.
`text_col`	Character, name of the text column if `x` is a data.frame. Ignored if `x` is a character vector. Default = "text".
`id_col`	Character, optional column name in the input data.frame to preserve as `id`. Default = NULL.

lang_guess_col	Character, optional column name in the input data.frame to preserve as lang_guess. Default = NULL.
targ_lang	Character, fallback language code when detection is uncertain.
model_path	Character, path to the fastText lid.176.bin model. Defaults to ~/.cache/easieRnmt/lid.176.bin.
prob_threshold	Numeric, threshold below which the detected language is replaced by und_label (or by the user-provided lang_guess). Default = 0.25.
und_label	Character, label for undefined language if confidence is below prob_threshold and no lang_guess column is provided. Default = "und".
max_char	Integer, maximum number of characters per text. Texts longer than this are truncated. Default = 5000.
chunk_size	Integer, optional. If provided, split each homogeneous language data.table into chunks of at most chunk_size rows.
tokenize_sentences	Logical, whether to split texts into sentences and chunks. Default = TRUE.
tokenize_when	Character, determines when to tokenize: "before" or "after".
threads	Integer, number of threads for fastText. Default = parallel::detectCores().
verbose	Logical, whether to print progress messages. Default = TRUE.
max_words	Integer, maximum number of words per sentence or chunk. Long sentences are split into chunks of this size. Default = 50.
...	Additional parameters passed on to clean_text().

Value

A named list of data.tables, each containing texts of one homogeneous language or language+part chunk.

---

replace_emoji_with_name

Replace Emojis with Names

Description

Replaces all emojis in a character vector with their textual names (in :name: style).

Usage

```
replace_emoji_with_name(text_vec)
```

Arguments

text_vec	Character vector of texts.

Value

Character vector with emojis replaced by names.

translate *Preprocess and Translate Texts*

Description

Detects languages, reprocesses uncertain cases, optionally splits long texts into sentences, splits the input into homogeneous language groups, and calls the Python translator (`easynmt_translate()`) on each group.

Usage

```
translate(
  x,
  text_col = "text",
  id_col = NULL,
  lang_guess_col = NULL,
  targ_lang,
  model_path = NULL,
  prob_threshold = 0.25,
  und_label = "und",
  max_char = 5000,
  threads = parallel::detectCores(),
  target_lang,
  model = "opus-mt",
  seed = 42L,
  batch_size = 20L,
  max_length_tl = 512L,
  beam_size = 1L,
  deterministic = TRUE,
  verbose = TRUE,
  check_translation = FALSE,
  n_retries = 3L,
  check_threshold = 0.6,
  return_string = FALSE,
  save_dir = NULL,
  tokenize_sentences = TRUE,
  uv_cache_dir = NULL,
  models_dir = NULL,
  ...
)
```

Arguments

x	Character vector or data.frame containing texts to clean.
text_col	Character, name of the text column if x is a data.frame. Ignored if x is a character vector. Default = "text".

id_col	Character, optional column name in the input data.frame to preserve as id. Default = NULL.
lang_guess_col	Character, optional column name in the input data.frame to preserve as lang_guess. Default = NULL.
targ_lang	Character, fallback language code when detection is uncertain.
model_path	Character, path to the fastText lid.176.bin model. Defaults to ~/.cache/easieRnmt/lid.176.bin.
prob_threshold	Numeric, threshold below which the detected language is replaced by und_label (or by the user-provided lang_guess). Default = 0.25.
und_label	Character, label for undefined language if confidence is below prob_threshold and no lang_guess column is provided. Default = "und".
max_char	Integer, maximum number of characters per text. Texts longer than this are truncated. Default = 5000.
threads	Integer, number of threads for fastText. Default = parallel::detectCores().
target_lang	Character, target language for translation.
model	Character, translation model (default "opus-mt").
seed	Integer, random seed (default 42).
batch_size	Integer, batch size for translation (default 20L).
max_length_tl	Integer, maximum token length (default 512L).
beam_size	Integer, beam size for translation (default 1L).
deterministic	Logical, enforce deterministic cudnn ops (default TRUE).
verbose	Logical, whether to print progress messages. Default = TRUE.
check_translation	Logical, perform retry check (default FALSE).
n_retries	Integer, number of retries if check fails (default 3L).
check_threshold	Numeric, threshold for ratio of unique tokens (target / source) below which retries are attempted (default 0.6).
return_string	Logical, if TRUE, return only the translated character vector. Default = FALSE.
save_dir	Optional character path. If provided, saves each processed subset as an .rds file with its language/part name.
tokenize_sentences	Logical, if TRUE, split long texts into sentences during preprocessing and reassemble them after translation. Default = TRUE.
uv_cache_dir	Character (optional). Directory used by uv to install python libraries, passed on to initialize_easynmt.
models_dir	Character (optional). Directory used to cache huggingface models, passed on to initialize_easynmt.
...	Additional parameters passed on to clean_text() during preprocessing.

Value

A data.table with original data plus translation results merged in.

Index

check_backend, 2
clean_text, 2

detect_languages, 4

initialize_easynmt, 5

preprocess, 6
py_require, *5*

replace_emoji_with_name, 7

translate, 8