Package: babeldown (via r-universe)

October 7, 2026

Title Helpers for Automatic Translation of Markdown-based Content

Version 0.0.0.9000

Description Provide workflows and guidance for automatic translation of Markdown-based R content using DeepL API.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

URL https://docs.ropensci.org/babeldown, https://github.com/ropensci-review-tools/babeldown

BugReports https://github.com/ropensci-review-tools/babeldown/issues

Depends R (>= 4.1)

Imports babelquarto (>= 0.0.0.9000), brio, cli, digest, glue, httr2, lifecycle, memoise, purrr, readr, rlang, rstudioapi, snakecase, tibble, tinkr (>= 0.2.0.9001), withr, xml2, yaml (>= 2.3.8)

Suggests blogdown, cld3, clipr, fs, gert, vcr, knitr, quarto, rmarkdown, rprojroot, sys, testthat (>= 3.0.0)

Remotes ropensci-review-tools/babelquarto, ropensci/tinkr

Config/testthat/edition 3

Config/ropensci/maintainer staff

Config/roxygen2/version 8.1.0.9000

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libxml2-dev libxslt-dev libssl-dev libx11-dev

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

Date/Publication 2026-10-06 09:31:53 UTC

RemoteUrl https://github.com/ropensci-review-tools/babeldown

RemoteRef main

RemoteSha 901b5e24a07c7b58fe24f300d9b78005b7a46a33

Contents

check_xml_structure 2
deepl_languages 2
deepl_translate 3
deepl_translate_clipboard 4
deepl_translate_hugo 5
deepl_translate_markdown_string 7
deepl_translate_quarto 8
deepl_translate_vtt 9
deepl_update 10
deepl_upsert_glossary 13
deepl_usage 14

Index **15**

`check_xml_structure` *Check that two files have the same XML structure*

Description

Useful if `deepl_branch_update()` fails with an error about the XML structure. Checkout the default branch, check the two files.

Usage

```
check_xml_structure(path1, path2)
```

Arguments

`path1`	First path
`path2`	Second path

`deepl_languages` *Languages supported by DeepL API*

Description

Languages supported by DeepL API

Usage

```
deepl_languages(type = c("target", "source"))
```

Arguments

`type`	Either "target" or "source"

Value

A data.frame of languages (language code as "language", name as "name", whether formality is supported as "supports_formality").

Examples

```
## Not run:
deepl_languages(type = "source")
deepl_languages(type = "target")

## End(Not run)
```

`deepl_translate` *Translate a Markdown file*

Description

Translate a Markdown file

Usage

```
deepl_translate(
  path,
  out_path,
  yaml_fields = c("title", "description"),
  glossary_name = NULL,
  source_lang = NULL,
  target_lang = NULL,
  formality = c("default", "more", "less", "prefer_more", "prefer_less")
)
```

Arguments

`path`	Path to the Markdown file to be translated (character).
`out_path`	Path where the new translated file should be saved (character).
`yaml_fields`	Vector of character names of YAML fields to translate.
`glossary_name`	Name of the glossary to be used, if any (character).
`source_lang`	Name or code of source language. See DeepL docs.
`target_lang`	Name or code of source language. See DeepL docs.
`formality`	Formality level to use (character), one of • "default" (default) • "less" – for a more informal language • "prefer_more" – for a more formal language if available, otherwise fallback to default formality • "prefer_less" – for a more informal language if available, otherwise fallback to default formality

Value

None

Examples

```
## Not run:
english_lines <- c(
  "## A cool section", "",
  "This is the first paragraph. `system.file()` is cool, right?", "",
  "Another paragraph. I really enjoy developing R packages.", "",
  "Do you enjoy debugging?"
)
file <- withr::local_tempfile()
brio::write_lines(english_lines, file)
out_path <- withr::local_tempfile()
babeldown::deepl_translate(
  path = file,
  out_path = out_path,
  source_lang = "EN",
  target_lang = "ES",
  formality = "less"
)
readLines(out_path)

## End(Not run)
```

---

deepl_translate_clipboard

Translate Markdown string from clipboard

Description

Read the string from the clipboard using the clipr package. Guesses the source language using the cld3 package.

Usage

```
deepl_translate_clipboard(
  target_lang,
  source_lang = NULL,
  glossary_name = NULL,
  formality = c("default", "more", "less", "prefer_more", "prefer_less")
)
```

Arguments

target_lang	Name or code of source language. See DeepL docs.

`source_lang` Name or code of source language. Guessed using cld3 if not provided. See DeepL docs.

`glossary_name` Name of the glossary to be used, if any (character).

`formality` Formality level to use (character), one of

- "default" (default)
- "less" – for a more informal language
- "prefer_more" – for a more formal language if available, otherwise fallback to default formality
- "prefer_less" – for a more informal language if available, otherwise fallback to default formality

Details

To make this function less chatty, set the `babeldown.quiet` option to `TRUE`.

Value

Translated Markdown string, also put on the clipboard.

Examples

```
## Not run:
clipr::write_clip("Je suis en vacances, en fait.")
deepl_translate_clipboard(target_lang = "EN-US")

clipr::write_clip("Je suis en vacances.")
deepl_translate_clipboard(target_lang = "EN-US")

## End(Not run)
```

`deepl_translate_hugo` *Translate a Hugo post*

Description

This translates the Markdown content including the "alt", "caption", "title" fields of shortcodes.

If it translates the title, it will update the slug.

This assumes the Hugo website uses

- YAML metadata at the top of posts;
- leaf bundles (each post in a folder, `leaf-bundle/index.md`);
- multilingualism so that a post in say Spanish lives in `leaf-bundle/index.es.md`.

Usage

```
deepl_translate_hugo(
  post_path = NULL,
  force = FALSE,
  yaml_fields = c("title", "description"),
  glossary_name = NULL,
  source_lang = NULL,
  target_lang = NULL,
  formality = c("default", "more", "less", "prefer_more", "prefer_less")
)
```

Arguments

post_path	Path to post. If RStudio IDE is installed, it will default to the currently open document. If you use Quarto or R Markdown for Hugo, translate the source file (qmd or Rmd) and then render it to Markdown.
force	Whether to overwrite the post in the target language.
yaml_fields	Vector of character names of YAML fields to translate.
glossary_name	Name of the glossary to be used, if any (character).
source_lang	Name or code of source language. See DeepL docs.
target_lang	Name or code of source language. See DeepL docs.
formality	Formality level to use (character), one of • "default" (default) • "less" – for a more informal language • "prefer_more" – for a more formal language if available, otherwise fallback to default formality • "prefer_less" – for a more informal language if available, otherwise fallback to default formality

Value

None

Examples

```
## Not run:
dir <- withr::local_tempdir()
blogdown::new_site(dir = dir)
deepl_translate_hugo(
  file.path(dir, "content", "post", "2016-12-30-hello-markdown", "index.md"),
  source_lang = "en",
  target_lang = "fr",
  formality = "less"
)
readLines(file.path(dir, "content", "post", "2016-12-30-hello-markdown", "index.fr.md"))

## End(Not run)
```

`deepl_translate_markdown_string` *Translate Markdown string*

Description

Translate Markdown string

Usage

```
deepl_translate_markdown_string(
  markdown_string,
  glossary_name = NULL,
  source_lang,
  target_lang,
  formality = c("default", "more", "less", "prefer_more", "prefer_less")
)
```

Arguments

`markdown_string`	Markdown string to translate
`glossary_name`	Name of the glossary to be used, if any (character).
`source_lang`	Name or code of source language. See DeepL docs.
`target_lang`	Name or code of source language. See DeepL docs.
`formality`	Formality level to use (character), one of • "default" (default) • "less" – for a more informal language • "prefer_more" – for a more formal language if available, otherwise fallback to default formality • "prefer_less" – for a more informal language if available, otherwise fallback to default formality

Value

Translated Markdown string

Examples

```
## Not run:
deepl_translate_markdown_string(
  "[So _incredibly_ **wonderful**](https://ropensci.org)!",
  source_lang = "EN",
  target_lang = "FR",
  formality = "less"
)
```

```
## End(Not run)
```

`deepl_translate_quarto` *Translate a Quarto book chapter*

Description

This assumes the book is set up à la babelquarto.

Usage

```
deepl_translate_quarto(
  book_path,
  chapter,
  force = FALSE,
  render = TRUE,
  yaml_fields = c("title", "description"),
  glossary_name = NULL,
  source_lang = NULL,
  target_lang = NULL,
  formality = c("default", "more", "less", "prefer_more", "prefer_less")
)
```

Arguments

`book_path`	Path to book source
`chapter`	Chapter name
`force`	Whether to overwrite the chapter in the target language.
`render`	Whether to run `babelquarto::render_bool()` after translation.
`yaml_fields`	Vector of character names of YAML fields to translate.
`glossary_name`	Name of the glossary to be used, if any (character).
`source_lang`	Name or code of source language. See DeepL docs.
`target_lang`	Name or code of source language. See DeepL docs.
`formality`	Formality level to use (character), one of • "default" (default) • "less" – for a more informal language • "prefer_more" – for a more formal language if available, otherwise fallback to default formality • "prefer_less" – for a more informal language if available, otherwise fallback to default formality

Value

None

Examples

```
## Not run:
temp_dir <- withr::local_tempdir()
babelquarto::quarto_multilingual_book(
  parent_dir = temp_dir,
  book_dir = "blop",
  main_language = "en",
  further_languages = "es"
)
book_path <- file.path(temp_dir, "blop")
deepl_translate_quarto(
  book_path = book_path,
  chapter = "intro.qmd",
  force = TRUE, # the existing chapter is a placeholder
  render = TRUE,
  source_lang = "EN",
  target_lang = "ES",
  formality = "less"
)
# have a look at the translation
readLines(file.path(book_path, "intro.es.qmd"))
# servr::httw(file.path(book_path, "_book"))

## End(Not run)
```

---

deepl_translate_vtt *Translate a VTT subtitles file*

Description

[Experimental]

Usage

```
deepl_translate_vtt(
  path,
  out_path,
  glossary_name = NULL,
  source_lang = NULL,
  target_lang = NULL,
  formality = c("default", "more", "less", "prefer_more", "prefer_less")
)
```

Arguments

path	Path to the Markdown file to be translated (character).
out_path	Path where the new translated file should be saved (character).
glossary_name	Name of the glossary to be used, if any (character).
source_lang	Name or code of source language. See DeepL docs.
target_lang	Name or code of source language. See DeepL docs.
formality	Formality level to use (character), one of • "default" (default) • "less" – for a more informal language • "prefer_more" – for a more formal language if available, otherwise fallback to default formality • "prefer_less" – for a more informal language if available, otherwise fallback to default formality

Value

None

Examples

```
## Not run:
vtt <- system.file("pecan.vtt", package = "babeldown")
temp_dir <- withr::local_tempdir()
deepl_translate_vtt(
  vtt,
  out_path = file.path(temp_dir, "pecan.es.vtt"),
  source_lang = "EN",
  target_lang = "ES",
  formality = "less"
)
head(readLines(file.path(temp_dir, "pecan.es.vtt")))

## End(Not run)
```

---

deepl_update *Update a translation of a file in a Git repo*

Description

Re-use existing translation where possible (at the node level: paragraph, heading, item of list at the first level, etc.) `deepl_update()` is based on the commit history in a single branch, `deepl_branch_update()` is based on the commit history in a branch against another (in a GitHub PR for instance), and requires a configuration file (see "Configuration" section).

Usage

```
deepl_update(
  path,
  out_path,
  yaml_fields = c("title", "description"),
  glossary_name = NULL,
  source_lang = NULL,
  target_lang = NULL,
  formality = c("default", "more", "less", "prefer_more", "prefer_less"),
  max_commits = 100L
)

deepl_branch_update(repo = ".", max_commits = 100)
```

Arguments

`path`	Path to the Markdown file to be translated (character).
`out_path`	Path where the new translated file should be saved (character).
`yaml_fields`	Vector of character names of YAML fields to translate.
`glossary_name`	Name of the glossary to be used, if any (character).
`source_lang`	Name or code of source language. See DeepL docs.
`target_lang`	Name or code of source language. See DeepL docs.
`formality`	Formality level to use (character), one of • "default" (default) • "less" – for a more informal language • "prefer_more" – for a more formal language if available, otherwise fallback to default formality • "prefer_less" – for a more informal language if available, otherwise fallback to default formality
`max_commits`	Maximal number of commits to go back to to find when the target and source files were updated. You might need to increase it if your project is very active.
`repo`	path to the Git repository.

Value

None

Limitations of `deepl_update()`

`deepl_update()` looks for the latest commit that updated the source file, and for the latest commit that updated the target file. If the target file was updated later than the source file, or at the same time, nothing happens: you might need to fix up the Git history with rebase for instance. Also note that if there were some commits irrelevant to the translation but containing edits to the source file, like typo fixes, too much of the target file will be translated automatically again.

Configuration for `deepl_branch_update()`

`deepl_branch_update()` is more automatic but needs configuration. In a branch, for any edits made to any Rmd/qmd/md file, it will update the translations of the Rmd/qmd/md file with the same name and a language code. For instance if a branch had commits updating `pkg_building.Rmd`, `deepl_branch_update()` will update `pkg_building.es.Rmd`. Do not run it if your PR updated both `pkg_building.Rmd` and `pkg_building.es.Rmd`. Make sure to read the section about Markdown formatting.

The configuration needed must live under `_deepl.yml` at the root of the Git repo. It must contains those fields:

- "preferences" that for any source-target pair, indicates the preference for the `formality`, `glossary` and `yaml_fields` arguments of `deepl_translate()`.
- "excludes" that indicates which files to exclude from the automatic detection (all Rmd/qmd/md files are otherwise considered). It is used as `glob` arguments in calls to `fs::dir_ls()`.
- "languages" that for any language code used in file names, indicates what DeepL API code to use when it is a source or target language.

Example (`_deepl.yml`):

```
preferences:
  - source: en
    target: es-419
    formality: less
    glossary: "glosario"
    yaml_fields: ["title"]
  - source: en
    target: pt-br
    formality: less
    yaml_fields: ["title", "description"]
  - source: pt
    target: en-us
    formality: prefer_less
    yaml_fields: ["title", "description"]
  - source: es
    target: en-us
    formality: prefer_less
    yaml_fields: ["title", "description"]

excludes: ["*.md"]

languages:
  - extension: es
    target: es-419
    source: es
  - extension: ''
    target: en-us
    source: en
  - extension: pt
```

```
target: pt-br
source: pt
```

`deepl_upsert_glossary` *Create or update glossary*

Description

Create or update glossary

Usage

```
deepl_upsert_glossary(filename, glossary_name = NULL, source_lang, target_lang)
```

Arguments

`filename`	Name of the glossary file. See DeepL docs on supported formats. babeldown is stricter: the columns need to be named like `source_lang` and `target_lang`.
`glossary_name`	Name for the glossary. Defaults to the filename without extension.
`source_lang`	Name or code of source language. See DeepL docs.
`target_lang`	Name or code of source language. See DeepL docs.

Value

glossary ID

Examples

```
## Not run:
deepl_upsert_glossary(
  system.file("example-es-en.csv", package = "babeldown"),
  glossary_name = "rstats-glosario",
  target_lang = "Spanish",
  source_lang = "English"
)

## End(Not run)
```

`deepl_usage` *Get DeepL API usage data*

Description

Get DeepL API usage data

Usage

```
deepl_usage()
```

Value

A list with DeepL API usage data, depending on the account type.

Index

check_xml_structure, 2

deepl_branch_update *(*deepl_update*)*, 10
deepl_languages, 2
deepl_translate, 3
deepl_translate_clipboard, 4
deepl_translate_hugo, 5
deepl_translate_markdown_string, 7
deepl_translate_quarto, 8
deepl_translate_vtt, 9
deepl_update, 10
deepl_upsert_glossary, 13
deepl_usage, 14