Package: babeldown (via r-universe)

December 20, 2024

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)

RoxygenNote 7.3.1

URL `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 (>= 0.2.0.9000), withr, xml2, yaml (>= 2.3.8)

Suggests blogdown, fs, gert, httptest2, knitr, quarto, rmarkdown, rprojroot, sys, testthat (>= 3.0.0)

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

Config/testthat/edition 3

Config/pak/sysreqs make libicu-dev libxml2-dev libssl-dev libx11-dev

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

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

RemoteRef main

RemoteSha d908e3f34632aacbe165667f057865aaf90532ae

Contents

`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_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, etc.)

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")
)
```

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

Details

The function 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 reorder the Git history with rebase for instance.

Value

None

`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