Package ‘deeplr’

May 20, 2025

Type Package

Title Interface to the 'DeepL' Translation API

Version 2.1.0

Description
A wrapper for the 'DeepL' API <https://developers.deepl.com/docs>, a web service for translating texts between different languages. A DeepL API developer account is required to use the service (see <https://www.deepl.com/pro#developer>).

Encoding UTF-8

URL https://www.deepl.com/translator

BugReports https://github.com/zumbov2/deeplr/issues

Imports utf8, httr, tibble, purrr, tokenizers, jsonlite, readr

Suggests dplyr

RoxygenNote 7.3.2

License MIT + file LICENSE

NeedsCompilation no

Author David Zumbach [aut, cre],
Paul C. Bauer [aut]

Maintainer David Zumbach <david.zumbach@gfzb.ch>

Repository CRAN

Date/Publication 2025-05-20 15:50:02 UTC

Contents

`deeplr-package` `deeplr` *package*

Description

An R wrapper for the DeepL Translator API

Details

See the README on GitHub

Author(s)

Maintainer: David Zumbach <david.zumbach@gfzb.ch>

Authors:

- Paul C. Bauer <mail@paulcbauer.eu>

See Also

Useful links:

- https://www.deepl.com/translator
- Report bugs at https://github.com/zumbov2/deeplr/issues

---

available_languages *List Supported Languages of the DeepL API Pro*

Description

available_languages returns a list of all languages supported by the DeepL API Pro.

Usage

```
available_languages(auth_key)
```

Arguments

auth_key	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. The function makes an API call to retrieve the list of supported languages and returns them in a structured format.

References

DeepL API Documentation on Supported Languages

Examples

```
## Not run:
available_languages()

## End(Not run)
```

`available_languages2` *List Supported Languages of the DeepL API Free*

Description

`available_languages2` returns a list of all languages supported by the DeepL API Free.

Usage

```
available_languages2(auth_key)
```

Arguments

`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. The function makes an API call to retrieve the list of supported languages and returns them in a structured format.

References

DeepL API Documentation on Supported Languages

Examples

```
## Not run:
available_languages2()

## End(Not run)
```

`create_glossary` *Create a Glossary with the DeepL API Pro*

Description

`create_glossary` creates a glossary for a language pair using the DeepL API Pro.

Usage

```
create_glossary(
  name,
  source_lang,
  target_lang,
  entries_source_lang,
  entries_target_lang,
  return_tibble = F,
  auth_key
)
```

Arguments

`name`	A string specifying the name to be associated with the glossary.
`source_lang`	A string specifying the source language code.
`target_lang`	A string specifying the target language code.
`entries_source_lang`	A character vector containing the glossary entries in the source language.
`entries_target_lang`	A character vector containing the glossary entries in the target language.
`return_tibble`	Logical. If `TRUE`, the returned result will be converted to a tibble.
`auth_key`	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. The function sends a request to create a glossary and returns the result in a structured format.

References

DeepL API Documentation on Glossaries

Examples

```
## Not run:
glossary_english <- c("Hello", "Goodbye")
glossary_swiss_german <- c("Grüezi", "Adiöö")

create_glossary(
  name = "My Glossary",
  source_lang = "en",
  target_lang = "de",
  entries_source_lang = glossary_english,
  entries_target_lang = glossary_swiss_german
)

## End(Not run)
```

create_glossary2 *Create a Glossary with the DeepL API Free*

Description

`create_glossary2` creates a glossary for a language pair using the DeepL API Free.

Usage

```
create_glossary2(
  name,
  source_lang,
  target_lang,
  entries_source_lang,
  entries_target_lang,
  return_tibble = F,
  auth_key
)
```

Arguments

`name`	A string specifying the name to be associated with the glossary.
`source_lang`	A string specifying the source language code.
`target_lang`	A string specifying the target language code.
`entries_source_lang`	A character vector containing the glossary entries in the source language.
`entries_target_lang`	A character vector containing the glossary entries in the target language.
`return_tibble`	Logical. If `TRUE`, the returned result will be converted to a tibble.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. The function sends a request to create a glossary and returns the result in a structured format.

References

DeepL API Documentation on Glossaries

Examples

```
## Not run:
glossary_english <- c("Hello", "Goodbye")
glossary_swiss_german <- c("Grüezi", "Adiöö")

create_glossary2(
  name = "My Glossary",
  source_lang = "en",
  target_lang = "de",
  entries_source_lang = glossary_english,
  entries_target_lang = glossary_swiss_german
)

## End(Not run)
```

`delete_glossary` *Delete a Glossary with the DeepL API Pro*

Description

`delete_glossary` deletes a glossary from your DeepL API Pro account using its unique ID.

Usage

```
delete_glossary(glossary_id, auth_key)
```

Arguments

`glossary_id`	A string specifying the unique ID of the glossary to be deleted.
`auth_key`	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

Deleting a glossary is permanent and cannot be undone. If needed, you can back up glossary entries in advance using `get_glossary_entries`.

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro.

For a list of available glossaries and their metadata, see `list_glossaries`.

Value

No return value. A confirmation message is printed upon successful deletion.

References

DeepL API Documentation on Glossary Deletion

Examples

```
## Not run:
glossary_id <- "example-glossary-id"
delete_glossary(glossary_id)

## End(Not run)
```

`delete_glossary2` *Delete a Glossary with the DeepL API Free*

Description

`delete_glossary2` deletes a glossary from your DeepL API Free account using its unique ID.

Usage

```
delete_glossary2(glossary_id, auth_key)
```

Arguments

`glossary_id`	A string specifying the unique ID of the glossary to be deleted.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

Deleting a glossary is permanent and cannot be undone. If needed, you can back up glossary entries in advance using `get_glossary_entries2`.

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free.

For a list of available glossaries and their metadata, see `list_glossaries2`.

Value

No return value. A confirmation message is printed upon successful deletion.

References

DeepL API Documentation on Glossary Deletion

Examples

```
## Not run:
glossary_id <- "example-glossary-id"
delete_glossary2(glossary_id)

## End(Not run)
```

`detect` *Detect the Language of a Text with the DeepL API Pro*

Description

`detect` identifies the language of a given text using the DeepL API Pro.

Usage

```
detect(text, auth_key)
```

Arguments

`text`	A character vector containing the texts to classify. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`auth_key`	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters submitted. To view all supported languages, use `available_languages`.

References

DeepL API Documentation

Examples

```
## Not run:
detect("My name is Hans.")

## End(Not run)
```

detect2 — *Detect the Language of a Text with the DeepL API Free*

Description

`detect2` identifies the language of a given text using the DeepL API Free.

Usage

```
detect2(text, auth_key)
```

Arguments

`text`	A character vector containing the texts to classify. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, you can translate or detect up to 500,000 characters per month at no cost. To view all supported languages, use `available_languages2`.

References

DeepL API Documentation

Examples

```
## Not run:
detect2("My name is Hans.")

## End(Not run)
```

get_glossary_entries *Retrieve Glossary Entries with the DeepL API Pro*

Description

`get_glossary_entries` retrieves all term pairs from a specified glossary using the DeepL API Pro. For a list of available glossaries, see `list_glossaries`.

Usage

```
get_glossary_entries(glossary_id, auth_key)
```

Arguments

`glossary_id`	A string specifying the unique ID of the glossary whose entries you want to retrieve.
`auth_key`	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

Glossaries are custom dictionaries consisting of source-target term pairs. The DeepL API returns these entries as tab-separated values (TSV). This function parses and converts them into a tidy tibble for further analysis.

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro.

Value

A tibble with two columns representing the source and target language terms. Column names are automatically inferred from the glossary's language pair.

References

DeepL API Documentation — Get Glossary Entries

Examples

```
## Not run:
glossary_id <- "your-glossary-id"
entries <- get_glossary_entries(glossary_id)

## End(Not run)
```

get_glossary_entries2 *Retrieve Glossary Entries with the DeepL API Free*

Description

`get_glossary_entries2` retrieves all term pairs from a specified glossary using the DeepL API Free. For a list of available glossaries, see `list_glossaries2`.

Usage

```
get_glossary_entries2(glossary_id, auth_key)
```

Arguments

`glossary_id`	A string specifying the unique ID of the glossary whose entries you want to retrieve.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

Glossaries are custom dictionaries consisting of source-target term pairs. The DeepL API returns these entries as tab-separated values (TSV). This function parses and converts them into a tidy tibble for further analysis.

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free.

Value

A tibble with two columns representing the source and target language terms. Column names are automatically inferred from the glossary's language pair.

References

DeepL API Documentation — Get Glossary Entries

Examples

```
## Not run:
glossary_id <- "your-glossary-id"
entries <- get_glossary_entries2(glossary_id)

## End(Not run)
```

`list_glossaries` *List All Glossaries from the DeepL API Pro*

Description

`list_glossaries` retrieves a list of all glossaries and their metadata associated with your DeepL API Pro account. Note that glossary entries themselves are not included.

Usage

```
list_glossaries(auth_key)
```

Arguments

`auth_key`	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro.

References

DeepL API Documentation on Glossaries

Examples

```
## Not run:
list_glossaries()

## End(Not run)
```

`list_glossaries2` *List All Glossaries from the DeepL API Free*

Description

`list_glossaries2` retrieves a list of all glossaries and their metadata associated with your DeepL API Free account. Note that glossary entries themselves are not included.

Usage

```
list_glossaries2(auth_key)
```

Arguments

`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free.

References

DeepL API Documentation on Glossaries

Examples

```
## Not run:
list_glossaries2()

## End(Not run)
```

`pimp` *Improve Texts via Round-Trip Translation with the DeepL API Pro*

Description

`pimp` translates a text into a helper language and then back to the original language using the DeepL API Pro. This method can be used to refine or rephrase text automatically.

Usage

```
pimp(text, source_lang, help_lang, auth_key)
```

Arguments

`text`	A character vector containing the texts to be improved. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language of the input text. If of length 1, the same language is applied to all elements.
`help_lang`	A string specifying the helper language used for the intermediate translation.
`auth_key`	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs based on the number of translated characters. To view all supported languages, use `available_languages`.

References

DeepL API Documentation on Translation

Examples

```
## Not run:
pimp(
  "In former times I lived in Zurich",
  source_lang = "EN",
  help_lang = "DE"
  )

## End(Not run)
```

pimp2 *Improve Texts via Round-Trip Translation with the DeepL API Free*

Description

pimp2 translates a text into a helper language and then back to the original language using the DeepL API Free. This method can be used to refine or rephrase text automatically.

Usage

```
pimp2(text, source_lang, help_lang, auth_key)
```

Arguments

`text`	A character vector containing the texts to be improved. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language of the input text. If of length 1, the same language is applied to all elements.
`help_lang`	A string specifying the helper language used for the intermediate translation.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, you can translate up to 500,000 characters per month at no cost. To view all supported languages, use `available_languages2`.

References

DeepL API Documentation on Translation

Examples

```
## Not run:
pimp2(
  text = "In former times I lived in Zurich",
  source_lang = "EN",
  help_lang = "DE"
)

## End(Not run)
```

`split_text` *Split Text into Byte-Limited Segments*

Description

`split_text` divides input text into smaller segments that do not exceed a specified maximum size in bytes. Segmentation is based on sentence or word boundaries.

Usage

```
split_text(text, max_size_bytes = 29000, tokenize = "sentences")
```

Arguments

`text`	A character vector containing the text(s) to be split.
`max_size_bytes`	An integer specifying the maximum size (in bytes) for each segment.
`tokenize`	A string indicating the level of tokenization. Must be either `"sentences"` or `"words"`.

Details

This function uses `tokenizers::tokenize_sentences` (or `tokenize_words` if specified) to split the text into natural language segments before assembling byte-limited blocks.

Value

A tibble with one row per text segment, containing the following columns:

- `text_id`: The index of the original text in the input vector.
- `segment_id`: A sequential ID identifying the segment number.
- `segment_text`: The resulting text segment, each within the specified byte limit.

Examples

```
## Not run:
long_text <- paste0(rep("This is a very long text. ", 10000), collapse = "")
split_text(long_text, max_size_bytes = 1000, tokenize = "sentences")

## End(Not run)
```

`supported_glossary_language_pairs` *List Supported Glossary Language Pairs with the DeepL API Pro*

Description

`supported_glossary_language_pairs` lists all language pairs supported for glossary creation in the DeepL API Pro.

Usage

```
supported_glossary_language_pairs(auth_key)
```

Arguments

`auth_key`	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro.

References

DeepL API Documentation — Supported Glossary Language Pairs

Examples

```
## Not run:
supported_glossary_language_pairs()

## End(Not run)
```

---

supported_glossary_language_pairs2

List Supported Glossary Language Pairs with the DeepL API Free

---

Description

`supported_glossary_language_pairs2` lists all language pairs supported for glossary creation in the DeepL API Free.

Usage

```
supported_glossary_language_pairs2(auth_key)
```

Arguments

`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free.

References

DeepL API Documentation — Supported Glossary Language Pairs

Examples

```
## Not run:
supported_glossary_language_pairs2()

## End(Not run)
```

`toChinese` *Translate Texts into Chinese with the DeepL API Pro*

Description

`toChinese` translates a text from any supported source language into Chinese using the DeepL API Pro. Use `available_languages` to list all supported languages.

Usage

```
toChinese(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality` Optional. Controls formality level of the translation (only for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries` to retrieve available glossaries.

`auth_key` A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toChinese("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toChinese(texts, get_detect = TRUE)

## End(Not run)
```

toChinese2 — *Translate Texts into Chinese with the DeepL API Free*

Description

toChinese2 translates a text from any supported source language into Chinese using the DeepL API Free. Use available_languages2 to list all supported languages.

Usage

```
toChinese2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

text	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
source_lang	A string specifying the source language. If NULL, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
split_sentences	Logical. If TRUE, the engine splits the input into sentences. For single-sentence inputs, consider setting to FALSE to prevent unwanted splitting.
preserve_formatting	Logical. If TRUE, formatting such as punctuation and casing is preserved.
get_detect	Logical. If TRUE, the detected language of the source text is included in the response.
context	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
model_type	Optional. Specifies the DeepL model to use: • "latency_optimized" – Default low-latency model. • "quality_optimized" – Higher quality, higher latency model (Pro only, limited language pairs).

- "prefer_quality_optimized" – Use quality-optimized when available, otherwise fallback.

formality
: Optional. Controls formality level of the translation (only for certain target languages):

  - "default" – Neutral.
  - "more" – More formal.
  - "less" – More informal.
  - "prefer_more" – Prefer formal, fallback to default.
  - "prefer_less" – Prefer informal, fallback to default.

glossary_id
: Optional. Glossary ID for custom translation. Must match the language pair and requires source_lang. Use list_glossaries2 to retrieve available glossaries.

auth_key
: A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If get_detect = FALSE, a character vector with translations is returned. If get_detect = TRUE, a tibble with the following columns is returned:

- translation: The translated text.
- source_lang: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toChinese2("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toChinese2(texts, get_detect = TRUE)

## End(Not run)
```

`toEnglish` *Translate Texts into English with the DeepL API Pro*

Description

`toEnglish` translates a text from any supported source language into English using the DeepL API Pro. Use `available_languages` to list all supported languages.

Usage

```
toEnglish(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- "prefer_quality_optimized" – Use quality-optimized when available, otherwise fallback.

formality	Optional. Controls formality level of the translation (only for certain target languages): • "default" – Neutral. • "more" – More formal. • "less" – More informal. • "prefer_more" – Prefer formal, fallback to default. • "prefer_less" – Prefer informal, fallback to default.
glossary_id	Optional. Glossary ID for custom translation. Must match the language pair and requires source_lang. Use list_glossaries to retrieve available glossaries.
auth_key	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If get_detect = FALSE, a character vector with translations is returned. If get_detect = TRUE, a tibble with the following columns is returned:

- translation: The translated text.
- source_lang: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toEnglish("Hallo Welt!")

texts <- c("Me llamo Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toEnglish(texts, get_detect = TRUE)

## End(Not run)
```

toEnglish2 *Translate Texts into English with the DeepL API Free*

Description

`toEnglish2` translates a text from any supported source language into English using the DeepL API Free. Use `available_languages2` to list all supported languages.

Usage

```
toEnglish2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality` Optional. Controls formality level of the translation (only for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries2` to retrieve available glossaries.

`auth_key` A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toEnglish2("Hallo Welt!")

texts <- c("Me llamo Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toEnglish2(texts, get_detect = TRUE)

## End(Not run)
```

toFrench *Translate Texts into French with the DeepL API Pro*

Description

toFrench translates a text from any supported source language into French using the DeepL API Pro. Use available_languages to list all supported languages.

Usage

```
toFrench(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

text	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
source_lang	A string specifying the source language. If NULL, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
split_sentences	Logical. If TRUE, the engine splits the input into sentences. For single-sentence inputs, consider setting to FALSE to prevent unwanted splitting.
preserve_formatting	Logical. If TRUE, formatting such as punctuation and casing is preserved.
get_detect	Logical. If TRUE, the detected language of the source text is included in the response.
context	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
model_type	Optional. Specifies the DeepL model to use: • "latency_optimized" – Default low-latency model. • "quality_optimized" – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality` Optional. Controls formality level of the translation (only for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries` to retrieve available glossaries.

`auth_key` A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toFrench("Hallo Welt!")

texts <- c("Me llamo Fred.", "I'm a doctor.", "Ich komme aus der Schweiz.")
toFrench(texts, get_detect = TRUE)

## End(Not run)
```

toFrench2 *Translate Texts into French with the DeepL API Free*

Description

toFrench2 translates a text from any supported source language into French using the DeepL API Free. Use available_languages2 to list all supported languages.

Usage

```
toFrench2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

text	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
source_lang	A string specifying the source language. If NULL, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
split_sentences	Logical. If TRUE, the engine splits the input into sentences. For single-sentence inputs, consider setting to FALSE to prevent unwanted splitting.
preserve_formatting	Logical. If TRUE, formatting such as punctuation and casing is preserved.
get_detect	Logical. If TRUE, the detected language of the source text is included in the response.
context	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
model_type	Optional. Specifies the DeepL model to use: • "latency_optimized" – Default low-latency model. • "quality_optimized" – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality`	Optional. Controls formality level of the translation (only for certain target languages): • `"default"` – Neutral. • `"more"` – More formal. • `"less"` – More informal. • `"prefer_more"` – Prefer formal, fallback to default. • `"prefer_less"` – Prefer informal, fallback to default.
`glossary_id`	Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries2` to retrieve available glossaries.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toFrench2("Hallo Welt!")

texts <- c("Me llamo Fred.", "I'm a doctor.", "Ich komme aus der Schweiz.")
toFrench2(texts, get_detect = TRUE)

## End(Not run)
```

toGerman — *Translate Texts into German with the DeepL API Pro*

Description

`toGerman` translates a text from any supported source language into German using the DeepL API Pro. Use `available_languages` to list all supported languages.

Usage

```
toGerman(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- "prefer_quality_optimized" – Use quality-optimized when available, otherwise fallback.

formality	Optional. Controls formality level of the translation (only for certain target languages): • "default" – Neutral. • "more" – More formal. • "less" – More informal. • "prefer_more" – Prefer formal, fallback to default. • "prefer_less" – Prefer informal, fallback to default.
glossary_id	Optional. Glossary ID for custom translation. Must match the language pair and requires source_lang. Use list_glossaries to retrieve available glossaries.
auth_key	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If get_detect = FALSE, a character vector with translations is returned. If get_detect = TRUE, a tibble with the following columns is returned:

- translation: The translated text.
- source_lang: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toGerman("Hello world!")

texts <- c("Me llamo Fred.", "Je suis médecin.", "I'm from Brisbane.")
toGerman(texts, get_detect = TRUE)

## End(Not run)
```

`toGerman2` *Translate Texts into German with the DeepL API Free*

Description

`toGerman2` translates a text from any supported source language into German using the DeepL API Free. Use `available_languages2` to list all supported languages.

Usage

```
toGerman2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- "prefer_quality_optimized" – Use quality-optimized when available, otherwise fallback.

formality	Optional. Controls formality level of the translation (only for certain target languages): • "default" – Neutral. • "more" – More formal. • "less" – More informal. • "prefer_more" – Prefer formal, fallback to default. • "prefer_less" – Prefer informal, fallback to default.
glossary_id	Optional. Glossary ID for custom translation. Must match the language pair and requires source_lang. Use list_glossaries2 to retrieve available glossaries.
auth_key	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If get_detect = FALSE, a character vector with translations is returned. If get_detect = TRUE, a tibble with the following columns is returned:

- translation: The translated text.
- source_lang: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toGerman2("Hello world!")

texts <- c("Me llamo Fred.", "Je suis médecin.", "I'm from Brisbane.")
toGerman2(texts, get_detect = TRUE)

## End(Not run)
```

toItalian — *Translate Texts into Italian with the DeepL API Pro*

Description

toItalian translates a text from any supported source language into Italian using the DeepL API Pro. Use available_languages to list all supported languages.

Usage

```
toItalian(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

text	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
source_lang	A string specifying the source language. If NULL, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
split_sentences	Logical. If TRUE, the engine splits the input into sentences. For single-sentence inputs, consider setting to FALSE to prevent unwanted splitting.
preserve_formatting	Logical. If TRUE, formatting such as punctuation and casing is preserved.
get_detect	Logical. If TRUE, the detected language of the source text is included in the response.
context	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
model_type	Optional. Specifies the DeepL model to use: • "latency_optimized" – Default low-latency model. • "quality_optimized" – Higher quality, higher latency model (Pro only, limited language pairs).

- "prefer_quality_optimized" – Use quality-optimized when available, otherwise fallback.

formality	Optional. Controls formality level of the translation (only for certain target languages): • "default" – Neutral. • "more" – More formal. • "less" – More informal. • "prefer_more" – Prefer formal, fallback to default. • "prefer_less" – Prefer informal, fallback to default.
glossary_id	Optional. Glossary ID for custom translation. Must match the language pair and requires source_lang. Use list_glossaries to retrieve available glossaries.
auth_key	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If get_detect = FALSE, a character vector with translations is returned. If get_detect = TRUE, a tibble with the following columns is returned:

- translation: The translated text.
- source_lang: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toItalian("Hallo Welt!")

texts <- c("Me llamo Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toItalian(texts, get_detect = TRUE)

## End(Not run)
```

toItalian2 *Translate Texts into Italian with the DeepL API Free*

Description

`toItalian2` translates a text from any supported source language into Italian using the DeepL API Free. Use `available_languages2` to list all supported languages.

Usage

```
toItalian2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- "prefer_quality_optimized" – Use quality-optimized when available, otherwise fallback.

formality	Optional. Controls formality level of the translation (only for certain target languages): • "default" – Neutral. • "more" – More formal. • "less" – More informal. • "prefer_more" – Prefer formal, fallback to default. • "prefer_less" – Prefer informal, fallback to default.
glossary_id	Optional. Glossary ID for custom translation. Must match the language pair and requires source_lang. Use list_glossaries2 to retrieve available glossaries.
auth_key	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If get_detect = FALSE, a character vector with translations is returned. If get_detect = TRUE, a tibble with the following columns is returned:

- translation: The translated text.
- source_lang: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toItalian2("Hallo Welt!")

texts <- c("Me llamo Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toItalian2(texts, get_detect = TRUE)

## End(Not run)
```

toJapanese *Translate Texts into Japanese with the DeepL API Pro*

Description

toJapanese translates a text from any supported source language into Japanese using the DeepL API Pro. Use available_languages to list all supported languages.

Usage

```
toJapanese(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

text	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
source_lang	A string specifying the source language. If NULL, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
split_sentences	Logical. If TRUE, the engine splits the input into sentences. For single-sentence inputs, consider setting to FALSE to prevent unwanted splitting.
preserve_formatting	Logical. If TRUE, formatting such as punctuation and casing is preserved.
get_detect	Logical. If TRUE, the detected language of the source text is included in the response.
context	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
model_type	Optional. Specifies the DeepL model to use: • "latency_optimized" – Default low-latency model. • "quality_optimized" – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality` Optional. Controls formality level of the translation (only for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries` to retrieve available glossaries.

`auth_key` A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toJapanese("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toJapanese(texts, get_detect = TRUE)

## End(Not run)
```

toJapanese2 *Translate Texts into Japanese with the DeepL API Free*

Description

`toJapanese2` translates a text from any supported source language into Japanese using the DeepL API Free. Use `available_languages2` to list all supported languages.

Usage

```
toJapanese2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality`	Optional. Controls formality level of the translation (only for certain target languages): • `"default"` – Neutral. • `"more"` – More formal. • `"less"` – More informal. • `"prefer_more"` – Prefer formal, fallback to default. • `"prefer_less"` – Prefer informal, fallback to default.
`glossary_id`	Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries2` to retrieve available glossaries.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toJapanese2("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toJapanese2(texts, get_detect = TRUE)

## End(Not run)
```

`toPortuguese` *Translate Texts into Portuguese with the DeepL API Pro*

Description

`toPortuguese` translates a text from any supported source language into Portuguese using the DeepL API Pro. Use `available_languages` to list all supported languages.

Usage

```
toPortuguese(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- "prefer_quality_optimized" – Use quality-optimized when available, otherwise fallback.

formality	Optional. Controls formality level of the translation (only for certain target languages): • "default" – Neutral. • "more" – More formal. • "less" – More informal. • "prefer_more" – Prefer formal, fallback to default. • "prefer_less" – Prefer informal, fallback to default.
glossary_id	Optional. Glossary ID for custom translation. Must match the language pair and requires source_lang. Use list_glossaries to retrieve available glossaries.
auth_key	A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable DEEPL_API_KEY. You can set this variable using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If get_detect = FALSE, a character vector with translations is returned. If get_detect = TRUE, a tibble with the following columns is returned:

- translation: The translated text.
- source_lang: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toPortuguese("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toPortuguese(texts, get_detect = TRUE)

## End(Not run)
```

toPortuguese2 — *Translate Texts into Portuguese with the DeepL API Free*

Description

toPortuguese2 translates a text from any supported source language into Portuguese using the DeepL API Free. Use available_languages2 to list all supported languages.

Usage

```
toPortuguese2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

text	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
source_lang	A string specifying the source language. If NULL, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
split_sentences	Logical. If TRUE, the engine splits the input into sentences. For single-sentence inputs, consider setting to FALSE to prevent unwanted splitting.
preserve_formatting	Logical. If TRUE, formatting such as punctuation and casing is preserved.
get_detect	Logical. If TRUE, the detected language of the source text is included in the response.
context	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
model_type	Optional. Specifies the DeepL model to use: • "latency_optimized" – Default low-latency model. • "quality_optimized" – Higher quality, higher latency model (Pro only, limited language pairs).

	• `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.
`formality`	Optional. Controls formality level of the translation (only for certain target languages): • `"default"` – Neutral. • `"more"` – More formal. • `"less"` – More informal. • `"prefer_more"` – Prefer formal, fallback to default. • `"prefer_less"` – Prefer informal, fallback to default.
`glossary_id`	Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries2` to retrieve available glossaries.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toPortuguese2("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toPortuguese2(texts, get_detect = TRUE)

## End(Not run)
```

toRussian *Translate Texts into Russian with the DeepL API Pro*

Description

`toRussian` translates a text from any supported source language into Russian using the DeepL API Pro. Use `available_languages` to list all supported languages.

Usage

```
toRussian(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality` Optional. Controls formality level of the translation (only for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries` to retrieve available glossaries.

`auth_key` A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toRussian("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toRussian(texts, get_detect = TRUE)

## End(Not run)
```

`toRussian2` *Translate Texts into Russian with the DeepL API Free*

Description

`toRussian2` translates a text from any supported source language into Russian using the DeepL API Free. Use `available_languages2` to list all supported languages.

Usage

```
toRussian2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

	• `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.
`formality`	Optional. Controls formality level of the translation (only for certain target languages): • `"default"` – Neutral. • `"more"` – More formal. • `"less"` – More informal. • `"prefer_more"` – Prefer formal, fallback to default. • `"prefer_less"` – Prefer informal, fallback to default.
`glossary_id`	Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries2` to retrieve available glossaries.
`auth_key`	A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toRussian2("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toRussian2(texts, get_detect = TRUE)

## End(Not run)
```

toSpanish *Translate Texts into Spanish with the DeepL API Pro*

Description

`toSpanish` translates a text from any supported source language into Spanish using the DeepL API Pro. Use `available_languages` to list all supported languages.

Usage

```
toSpanish(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality` Optional. Controls formality level of the translation (only for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries` to retrieve available glossaries.

`auth_key` A string representing the authentication key for the DeepL API Pro. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Pro account at DeepL API Pro. This service may incur costs depending on the number of characters translated.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toSpanish("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toSpanish(texts, get_detect = TRUE)

## End(Not run)
```

toSpanish2 *Translate Texts into Spanish with the DeepL API Free*

Description

`toSpanish2` translates a text from any supported source language into Spanish using the DeepL API Free. Use `available_languages2` to list all supported languages.

Usage

```
toSpanish2(
  text,
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	A character vector containing the text(s) to be translated. Only UTF-8 encoded plain text is supported. Each element may contain multiple sentences but should not exceed 30 kB.
`source_lang`	A string specifying the source language. If `NULL`, the API will auto-detect the language. If of length 1, the same source language is applied to all elements.
`split_sentences`	Logical. If `TRUE`, the engine splits the input into sentences. For single-sentence inputs, consider setting to `FALSE` to prevent unwanted splitting.
`preserve_formatting`	Logical. If `TRUE`, formatting such as punctuation and casing is preserved.
`get_detect`	Logical. If `TRUE`, the detected language of the source text is included in the response.
`context`	Optional string providing contextual information to improve translation quality, especially for short or ambiguous text. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher quality, higher latency model (Pro only, limited language pairs).

- `"prefer_quality_optimized"` – Use quality-optimized when available, otherwise fallback.

`formality` Optional. Controls formality level of the translation (only for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for custom translation. Must match the language pair and requires `source_lang`. Use `list_glossaries2` to retrieve available glossaries.

`auth_key` A string representing the authentication key for the DeepL API Free. If not provided, the function will attempt to retrieve the key from the environment variable `DEEPL_API_KEY`. You can set this variable using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must obtain an authentication key by registering for a DeepL API Free account at DeepL API Free. With the Free API, up to 500,000 characters per month can be translated at no cost.

Value

If `get_detect = FALSE`, a character vector with translations is returned. If `get_detect = TRUE`, a tibble with the following columns is returned:

- `translation`: The translated text.
- `source_lang`: The detected or specified source language.

References

DeepL API Documentation — Translate

Examples

```
## Not run:
toSpanish2("Hallo Welt!")

texts <- c("My name is Fred.", "Je suis médecin.", "Ich komme aus der Schweiz.")
toSpanish2(texts, get_detect = TRUE)

## End(Not run)
```

`translate` *Translate Texts Using the DeepL API Pro*

Description

Translates UTF-8 encoded plain text between supported languages using the DeepL API Pro. A list of supported source and target languages is available at https://developers.deepl.com/docs/getting-started/supported-languages. An authentication key is required to use this service. Charges may apply based on the number of characters translated.

Usage

```
translate(
  text,
  target_lang = "EN",
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	Character vector. The text(s) to translate. Each element can contain multiple sentences but must not exceed 30 kB. Only UTF-8 plain text is supported.
`target_lang`	Character vector. Target language(s) for translation. If length 1, all texts are translated into the same language.
`source_lang`	Character vector or `NULL`. Source language(s). If `NULL`, the language is auto-detected. If of length 1, it is applied to all texts.
`split_sentences`	Logical. If `TRUE` (default), DeepL splits input into sentences before translating. Set to `FALSE` to avoid unintended splits in short texts.
`preserve_formatting`	Logical. If `TRUE`, preserves some text formatting (e.g., punctuation and capitalization).
`get_detect`	Logical. If `TRUE`, returns a tibble including detected source languages along with translations.
`context`	Optional. Contextual text to improve translation quality, especially for short or ambiguous inputs. Context is not translated and does not count toward character limits.

`model_type` Optional. Specifies the DeepL model to use:

- `"latency_optimized"` – Default low-latency model.
- `"quality_optimized"` – Higher-quality, higher-latency model (Pro only, limited languages).
- `"prefer_quality_optimized"` – Uses quality-optimized when available; otherwise falls back.

`formality` Optional. Controls the formality level of the translation (only supported for certain target languages):

- `"default"` – Neutral.
- `"more"` – More formal.
- `"less"` – More informal.
- `"prefer_more"` – Prefer formal, fallback to default.
- `"prefer_less"` – Prefer informal, fallback to default.

`glossary_id` Optional. Glossary ID for translation. Must match the language pair and requires `source_lang`. Use `list_glossaries` to retrieve IDs.

`auth_key` Character. Your DeepL API authentication key. If missing, the function uses the `DEEPL_API_KEY` environment variable. You can set it using `Sys.setenv(DEEPL_API_KEY = "your_key")` or in your `.Renviron` file.

Details

Register for a DeepL API Pro key at `https://www.deepl.com/pro#developer`. Only texts passed via the `text` argument count toward your monthly quota.

Value

If `get_detect = FALSE`, returns a character vector of translated texts. If `get_detect = TRUE`, returns a tibble with:

- `translation` – Translated text.
- `source_lang` – Detected or provided source language.

References

DeepL API Documentation — Translate

See Also

`list_glossaries`

Examples

```
## Not run:
translate("I like to translate texts.", target_lang = "DE")

translate(
  c("I like to translate texts.", "Ich übersetze gerne Texte."),
  target_lang = "FR"
  )

translate("I like to translate texts.", target_lang = c("FR", "DE", "IT"))

translate(
  c("I like to translate texts.", "Ich übersetze gerne Texte."),
  target_lang = c("FR", "IT")
  )

## End(Not run)
```

---

translate2 *Translate Texts Using the DeepL API Free*

Description

Translates UTF-8 encoded plain text between supported languages using the DeepL API Free. A list of supported source and target languages is available at https://developers.deepl.com/docs/getting-started/supported-languages. An authentication key is required. The Free plan allows up to 500,000 characters per month.

Usage

```
translate2(
  text,
  target_lang = "EN",
  source_lang = NULL,
  split_sentences = TRUE,
  preserve_formatting = FALSE,
  get_detect = FALSE,
  context = NULL,
  model_type = NULL,
  formality = NULL,
  glossary_id = NULL,
  auth_key
)
```

Arguments

`text`	Character vector. The text(s) to translate. Each element can contain multiple sentences but must not exceed 30 kB. Only UTF-8 plain text is supported.
`target_lang`	Character vector. Target language(s) for translation. If length 1, all texts are translated into the same language.
`source_lang`	Character vector or `NULL`. Source language(s). If `NULL`, the language is auto-detected. If of length 1, it is applied to all texts.

`split_sentences`	Logical. If `TRUE` (default), DeepL splits input into sentences before translating. Set to `FALSE` to avoid unintended splits in short texts.
`preserve_formatting`	Logical. If `TRUE`, preserves some text formatting (e.g., punctuation and capitalization).
`get_detect`	Logical. If `TRUE`, returns a tibble including detected source languages along with translations.
`context`	Optional. Contextual text to improve translation quality, especially for short or ambiguous inputs. Context is not translated and does not count toward character limits.
`model_type`	Optional. Specifies the DeepL model to use: • `"latency_optimized"` – Default low-latency model. • `"quality_optimized"` – Higher-quality, higher-latency model (Pro only, limited languages). • `"prefer_quality_optimized"` – Uses quality-optimized when available; otherwise falls back.
`formality`	Optional. Controls the formality level of the translation (only supported for certain target languages): • `"default"` – Neutral. • `"more"` – More formal. • `"less"` – More informal. • `"prefer_more"` – Prefer formal, fallback to default. • `"prefer_less"` – Prefer informal, fallback to default.
`glossary_id`	Optional. Glossary ID for translation. Must match the language pair and requires `source_lang`. Use `list_glossaries2` to retrieve IDs.
`auth_key`	Character. Your DeepL API authentication key. If missing, the function uses the `DEEPL_API_KEY` environment variable. You can set it using `Sys.setenv(DEEPL_API_KEY = "your_key")` or in your `.Renviron` file.

Details

Register for a free DeepL API key at `https://www.deepl.com/pro#developer`. Only texts passed via the `text` argument count toward your monthly character quota.

Value

If `get_detect = FALSE`, returns a character vector of translated texts. If `get_detect = TRUE`, returns a tibble with:

- `translation` – Translated text.
- `source_lang` – Detected or provided source language.

References

DeepL API Documentation — Translate

See Also

list_glossaries2

Examples

```
## Not run:
translate2("I like to translate texts.", target_lang = "DE")

translate2(
  c("I like to translate texts.", "Ich übersetze gerne Texte."),
  target_lang = "FR"
  )

translate2("I like to translate texts.", target_lang = c("FR", "DE", "IT"))

translate2(
  c("I like to translate texts.", "Ich übersetze gerne Texte."),
  target_lang = c("FR", "IT")
  )

## End(Not run)
```

---

usage — *Retrieve Usage Data from a DeepL API Pro Account*

Description

usage returns the character usage and configured character limit for the current billing period of a DeepL API Pro account.

Usage

```
usage(auth_key)
```

Arguments

auth_key	Character. Your DeepL API authentication key. If missing, the function uses the DEEPL_API_KEY environment variable. You can set it using Sys.setenv(DEEPL_API_KEY = "your_key") or define it in your .Renviron file for persistent use.

Details

To use this function, you must register for a DeepL API Pro account at https://www.deepl.com/pro#developer.

Value

A named list or structured object containing:

- `character_count` – Number of characters used in the current billing period.
- `character_limit` – Total character limit for the current billing period.

References

DeepL API Documentation — Usage

Examples

```
## Not run:
usage()

## End(Not run)
```

---

usage2 *Retrieve Usage Data from a DeepL API Free Account*

Description

`usage2` returns the character usage and configured character limit for the current billing period of a DeepL API Free account.

Usage

```
usage2(auth_key)
```

Arguments

`auth_key`	Character. Your DeepL API authentication key. If missing, the function uses the `DEEPL_API_KEY` environment variable. You can set it using `Sys.setenv(DEEPL_API_KEY = "your_key")` or define it in your `.Renviron` file for persistent use.

Details

To use this function, you must register for a DeepL API Free account at `https://www.deepl.com/pro#developer`. The Free plan includes up to 500,000 characters per month.

Value

A named list or structured object containing:

- `character_count` – Number of characters used in the current billing period.
- `character_limit` – Total character limit for the current billing period.

References

DeepL API Documentation — Usage

Examples

```
## Not run:
usage2()

## End(Not run)
```

Index