Package ‘fuzzylink’

February 23, 2026

Title Probabilistic Record Linkage Using Pretrained Text Embeddings

Version 0.4.1

Description Links datasets through fuzzy string matching using pretrained text embeddings. Produces more accurate record linkage when lexical string distance metrics are a poor guide to match quality (e.g., ``Patricia'' is more lexically similar to ``Patrick'' than it is to ``Trish''). Capable of performing multilingual record linkage. Methods are described in Ornstein (2025) <doi:10.1017/pan.2025.10016>.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Imports stats, utils, dplyr, Rfast, reshape2, stringdist, stringr, httr, jsonlite, httr2 (>= 1.2.1), ranger, ellmer (>= 0.4.0)

Depends R (>= 4.1.0)

URL `https://joeornstein.github.io/software/fuzzylink/`

BugReports `https://github.com/joeornstein/fuzzylink/issues`

NeedsCompilation no

Author Joe Ornstein [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0002-5704-2098`>)

Maintainer Joe Ornstein <`jornstein@uga.edu`>

Repository CRAN

Date/Publication 2026-02-23 19:00:02 UTC

Contents

`anthropic_api_key` *Install an ANTHROPIC API KEY in Your* `.Renviron` *File for Repeated Use*

Description

This function will add your Anthropic API key to your `.Renviron` file so it can be called securely without being stored in your code. After you have installed your key, it can be called any time by typing `Sys.getenv("ANTHROPIC_API_KEY")` and will be automatically called in package functions. If you do not have an `.Renviron` file, the function will create one for you. If you already have an `.Renviron` file, the function will append the key to your existing file, while making a backup of your original file for disaster recovery purposes.

Usage

```
anthropic_api_key(key, overwrite = FALSE, install = FALSE)
```

Arguments

`key`	The API key provided to you from Anthropic formatted in quotes. A key can be acquired at `https://console.anthropic.com/settings/keys`
`overwrite`	If this is set to TRUE, it will overwrite an existing ANTHROPIC_API_KEY that you already have in your `.Renviron` file.
`install`	if TRUE, will install the key in your `.Renviron` file for use in future sessions. Defaults to FALSE.

Value

No return value, called for side effects.

Examples

```
## Not run:
anthropic_api_key("111111abc", install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("ANTHROPIC_API_KEY")

## End(Not run)

## Not run:
# If you need to overwrite an existing key:
anthropic_api_key("111111abc", overwrite = TRUE, install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
```

```
Sys.getenv("ANTHROPIC_API_KEY")

## End(Not run)
```

`check_match` *Test whether two strings match with an LLM prompt.*

Description

Test whether two strings match with an LLM prompt.

Usage

```
check_match(
  string1,
  string2,
  model = "gpt-5.2",
  record_type = "entity",
  instructions = NULL,
  openai_api_key = Sys.getenv("OPENAI_API_KEY"),
  parallel = TRUE
)
```

Arguments

`string1`	A string or vector of strings
`string2`	A string or vector of strings
`model`	Which LLM to prompt; defaults to 'gpt-5.2'. Also accepts Mistral models (e.g. 'mistral-large-latest') and Anthropic Claude models (e.g. 'claude-sonnet-4-5-20250929').
`record_type`	A character describing what type of entity `string1` and `string2` represent. Should be a singular noun (e.g. "person", "organization", "interest group", "city").
`instructions`	A string containing additional instructions to include in the LLM prompt.
`openai_api_key`	Your OpenAI API key. By default, looks for a system environment variable called "OPENAI_API_KEY" (recommended option). Otherwise, it will prompt you to enter the API key as an argument.
`parallel`	TRUE to submit API requests in parallel. Setting to FALSE can reduce rate limit errors at the expense of longer runtime.

Value

A vector the same length as `string1` and `string2`. "Yes" if the pair of strings match, "No" otherwise.

Examples

```
## Not run:
check_match('UPS', 'United Parcel Service')
check_match('UPS', 'United States Postal Service')
check_match(c('USPS', 'USPS', 'USPS'),
            c('Post Office', 'United Parcel', 'US Postal Service'))

## End(Not run)
```

`dot` *Compute the dot product between two vectors*

Description

Compute the dot product between two vectors

Usage

```
dot(vec1, vec2)
```

Arguments

`vec1`	A numeric vector
`vec2`	Another numeric vector

Value

A numeric

Examples

```
dot(c(0,1), c(1,0))
```

`fuzzylink` *Probabilistic Record Linkage Using Pretrained Text Embeddings*

Description

Probabilistic Record Linkage Using Pretrained Text Embeddings

Usage

```
fuzzylink(
  dfA,
  dfB,
  by,
  blocking.variables = NULL,
  verbose = TRUE,
  record_type = "entity",
  instructions = NULL,
  model = "gpt-5.2",
  openai_api_key = Sys.getenv("OPENAI_API_KEY"),
  embedding_dimensions = 256,
  embedding_model = "text-embedding-3-large",
  learner = "glm",
  fmla = match ~ sim + jw,
  max_labels = 10000,
  parallel = TRUE,
  return_all_pairs = FALSE
)
```

Arguments

`dfA`, `dfB`	A pair of data frames or data frame extensions (e.g. tibbles)
`by`	A character denoting the name of the variable to use for fuzzy matching
`blocking.variables`	A character vector of variables that must match exactly in order to match two records
`verbose`	TRUE to print progress updates, FALSE for no output
`record_type`	A character describing what type of entity the by variable represents. Should be a singular noun (e.g. "person", "organization", "interest group", "city").
`instructions`	A string containing additional instructions to include in the LLM prompt during validation.
`model`	Which LLM to prompt when validating matches; defaults to 'gpt-5.2'. Also accepts Mistral models (e.g. 'mistral-large-latest') and Anthropic Claude models (e.g. 'claude-sonnet-4-5-20250929').
`openai_api_key`	Your OpenAI API key. By default, looks for a system environment variable called "OPENAI_API_KEY" (recommended option). Otherwise, it will prompt you to enter the API key as an argument.
`embedding_dimensions`	The dimension of the embedding vectors to retrieve. Defaults to 256
`embedding_model`	Which pretrained embedding model to use; defaults to 'text-embedding-3-large' (OpenAI), but will also accept 'mistral-embed' (Mistral).
`learner`	Which supervised learner should be used to predict match probabilities. Defaults to logistic regression ('glm'), but will also accept random forest ('ranger').

`fmla`	By default, logistic regression model predicts whether two records match as a linear combination of embedding similarity and Jaro-Winkler similarity (`match ~ sim + jw`). Change this input for alternate specifications.
`max_labels`	The maximum number of LLM prompts to submit when labeling record pairs. Defaults to 10,000
`parallel`	TRUE to submit API requests in parallel. Setting to FALSE can reduce rate limit errors at the expense of longer runtime.
`return_all_pairs`	If TRUE, returns *every* within-block record pair from dfA and dfB, not just validated pairs. Defaults to FALSE.

Value

A dataframe with all rows of `dfA` joined with any matches from `dfB`

Examples

```
## Not run:
dfA <- data.frame(state.x77)
dfA$name <- rownames(dfA)
dfB <- data.frame(name = state.abb, state.division)
df <- fuzzylink(dfA, dfB,
                by = 'name',
                record_type = 'US state government',
                instructions = 'The second dataset contains US postal codes.')

## End(Not run)
```

`get_embeddings` *Get pretrained text embeddings*

Description

Get pretrained text embeddings from the OpenAI or Mistral API. Automatically batches requests to handle rate limits.

Usage

```
get_embeddings(
  text,
  model = "text-embedding-3-large",
  dimensions = 256,
  openai_api_key = Sys.getenv("OPENAI_API_KEY"),
  parallel = TRUE
)
```

Arguments

`text`	A character vector
`model`	Which embedding model to use. Defaults to 'text-embedding-3-large'.
`dimensions`	The dimension of the embedding vectors to return. Defaults to 256. Note that the 'mistral-embed' model will always return 1024 vectors.
`openai_api_key`	Your OpenAI API key. By default, looks for a system environment variable called "OPENAI_API_KEY".
`parallel`	TRUE to submit API requests in parallel. Setting to FALSE can reduce rate limit errors at the expense of longer runtime.

Value

A matrix of embedding vectors (one per row).

Examples

```
## Not run:
embeddings <- get_embeddings(c('dog', 'cat', 'canine', 'feline'))
embeddings['dog',] |> dot(embeddings['canine',])
embeddings['dog',] |> dot(embeddings['feline',])

## End(Not run)
```

`get_similarity_matrix` *Create matrix of embedding similarities*

Description

Create a matrix of pairwise similarities between each string in `strings_A` and `strings_B`.

Usage

```
get_similarity_matrix(embeddings, strings_A = NULL, strings_B = NULL)
```

Arguments

`embeddings`	A matrix of text embeddings
`strings_A`	A string vector
`strings_B`	A string vector

Value

A matrix of cosine similarities between the embeddings of strings_A and the embeddings of strings_B

Examples

```
## Not run:
embeddings <- get_embeddings(c('UPS', 'USPS', 'Postal Service'))
get_similarity_matrix(embeddings)
get_similarity_matrix(embeddings, 'Postal Service')
get_similarity_matrix(embeddings, 'Postal Service', c('UPS', 'USPS'))

## End(Not run)
```

`mistral_api_key` *Install a MISTRAL API KEY in Your* `.Renviron` *File for Repeated Use*

Description

This function will add your Mistral API key to your `.Renviron` file so it can be called securely without being stored in your code. After you have installed your key, it can be called any time by typing `Sys.getenv("MISTRAL_API_KEY")` and will be automatically called in package functions. If you do not have an `.Renviron` file, the function will create one for you. If you already have an `.Renviron` file, the function will append the key to your existing file, while making a backup of your original file for disaster recovery purposes.

Usage

```
mistral_api_key(key, overwrite = FALSE, install = FALSE)
```

Arguments

`key`	The API key provided to you from Mistral formated in quotes. A key can be acquired at https://console.mistral.ai/api-keys/
`overwrite`	If this is set to TRUE, it will overwrite an existing MISTRAL_API_KEY that you already have in your `.Renviron` file.
`install`	if TRUE, will install the key in your `.Renviron` file for use in future sessions. Defaults to FALSE.

Value

No return value, called for side effects.

Examples

```
## Not run:
mistral_api_key("111111abc", install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("MISTRAL_API_KEY")

## End(Not run)
## Not run:
# If you need to overwrite an existing key:
mistral_api_key("111111abc", overwrite = TRUE, install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("MISTRAL_API_KEY")

## End(Not run)
```

openai_api_key *Install an OPENAI API KEY in Your* `.Renviron` *File for Repeated Use*

Description

This function will add your OpenAI API key to your `.Renviron` file so it can be called securely without being stored in your code. After you have installed your key, it can be called any time by typing `Sys.getenv("OPENAI_API_KEY")` and will be automatically called in package functions. If you do not have an `.Renviron` file, the function will create one for you. If you already have an `.Renviron` file, the function will append the key to your existing file, while making a backup of your original file for disaster recovery purposes.

Usage

```
openai_api_key(key, overwrite = FALSE, install = FALSE)
```

Arguments

`key`	The API key provided to you from OpenAI formated in quotes.
`overwrite`	If this is set to TRUE, it will overwrite an existing OPENAI_API_KEY that you already have in your `.Renviron` file.
`install`	if TRUE, will install the key in your `.Renviron` file for use in future sessions. Defaults to FALSE.

Value

No return value, called for side effects.

Examples

```
## Not run:
openai_api_key("111111abc", install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("OPENAI_API_KEY")
```

```
## End(Not run)

## Not run:
# If you need to overwrite an existing key:
openai_api_key("111111abc", overwrite = TRUE, install = TRUE)
# First time, reload your environment so you can use the key without restarting R.
readRenviron("~/.Renviron")
# You can check it with:
Sys.getenv("OPENAI_API_KEY")

## End(Not run)
```

Index