Package 'levitate'

May 25, 2021

Type Package

Title Fuzzy String Comparison

Version 0.1.0

Description Provides string similarity calculations inspired by the Python 'fuzzywuzzy' package. Compare strings by edit distance, similarity ratio, best matching substring, ordered token matching and set-based token matching. A range of edit distance measures are available thanks to the 'stringdist' package.

License GPL-3

URL https://lewinfox.github.io/levitate/, https://github.com/lewinfox/levitate/

BugReports https://github.com/lewinfox/levitate/issues

Depends R (>= 2.10)

Imports cli, glue, rlang, stringdist, stringr

Suggests knitr, pkgdown, rmarkdown, testthat

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Lewin Appleton-Fox [aut, cre, cph]

Maintainer Lewin Appleton-Fox <lewin.a.f@gmail.com>

Repository CRAN

Date/Publication 2021-05-25 08:00:02 UTC

R topics documented:

`lev_distance` *String distance metrics*

Description

Uses `stringdist::stringdistmatrix()` to compute a range of string distance metrics.

Usage

```
lev_distance(a, b, pairwise = TRUE, useNames = TRUE, ...)
```

Arguments

`a`	The input strings
`b`	The input strings
`pairwise`	Boolean. If `TRUE`, only the pairwise distances between `a` and `b` will be computed, rather than the combinations of all elements.
`useNames`	Boolean. Use input vectors as row and column names?
`...`	Additional arguments to be passed to `stringdist::stringdistmatrix()` or `stringdist::stringsimmatrix()`.

Value

A numeric scalar, vector or matrix depending on the length of the inputs. See "Details".

Details

This is a thin wrapper around `stringdist::stringdistmatrix()` and mainly exists to coerce the output into the simplest possible format (via `lev_simplify_matrix()`).

The function will return the simplest possible data structure permitted by the length of the inputs `a` and `b`. This will be a scalar if `a` and `b` are length 1, a vector if either (but not both) is length > 1, and a matrix otherwise.

Other options

In addition to `useNames` `stringdist::stringdistmatrix()` provides a range of options to control the matching, which can be passed using `...`. Refer to the `stringdist` documentation for more information.

Examples

```
lev_distance("Bilbo", "Frodo")

lev_distance("Bilbo", c("Frodo", "Merry"))

lev_distance("Bilbo", c("Frodo", "Merry"), useNames = FALSE)

lev_distance(c("Bilbo", "Gandalf"), c("Frodo", "Merry"))
```

---

lev_partial_ratio *Ratio of the best-matching substring*

Description

Find the best `lev_ratio()` between substrings.

Usage

```
lev_partial_ratio(a, b, pairwise = TRUE, useNames = TRUE, ...)
```

Arguments

a	The input strings
b	The input strings
pairwise	Boolean. If TRUE, only the pairwise distances between a and b will be computed, rather than the combinations of all elements.
useNames	Boolean. Use input vectors as row and column names?
...	Additional arguments to be passed to `stringdist::stringdistmatrix()` or `stringdist::stringsimmatrix()`.

Value

A numeric scalar, vector or matrix depending on the length of the inputs.

Details

If string a has length `len_a` and is shorter than string b, this function finds the highest `lev_ratio()` of all the `len_a`-long substrings of b (and vice versa).

Examples

```
lev_ratio("Bruce Springsteen", "Bruce Springsteen and the E Street Band")

# Here the two "Bruce Springsteen" strings will match perfectly.
lev_partial_ratio("Bruce Springsteen", "Bruce Springsteen and the E Street Band")
```

`lev_ratio` *String similarity ratio*

Description

String similarity ratio

Usage

```
lev_ratio(a, b, pairwise = TRUE, useNames = TRUE, ...)
```

Arguments

`a`	The input strings
`b`	The input strings
`pairwise`	Boolean. If `TRUE`, only the pairwise distances between `a` and `b` will be computed, rather than the combinations of all elements.
`useNames`	Boolean. Use input vectors as row and column names?
`...`	Additional arguments to be passed to `stringdist::stringdistmatrix()` or `stringdist::stringsimmatrix()`.

Value

A numeric scalar, vector or matrix depending on the length of the inputs.

Details

This is a thin wrapper around `stringdist::stringsimmatrix()` and mainly exists to coerce the output into the simplest possible format (via `lev_simplify_matrix()`).

The function will return the simplest possible data structure permitted by the length of the inputs `a` and `b`. This will be a scalar if `a` and `b` are length 1, a vector if either (but not both) is length > 1, and a matrix otherwise.

Examples

```
lev_ratio("Bilbo", "Frodo")

lev_ratio("Bilbo", c("Frodo", "Merry"))

lev_ratio("Bilbo", c("Frodo", "Merry"), useNames = FALSE)

lev_ratio(c("Bilbo", "Gandalf"), c("Frodo", "Merry"))
```

`lev_token_set_ratio` *Matching based on common tokens*

Description

Compare stings based on shared tokens.

Usage

```
lev_token_set_ratio(a, b, pairwise = TRUE, useNames = TRUE, ...)
```

Arguments

`a`	The input strings
`b`	The input strings
`pairwise`	Boolean. If `TRUE`, only the pairwise distances between `a` and `b` will be computed, rather than the combinations of all elements.
`useNames`	Boolean. Use input vectors as row and column names?
`...`	Additional arguments to be passed to `stringdist::stringdistmatrix()` or `stringdist::stringsimmatrix()`.

Value

A numeric scalar, vector or matrix depending on the length of the inputs.

Details

Similar to `lev_token_sort_ratio()` this function breaks the input down into tokens. It then identifies any common tokens between strings and creates three new strings:

```
x <- {common_tokens}
y <- {common_tokens}{remaining_unique_tokens_from_string_a}
z <- {common_tokens}{remaining_unique_tokens_from_string_b}
```

and performs three pairwise `lev_ratio()` calculations between them (`x` vs `y`, `y` vs `z` and `x` vs `z`). The highest of those three ratios is returned.

See Also

`lev_token_sort_ratio()`

Examples

```
x <- "the quick brown fox jumps over the lazy dog"
y <- "my lazy dog was jumped over by a quick brown fox"

lev_ratio(x, y)

lev_token_sort_ratio(x, y)

lev_token_set_ratio(x, y)
```

`lev_token_sort_ratio` *Ordered token matching*

Description

Compares strings by tokenising them, sorting the tokens alphabetically and then computing the `lev_ratio()` of the result. This means that the order of words is irrelevant which can be helpful in some circumstances.

Usage

```
lev_token_sort_ratio(a, b, pairwise = TRUE, useNames = TRUE, ...)
```

Arguments

`a`	The input strings
`b`	The input strings
`pairwise`	Boolean. If `TRUE`, only the pairwise distances between `a` and `b` will be computed, rather than the combinations of all elements.
`useNames`	Boolean. Use input vectors as row and column names?
`...`	Additional arguments to be passed to `stringdist::stringdistmatrix()` or `stringdist::stringsimmatrix()`.

Value

A numeric scalar, vector or matrix depending on the length of the inputs.

See Also

`lev_token_set_ratio()`

Examples

```
x <- "Episode IV - Star Wars: A New Hope"
y <- "Star Wars Episode IV - New Hope"

# Because the order of words is different the simple approach gives a low match ratio.
lev_ratio(x, y)

# The sorted token approach ignores word order.
lev_token_sort_ratio(x, y)
```

Index