Package ‘levitate’

October 1, 2023

Type Package

Title Fuzzy String Comparison

Version 0.2.0

Description Provides string similarity calculations inspired by the Python ‘thefuzz’ 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://github.com/lewinfox/levitate/,
https://www.lewinfox.com/levitate/

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

Depends R (>= 2.10)

Imports rlang, stringdist

Suggests glue, knitr, pkgdown, rmarkdown, styler, testthat

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2023-10-01 00:00:02 UTC
**R topics documented:**

lev_best_match 

lev_best_match ..... 2
lev_distance ..... 3
lev_partial_ratio ..... 4
lev_ratio ..... 5
lev_score_multiple ..... 6
lev_token_set_ratio ..... 6
lev_token_sort_ratio ..... 8
lev_weighted_token_ratio ..... 9
lev_weighted_token_set_ratio ..... 10
lev_weighted_token_sort_ratio ..... 10

**Index**

lev_best_match

**lev_best_match**

Get the best matched string from a list of candidates

**Description**

Given an input string and multiple candidates, return the candidate with the best score as calculated by .fn.

**Usage**

lev_best_match(input, candidates, .fn = lev_ratio, ..., decreasing = TRUE)

**Arguments**

input 
A single string

candidates 
One or more candidate strings to score

.fn 
The scoring function to use, as a string or function object. Defaults to lev_ratio().

... 
Additional arguments to pass to .fn.

decreasing 
If TRUE (the default), the candidate with the highest score is ranked first. If using a comparison .fn that computes distance rather than similarity, or if you want the worst match to be returned first, set this to FALSE.

**Value**

A string

**See Also**

lev_score_multiple()

**Examples**

lev_best_match("bilbo", c("frodo", "gandalf", "legolas"))
Description

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

Usage

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

Arguments

- `a, 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

```r
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, 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, 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_score_multiple  
*Score multiple candidate strings against a single input*

**Description**

Given a single input string and multiple candidates, compute scores for each candidate.

**Usage**

```r
lev_score_multiple(input, candidates, .fn = lev_ratio, ..., decreasing = TRUE)
```

**Arguments**

- `input`: A single string
- `candidates`: One or more candidate strings to score
- `.fn`: The scoring function to use, as a string or function object. Defaults to `lev_ratio()`. Additional arguments to pass to `.fn`.
- `decreasing`: If TRUE (the default), the candidate with the highest score is ranked first. If using a comparison `.fn` that computes *distance* rather than similarity, or if you want the worst match to be returned first, set this to FALSE.

**Value**

A list where the keys are candidates and the values are the scores. The list is sorted according to the decreasing parameter, so by default higher scores are first.

**See Also**

`lev_best_match()`

**Examples**

```r
lev_score_multiple("bilbo", c("frodo", "gandalf", "legolas"))
```

---

lev_token_set_ratio  
*Matching based on common tokens*

**Description**

Compare strings based on shared tokens.

**Usage**

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

- `a, 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

```r
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 \( \text{lev_ratio}() \) of the result. This means that the order of words is irrelevant which can be helpful in some circumstances.

Usage

\[
\text{lev_token_sort_ratio}(a, b, \text{pairwise = TRUE}, \text{useNames = TRUE}, \ldots)
\]

Arguments

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

Value

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

See Also

\text{lev_token_set_ratio()}

Examples

```r
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)
```
**lev_weighted_token_ratio**

*Weighted token similarity measure*

**Description**

Computes similarity but allows you to assign weights to specific tokens. This is useful, for example, when you have a frequently-occurring string that doesn’t contain useful information. See examples.

**Usage**

```r
lev_weighted_token_ratio(a, b, weights = list(), ...)
```

**Arguments**

- `a, b` The input strings
- `weights` List of token weights. For example, `weights = list(foo = 0.9, bar = 0.1)`. Any tokens omitted from `weights` will be given a weight of 1.
- `...` Additional arguments to be passed to `stringdist::stringdistmatrix()` or `stringdist::stringsimmatrix()`.

**Value**

A float

**Details**

The algorithm used here is as follows:

- Tokenise the input strings
- Compute the edit distance between each pair of tokens
- Compute the maximum edit distance between each pair of tokens
- Apply any weights from the `weights` argument
- Return \(1 - \frac{\text{sum(weighted_edit_distances)}}{\text{sum(weighted_max_edit_distance)}}\)

**See Also**

Other weighted token functions: `lev_weighted_token_set_ratio()`, `lev_weighted_token_sort_ratio()`

**Examples**

```r
lev_weighted_token_ratio("jim ltd", "tim ltd")
```

```r
lev_weighted_token_ratio("tim ltd", "jim ltd", weights = list(ltd = 0.1))
```
lev_weighted_token_set_ratio

Weighted version of lev_token_set_ratio()

Description

Weighted version of lev_token_set_ratio()

Usage

lev_weighted_token_set_ratio(a, b, weights = list(), ...)

Arguments

a, b  The input strings
weights  List of token weights. For example, weights = list(foo = 0.9, bar = 0.1). Any tokens omitted from weights will be given a weight of 1.
...  Additional arguments to be passed to stringdist::stringdistmatrix() or stringdist::stringsimmatrix().

Value

Float

See Also

lev_token_set_ratio()

Other weighted token functions: lev_weighted_token_ratio(), lev_weighted_token_sort_ratio()

lev_weighted_token_sort_ratio

Weighted version of lev_token_sort_ratio()

Description

This function tokenises inputs, sorts tokens and computes similarities for each pair of tokens. Similarity scores are weighted based on the weights argument, and a total similarity score is returned in the same manner as lev_weighted_token_ratio().

Usage

lev_weighted_token_sort_ratio(a, b, weights = list(), ...)
lev_weighted_token_sort_ratio

Arguments

a, b The input strings
weights List of token weights. For example, weights = list(foo = 0.9, bar = 0.1). Any tokens omitted from weights will be given a weight of 1.
...
Additional arguments to be passed to stringdist::stringdistmatrix() or stringdist::stringsimmatrix().

Value

Float

See Also

lev_token_sort_ratio()

Other weighted token functions: lev_weighted_token_ratio(), lev_weighted_token_set_ratio()
Index

* weighted token functions
  lev_weighted_token_ratio, 9
  lev_weighted_token_set_ratio, 10
  lev_weighted_token_sort_ratio, 10

lev_best_match, 2
lev_best_match(), 6
lev_distance, 3
lev_partial_ratio, 4
lev_ratio, 5
lev_ratio(), 2, 4, 6–8
lev_score_multiple, 6
lev_score_multiple(), 2
lev_simplify_matrix(), 3, 5
lev_token_set_ratio, 6
lev_token_set_ratio(), 8, 10
lev_token_sort_ratio, 8
lev_token_sort_ratio(), 7, 11
lev_weighted_token_ratio, 9, 10, 11
lev_weighted_token_ratio(), 10
lev_weighted_token_set_ratio, 9, 10, 11
lev_weighted_token_sort_ratio, 9, 10, 10

string distance metrics, 3
stringdist::stringdistmatrix(), 3–5, 7–11
stringdist::stringsimmatrix(), 3–5, 7–11