stringdist_api

Generated by Doxygen 1.8.13

Contents

Chapter 1

Stringdist C API

Author

Mark van der Loo, Jan van der Laan, R Core Team, Paul Hsieh, Chris Muir

Version

`R package stringdist` version `0.9.5.0` and higher.

1.1 Using the stringdist C API

To call the functions described here from your package you need to:

1. Make sure that `stringdist` is installed.
2. Add `stringdist` to `Imports` (or `Depends`) and `LinkingTo` in the `DESCRIPTION` file.
3. In your source file under the package's `/src` directory, add the line

```
#include <stringdist_api.h>
```

An example of a published package using this API is `refinr`. A minimal example can be found `here`.

1.2 Character encoding

All `character` vector input is expected to be in `UTF-8` (this also allows `ASCII`). Distance computations are based on UTF `code points` unless `useBytes` is `TRUE`, in which case distances are computed over byte sequences. Using non-UTF-8 encoded strings is untested and is highly likely to result in errors.

1.3 Thread safety

It is not safe to call functions from `stringdist` C API from multiple concurrent threads.

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

File Documentation

3.1 stringdist_api.h File Reference

Functions exported from the stringdist package.

```
#include <R_ext/Rdynload.h>
```

Functions

- SEXP attribute_hidden sd_amatch (SEXP x, SEXP table, SEXP method, SEXP nomatch, SEXP matchNA, SEXP weight, SEXP p, SEXP bt, SEXP q, SEXP maxDistance, SEXP useBytes, SEXP nthrd)

  Find the location of values in `x` in `table` by approximate matching.
- SEXP attribute_hidden sd_get_qgrams (SEXP a, SEXP qq)

  Compute q-gram counts.
- SEXP attribute_hidden sd_lower_tri (SEXP a, SEXP method, SEXP weight, SEXP p, SEXP bt, SEXP q, SEXP useBytes, SEXP nthrd)

  Lower tridiagonal elements of distance matrix.
- SEXP attribute_hidden sd_soundex (SEXP x, SEXP useBytes)

  Compute soundex code.
- SEXP attribute_hidden sd_stringdist (SEXP a, SEXP b, SEXP method, SEXP weight, SEXP p, SEXP bt, SEXP q, SEXP useBytes, SEXP nthrd)

  compute string distances

3.1.1 Detailed Description

Functions exported from the stringdist package.

3.1.2 Function Documentation

3.1.2.1 sd_amatch()

```
SEXP attribute_hidden sd_amatch (
              SEXP x,
              SEXP table,
              SEXP method,
              SEXP nomatch,
              SEXP matchNA,
              SEXP weight,
              SEXP p,
              SEXP bt,
              SEXP q,
              SEXP maxDistance,
              SEXP useBytes,
              SEXP nthrd )
```

Find the location of values in `x` in `table` by approximate matching.

Parameters

x	`[character]` vector.
table	`[character]` vector (lookup table)
method	`[integer]` scalar, indicating the distance method as follows • 0: Optimal String Alignment (`"osa"`) • 1: Levenshtein (`"lv"`) • 2: Damerau-Levenshtein (`"dl"`) • 3: Hamming (`"hamming"`) • 4: Longest Common Substring (`"lcs"`) • 5: q-gram (`"qgram"`) • 6: cosine (`"cosine"`) • 7: Jaccard (`"jaccard"`) • 8: Jaro-Winkler (`"jw"`) • 9: Soundex (`"soundex"`)
nomatch	`[integer]` The value to be returned when no match is found.
matchNA	Should `NA`s be matched? Default behaviour mimics the behaviour of base `match`, meaning that `NA` matches `NA` (see also the note on `NA` handling below).
weight	`[numeric]` vector. Edit penalty For `method='osa'` or `'dl'`, the penalty for deletion, insertion, substitution and transposition, in that order. When `method='lv'`, the penalty for transposition is ignored. When `method='jw'`, the weights associated with characters of `a`, characters from `b` and the transposition weight, in that order. Weights must be positive and not exceed 1. `weight` is ignored completely for other methods
q	`[integer]` scalar. Size of the q-gram; must be nonnegative. Only applies to `method='qgram'`, `'jaccard'` or `'cosine'`.
maxDistance	`[numeric]` scalar. The maximum distance allowed for matching.
p	`[numeric]` scalar. Penalty factor for Jaro-Winkler distance. The valid range for `p` is `0 <= p <= 0.25`. If `p=0` (default), the Jaro-distance is returned. Applies only to `method='jw'`.
bt	`[numeric]` vector. Winkler's boost threshold. Winkler's penalty factor is only applied when the Jaro distance is larger than `bt`. Applies only to `method='jw'` and `p>0`.

Parameters

useBytes	Perform byte-wise comparison (i.e. do not translate UTF-8 to integer prior to distance calculation)
nthread	`[integer]` scalar. Maximum number of threads to use.

Returns

`[integer]` vector of `length(x)` with indices in `table`.

3.1.2.2 sd_get_qgrams()

```
SEXP attribute_hidden sd_get_qgrams (
             SEXP a,
             SEXP qq )
```

Compute q-gram counts.

Parameters

a	`[character]` vector
qq	`[integer]` scalar.

Returns

A `[numeric]` vector of `length(a)*n_qgrams`, where `n_qrams` is the number of different `qgrams` observed in the elements of `a`. The output vector has an attribute called `qgrams`, which is an integer vector of size `q*n_qgrams` containing integer (UTF-32) labels for the q-grams sequentially.

3.1.2.3 sd_lower_tri()

```
SEXP attribute_hidden sd_lower_tri (
             SEXP a,
             SEXP method,
             SEXP weight,
             SEXP p,
             SEXP bt,
             SEXP q,
             SEXP useBytes,
             SEXP nthrd )
```

Lower tridiagonal elements of distance matrix.

Parameters

a	`[character]` vector

Parameters

method	`[integer]` scalar, indicating the distance method as follows • 0: Optimal String Alignment (`"osa"`) • 1: Levenshtein (`"lv"`) • 2: Damerau-Levenshtein (`"dl"`) • 3: Hamming (`"hamming"`) • 4: Longest Common Substring (`"lcs"`) • 5: q-gram (`"qgram"`) • 6: cosine (`"cosine"`) • 7: Jaccard (`"jaccard"`) • 8: Jaro-Winkler (`"jw"`) • 9: Soundex (`"soundex"`)
weight	`[numeric]` vector. Edit penalty For `method='osa'` or `'dl'`, the penalty for deletion, insertion, substitution and transposition, in that order. When `method='lv'`, the penalty for transposition is ignored. When `method='jw'`, the weights associated with characters of `a`, characters from `b` and the transposition weight, in that order. Weights must be positive and not exceed 1. `weight` is ignored completely for other methods
q	`[integer]` scalar. Size of the q-gram; must be nonnegative. Only applies to `method='qgram'`, `'jaccard'` or `'cosine'`.
p	`[numeric]` scalar. Penalty factor for Jaro-Winkler distance. The valid range for `p` is `0 <= p <= 0.25`. If `p=0` (default), the Jaro-distance is returned. Applies only to `method='jw'`.
bt	`[numeric]` vector. Winkler's boost threshold. Winkler's penalty factor is only applied when the Jaro distance is larger than `bt`. Applies only to `method='jw'` and `p>0`.
useBytes	Perform byte-wise comparison (i.e. do not translate UTF-8 to integer prior to distance calculation)
nthread	`[integer]` scalar. Maximum number of threads to use.

Returns

A `[numeric]` vector of length `n*(n-1)/2`, where `n=length(a)`. It contains the positive values of consequtive columns of the distance matrix. Also see the R-code in `stringdist:::lower_tri`.

3.1.2.4 sd_soundex()

```
SEXP attribute_hidden sd_soundex (
             SEXP x,
             SEXP useBytes )
```

Compute soundex code.

Parameters

in	*x*	`[character]` vector
in	*useBytes*	`[logical]` scalar.

Returns

A `list` with `length(x)` element. Each element is a length 4 integer vector representing a 4-character soundex code. The integers are ASCII code points.

3.1.2.5 sd_stringdist()

```
SEXP attribute_hidden sd_stringdist (
             SEXP a,
             SEXP b,
             SEXP method,
             SEXP weight,
             SEXP p,
             SEXP bt,
             SEXP q,
             SEXP useBytes,
             SEXP nthrd )
```

compute string distances

Parameters

a	`[character]` vector
b	`[character]` vector
method	`[integer]` scalar, indicating the distance method as follows • 0: Optimal String Alignment (`"osa"`) • 1: Levenshtein (`"lv"`) • 2: Damerau-Levenshtein (`"dl"`) • 3: Hamming (`"hamming"`) • 4: Longest Common Substring (`"lcs"`) • 5: q-gram (`"qgram"`) • 6: cosine (`"cosine"`) • 7: Jaccard (`"jaccard"`) • 8: Jaro-Winkler (`"jw"`) • 9: Soundex (`"soundex"`)
weight	`[numeric]` vector. Edit penalty For `method='osa'` or `'dl'`, the penalty for deletion, insertion, substitution and transposition, in that order. When `method='lv'`, the penalty for transposition is ignored. When `method='jw'`, the weights associated with characters of `a`, characters from `b` and the transposition weight, in that order. Weights must be positive and not exceed 1. `weight` is ignored completely for other methods
q	`[integer]` scalar. Size of the q-gram; must be nonnegative. Only applies to `method='qgram'`, `'jaccard'` or `'cosine'`.
p	`[numeric]` scalar. Penalty factor for Jaro-Winkler distance. The valid range for `p` is `0 <= p <= 0.25`. If `p=0` (default), the Jaro-distance is returned. Applies only to `method='jw'`.
bt	`[numeric]` vector. Winkler's boost threshold. Winkler's penalty factor is only applied when the Jaro distance is larger than `bt`. Applies only to `method='jw'` and `p>0`.
useBytes	Perform byte-wise comparison (i.e. do not translate UTF-8 to integer prior to distance calculation)
nthread	`[integer]` scalar. Maximum number of threads to use.

Returns

A `[numeric]` vector of length `max(length(a),length(b))` where the shortest vector is recycled over the longer (no warnings are given when the longer length is not an integer multiple of the shorter length).

Index