Package: lingdist (via r-universe)

September 8, 2024

Type Package

Title Fast Linguistic Distance and Alignment Computation

Version 1.0

Date 2023-10-12

Description A fast generalized edit distance and string alignment computation mainly for linguistic aims. As a generalization to the classic edit distance algorithms, the package allows users to define custom cost for every symbol's insertion, deletion, and substitution. The package also allows character combinations in any length to be seen as a single symbol which is very useful for International Phonetic Alphabet (IPA) transcriptions with diacritics. In addition to edit distance result, users can get detailed alignment information such as all possible alignment scenarios between two strings which is useful for testing, illustration or any further usage. Either the distance matrix or its long table form can be obtained and tools to do such conversions are provided. All functions in the package are implemented in 'C++' and the distance matrix computation is parallelized leveraging the 'RcppThread' package.

License GPL (>= 2)

Imports Rcpp (>= 1.0.10)

LinkingTo Rcpp,RcppThread

URL https://github.com/fncokg/lingdist

BugReports https://github.com/fncokg/lingdist/issues

Encoding UTF-8

RoxygenNote 7.2.3

Repository https://fncokg.r-universe.dev

RemoteUrl https://github.com/fncokg/lingdist

RemoteRef HEAD

RemoteSha 37d25a0d9d1e4c24ea25833e4c125ab771abba7d

Contents

`check_cost_defined` *Check whether there's missing characters in the cost matrix.*

Description

Check whether there's missing characters in the cost matrix and return the missing characters.

Usage

```
check_cost_defined(data, cost_mat, delim = "")
```

Arguments

`data`	DataFrame to be computed.
`cost_mat`	Cost matrix to be checked.
`delim`	The delimiter separating atomic symbols.

Value

A string vector containing the missing characters, empty indicating there's no missing characters.

Examples

```
df <- as.data.frame(rbind(a=c("a_bc_d","d_bc_a"),b=c("b_bc_d","d_bc_a")))
cost.mat <- data.frame()
chars.not.found <- check_cost_defined(df, cost.mat, "_")
```

edit_dist_df *Compute edit distance between all row pairs of a dataframe*

Description

Compute average edit distance between all row pairs of a dataframe, empty or NA cells are ignored. If all values in a row are not valid strings, all average distances involving this row is set to -1.

Usage

```
edit_dist_df(
  data,
  cost_mat = NULL,
  delim = "",
  squareform = FALSE,
  symmetric = TRUE,
  parallel = FALSE,
  n_threads = 2L
)
```

Arguments

data	DataFrame with n rows and m columns indicating there are n languages or dialects to involve in the calculation and there are at most m words to base on, in which the rownames are the language ids.
cost_mat	Dataframe in squareform indicating the cost values when one symbol is deleted, inserted or substituted by another. Rownames and colnames are symbols. ‘cost_mat[char1,"_NULL_"]‘ indicates the cost value of deleting char1 and ‘cost_mat["_NULL_",char1]‘ is the cost value of inserting it. When an operation is not defined in the cost_mat, it is set 0 when the two symbols are the same, otherwise 1.
delim	The delimiter separating atomic symbols.
squareform	Whether to return a dataframe in squareform.
symmetric	Whether to the result matrix is symmetric. This depends on whether the ‘cost_mat‘ is symmetric.
parallel	Whether to parallelize the computation.
n_threads	The number of threads is used to parallelize the computation. Only meaningful if ‘parallel‘ is TRUE.

Value

A dataframe in long table form if ‘squareform‘ is FALSE, otherwise in squareform. If ‘symmetric‘ is TRUE, the long table form has C_n^2 rows otherwise n^2 rows.

Examples

```
df <- as.data.frame(rbind(a=c("a_bc_d","d_bc_a"),b=c("b_bc_d","d_bc_a")))
cost.mat <- data.frame()
result <- edit_dist_df(df, cost_mat=cost.mat, delim="_")
result <- edit_dist_df(df, cost_mat=cost.mat, delim="_", squareform=TRUE)
result <- edit_dist_df(df, cost_mat=cost.mat, delim="_", parallel=TRUE, n_threads=4)
```

`edit_dist_string` *Compute edit distance between two strings*

Description

Compute edit distance between two strings and get all possible alignment scenarios. Custom cost matrix is supported. Symbols separated by custom delimiters are supported.

Usage

```
edit_dist_string(
  str1,
  str2,
  cost_mat = NULL,
  delim = "",
  return_alignments = FALSE
)
```

Arguments

`str1`	String to be compared.
`str2`	String to be compared.
`cost_mat`	Dataframe in squareform indicating the cost values when one symbol is deleted, inserted or substituted by another. Rownames and colnames are symbols. ‘cost_mat[char1,"_NULL_"]‘ indicates the cost value of deleting char1 and ‘cost_mat["_NULL_",char1]‘ is the cost value of inserting it. When an operation is not defined in the cost_mat, it is set 0 when the two symbols are the same, otherwise 1.
`delim`	The delimiter in ‘str1‘ and ‘str2‘ separating atomic symbols.
`return_alignments`	Whether to return alignment details

Value

A list contains ‘distance‘ attribution storing the distance result. If ‘return_alignments‘ is TRUE, then a ‘alignments‘ attribution is present which is a list of dataframes with each storing a possible best alignment scenario.

Examples

```
cost.mat <- data.frame()
dist <- edit_dist_string("leaf","leaves")$distance
dist <- edit_dist_string("ph_l_i_z","p_l_i_s",cost_mat=cost.mat,delim="_")$distance
alignments <- edit_dist_string("ph_l_i_z","p_l_i_s",delim="_",return_alignments=TRUE)$alignments
```

---

generate_default_cost_matrix

Generate a default cost matrix

Description

generate a default cost matrix contains all possible characters in the raw data with all diagonal values set to 0 and others set to 1. This avoids you constructing the matrix from scratch.

Usage

```
generate_default_cost_matrix(data, delim = "")
```

Arguments

data	DataFrame to be computed.
delim	The delimiter separating atomic symbols.

Value

Cost matrix contains all possible characters in the raw data with all diagonal values set to 0 and others set to 1.

Examples

```
df <- as.data.frame(rbind(a=c("a_bc_d","d_bc_a"),b=c("b_bc_d","d_bc_a")))
default.cost <- generate_default_cost_matrix(df, "_")
```

---

long2squareform

Convert long table to square form

Description

Convert a distance dataframe in long table form to a square matrix form.

Usage

```
long2squareform(data, symmetric = TRUE)
```

Arguments

`data`	Dataframe in long table form. The first and second columns are labels and the third column stores the distance values.
`symmetric`	Whether the distance matrix are symmetric (if cost matrix is not, then the distance matrix is also not).

Value

Dataframe in square matrix form, rownames and colnames are labels. If the long table only contains C_n^2 rows and 'symmetric' is set to FALSE, then only lower triangle positions in the result is filled.

Examples

```
data <- as.data.frame(list(chars1=c("a","a","b"),chars2=c("b","c","c"),dist=c(1,2,3)))
mat <- long2squareform(data)
```

Index