

Package: lingdist (via r-universe)

October 18, 2026

Type Package

Title Fast Linguistic Distance and Alignment Computation

Version 2.4.0

Date 2026-3-8

Description A fast linguistic distance and alignment computation package. It implements generalized edit distance, Pointwise Mutual Information (PMI) distance, and Weighted Jaccard Distance (WJD). For generalized edit distance, the package allows users to define custom cost for every symbol's insertion, deletion, and substitution, and supports treating character combinations as single symbols (e.g., IPA with diacritics). It also provides detailed alignment information. PMI distance automatically learns costs from data. WJD is suitable for hierarchical categorical data with multiple forms. All functions are implemented in 'C++' and distance matrix computation is parallelized leveraging the 'RcppThread' package.

License GPL (>= 2)

Imports Rcpp (>= 1.1.0), RcppThread (>= 2.3.0)

LinkingTo Rcpp, RcppThread

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

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

Encoding UTF-8

RoxygenNote 7.3.2

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

Date/Publication 2026-08-19 12:59:47 UTC

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

RemoteRef HEAD

RemoteSha c05ebc90001c8597b51fa546dd0c31cd5c7ddfe9

Contents

check_cost_mat_symbols	2
generate_default_cost_matrix	3
long2squareform	3
pw_edit_dist	4
pw_pmi_dist	6
pw_wjd	7
string_edit_dist	8
Index	11

check_cost_mat_symbols
<i>Check which symbols in ‘data’ are missing from a cost matrix</i>

Description

Given a cost matrix (square-form DataFrame) and a data table of tokenized strings, return a character vector of symbols that appear in ‘data’ but are not present in the cost matrix row/column names. This is useful to validate a user-supplied cost matrix before running distance computations.

Usage

```
check_cost_mat_symbols(cost_mat, data, delim)
```

Arguments

cost_mat	DataFrame representing a cost matrix in square form. Row names and column names should be the symbols included in the matrix.
data	DataFrame containing the tokenized strings; used to extract symbols to check.
delim	String delimiter used in ‘data’ to split atomic symbols (empty string means split by characters).

Value

A character vector of symbols that are present in ‘data’ but missing from ‘cost_mat’.

Examples

```
cost.mat <- data.frame()
df <- as.data.frame(rbind(a=c("ph_l_i_z", "k_o_l"), b=c("b_l_i_s", "k_a:l")))
missing <- check_cost_mat_symbols(cost.mat, df, "_")
```

```
generate_default_cost_matrix
```

Generate a default cost matrix

Description

Generate a default cost matrix containing all possible characters in the raw data with all diagonal values set to 0 and others set to 1. This avoids 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 containing 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("ph_l_i_z", "k_o_l"), b=c("b_l_i_s", "k_a:_l")))
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. Values in the diagonal positions are automatically filled with 'default_diag'. If you want self-defined diagonal values, define them in the long table.

Usage

```
long2squareform(data, symmetric = TRUE, default_diag = 0)
```

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 is symmetric (if cost matrix is not, then the distance matrix is also not).
default_diag	The default value to fill in the diagonal positions. By default it is 0.0.

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 are 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)
```

pw_edit_dist	<i>Compute edit distance between all row pairs of a dataframe</i>
--------------	---

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

```
pw_edit_dist(
  data,
  cost_mat = NULL,
  delim = "",
  detailed = FALSE,
  normalize_method = "longest",
  squareform = FALSE,
  symmetric = TRUE,
  parallel = FALSE,
  n_threads = 2L,
  check_missing_cost = TRUE,
  default_sub_cost = 1,
  default_ins_del_cost = 1,
  quiet = FALSE
)
```

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. When 'cost_mat' is NULL, general cost rules are used: substitution cost is 'default_sub_cost' if the two symbols are different and 0 if they are the same; insertion and deletion cost is 'default_ins_del_cost'.
delim	The delimiter separating atomic symbols.
detailed	Whether to return detailed information.
normalize_method	Method to normalize the distance. Default is "longest".
squareform	Whether to return a dataframe in squareform.
symmetric	Whether 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.
check_missing_cost	Whether to check if all symbols in 'data' are defined in 'cost_mat'. If TRUE, an warning message is printed when there are missing symbols. You are recommended to set this to 'TRUE' unless you are sure all symbols are defined in 'cost_mat' and you want to skip the check for performance consideration.
default_sub_cost	Default substitution cost when 'cost_mat' is NULL.
default_ins_del_cost	Default insertion and deletion cost when 'cost_mat' is NULL.
quiet	Whether to suppress all output messages.

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("ph_l_i_z","k_o_l"),b=c("b_l_i_s", "k_a:l")))
cost.mat <- data.frame()
result <- pw_edit_dist(df, delim="_")
result <- pw_edit_dist(df, delim="_", default_sub_cost=2.0, default_ins_del_cost=1.5)
result <- pw_edit_dist(df, cost_mat=cost.mat, delim="_")
result <- pw_edit_dist(df, cost_mat=cost.mat, delim="_", squareform=TRUE)
result <- pw_edit_dist(df, cost_mat=cost.mat, delim="_", parallel=TRUE, n_threads=4, check_missing_cost=FALSE)
```

pw_pmi_dist

*Compute PMI distance between all row pairs of a dataframe***Description**

Compute PMI (Pointwise Mutual Information) distance between all row pairs of a dataframe.

Usage

```
pw_pmi_dist(
  data,
  delim = "",
  detailed = FALSE,
  normalize_method = "longest",
  squareform = FALSE,
  parallel = FALSE,
  n_threads = 4L,
  max_epochs = 20L,
  tol = 1e-04,
  alignment_max_paths = 3L,
  quiet = FALSE
)
```

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.
delim	The delimiter separating atomic symbols.
detailed	Whether to return detailed information.
normalize_method	Method to normalize the distance. Default is "longest".
squareform	Whether to return a dataframe in squareform.
parallel	Whether to parallelize the computation.
n_threads	The number of threads is used to parallelize the computation. Only meaningful if 'parallel' is TRUE.
max_epochs	Maximum number of epochs.
tol	Tolerance for convergence.
alignment_max_paths	Maximum number of paths to consider in alignment. There may be multiple optimal alignment paths between two strings; this parameter limits how many of them are considered when updating the cost matrix in each epoch.
quiet	Whether to suppress all output messages.

Value

A list containing the following components:

result	A dataframe of distances, either in long table form or square form.
cost	The final cost matrix used for distance calculation. Note: this is NOT the cost matrix after the last iteration, but the one before that, which is used to compute the final distances. That is, when you finished the iteration after 10 epochs, the cost matrix used to compute distances is actually the one being updated after the 9th epoch.
sum_diff	The sum of absolute differences between the cost matrices of the last two iterations.
mean_diff	The mean of absolute differences between the cost matrices of the last two iterations.

Examples

```
df <- as.data.frame(rbind(a=c("ph_l_i_z", "k_o_l"), b=c("b_l_i_s", "k_a:_l")))
result <- pw_pmi_dist(df, delim="_")
result <- pw_pmi_dist(df, delim="_", squareform=TRUE)
result <- pw_pmi_dist(df, delim="_", parallel=TRUE, n_threads=4)
```

pw_wjd	<i>Compute Weighted Jaccard Distance between all row pairs of a dataframe</i>
--------	---

Description

Compute Weighted Jaccard Distance (WJD) between all row pairs of a dataframe. This metric is suitable for categorical data with hierarchical structure and multiple forms. For example, a cell value "A_2_a#A_3#B_5_c" indicates there are 3 word forms for this lexical item. The first form "A_2_a" belongs to category A, subcategory 2, and sub-subcategory a. The second form "A_3" belongs to category A and subcategory 3. The third form "B_5_c" belongs to category B, subcategory 5, and sub-subcategory c. The delimiters can be customized via 'form_delim' and 'cate_delim'.

Usage

```
pw_wjd(
  data,
  cate_level_weights = NULL,
  multi_form_weights = NULL,
  form_delim = "#",
  cate_delim = "_",
  detailed = FALSE,
  squareform = FALSE,
  parallel = FALSE,
  n_threads = 2L,
  quiet = FALSE
)
```

Arguments

<code>data</code>	DataFrame with <code>n</code> rows and <code>m</code> columns indicating there are <code>n</code> languages or dialects to involve in the calculation and there are at most <code>m</code> words to base on, in which the rownames are the language ids.
<code>cate_level_weights</code>	Numeric vector of weights for different levels of categories for a single word form. For example, if a word form is "A_2_a", the first weight applies to "A", the second to "2", and the third to "a". If not provided, default weights are used.
<code>multi_form_weights</code>	Numeric vector of weights for different word forms of a single lexical item (ordered). For example, if a lexical item is "A_2_a#A_3#B_5_c", the first weight applies to "A_2_a", the second to "A_3", etc. If not provided, default weights are used.
<code>form_delim</code>	The delimiter separating different word forms of a single lexical item. Default is "#".
<code>cate_delim</code>	The delimiter separating different levels of categories within a word form. Default is "_".
<code>detailed</code>	Whether to return detailed information.
<code>squareform</code>	Whether to return a dataframe in squareform.
<code>parallel</code>	Whether to parallelize the computation.
<code>n_threads</code>	The number of threads is used to parallelize the computation. Only meaningful if 'parallel' is TRUE.
<code>quiet</code>	Whether to suppress all output messages.

Value

A dataframe in long table form if 'squareform' is FALSE, otherwise in squareform.

Examples

```
df <- as.data.frame(rbind(a=c("A_1#B_2", "C_3"), b=c("A_1", "C_3_x")))
result <- pw_wjd(df, form_delim="#", cate_delim="_")
```

<code>string_edit_dist</code>	<i>Compute edit distance between two strings</i>
-------------------------------	--

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

```
string_edit_dist(
  str1,
  str2,
  cost_mat = NULL,
  delim = "",
  normalize_method = "longest",
  return_alignments = FALSE,
  default_sub_cost = 1,
  default_ins_del_cost = 1,
  quiet = 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. When 'cost_mat' is NULL, general cost rules are used: substitution cost is 'default_sub_cost' if the two symbols are different and 0 if they are the same; insertion and deletion cost is 'default_ins_del_cost'.
delim	The delimiter in 'str1' and 'str2' separating atomic symbols.
normalize_method	Method to normalize the distance. Default is "longest".
return_alignments	Whether to return alignment details
default_sub_cost	Default substitution cost when 'cost_mat' is NULL.
default_ins_del_cost	Default insertion and deletion cost when 'cost_mat' is NULL.
quiet	Whether to suppress all output messages.

Value

A list containing a 'distance' element storing the distance result. If 'return_alignments' is TRUE, then an 'alignments' element is present which is a list of dataframes with each storing a possible best alignment scenario.

See Also

[generate_default_cost_matrix()]

Examples

```
cost.mat <- data.frame()
dist <- string_edit_dist("leaf", "leaves")$distance
res <- string_edit_dist("ph_l_i_z", "p_l_i_s", cost.mat=cost.mat, delim="_")
alignments <- res$alignments
```

Index

`check_cost_mat_symbols`, [2](#)

`generate_default_cost_matrix`, [3](#)

`long2squareform`, [3](#)

`pw_edit_dist`, [4](#)

`pw_pmi_dist`, [6](#)

`pw_wjd`, [7](#)

`string_edit_dist`, [8](#)