Package 'text.alignment'

October 14, 2022

Type Package

Title Text Alignment with Smith-Waterman

Version 0.1.3

Maintainer Jan Wijffels <jwijffels@bnosac.be>

Description Find similarities between texts using the Smith-Waterman algorithm. The algorithm performs local sequence alignment and determines similar regions between two strings.
The Smith-Waterman algorithm is explained in the paper: ``Identification of common molecular subsequences'' by T.F.Smith and M.S.Waterman (1981), available at <doi:10.1016/0022-2836(81)90087-5>.
This package implements the same logic for sequences of words and letters instead of molecular sequences.

License MIT + file LICENSE

URL https://github.com/DIGI-VUB/text.alignment

Encoding UTF-8

Imports Rcpp (>= 0.11.5)

LinkingTo Rcpp

RoxygenNote 7.1.2

Suggests knitr, markdown

VignetteBuilder knitr

NeedsCompilation yes

Author Jan Wijffels [aut, cre, cph] (Rewrite of functionalities from the textreuse R package),
Vrije Universiteit Brussel - DIGI: Brussels Platform for Digital Humanities [cph],
Lincoln Mullen [ctb, cph]

Repository CRAN

Date/Publication 2022-08-17 10:50:02 UTC

R topics documented:

`smith_waterman` *Align text using Smith-Waterman*

Description

Align text using the Smith-Waterman algorithm. The Smith–Waterman algorithm performs local sequence alignment. It finds similar regions between two strings.
Similar regions are a sequence of either characters or words which are found by matching the characters or words of 2 sequences of strings.
If the word/letter is the same in each text, the alignment score is increased with the match score, while if they are not the same the local alignment score drops by the gap score. If one of the 2 texts contains extra words/letters, the score drops by the mismatch score.

Usage

```
smith_waterman(
  a,
  b,
  type = c("characters", "words"),
  match = 2L,
  mismatch = -1L,
  gap = -1L,
  lower = TRUE,
  similarity = function(x, y) ifelse(x == y, 2L, -1L),
  tokenizer,
  collapse,
  edit_mark = "#",
  implementation = c("R", "Rcpp"),
  ...
)
```

Arguments

`a`	a character string of length one
`b`	a character string of length one
`type`	either 'characters' or 'words' indicating to align based on a sequence of characters or a sequence of words. Defaults to 'characters'.

match	integer value of a score to assign a match (a letter/word from a and b which are the same during alignment). This value should be bigger than zero. Defaults to 2.
mismatch	integer value of a score to assign a mismatch (leave out 1 word / 1 letter from 1 of the 2 input strings during alignment). This value should be smaller or equal to zero. Defaults to -1.
gap	integer value of a score to assign a gap (drop 1 word / letter from each of the 2 input strings during alignment). This value should be smaller or equal to zero. Defaults to -1.
lower	logical indicating to lowercase text before doing the alignment. Defaults to TRUE.
similarity	optionally, a function to compare 2 characters or words. This function should have 2 arguments x and y with the 2 letters / words to compare and should return 1 number indicating the similarity between x and y. See the examples.
tokenizer	a function to tokenise text into either a sequence of characters or a sequence of words. Defaults to `tokenize_letters` if type is `'characters'` and `tokenize_spaces_punct` if type is `'words'`
collapse	separator used to combined characters / words back together in the output. Defaults to '' for type 'characters' and a space for type 'words'
edit_mark	separator to indicated a gap/mismatch between sequences. Defaults to the hashtag symbol.
implementation	either 'R' or 'Rcpp' indicating to perform the alignment in Rcpp or with plain R code. Defaults to 'R'.
...	extra arguments passed on to `tokenizer`

Details

The code uses similar code as the `textreuse::local_align` function and also allows to align character sequences next to aligning word sequences

Value

an object of class smith_waterman which is a list with elements

- type: The alignment `type`
- edit_mark:The `edit_mark`
- sw: The Smith-Waterman local alignment score
- similarity: Score between 0 and 1, calculated as the Smith-Waterman local alignment score / (the number of letters/words in the shortest text times the match weight)
- weights: The list of weights provided to the function: match, mismatch and gap
- matches: The number of matches found during alignment
- mismatches: The number of mismatches found during alignment
- a: A list with alignment information from the text provided in `a`. The list elements documented below

- b: A list with alignment information from the text provided in b. The list elements documented below

Elements a and b are both lists which contain

- text: The provided character string of either a or b
- tokens: A character vector of the tokenised texts of a or b
- n: The length of `tokens`
- similarity: The similarity to a calculated as the Smith-Waterman local alignment score / (the number of letters/words in the a or b text times the match weight)
- alignment: A list with the following elements
  - text: The aligned text from either a or b where gaps/mismatches are filled up with the `edit_mark` symbol
  - tokens: The character vector of tokens which form the aligned `text`
  - n: The length of the aligned `text`
  - gaps: The number of gaps during alignment
  - from: The starting position in the full tokenised `tokens` element from either a or b where the aligned text is found. See the example.
  - to: The end position in the full tokenised `tokens` element from either a or b where the aligned text is found. See the example.

See Also

https://en.wikipedia.org/wiki/Smith-Waterman_algorithm

Examples

```
## align sequence of letters
smith_waterman("Joske Vermeulen", "Jiske Vermoelen")
smith_waterman("Joske Vermeulen", "Ik zoek naar Jiske Versmoelen, waar is die te vinden")
smith_waterman("Joske", "Jiske")
smith_waterman("Joske", "Jiske",
             similarity = function(x, y) ifelse(x == y | (x == "o" & y == "i"), 2L, -1L))

## align sequence of words
a <- "The answer is blowin' in the wind."
b <- "As the Bob Dylan song says, the answer is blowing in the wind."
smith_waterman(a, b)
smith_waterman(a, b, type = "characters")
smith_waterman(a, b, type = "words")
smith_waterman(a, b, type = "words", similarity = function(x, y) adist(x, y))
smith_waterman(a, b, type = "words",
               tokenizer = function(x) unlist(strsplit(x, "[[:space:]]")))
x <- smith_waterman(a, b, type = "words")
x$b$tokens[x$b$alignment$from:x$b$alignment$to]

# examples on aligning text files
a <- system.file(package = "text.alignment", "extdata", "example1.txt")
a <- readLines(a)
```

```
a <- paste(a, collapse = "\n")
b <- system.file(package = "text.alignment", "extdata", "example2.txt")
b <- readLines(b)
b <- paste(b, collapse = "\n")
smith_waterman(a, b, type = "characters")
smith_waterman(a, b, type = "words")
smith_waterman("Gistel Hof", b, type = "characters")
smith_waterman("Bailiestraat", b, type = "characters")
smith_waterman("Lange rei", b, type = "characters")

# examples on extracting where elements were found
x <- smith_waterman("Lange rei", b)
x$b$tokens[x$b$alignment$from:x$b$alignment$to]
as.data.frame(x)
as.data.frame(x, alignment_id = "alignment-ab")

x <- lapply(c("Lange rei", "Gistel Hof", NA, "Test"), FUN = function(a){
  x <- smith_waterman(a, b)
  x <- as.data.frame(x)
  x
})
x <- do.call(rbind, x)
x
```

`smith_waterman_misaligned` *Extract misaligned elements*

Description

Extract misaligned elements from the Smith-Waterman alignment, namely

- before_alignment: Sections in a or b which were occurring before the alignment
- wrong_alignment: Sections in a or b which were mismatched in the alignment
- after_alignment: Sections in a or b which were occurring after the alignment

Usage

```
smith_waterman_misaligned(x, type = c("a", "b"))
```

Arguments

x	an object of class `smith_waterman` as returned by `smith_waterman`
type	either 'a' or 'b' indicating to return elements misaligned from a or from b

Value

a list of character vectors of misaligned elements

- before_alignment: Sections in a or b which were occurring before the alignment
- wrong_alignment: Sections in a or b which were mismatched in the alignment
- after_alignment: Sections in a or b which were occurring after the alignment
- combined: The combination of `before_alignment`, `wrong_alignment` and `after_alignment`
- partial: Logical, where TRUE indicates that there is at least a partial alignment and FALSE indicating no alignment between a and b was done (alignment score of 0)

Examples

```
sw <- smith_waterman("ab test xy", "cd tesst ab", type = "characters")
sw
misses <- smith_waterman_misaligned(sw, type = "a")
str(misses)
misses <- smith_waterman_misaligned(sw, type = "b")
str(misses)

a <- system.file(package = "text.alignment", "extdata", "example1.txt")
a <- readLines(a)
a <- paste(a, collapse = "\n")
b <- system.file(package = "text.alignment", "extdata", "example2.txt")
b <- readLines(b)
b <- paste(b, collapse = "\n")
sw <- smith_waterman(a, b, type = "characters")
smith_waterman_misaligned(sw, type = "a")
smith_waterman_misaligned(sw, type = "b")
```

---

smith_waterman_pairwise

Perform multiple alignments using Smith-Waterman

Description

Utility function to perform all pairwise combinations of alignments between text.

Usage

```
smith_waterman_pairwise(a, b, FUN = identity, ...)
```

Arguments

a	a data.frame with columns doc_id and text. Or a character vector where the names of the character vector respresent a doc_id and the character vector corresponds to the text.

b	a data.frame with columns doc_id and text. Or a character vector where the names of the character vector respresent a doc_id and the character vector corresponds to the text.
FUN	a function to apply on an object of class smith_waterman which has done the pairwise alignment. Defaults to identity. Other options are as.data.frame or your own function. See the examples.
...	other arguments passed on to smith_waterman

Value

a list of pairwise Smith-Waterman comparisons after which the FUN argument is applied on all of these pairwise alignments. The output of the result of FUN is enriched by adding a list element a_doc_id and b_doc_id which correspond to the doc_id's provided in a and b and which can be used in order to identify the match.

See Also

smith_waterman

Examples

```
x <- data.frame(doc_id = c(1, 2),
                text = c("This is some text", "Another set of texts."),
                stringsAsFactors = FALSE)
y <- data.frame(doc_id = c(1, 2, 3),
                text = c("were as some thing", "else, another set", NA_character_),
                stringsAsFactors = FALSE)
alignments <- smith_waterman_pairwise(x, y)
alignments
alignments <- smith_waterman_pairwise(x, y, FUN = as.data.frame)
do.call(rbind, alignments)
alignments <- smith_waterman_pairwise(x, y,
                                      FUN = function(x) list(sim = x$similarity))
do.call(rbind, alignments)

x <- c("1" = "This is some text", "2" = "Another set of texts.")
y <- c("1" = "were as some thing", "2" = "else, another set", "3" = NA_character_)
alignments <- smith_waterman_pairwise(x, y)
```

---

tokenize_letters *Tokenise text into a sequence of characters*

Description

Tokenise text into a sequence of characters

Usage

```
tokenize_letters(x)
```

Arguments

x	a character string of length 1

Value

a character vector with the sequence of characters in x

See Also

strsplit

Examples

```
tokenize_letters("This function just chunks up text in letters\nOK?")
tokenize_letters("Joske  Vermeulen")
```

---

tokenize_spaces_punct *Tokenise text into a sequence of words*

Description

Tokenise text into a sequence of words. The function uses strsplit to split text into words by using the [:space:] and [:punct:] character classes.

Usage

```
tokenize_spaces_punct(x)
```

Arguments

x	a character string of length 1

Value

a character vector with the sequence of words in x

See Also

strsplit

Examples

```
tokenize_spaces_punct("This just splits. Text.alongside\nspaces right?")
tokenize_spaces_punct("Also .. multiple punctuations or ??marks")
tokenize_spaces_punct("Joske  Vermeulen")
```

Index