Package 'lexRankr'

July 22, 2025

Type Package

Title Extractive Summarization of Text with the LexRank Algorithm

Version 0.5.2

Author Adam Spannbauer [aut, cre], Bryan White [ctb]

Maintainer Adam Spannbauer <spannbaueradam@gmail.com>

Description An R implementation of the LexRank algorithm described by G. Erkan and D. R. Radev (2004) <DOI:10.1613/jair.1523>.

License MIT + file LICENSE

URL https://github.com/AdamSpannbauer/lexRankr/

BugReports https://github.com/AdamSpannbauer/lexRankr/issues/

LazyData TRUE

RoxygenNote 6.1.1

Imports SnowballC, igraph, Rcpp

Depends R (>= 2.10)

LinkingTo Rcpp

Suggests covr, testthat, R.rsp

VignetteBuilder R.rsp

Encoding UTF-8

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-03-17 21:00:03 UTC

Contents

`bind_lexrank_` *Bind lexrank scores to a dataframe of text*

Description

Bind lexrank scores to a dataframe of sentences or to a dataframe of tokens with sentence ids

Usage

```
bind_lexrank_(tbl, text, doc_id, sent_id = NULL, level = c("sentences",
  "tokens"), threshold = 0.2, usePageRank = TRUE, damping = 0.85,
  continuous = FALSE, ...)

bind_lexrank(tbl, text, doc_id, sent_id = NULL, level = c("sentences",
  "tokens"), threshold = 0.2, usePageRank = TRUE, damping = 0.85,
  continuous = FALSE, ...)
```

Arguments

`tbl`	dataframe containing column of sentences to be lexranked
`text`	name of column containing sentences or tokens to be lexranked
`doc_id`	name of column containing document ids corresponding to `text`
`sent_id`	Only needed if `level` is "tokens". name of column containing sentence ids corresponding to `text`
`level`	the parsed level of the text column to be lexranked. i.e. is `text` a column of "sentences" or "tokens"? The "tokens" level is provided to allow users to implement custom tokenization. Note: even if the input `level` is "tokens" lexrank scores are assigned at the sentence level.
`threshold`	The minimum simililarity value a sentence pair must have to be represented in the graph where lexRank is calculated.
`usePageRank`	`TRUE` or `FALSE` indicating whether or not to use the page rank algorithm for ranking sentences. If `FALSE`, a sentences unweighted centrality will be used as the rank. Defaults to `TRUE`.
`damping`	The damping factor to be passed to page rank algorithm. Ignored if `usePageRank` is `FALSE`.
`continuous`	`TRUE` or `FALSE` indicating whether or not to use continuous LexRank. Only applies if `usePageRank==TRUE`. If `TRUE`, `threshold` will be ignored and lexRank will be computed using a weighted graph representation of the sentences. Defaults to `FALSE`.

... tokenizing options to be passed to lexRankr::tokenize. Ignored if `level` is "sentences"

Value

A dataframe with an additional column of lexrank scores (column is given name lexrank)

Examples

```
df <- data.frame(doc_id = 1:3,
                 text = c("Testing the system. Second sentence for you.",
                          "System testing the tidy documents df.",
                          "Documents will be parsed and lexranked."),
                 stringsAsFactors = FALSE)

## Not run:
library(magrittr)

df %>%
  unnest_sentences(sents, text) %>%
  bind_lexrank(sents, doc_id, level = "sentences")

df %>%
  unnest_sentences(sents, text) %>%
  bind_lexrank_("sents", "doc_id", level = "sentences")

df <- data.frame(doc_id  = c(1, 1, 1, 1, 1, 1, 1, 2, 2, 2,
                             2, 2, 2, 3, 3, 3, 3, 3, 3),
                 sent_id = c(1, 1, 1, 2, 2, 2, 2, 1, 1, 1,
                             1, 1, 1, 1, 1, 1, 1, 1, 1),
                 tokens = c("testing", "the", "system", "second",
                            "sentence", "for", "you", "system",
                            "testing", "the", "tidy", "documents",
                            "df", "documents", "will", "be", "parsed",
                            "and", "lexranked"),
                 stringsAsFactors = FALSE)

df %>%
  bind_lexrank(tokens, doc_id, sent_id, level = 'tokens')

## End(Not run)
```

`lexRank` *Extractive text summarization with LexRank*

Description

Compute LexRanks from a vector of documents using the page rank algorithm or degree centrality the methods used to compute lexRank are discussed in "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization."

Usage

```
lexRank(text, docId = "create", threshold = 0.2, n = 3,
  returnTies = TRUE, usePageRank = TRUE, damping = 0.85,
  continuous = FALSE, sentencesAsDocs = FALSE, removePunc = TRUE,
  removeNum = TRUE, toLower = TRUE, stemWords = TRUE,
  rmStopWords = TRUE, Verbose = TRUE)
```

Arguments

text	A character vector of documents to be cleaned and processed by the LexRank algorithm
docId	A vector of document IDs with length equal to the length of text. If docId == "create" then doc IDs will be created as an index from 1 to n, where n is the length of text.
threshold	The minimum simil value a sentence pair must have to be represented in the graph where lexRank is calculated.
n	The number of sentences to return as the extractive summary. The function will return the top n lexRanked sentences. See returnTies for handling ties in lexRank.
returnTies	TRUE or FALSE indicating whether or not to return greater than n sentence IDs if there is a tie in lexRank. If TRUE, the returned number of sentences will not be limited to n, but rather will return every sentence with a top 3 score. If FALSE, the returned number of sentences will be <=n. Defaults to TRUE.
usePageRank	TRUE or FALSE indicating whether or not to use the page rank algorithm for ranking sentences. If FALSE, a sentences unweighted centrality will be used as the rank. Defaults to TRUE.
damping	The damping factor to be passed to page rank algorithm. Ignored if usePageRank is FALSE.
continuous	TRUE or FALSE indicating whether or not to use continuous LexRank. Only applies if usePageRank==TRUE. If TRUE, threshold will be ignored and lexRank will be computed using a weighted graph representation of the sentences. Defaults to FALSE.
sentencesAsDocs	TRUE or FALSE, indicating whether or not to treat sentences as documents when calculating tfidf scores for similarity. If TRUE, inverse document frequency will be calculated as inverse sentence frequency (useful for single document extractive summarization).
removePunc	TRUE or FALSE indicating whether or not to remove punctuation from text while tokenizing. If TRUE, punctuation will be removed. Defaults to TRUE.
removeNum	TRUE or FALSE indicating whether or not to remove numbers from text while tokenizing. If TRUE, numbers will be removed. Defaults to TRUE.
toLower	TRUE or FALSE indicating whether or not to coerce all of text to lowercase while tokenizing. If TRUE, text will be coerced to lowercase. Defaults to TRUE.
stemWords	TRUE or FALSE indicating whether or not to stem resulting tokens. If TRUE, the outputted tokens will be tokenized using SnowballC::wordStem(). Defaults to TRUE.

rmStopWords	TRUE, FALSE, or character vector of stopwords to remove from tokens. If TRUE, words in lexRankr::smart_stopwords will be removed prior to stemming. If FALSE, no stopword removal will occur. If a character vector is passed, this vector will be used as the list of stopwords to be removed. Defaults to TRUE.
Verbose	TRUE or FALSE indicating whether or not to cat progress messages to the console while running. Defaults to TRUE.

Value

A 2 column dataframe with columns sentenceId and value. sentence contains the ids of the top n sentences in descending order by value. value contains page rank score (if usePageRank==TRUE) or degree centrality (if usePageRank==FALSE).

References

http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume22/erkan04a-html/erkan04a.html

Examples

```
lexRank(c("This is a test.","Tests are fun.",
"Do you think the exam will be hard?","Is an exam the same as a test?",
"How many questions are going to be on the exam?"))
```

---

lexRankFromSimil *Compute LexRanks from pairwise sentence similarities*

Description

Compute LexRanks from sentence pair similarities using the page rank algorithm or degree centrality the methods used to compute lexRank are discussed in "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization."

Usage

```
lexRankFromSimil(s1, s2, simil, threshold = 0.2, n = 3,
  returnTies = TRUE, usePageRank = TRUE, damping = 0.85,
  continuous = FALSE)
```

Arguments

s1	A character vector of sentence IDs corresponding to the s2 and simil arguments
s2	A character vector of sentence IDs corresponding to the s1 and simil arguments
simil	A numeric vector of similarity values that represents the similarity between the sentences represented by the IDs in s1 and s2.
threshold	The minimum simil value a sentence pair must have to be represented in the graph where lexRank is calculated.

n	The number of sentences to return as the extractive summary. The function will return the top n lexRanked sentences. See returnTies for handling ties in lexRank.
returnTies	TRUE or FALSE indicating whether or not to return greater than n sentence IDs if there is a tie in lexRank. If TRUE, the returned number of sentences will not be limited to n, but rather will return every sentence with a top 3 score. If FALSE, the returned number of sentences will be <=n. Defaults to TRUE.
usePageRank	TRUE or FALSE indicating whether or not to use the page rank algorithm for ranking sentences. If FALSE, a sentences unweighted centrality will be used as the rank. Defaults to TRUE.
damping	The damping factor to be passed to page rank algorithm. Ignored if usePageRank is FALSE.
continuous	TRUE or FALSE indicating whether or not to use continuous LexRank. Only applies if usePageRank==TRUE. If TRUE, threshold will be ignored and lexRank will be computed using a weighted graph representation of the sentences. Defaults to FALSE.

Value

A 2 column dataframe with columns sentenceId and value. sentenceId contains the ids of the top n sentences in descending order by value. value contains page rank score (if usePageRank==TRUE) or degree centrality (if usePageRank==FALSE).

References

http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume22/erkan04a-html/erkan04a.html

Examples

```
lexRankFromSimil(s1=c("d1_1","d1_1","d1_2"), s2=c("d1_2","d2_1","d2_1"), simil=c(.01,.03,.5))
```

---

sentenceParse *Parse text into sentences*

Description

Parse the elements of a character vector into a dataframe of sentences with additional identifiers.

Usage

```
sentenceParse(text, docId = "create")
```

Arguments

text	Character vector to be parsed into sentences
docId	A vector of document IDs with length equal to the length of text. If docId == "create" then doc IDs will be created as an index from 1 to n, where n is the length of text.

Value

A data frame with 3 columns and n rows, where n is the number of sentences found by the routine. Column 1: docId document id for the sentence. Column 2: sentenceId sentence id for the sentence. Column 3: sentence the sentences found in the routine.

Examples

```
sentenceParse("Bill is trying to earn a Ph.D.", "You have to have a 5.0 GPA.")
sentenceParse(c("Bill is trying to earn a Ph.D.", "You have to have a 5.0 GPA."),
              docId=c("d1","d2"))
```

---

sentenceSimil *Compute distance between sentences*

Description

Compute distance between sentences using modified idf cosine distance from "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization". Output can be used as input to lexRankFromSimil.

Usage

```
sentenceSimil(sentenceId, token, docId = NULL, sentencesAsDocs = FALSE)
```

Arguments

sentenceId	A character vector of sentence IDs corresponding to the docId and token arguments
token	A character vector of tokens corresponding to the docId and sentenceId arguments
docId	A character vector of document IDs corresponding to the sentenceId and token arguments. Can be NULL if sentencesAsDocs is TRUE.
sentencesAsDocs	TRUE or FALSE, indicating whether or not to treat sentences as documents when calculating tfidf scores. If TRUE, inverse document frequency will be calculated as inverse sentence frequency (useful for single document extractive summarization)

Value

A 3 column dataframe of pairwise distances between sentences. Columns: sent1 (sentence id), sent2 (sentence id), & dist (distance between sent1 and sent2).

References

http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume22/erkan04a-html/erkan04a.html

Examples

```
sentenceSimil(docId=c("d1","d1","d2","d2"),
              sentenceId=c("d1_1","d1_1","d2_1","d2_1"),
              token=c("i", "ran", "jane", "ran"))
```

`sentenceTokenParse` *Parse text into sentences and tokens*

Description

Parse a character vector of documents into into both sentences and a clean vector of tokens. The resulting output includes IDs for document and sentence for use in other `lexRank` functions.

Usage

```
sentenceTokenParse(text, docId = "create", removePunc = TRUE,
  removeNum = TRUE, toLower = TRUE, stemWords = TRUE,
  rmStopWords = TRUE)
```

Arguments

`text`	A character vector of documents to be parsed into sentences and tokenized.
`docId`	A character vector of document Ids the same length as `text`. If `docId=="create"` document Ids will be created.
`removePunc`	`TRUE` or `FALSE` indicating whether or not to remove punctuation from `text` while tokenizing. If `TRUE`, punctuation will be removed. Defaults to `TRUE`.
`removeNum`	`TRUE` or `FALSE` indicating whether or not to remove numbers from `text` while tokenizing. If `TRUE`, numbers will be removed. Defaults to `TRUE`.
`toLower`	`TRUE` or `FALSE` indicating whether or not to coerce all of `text` to lowercase while tokenizing. If `TRUE`, `text` will be coerced to lowercase. Defaults to `TRUE`.
`stemWords`	`TRUE` or `FALSE` indicating whether or not to stem resulting tokens. If `TRUE`, the outputted tokens will be tokenized using `SnowballC::wordStem()`. Defaults to `TRUE`.
`rmStopWords`	`TRUE`, `FALSE`, or character vector of stopwords to remove from tokens. If `TRUE`, words in `lexRankr::smart_stopwords` will be removed prior to stemming. If `FALSE`, no stopword removal will occur. If a character vector is passed, this vector will be used as the list of stopwords to be removed. Defaults to `TRUE`.

Value

A list of dataframes. The first element of the list returned is the `sentences` dataframe; this dataframe has columns `docId`, `sentenceId`, & `sentence` (the actual text of the sentence). The second element of the list returned is the `tokens` dataframe; this dataframe has columns `docId`, `sentenceId`, & `token` (the actual text of the token).

Examples

```
sentenceTokenParse(c("Bill is trying to earn a Ph.D.", "You have to have a 5.0 GPA."),
                   docId=c("d1","d2"))
```

---

sentence_parser *Utility to parse sentences from text*

Description

Utility to parse sentences from text; created to have a central shared sentence parsing function

Usage

```
sentence_parser(text)
```

Arguments

text	Character vector to be parsed into sentences

Value

A list with length equal to 'length(text)'; list elements are character vectors of text parsed with sentence regex

---

smart_stopwords *SMART English Stopwords*

Description

English stopwords from the SMART information retrieval system (as documented in Appendix 11 of http://jmlr.csail.mit.edu/papers/volume5/lewis04a/)

Usage

```
smart_stopwords
```

Format

a character vector with 571 elements

Source

http://jmlr.csail.mit.edu/papers/volume5/lewis04a/

`tokenize` *Tokenize a character vector Parse the elements of a character vector into a list of cleaned tokens.*

Description

Tokenize a character vector Parse the elements of a character vector into a list of cleaned tokens.

Usage

```
tokenize(text, removePunc = TRUE, removeNum = TRUE, toLower = TRUE,
  stemWords = TRUE, rmStopWords = TRUE)
```

Arguments

`text`	The character vector to be tokenized
`removePunc`	`TRUE` or `FALSE` indicating whether or not to remove punctuation from `text`. If `TRUE`, punctuation will be removed. Defaults to `TRUE`.
`removeNum`	`TRUE` or `FALSE` indicating whether or not to remove numbers from `text`. If `TRUE`, numbers will be removed. Defaults to `TRUE`.
`toLower`	`TRUE` or `FALSE` indicating whether or not to coerce all of `text` to lowercase. If `TRUE`, `text` will be coerced to lowercase. Defaults to `TRUE`.
`stemWords`	`TRUE` or `FALSE` indicating whether or not to stem resulting tokens. If `TRUE`, the outputted tokens will be tokenized using `SnowballC::wordStem()`. Defaults to `TRUE`.
`rmStopWords`	`TRUE`, `FALSE`, or character vector of stopwords to remove. If `TRUE`, words in `lexRankr::smart_stopwords` will be removed prior to stemming. If `FALSE`, no stopword removal will occur. If a character vector is passed, this vector will be used as the list of stopwords to be removed. Defaults to `TRUE`.

Examples

```
tokenize("Mr. Feeny said the test would be on Sat. At least I'm 99.9% sure that's what he said.")
tokenize("Bill is trying to earn a Ph.D. in his field.", rmStopWords=FALSE)
```

`unnest_sentences_` *Split a column of text into sentences*

Description

Split a column of text into sentences

Usage

```
unnest_sentences_(tbl, output, input, doc_id = NULL,
  output_id = "sent_id", drop = TRUE)

unnest_sentences(tbl, output, input, doc_id = NULL,
  output_id = "sent_id", drop = TRUE)
```

Arguments

tbl	dataframe containing column of text to be split into sentences
output	name of column to be created to store parsed sentences
input	name of input column of text to be parsed into sentences
doc_id	column of document ids; if not provided it will be assumed that each row is a different document
output_id	name of column to be created to store sentence ids
drop	whether original input column should get dropped

Value

A data.frame of parsed sentences and sentence ids

Examples

```
df <- data.frame(doc_id = 1:3,
                 text = c("Testing the system. Second sentence for you.",
                          "System testing the tidy documents df.",
                          "Documents will be parsed and lexranked."),
                 stringsAsFactors=FALSE)

unnest_sentences(df, sents, text)
unnest_sentences_(df, "sents", "text")

## Not run:
library(magrittr)

df %>%
  unnest_sentences(sents, text)

## End(Not run)
```

Index