Package 'inzightta'

August 16, 2019

Title iNZight Text Analytics

Version 0.0.0.9000

Description Provides text analytics functions for the importation, analysis, and visualisation of text. This package is designed specifically for output in shiny, with the analytical functions all working well with dplyr tools.

License GPL-3

Encoding UTF-8

LazyData true

Imports readr, tibble, stringr, dplyr, readxl, purrr, tidytext, textstem, magrittr, stats, textrank, lexRankr

RoxygenNote 6.1.1

R topics documented:

`aggregate_sentiment` *Get statistics for sentiment over some group, such as sentence.*

Description

Get statistics for sentiment over some group, such as sentence.

Usage

```
aggregate_sentiment(.data, aggregate_on, statistic)
```

Arguments

`.data`	character vector of words
`aggregate_on`	vector to aggregate .data over; ideally, sentence_id, but could be chapter, document, etc.
`statistic`	function that accepts na.rm argument; e.g. mean, median, sd.

determine_stopwords *determine stopword status*

Description

determine stopword status

Usage

```
determine_stopwords(.data, ...)
```

Arguments

.data	vector of words
...	arguments of get_sw

Value

a [tibble][tibble::tibble-package] equivalent to the input dataframe, with an additional stopword column

format_data *formats imported data into an analysis-ready format*

Description

formats imported data into an analysis-ready format

Usage

```
format_data(data)
```

Arguments

data	a tibble formatted with a text and (optional) group column

Value

a [tibble][tibble::tibble-package] formatted such that columns correspond to identifiers of group, line, sentence, word (groups ignored)

get_bigram *Determine bigrams*

Description

Determine bigrams

Usage

```
get_bigram(.data)
```

Arguments

.data	character vector of words

Value

character vector of bigrams

get_chapters *sections text based on chapters*

Description

sections text based on chapters

Usage

```
get_chapters(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with chapter numbers

get_filetype *Get filetype*

Description

Get filetype

Usage

```
get_filetype(filepath)
```

Arguments

filepath	string filepath of document

Value

filetype (string) - NA if no extension

get_parts *sections text based on parts*

Description

sections text based on parts

Usage

```
get_parts(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with part numbers

get_search *creates a search closure to section text*

Description

creates a search closure to section text

Usage

```
get_search(search)
```

Arguments

search	a string regexp for the term to seperate on, e.g. "Chapter"

Value

closure over search expression

get_sections *sections text based on sections*

Description

sections text based on sections

Usage

```
get_sections(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with section numbers

get_sw *Gets stopwords from a default list and user-provided list*

Description

Gets stopwords from a default list and user-provided list

Usage

```
get_sw(lexicon = "snowball", addl = NA)
```

Arguments

lexicon	a string name of a stopword list, one of "smart", "snowball", or "onix"
addl	user defined character vector of additional stopwords, each element being a stopword

Value

a [tibble][tibble::tibble-package] with one column named "word"

get_valid_input *helper function to get valid input (recursively)*

Description

helper function to get valid input (recursively)

Usage

```
get_valid_input(options, init = TRUE)
```

Arguments

options	vector of options that valid input should be drawn from
init	whether this is the initial attempt, used only as recursive information

Value

readline output that exists in the vector of options

`ifexp` *scheme-like if expression, without restriction of returning same-size table of .test, as ifelse() does*

Description

scheme-like if expression, without restriction of returning same-size table of .test, as ifelse() does

Usage

```
ifexp(.test, true, false)
```

Arguments

`.test`	predicate to test
`true`	expression to return if .test evals to TRUE
`false`	expression to return if .test evals to TRUE

Value

either true or false

`import_base_file` *Base case for file import*

Description

Base case for file import

Usage

```
import_base_file(filepath)
```

Arguments

`filepath`	string filepath of file for import

Value

imported file with document id

import_csv *Import csv file*

Description

Import csv file

Usage

```
import_csv(filepath)
```

Arguments

filepath	a string indicating the relative or absolute filepath of the file to import

Value

a [tibble][tibble::tibble-package] of each row corrresponding to a line of the text file, with the column named "text"

import_excel *Import excel file*

Description

Import excel file

Usage

```
import_excel(filepath)
```

Arguments

filepath	a string indicating the relative or absolute filepath of the file to import

Value

a [tibble][tibble::tibble-package] of each row corrresponding to a line of the text file, with the column named "text"

`import_files` *Import any number of files*

Description

Import any number of files

Usage

```
import_files(filepaths)
```

Arguments

`filepaths`	char vector of filepaths

Value

a [tibble][tibble::tibble-package] imported files with document id

`import_txt` *Import text file*

Description

Import text file

Usage

```
import_txt(filepath)
```

Arguments

`filepath`	a string indicating the relative or absolute filepath of the file to import

Value

a [tibble][tibble::tibble-package] of each row corrresponding to a line of the text file, with the column named "text"

index_bigram *get bigram at index i of list1 & 2*

Description

get bigram at index i of list1 & 2

Usage

```
index_bigram(i, list1, list2)
```

Arguments

i	numeric index to attain bigram at
list1	list or vector for first bigram token
list2	list or vector for second bigram token

Value

bigram of list1 and list2 at index i, skipping NA's

keywords_tr *Determine textrank score for vector of words*

Description

Determine textrank score for vector of words

Usage

```
keywords_tr(.data)
```

Arguments

.data	character vector of words

Value

vector of scores for each word

key_sentences *get score for key sentences as per Lexrank*

Description

get score for key sentences as per Lexrank

Usage

```
key_sentences(.data, aggregate_on)
```

Arguments

.data	character vector of words
aggregate_on	vector to aggregate .data over; ideally, sentence_id

table_textcol *Interactively determine and automatically mark the text column of a table*

Description

Interactively determine and automatically mark the text column of a table

Usage

```
table_textcol(data)
```

Arguments

data	dataframe with column requiring marking

Value

same dataframe with text column renamed to "text"

term_count *Determine the number of terms at each aggregate level*

Description

Determine the number of terms at each aggregate level

Usage

```
term_count(.data, aggregate_on)
```

Arguments

.data	character vector of terms
aggregate_on	vector to split .data on for insight

Value

vector of number of terms for each aggregate level, same length as .data

term_freq *Determine term frequency*

Description

Determine term frequency

Usage

```
term_freq(.data)
```

Arguments

.data	character vector of terms

Value

numeric vector of term frequencies

text_prep — *takes imported one-line-per-row data and prepares it for later analysis*

Description

takes imported one-line-per-row data and prepares it for later analysis

Usage

```
text_prep(.data, lemmatize = TRUE, stopwords = TRUE,
  sw_lexicon = "snowball", addl_stopwords = NA)
```

Arguments

.data	tibble with one line of text per row
lemmatize	boolean, whether to lemmatize or not
stopwords	boolean, whether to remove stopwords or not
sw_lexicon	string, lexicon with which to remove stopwords
addl_stopwords	char vector of user-supplied stopwords

Value

a [tibble][tibble::tibble-package] with one token per line, stopwords removed leaving NA values, column for analysis named "text"

ungroup_by — *helper function to ungroup for dplyr. functions equivalently to group_by() but with standard (string) evaluation*

Description

helper function to ungroup for dplyr. functions equivalently to group_by() but with standard (string) evaluation

Usage

```
ungroup_by(x, ...)
```

Arguments

x	tibble to perform function on
...	string of groups to ungroup on

Value

x with ... no longer grouped upon

`word_sentiment` *Determine sentiment of words*

Description

Determine sentiment of words

Usage

```
word_sentiment(.data, lexicon = "afinn")
```

Arguments

`.data`	vector of words
`lexicon`	sentiment lexicon to use, based on the corpus provided by tidytext

Value

vector with sentiment score of each word in the vector

Index