Package 'inzightta'

September 24, 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 the shiny program, 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, ggpage, ggplot2, forcats, shiny, ggwordcloud

RoxygenNote 6.1.1

NeedsCompilation no

Author Jason Cairns [aut, cre]

Maintainer Jason Cairns <jcai849@aucklanduni.ac.nz>

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, lexicon = "afinn",
  statistic = mean)
```

Arguments

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

Value

sentiment of same length as input vector aggregated over the aggregate_on vector

bind_aggregation *bind aggregate terms together*

Description

bind aggregate terms together

Usage

```
bind_aggregation(data, aggregate_on)
```

Arguments

data	vector of terms
aggregate_on	vector of aggregations

Value

data with every aggregation bound, as in a sentence

concat_walk *concat list 1 and 2, moving past NA values*

Description

concat list 1 and 2, moving past NA values

Usage

```
concat_walk(list1, list2)
```

Arguments

list1	list or vector for first bigram token
list2	list or vector for second bigram token

Value

paste of list1 and list2, skipping NA's

concat_walk_i *concat list 1 and 2 at index, skipping NA values*

Description

concat list 1 and 2 at index, skipping NA values

Usage

```
concat_walk_i(i, list1, list2)
```

Arguments

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

Value

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

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

dist_density *output a histogram of the distribution of some function of words*

Description

output a histogram of the distribution of some function of words

Usage

```
dist_density(.data, col_name)
```

Arguments

.data	the standard dataframe, modified so the last column is the output of some insight function (eg. output from term_freq)
col_name	symbol name of the column insight was performed on

dist_hist *output a histogram of the distribution of some function of words*

Description

output a histogram of the distribution of some function of words

Usage

```
dist_hist(.data, col_name)
```

Arguments

.data	the standard dataframe, modified so the last column is the output of some insight function (eg. output from term_freq)
col_name	symbol name of the column insight was performed on

format_data *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

```
format_data(.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"

get_aggregate_insight *perform group-aware aggregate operations on the data*

Description

perform group-aware aggregate operations on the data

Usage

```
get_aggregate_insight(.data, operations, aggregate_on, ...)
```

Arguments

.data	dataframe of terms as per output of format_data
operations	character vector of operations to perform
aggregate_on	character name of the column to perform aggregate operations on
...	additional arguments to the operation - only sensible for singular operations

Value

.data with operation columns added

get_bigram *Determine bigrams*

Description

Determine bigrams

Usage

```
get_bigram(.data)
```

Arguments

.data	character vector of words

Value

character vector of bigrams

get_books *sections text based on book*

Description

sections text based on book

Usage

```
get_books(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with book numbers

get_cantos *sections text based on cantos*

Description

sections text based on cantos

Usage

```
get_cantos(.data)
```

Arguments

.data	vector to section

Value

vector of same length as .data with canto numbers

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_ngram *Returns the n-grams, skipping NA values*

Description

Returns the n-grams, skipping NA values

Usage

```
get_ngram(.data, n)
```

Arguments

.data	vector to get n-grams from
n	number of n-grams to attain

Value

n-gram vector without NA values

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_term_insight *perform group-aware term operations on the data*

Description

perform group-aware term operations on the data

Usage

```
get_term_insight(.data, operations, ...)
```

Arguments

.data	dataframe of terms as per output of format_data
operations	character vector of term operations to perform
...	additional arguments to the operation - only sensible for singular operations

Value

.data with operation columns added

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

get_vis *create a group-aware visualisation*

Description

create a group-aware visualisation

Usage

```
get_vis(.data, vis, col, facet_by = "", scale_fixed = TRUE, ...)
```

Arguments

.data	the standard dataframe, modified so the last column is the output of some insight function (eg. output from term_freq)
vis	character name of visualisation function
col	character name of the column to get insight from
facet_by	character name of the column to facet by
scale_fixed	force scales to be fixed in a facet
...	additional arguments to the visualisation

`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"

keywords_tr *Determine textrank score for vector of words*

Description

Determine textrank score for vector of words

Usage

```
keywords_tr(.data, summ_method)
```

Arguments

.data	character vector of words
summ_method	method to use for summarisation: textrank or lexrank. Doesn't do anything yet

Value

vector of scores for each word

key_aggregates *get score for key sentences as per Lexrank*

Description

get score for key sentences as per Lexrank

Usage

```
key_aggregates(.data, aggregate_on, summ_method)
```

Arguments

.data	character vector of words
aggregate_on	vector to aggregate .data over; ideally, sentence_id
summ_method	method to use for summarisation: textrank or lexrank. Doesn't do anything yet

Value

lexrank scores of aggregates

ma_term_sentiment *Determine the lagged sentiment of terms*

Description

Determine the lagged sentiment of terms

Usage

```
ma_term_sentiment(.data, lexicon = "afinn", lag = 10,
  statistic = mean)
```

Arguments

.data	vector of terms
lexicon	sentiment lexicon to use, based on the corpus provided by tidytext
lag	how many (inclusive) terms to compute statistic over
statistic	base statistic used to summarise the data, capable of taking an na.rm argument

Value

vector with lagged sentiment score of each term in the input vector

ngram_freq *NOT FOR PRODUCTION - STILL IN TESING. Returns the count of n-grams, skipping NA values*

Description

NOT FOR PRODUCTION - STILL IN TESING. Returns the count of n-grams, skipping NA values

Usage

```
ngram_freq(.data, n)
```

Arguments

.data	vector to get n-grams from
n	number of n-grams to attain

Value

count of each associated n-gram

score_barplot *output a ggplot column graph of the top texts from some insight function*

Description

output a ggplot column graph of the top texts from some insight function

Usage

```
score_barplot(.data, y, n = 15, x = text, desc = FALSE)
```

Arguments

.data	a dataframe containing "text" and insight columns as per the output of the get_(term|aggregate)_insight wrapper function
y	symbol name of the column insight was outputted to
n	number of bars to display
x	symbol name of column for insight labels
desc	bool: show bars in descending order

score_wordcloud *output a ggplot wordcloud graph of the top texts from some insight function*

Description

output a ggplot wordcloud graph of the top texts from some insight function

Usage

```
score_wordcloud(.data, y, n = 15, x = text, shape = "circle")
```

Arguments

.data	a dataframe containing "text" and insight columns as per the output of the get_(term|aggregate)_insight wrapper function
y	symbol name of the column insight was outputted to
n	number of words to display
x	symbol name of column for insight labels
shape	character: shape of the wordcloud

section *Adds section column to dataframe*

Description

Adds section column to dataframe

Usage

```
section(.data, section_by)
```

Arguments

.data	dataframe formatted as per output of prep process
section_by	character name of what to section over

Value

input dataframe with additional section column

shorten *Shorten some text up to n characters*

Description

Shorten some text up to n characters

Usage

```
shorten(.data, n)
```

Arguments

.data	character vector
n	wrap length of text

Value

shortened form of .data

struct_pageview *Colours a ggpage based on an insight function*

Description

Colours a ggpage based on an insight function

Usage

```
struct_pageview(.data, col_name, num_terms, term_index, palette)
```

Arguments

.data	a dataframe containing "word" and insight columns as per the output of the get_(term|aggregate)_insight wrapper function
col_name	symbol name of the insight column intended to colour plot
num_terms	the number of terms to visualise
term_index	which term to start the visualisation from
palette	determine coloration of palette (not yet implemented)

Value

ggplot object as per ggpage

struct_time_series *output a ggplot time series plot of some insight function*

Description

output a ggplot time series plot of some insight function

Usage

```
struct_time_series(.data, y)
```

Arguments

.data	a dataframe containing "text" and insight columns as per the output of the get_(term|aggregate)_insight wrapper function
y	symbol name of the column insight was outputted to

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_cooccurance *Determine term cooccurances - extremely slow*

Description

Determine term cooccurances - extremely slow

Usage

```
term_cooccurance(.data, term, aggregate_on)
```

Arguments

.data	character vector of terms
term	character to find correlations with
aggregate_on	vector to aggregate .data over; ideally, sentence_id, but could be chapter, document, etc.

Value

numeric vector of term correlations as per phi_coef

term_corr *Determine term correlations - extremely slow*

Description

Determine term correlations - extremely slow

Usage

```
term_corr(.data, term, aggregate_on)
```

Arguments

.data	character vector of terms
term	character to find correlations with
aggregate_on	vector to aggregate .data over; ideally, sentence_id, but could be chapter, document, etc.

Value

numeric vector of term correlations as per phi_coef

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

term_sentiment *Determine sentiment of terms*

Description

Determine sentiment of terms

Usage

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

Arguments

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

Value

vector with sentiment score of each word in the vector

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

Index