Package 'opitools'

July 29, 2021

Type Package

Title Analyzing the Opinions in a Big Text Document

Version 1.8.0

Author Monsuru Adepeju [cre, aut],

Maintainer Monsuru Adepeju <monsuur2010@yahoo.com>

Description Designed for performing impact analysis of opinions in a digital text document (DTD). The package allows a user to assess the extent to which a theme or subject within a document impacts the overall opinion expressed in the document. The package can be applied to a wide range of opinion-based DTD, including commentaries on social media platforms (such as 'Facebook', 'Twitter' and 'Youtube'), online products reviews, and so on. The utility of 'opitools' was originally demonstrated in Adepeju and Jimoh (2021) <doi:10.31235/osf.io/c32qh> in the assessment of COVID-19 impacts on neighbourhood policing using Twitter data. Further examples can be found in the vignette of the package.

Language en-US

License GPL-3

URL https://github.com/MAnalytics/opitools

BugReports https://github.com/MAnalytics/opitools/issues/1

Depends R (>= 4.0.0)

Encoding UTF-8

LazyData true

Imports ggplot2, tibble, tidytext, magrittr, dplyr, stringr, purrr, tidyr, likert, tm, wordcloud2, forcats, cowplot

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat, rvest, kableExtra

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2021-07-29 15:30:02 UTC

R topics documented:

`covid_theme` *keywords relating to COVID-19 pandemics*

Description

A list of keywords relating to the COVID-19 pandemic

Usage

```
covid_theme
```

Format

A dataframe containing one variable:

- keys: list of keywords

debate_dtd *Comments on a video of a political debate.*

Description

A DTD containing individual comments on a video showing the first debate between two US presidential nominees (Donald Trump and Hillary Clinton) in Sept. 2016. (Credit: NBC News).

Usage

```
debate_dtd
```

Format

A dataframe containing one variable

- text: individual text records

Details

The DTD only include the comments within the first 24hrs in which the video was posted. All individual comments in which the names of both candidates are mentioned are filtered out.

opi_impact *Statistical assessment of impacts of a specified theme from a DTD.*

Description

This function assesses the impacts of a theme (or subject) on the overall opinion computed for a DTD Different themes in a DTD can be identified by the keywords used in the DTD. These keywords (or words) can be extracted by any analytical means available to the users, e.g. `word_imp` function. The keywords must be collated and supplied this function through the `theme_keys` argument (see below).

Usage

```
opi_impact(textdoc, theme_keys=NULL, metric = 1,
fun = NULL, nsim = 99, alternative="two.sided",
quiet=TRUE)
```

Arguments

textdoc	An n x 1 list (dataframe) of individual text records, where n is the total number of individual records.
theme_keys	(a list) A one-column dataframe (of any number of length) containing a list of keywords relating to the theme or secondary subject to be investigated. The keywords can also be defined as a vector of characters.
metric	(an integer) Specify the metric to utilize for the calculation of opinion score. Default: 1. See detailed documentation in the opi_score function.
fun	A user-defined function given that parameter metric (above) is set equal to 5. See detailed documentation in the opi_score function.
nsim	(an integer) Number of replicas (ESD) to generate. See detailed documentation in the opi_sim function. Default: 99.
alternative	(a character) Default: "two.sided", indicating a two-tailed test. A user can override this default value by specifying "less" or "greater" to run the analysis as one-tailed test when the observed score is located at the lower or upper regions of the expectation distribution, respectively. Note: for metric=1, the alternative parameter should be set equal to "two.sided" because the opinion score is bounded by both positive and negative values. For an opinion score bounded by positive values, such as when metric = 2, 3 or 4, the alternative parameter should be set as "greater", and set as "less" otherwise. If metric parameter is set equal to 5, with a user-defined opinion score function (i.e. fun not NULL), the user is required to determine the limits of the opinion scores, and set the alternative argument appropriately.
quiet	(TRUE or FALSE) To suppress processing messages. Default: TRUE.

Details

This function calculates the statistical significance value (p-value) of an opinion score by comparing the observed score (from the opi_score function) with the expected scores (distribution) (from the opi_sim function). The formula is given as p = (S.beat+1)/(S.total+1), where S_total is the total number of replicas (nsim) specified, S.beat is number of replicas in which their expected scores are than the observed score (See further details in Adepeju and Jimoh, 2021).

Value

Details of statistical significance of impacts of a secondary subject B on the opinion concerning the primary subject A.

References

(1) Adepeju, M. and Jimoh, F. (2021). An Analytical Framework for Measuring Inequality in the Public Opinions on Policing – Assessing the impacts of COVID-19 Pandemic using Twitter Data. https://doi.org/10.31235/osf.io/c32qh

Examples

```
# Application in marketing:

#`data` -> 'reviews_dtd'
#`theme_keys` -> 'refreshment_theme'

#RQ2a: "Do the refreshment outlets impact customers'
#opinion of the services at the Piccadilly train station?"

##execute function
output <- opi_impact(textdoc = reviews_dtd,
          theme_keys=refreshment_theme, metric = 1,
          fun = NULL, nsim = 99, alternative="two.sided",
          quiet=TRUE)

#To print results
print(output)

#extracting the pvalue in order to answer RQ2a
output$pvalue
```

`opi_score` *Opinion score of a digital text document (DTD)*

Description

Given a DTD, this function computes the overall opinion score based on the proportion of text records classified as expressing positive, negative or a neutral sentiment. The function first transforms the text document into a tidy-format dataframe, described as the observed sentiment document (OSD) (Adepeju and Jimoh, 2021), in which each text record is assigned a sentiment class based on the summation of all sentiment scores expressed by the words in the text record.

Usage

```
opi_score(textdoc, metric = 1, fun = NULL)
```

Arguments

`textdoc`	An `n` x `1` list (dataframe) of individual text records, where `n` is the total number of individual records.
`metric`	(an integer) Specify the metric to utilize for the calculation of opinion score. Valid values include `1, 2, ..., 5`. Assuming `P`, `N` and `O` represent positive, negative, and neutral record sentiments, respectively, the followings are the details of the opinion score function represented by the numerical arguments above: `1`: Polarity (percentage difference) `((P -N)/(P + N))*100`, (Bound: -100%, +100%);

2: Polarity (proportional difference) ((abs(P -N) / (P + N + O))*100, (Bound: 0, +100%); 3: Positivity (P/ (P + N + O))*100, (Bound: 0, +100%); 4: Negativity (N / (P + N + O))*100, (Bound: 0, +100%) (Malshe, A. 2019; Lowe et al. 2011). 5: To pass a user-defined opinion score function (also see the fun parameter below.

fun — A user-defined function given that metric parameter (above) is set equal to 5. For example, given a defined opinion score function myfun <- function(P, N, O){ ("some tasks to do"); return("a value")}, the input argument of fun parameter then becomes fun = myfun. Default: NULL.

Details

An opinion score is derived from all the sentiments (i.e. positive, negative (and neutral) expressed within a text document. We deploy a lexicon-based approach (Taboada et al. 2011) using the AFINN lexicon (Nielsen, 2011).

Value

Returns an opi_object containing details of the opinion measures from the text document.

References

(1) Adepeju, M. and Jimoh, F. (2021). An Analytical Framework for Measuring Inequality in the Public Opinions on Policing – Assessing the impacts of COVID-19 Pandemic using Twitter Data. https://doi.org/10.31235/osf.io/c32qh (2) Malshe, A. (2019) Data Analytics Applications. Online book available at: https://ashgreat.github.io/analyticsAppBook/index.html. Date accessed: 15th December 2020. (3) Taboada, M.et al. (2011). Lexicon-based methods for sentiment analysis. Computational linguistics, 37(2), pp.267-307. (4) Lowe, W. et al. (2011). Scaling policy preferences from coded political texts. Legislative studies quarterly, 36(1), pp.123-155. (5) Razorfish (2009) Fluent: The Razorfish Social Influence Marketing Report. Accessed: 24th February, 2021. (6) Nielsen, F. A. (2011), "A new ANEW: Evaluation of a word list for sentiment analysis in microblogs", Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages (2011) 93-98.

Examples

```
# Use police/pandemic posts on Twitter
# Experiment with a standard metric (e.g. metric 1)
score <- opi_score(textdoc = policing_dtd, metric = 1, fun = NULL)
#print result
print(score)

#Example using a user-defined opinion score -
#a demonstration with a component of SIM opinion
#Score function (by Razorfish, 2009). The opinion
#function can be expressed as:

myfun <- function(P, N, O){
  score <- (P + O - N)/(P + O + N)
return(score)
```

```
}

#Run analysis
score <- opi_score(textdoc = policing_dtd, metric = 5, fun = myfun)
#print results
print(score)
```

---

opi_sim *Simulates the opinion expectation distribution of a digital text document.*

Description

This function simulates the expectation distribution of the observed opinion score (computed using the opi_score function). The resulting tidy-format dataframe can be described as the expected sentiment document (ESD) (Adepeju and Jimoh, 2021).

Usage

```
opi_sim(osd_data, nsim=99, metric = 1, fun = NULL, quiet=TRUE)
```

Arguments

osd_data	A list (dataframe). An n x 3 OSD, in which n represents the length of the text records that have been successfully classified as expressing positive, negative or a neutral sentiment. Column 1 of the OSD is the text record ID, column 2 shows the sentiment classes (i.e. positive, negative, or neutral), while column 3 contains two variables: present and absent indicating records that include and records that do not include any of the specified theme keywords, respectively.
nsim	(an integer) Number of replicas (ESD) to simulate. Recommended values are: 99, 999, 9999, and so on. Since the run time is proportional to the number of replicas, a moderate number of simulation, such as 999, is recommended. Default: 99.
metric	(an integer) Specify the metric to utilize for the calculation of the opinion score. Default: 1. See details in the documentation of opi_score function. The input argument here must correspond to that of opi_score function in order to compute a statistical significance value (p-value).
fun	A user-defined function given that parameter metric is set equal to 5. See details in the documentation of the opi_score function.
quiet	(TRUE or FALSE) To suppress processing messages. Default: TRUE.

Details

Employs non-parametric randomization testing approach in order to generate the expectation distribution of the observed opinion scores (see details in Adepeju and Jimoh 2021).

Value

Returns a list of expected opinion scores with length equal to the number of simulation (`nsim`) specified.

References

(1) Adepeju, M. and Jimoh, F. (2021). An Analytical Framework for Measuring Inequality in the Public Opinions on Policing – Assessing the impacts of COVID-19 Pandemic using Twitter Data. https://doi.org/10.31235/osf.io/c32qh

Examples

```
#Prepare an osd data from the output
#of `opi_score` function.

score <- opi_score(textdoc = policing_dtd,
                     metric = 1, fun = NULL)
#extract OSD
OSD <- score$OSD
#note that `OSD` is shorter in length
#than `policing_dtd`, meaning that some
#text records were not classified

#Bind a fictitious indicator column
osd_data2 <- data.frame(cbind(OSD,
           keywords = sample(c("present","absent"), nrow(OSD),
           replace=TRUE, c(0.35, 0.65))))

#generate expected distribution
exp_score <- opi_sim(osd_data2, nsim=99, metric = 1,
                                 fun = NULL, quiet=TRUE)
#preview the distribution
hist(exp_score)
```

`osd_data` *Observed sentiment document (OSD).*

Description

A tidy-format list (dataframe) showing the resulting classification of each text record into positive, negative or neutral sentiment. The second column of the dataframe consists of labels variables `present` and `absent` to indicate whether any of the secondary keywords exist in a text record.

Usage

```
osd_data
```

Format

A dataframe with the following variables:

- ID: numeric id of text record with valid resultant sentiments score and classification.
- sentiment: Containing the sentiment classes.
- keywords: Indicator to show whether a secondary keyword in present or absent in a text record.

---

policing_dtd *Twitter posts on police/policing*

Description

A text document (an DTD) containing twitter posts (for an anonymous geographical location 'A') on police/policing. The DTD also includes posts that express sentiments on policing in relation to the COVID-19 pandemic (Secondary subject B)

Usage

```
policing_dtd
```

Format

A dataframe containing one variable

- text: individual text records

---

refreshment_theme *Keywords relating to facilities at train stations*

Description

List of words relating to refreshments that can be found at the Piccadilly Train Station (Manchester)

Usage

```
refreshment_theme
```

Format

A dataframe containing one variable:

- keys: list of keywords

reviews_dtd *Customer reviews from tripadvisor website*

Description

A text document (an DTD) containing the customer reviews of the Piccadilly train station (Manchester) downloaded from the www.tripadvisor.co.uk'. The reviews cover from July 2016 to March 2021.

Usage

```
reviews_dtd
```

Format

A dataframe containing one variable

- text: individual text records

signage_theme *Keywords relating to signages at train stations*

Description

List of signages at the Piccadilly Train Station (Manchester)

Usage

```
signage_theme
```

Format

A dataframe containing one variable:

- keys: list of keywords

tweets *Fake Twitter posts on police/policing 2*

Description

A text document (an DTD) containing twitter posts (for an anonymous geographical location 2) on police/policing (primary subject A). The DTD includes posts that express sentiments on policing in relation to the COVID-19 pandemic (Secondary subject B)

Usage

```
tweets
```

Format

A dataframe with the following variables:

- text: individual text records
- group: real/arbitrary groups of text records

word_distrib *Words Distribution*

Description

This function examines whether the distribution of word frequencies in a text document follows the Zipf distribution (Zipf 1934). The Zipf's distribution is considered the ideal distribution of a perfect natural language text.

Usage

```
word_distrib(textdoc)
```

Arguments

textdoc	n x 1 list (dataframe) of individual text records, where n is the number of individual records.

Details

The Zipf's distribution is most easily observed by plotting the data on a log-log graph, with the axes being log(word rank order) and log(word frequency). For a perfect natural language text, the relationship between the word rank and the word frequency should have a negative slope with all points falling on a straight line. Any deviation from the straight line can be considered an imperfection attributable to the texts within the document.

Value

A list of word ranks and their respective frequencies, and a plot showing the relationship between the two variables.

References

Zipf G (1936). The Psychobiology of Language. London: Routledge; 1936.

Examples

```
#Get an \code{n} x 1 text document
tweets_dat <- data.frame(text=tweets[,1])
plt = word_distrib(textdoc = tweets_dat)

plt
```

`word_imp` *Importance of words (terms) embedded in a text document*

Description

Produces a wordcloud which represents the level of importance of each word (across different text groups) within a text document, according to a specified measure.

Usage

```
word_imp(textdoc, metric= "tf",
words_to_filter=NULL)
```

Arguments

`textdoc`	An n x 1 list (dataframe) of individual text records, where n is the total number of individual records. An n x code2 dataframe can also be supplied, in which the second column represents the labels of the pre-defined groupings of the text records, e.g. labels of geographical areas where each text record originates. For an n x 1 dataframe, an arbitrary grouping is automatically imposed.
`metric`	(character) The measure for determining the level of importance of each word within the text document. Options include `'tf'` representing term frequency and `'tf-idf'` representing term frequency inverse document frequency (Silge & Robinson, 2016).
`words_to_filter`	A pre-defined vector of words (terms) to filter out from the DTD prior to highlighting words importance. default: `NULL`. This parameter helps to eliminate non-necessary words that may be too dominant in the results.

Details

The function determines the most important words across various grouping of a text document. The measure options include the `tf` and `tf-idf`. The idea of `tf` is to rank words in the order of their number of occurrences across the text document, whereas `tf-idf` finds words that are not used very much, but appear across many groups in the document.

Value

Graphical representation of words importance according to a specified metric. A wordcloud is used to represent words importance if `tf` is specified, while facet wrapped histogram is used if `tf-idf` is specified. A wordcloud is represents each word with a size corresponding to its level of importance. In the facet wrapped histograms words are ranked in each group (histogram) in their order of importance.

References

Silge, J. and Robinson, D. (2016) tidytext: Text mining and analysis using tidy data principles in R. Journal of Open Source Software, 1, 37.

Examples

```
#words to filter out
wf <- c("police","policing")
output <- word_imp(textdoc = policing_dtd, metric= "tf",
words_to_filter= wf)
```

Index