Package 'opitools'

March 4, 2021

Type Package

Title A Tool for Analyzing Opinions in a Text Document

Version 1.0.2

Author Monsuru Adepeju [cre, aut],

Maintainer Monsuru Adepeju <monsuur2010@yahoo.com>

Description This tool analyzes the opinions inherent in a text document relating to a specific subject (A), and assesses the impacts that opinion expressed with respect to another subject (B) have on subject A. This package is specifically designed for application to social media datasets, such as Twitter and Facebook. The utility of the package has been demonstrated in Adepeju, M. and Jimoh, F. (2021) <doi:10.31235/osf.io/c32qh> in the assessment of impacts of COVID-19 pandemic on public opinions concerning neighborhood policing across England and Wales.

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

Roxygen list(markdown = TRUE)

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

RoxygenNote 7.1.1

Suggests knitr,
rmarkdown

VignetteBuilder knitr

R topics documented:

`covid_keys` *COVID-19 pandemic related keywords*

Description

A list of keywords relating to the COVID-19 pandemic

Usage

```
covid_keys
```

Format

A list (dataframe)

`opi_impact` *Impact analysis of subject B on the opinion expressed concerning subject A in a text document*

Description

This function assesses the impacts of a subject B (a secondary subject) on the opinion concerning subject A (the primary subject) in a text document. Keywords relating to the secondary subject, can be identified using any analytical techniques, such as the frequency analysis. The keywords should then be collated provide as input into this function (see below). The subject A (primary subject) is usually the main theme of the text document. For instance, by downloading Twitter data that include a set of related hashtags; e.g. '#police', '#policing' and/or '#law enforcement', then "Police or Policing" forms the primary subject of the downloaded text document.

Usage

```
opi_impact(textdoc, sec_keywords=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.
sec_keywords	(a list) A one-column dataframe (of any number of length) containing a list of keywords relating to the secondary subject (subject B).
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 boundary 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

```
#test document: 'policing_otd'
#list of keywords: 'covid_keys'

output <- opi_impact(textdoc = policing_otd,
          sec_keywords=covid_keys, metric = 1,
          fun = NULL, nsim = 99, alternative="two.sided",
          quiet=TRUE)
```

```
#check output variables
print(output)

#to access the pvalue
output$pvalue
```

---

opi_score *Opinion score (of the main subject matter) of a text document*

Description

Given a text document (concerning a subject A), this function computes the overall opinion score based on the proportion of text records classified as expressing positive, negative or a neutral sentiment about the subject. 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 sum of all sentiments 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. Available values in this package are: `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_otd, metric = 1, fun = NULL)
#print result details
print(score)
#preview results
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_otd, metric = 5, fun = myfun)
#preview results
print(score)
```

`opi_sim` *To simulate the opinion expectation distribution of a text document.*

Description

Given a text document concerning two identified subjects (primary subject `A` and secondary subject `B`), this function simulates the expectation distribution of the observed opinion score (computed using the `opi_score` function). The resulting tidy-format dataframe is 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 secondary 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_otd,
                     metric = 1, fun = NULL)
#extract OSD
OSD <- score$OSD
#note that `OSD` is shorter in length
#than `policing_otd`, meaning that some
#text records were not classified

#Bind a fictitious indicator column
set.seed(1000)
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 classification of text records into positive, negative or neutral sentiments. The second column of the dataframe consists of label variables: `present` and `absent` which indicate whether any of the secondary keywords exist in a text record.

Usage

```
osd_data
```

Format

A list (dataframe)

`policing_otd` *Fake Twitter posts on police/policing 1*

Description

A sample text document (an OTD) containing twitter posts (from an anonymous geographical location 1). The primary subject of the document is the "process of policing". The `OTD` is believed to also include posts that express sentiments on COVID-19 pandemic (secondary subject B) in relation to the primary subject.

Usage

```
policing_otd
```

Format

A list (dataframe)

tweets *Fake Twitter posts on police/policing 2*

Description

A sample text document (an OTD) containing twitter posts (from an anonymous geographical location 2). The primary subject of the document is the "process of policing" (as the primary subject). The OTD is believed to also include posts that express sentiments on the COVID-19 pandemic (Secondary subject B) in relation to the primary subject.

Usage

```
tweets
```

Format

A list(dataframe)

word_distrib *Word Distribution*

Description

This function examines whether the distribution of word frequency 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
```

Index