Package ‘opitools’

February 22, 2021

Type Package

Title An R-package for analyzing Opinions in a Text Document

Version 1.0.0

Author Monsuru Adepeju [cre, aut],

Maintainer Monsuru Adepeju <monsuur2010@yahoo.com>

Description This package 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.

Language en-US

License GPL-3

URL https://cran.r-project.rg/web/packages/opitools/index.html

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

Depends R (>= 3.5.0)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Imports ggplot2,
tibble,
tidytext,
magrittr,
dplyr,
plyr,
tm,
stringr,
purrr,
tidyr,
utils,
likert,
cowplot,
data.table

RoxygenNote 7.1.0

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*

Description

This function assesses the impacts of a subject B (henceforth referred to as secondary subject) on the opinion concerning subject A in a text document. Keywords relating to the secondary subject, either identified analytically (e.g. using `tf_idf` function) or collated manually, are provided as input into the function (see below). The subject A (primary subject) is the subject matter upon which the text document is based. For instance, by downloading Twitter data that include hashtags: '#police', '#policing' and/or '#law enforcement', then "Police/Policing" becomes the primary subject of the text document.

Usage

```
opi_impact(textdoc, sec_keywords=NULL, metric = 1,
fun = NULL, nsim = 99, alternative="two.sided", pplot=FALSE,
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`	list A one-column dataframe (of any number of rows) containing a list of keywords relating to the secondary subject (`B`).
`metric`	integer Metric to utilize for the calculation of the opinion score. Default: `1`. See detailed documentation in the `opi_score` function.
`fun`	A user-defined function provided parameter `metric` is set as `5`. See detailed documentation in the `opi_score` function.
`nsim`	integer Number of replicas (ESD) to generate. See detailed documentation in the `opi_sim` function. Default: `99`.
`alternative`	character Default: `"two.sided"`, indicating a two-tailed test. A user can override this by specifying “`less`” or “`greater`” to run the analysis as a one-tailed test with the observed score being located at the lower or upper regions of the distribution, respectively. Note: for `metric=1` (see above), the `alternative` parameter should be set as `"two.sided"` because the opinion score is bounded by both negative and positive values. For a positively bounded opinion score, such as when `metric = 2,3 or 4`, the the `alternative` parameter should be "greater", and "less" otherwise.
`pplot`	logical To display graphical plot showing the proportion of text records containing (or not containing) any of the specified secondary keywords.
`quiet`	(TRUE or FALSE) To suppress processing and warning messages. Default: `TRUE`.

Details

This function calculate the statistical significance value (`p-value`) by comparing the observed opinion scores (from the `opi_score` function) with the expected opinion 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 created, 'S.beat' is number of replicas with the expected score greater than the observed score. If a user-defined opinion score function is specified, he/she needs to determine whether a one-tailed or two-tailed comparison is required. (see explanations above).

Value

Details of statistical significance of impacts of secondary subject B on the opinion concerning 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

`opi_score` *Opinion score of the subject matter in a text document*

Description

Given a text document (concerning a subject A), this function compute the overall opinion score based on the proportion of text records classified as expressing positive, negative or neutral sentiment. The function first transforms the text document into a tidy-format dataframe, referred to as observed sentiment document (OSD), in which each text record is assigned a sentiment class based on the sum of all sentiments expressed by 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`	integer Metric to utilize for the calculation of the opinion score. Available values are: `1,2,...,5`. Assuming `P`, `N` and `O` represent positive, negative, and neutral text records, respectively, the followings are few examples of opinion scores from the literature: `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 function as argument into the `fun` parameter below.
`fun`	A user-defined function provided parameter `metric` above is set as `5`. For example, given the function `myfun` <- function(P, N, O) ("some tasks to do"); return("a value"), the `fun` parameter is then set as `fun = myfun`. Default: `NULL` i.e. when `metric` parameter is not `5`.

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) Malshe, A. (2019) Data Analytics Applications. Online book available at: https://ashgreat.github.io/analyticsAppBoo
Date accessed: 15th December 2020. (2) Taboada, M.et al. (2011). Lexicon-based methods for sentiment analysis. Computational linguistics, 37(2), pp.267-307. (3) Lowe, W. et al. (2011). Scaling policy preferences from coded political texts. Legislative studies quarterly, 36(1), pp.123-155.

`opi_sim` *To simulate the expected sentiment (opinion) distribution*

Description

Given a text document with two underlying subjects A and B, this function simulates the expected distribution of the observed opinion score from the `opi_score`. The resulting tidy-format dataframe is referred to 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 number of number of text records that have been successfully classified as positive, negative or neutral. Column `1` of the OSD is the 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 consist and records that do not consist, respectively, of any of the identified secondary keywords.
`nsim`	integer Number of replicas (ESD) to generate. Recommended values: 99, 999, 9999, and so on. Since the run time is proportional to the number of replicas, a lower number of simulation is recommended. Default: `99`.
`metric`	integer Metric to utilize for the calculation of the opinion score. Default: `1`. See detailed documentation in the `opi_score` function. The argument selected here must correspond to that of `opi_score` function in order to compute a statistical significance value (p-value).
`fun`	A user-defined function provided parameter `metric` is set as `5`. See detailed documentation in the `opi_score` function.
`quiet`	(TRUE or FALSE) To suppress processing and Warning messages. Default: `TRUE`.

Details

Uses randomization testing approach in order to generate expected distribution of the observed opinion scores (see details in Adepeju, M. and Jimoh, F., 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

`osd_data` *Observed sentiment document (OSD)*

Description

A tidy-format list (dataframe) showing the resulting classification of each text records 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 list (dataframe)

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

Description

A text document (an OTD) containing twitter posts (for an anonymous geographical location 1) on police/policing (primary subject A). The OTD includes posts that express sentiments on policing in relation to the COVID-19 pandemic (Secondary subject B)

Usage

```
policing_otd
```

Format

A list (dataframe)

`tf_idf` *Highlight the top-most important topics in a text document*

Description

This function identifies the most important words across different text groups in a text document, according to the `tf-idf` measure (Silge & Robinson, 2016).

Usage

```
tf_idf(textdoc, n_top=10, showplot=FALSE)
```

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, but the second column must contain pre-defined group labels of the text records, e.g. group labels according to geographical locations. For an `n` x code1 dataframe, the function automatically impose arbitrary group labels on the text document, based on the length `n` of the document. Default: `FALSE`
`n_top`	integer maximum number of top-most important words per group to display. Default value: `10`
`showplot`	To display graphical plot showing ranks of top-most important words by group. Default: `FALSE`

Details

The function utilizes the `tf-idf` measure in order to determine the most important words across various text groups in a document. The idea of `tf-idf` is to find words that are not used very much, but appear across many groups in the document. using this function, a user may be able to identify certain keywords that indicates some underlying (secondary) subject that are been discussed in relation to the original (primary) subject of the text document.

Value

A graphical display showing top-most important words in each group, according to the `tf-idf` measure.

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.

`tweets` *Fake Twitter posts on police/policing 2*

Description

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

Usage

```
tweets
```

Format

A list(dataframe)

`word_distrib` *Term Distribution*

Description

This function examines whether the word frequency in a text document follows the Zipf distribution (Zipf 1934). The Zipf's distribution is considered the 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 rank and the frequency should have a negative slope with all points falling on a straight line. Any deviation from a straight line can be considered an element of imperfection from the text document.

Value

A graphical plot showing the rank-frequency graph.

References

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

Index