Package: saotd (via r-universe)

September 9, 2024

Type Package

Title Sentiment Analysis of Twitter Data

Version 0.3.1

Date 2023-09-03

Maintainer Evan Munson <evan.l.munson@gmail.com>

BugReports https://github.com/evan-l-munson/saotd/issues

Description This analytic is an in initial foray into sentiment analysis. This analytic will allow a user to access the Twitter API (once they create their own developer account), ingest tweets of their interest, clean / tidy data, perform topic modeling if interested, compute sentiment scores utilizing the Bing Lexicon, and output visualizations.

License GPL (>= 2)

URL https://github.com/evan-l-munson/saotd

Language en-US

Imports dplyr, widyr, stringr, tidytext, rtweet, tidyr, igraph, ggplot2, ggraph, scales, reshape2, lubridate, utils, stats, magrittr, ldatuning, topicmodels

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0), knitr, rmarkdown, httr, base64enc, tibble, covr

Depends R (>= 3.5.0)

VignetteBuilder knitr

SystemRequirements GSL (>=2.4), MPFR (>= 4.0.0), udunits2 (>=2.2.26-3)

Encoding UTF-8

LazyLoad true

Config/testthat/edition 3

Repository https://evan-l-munson.r-universe.dev

RemoteUrl https://github.com/evan-l-munson/saotd

RemoteRef HEAD

RemoteSha 80829121690ad44573fac1860011d2eb9c4d4078

Contents

`bigram` *Twitter Bi-Grams*

Description

Determines and displays the text Bi-Grams within the Twitter data in sequence from the most used to the least used. A Bi-Gram is a combination of two consecutive words.

Usage

```
bigram(DataFrame)
```

Arguments

`DataFrame`	Data Frame of Twitter Data.

Value

A tibble.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
TD_Bigram <- bigram(DataFrame = data)
TD_Bigram

## End(Not run)
```

`bigram_network` *Twitter Bi-Gram Network*

Description

Displays the Bi-Gram Network. Bi-Gram networks builds on computed Bi-Grams. Bi-Gram networks serve as a visualization tool that displays the relationships between the words simultaneously as opposed to a tabular display of Bi-Gram words.

Usage

```
bigram_network(
  BiGramDataFrame,
  number,
  layout = "fr",
  edge_color = "royalblue",
  node_color = "black",
  node_size = 3,
  set_seed = 1234
)
```

Arguments

`BiGramDataFrame`	Data Frame of Bi-Grams.
`number`	The minimum desired number of Bi-Gram occurrences to be displayed (number = 300, would display all Bi-Grams that have at least 300 instances).
`layout`	Desired layout from the ‘ggraph‘ package. Acceptable layouts: "star", "circle", "gem", "dh", "graphopt", "grid", "mds", "randomly", "fr", "kk", "drl", "lgl"
`edge_color`	User desired edge color.
`node_color`	User desired node color.
`node_size`	User desired node size.
`set_seed`	Seed for reproducible results.

Value

A ggraph plot.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
TD_Bigram <- bigram(DataFrame = data)
TD_Bigram_Network <- bigram_network(BiGramDataFrame = TD_Bigram,
                                    number = 300,
                                    layout = "fr",
                                    edge_color = "royalblue",
                                    node_color = "black",
                                    node_size = 3,
                                    set_seed = 1234)

TD_Bigram_Network

## End(Not run)
```

`merge_terms` *Merge Terms*

Description

Function to merge terms within a data frame and prevent redundancy in the analysis. For example many users may refer to the same entity in multiple different ways: President Trump, The U.S. President, POTUS, Trump, President Donald Trump, Donald Trump, etc. While each entry is different, they all refer to the same individual. Using Merge Terms will allow all be converted into a single term.

Usage

```
merge_terms(DataFrame, term, term_replacement, ignore_case = TRUE)
```

Arguments

`DataFrame`	Data Frame of Twitter Data.
`term`	Term selected for merging.
`term_replacement`	Desired replacement term.
`ignore_case`	True is the default setting and will ignore case sensitivity of the selected terms. Selecting FALSE will maintain case sensitivity.

Value

A Tibble with user selected term replacement.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
data <- merge_terms(DataFrame = data,
                    term = "ice cream",
                    term_replacement = "ice_cream")
data

## End(Not run)
```

`number_topics` *Number Topics*

Description

Determines the optimal number of Latent topics within a data frame by tuning the Latent Dirichlet Allocation (LDA) model parameters. Uses the 'ldatuning' package and outputs an ldatuning plot. __This process can be time consuming depending on the size of the input data frame.__

Usage

```
number_topics(
  DataFrame,
  num_cores = 1L,
  min_clusters = 2,
  max_clusters = 12,
  skip = 2,
  set_seed = 1234
)
```

Arguments

`DataFrame`	Data Frame of Twitter Data.
`num_cores`	The number of CPU cores to processes models simultaneously (2L for dual core processor).
`min_clusters`	Lower range for the number of clusters.
`max_clusters`	Upper range for the number of clusters.
`skip`	Integer; The number of clusters to skip between entries.
`set_seed`	Seed for reproducible results.

Value

A Tidy DataFrame.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
LDA_Topic_Plot <- number_topics(DataFrame = data,
                                num_cores = 2L,
                                min_clusters = 2,
                                max_clusters = 12,
                                skip = 2,
                                set_seed = 1234)

LDA_Topic_Plot

## End(Not run)
```

---

posneg_words *Twitter Positive and Negative Words*

Description

Determines and displays the most positive and negative words within the twitter data.

Usage

```
posneg_words(DataFrameTidy, num_words, filterword = NULL)
```

Arguments

DataFrameTidy	DataFrame of Twitter Data that has been tidy'd.
num_words	Desired number of words to be returned.
filterword	Word or words to be removed.

Value

A ggplot

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
posneg <- posneg_words(DataFrameTidy = tidy_data,
                       n = 10)
posneg

data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
```

```
posneg <- posneg_words(DataFrameTidy = tidy_data,
                       n = 10,
                       filterword = "fail")
posneg

data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
posneg <- posneg_words(DataFrameTidy = tidy_data,
                       n = 10,
                       filterword = c("fail", "urgent"))
posneg

## End(Not run)
```

`raw_tweets` *Twitter Data Set*

Description

Dataset from a [Twitter US Airline Sentiment] (https://www.kaggle.com/crowdflower/twitter-airline-sentiment) Kaggle competition, from December 2017. The dataset contains 14,487 tweets from 6 different hashtags (2,604 x #American, 2,220 x #Delta, 2,420 x #Southwest, 3,822 x #United, 2,913 x #US Airways, 504 x #Virgin America).

Usage

```
data(raw_tweets)
```

Format

A `tribble` with 14,483 rows and 6 variables.

id ID of this status.

hashtags Hashtag that the individual tweet was acquired from.

screenName Screen name of the user who posted this status.

text The text of the status.

created_at When this status was created.

key Unique key based on the tweets originators user id and the created date time group.

trigram *Twitter Tri-Grams*

Description

Determines and displays the text Tri-Grams within the Twitter data in sequence from the most used to the least used. A Tri-Gram is a combination of three consecutive words.

Usage

```
trigram(DataFrame)
```

Arguments

DataFrame	Data Frame of Twitter Data.

Value

A tribble.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
TD_Trigram <- trigram(DataFrame = data)
TD_Trigram

## End(Not run)
```

tweet_acquire *Acquire Twitter Tweets*

Description

Function will enable a user to access the Twitter API through the [Twitter Developers Account](https://dev.twitter.com/) site. Once a user has a Twitter developers account and has received their individual consumer key, consumer secret key, access token, and access secret they can acquire Tweets based on a list of hashtags and a requested number of entries per query.

Usage

```
tweet_acquire(
  twitter_app,
  consumer_api_key,
  consumer_api_secret_key,
  access_token,
  access_token_secret,
  query,
  num_tweets,
  reduced_tweets = TRUE,
  distinct = TRUE
)
```

Arguments

twitter_app	The name of user created Twitter Application.
consumer_api_key	Twitter Application management consumer API key.
consumer_api_secret_key	Twitter Application management consumer API secret key. Application must have Read and write access level and Callback URL of http://127.0.0.1:1410.
access_token	Twitter Application management access token (apps.twitter.com).
access_token_secret	Twitter Application management access secret token (apps.twitter.com).
query	A single query or a list of queries the user has specified. Character string, not to exceed 500 characters. To search for tweets containing at least one of multiple possible terms, separate each search term with spaces and "OR" (in caps). For example, the search q = "data science" looks for tweets containing both "data" and "science" located anywhere in the tweets and in any order. When "OR" is entered between search terms, query = "data OR science", Twitter's REST API should return any tweet that contains either "data" or "science."
num_tweets	Number of Tweets to be acquired per each hashtag.
reduced_tweets	Logical. If reduced_tweets = TRUE, the data frame returned to the user will be significantly reduced specifically for use in the 'saotd' package. If reduced_tweets = FALSE, the full results from the Twitter API will be returned.
distinct	Logical. If distinct = TRUE, the function removes multiple Tweets that originate from the same Twitter id at the exact same time.

Value

A Data Frame with tweets and meta data.

Examples

```
## Not run:
twitter_app <- "super_app"
consumer_api_key <- "XXXXXXXXXXXXXXXXXXXXXXXXX"
```

```
consumer_api_secret_key <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
access_token <- "XXXXXXXXXXXXXXXXXX-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
access_token_secret <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

tweets <- tweet_acquire(
  twitter_app = "twitter_app",
  consumer_api_key = consumer_api_key,
  consumer_api_secret_key = consumer_api_secret_key,
  access_token = access_token,
  access_token_secret = access_token_secret,
  query = "#icecream",
  num_tweets = 100,
  distinct = TRUE)

Or the Twitter API keys and tokens can be saved as an .Renviron file in the
working directory.  If using a `.Renviron` file, the data should be saved
like the below example:

consumer_api_key=XXXXXXXXXXXXXXXXXXXXXXXXX
consumer_api_secret_key=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
access_token=XXXXXXXXXXXXXXXXXX-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
access_token_secret=XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

The `tweet_acquire` function would access the keys and tokens using the
`Sys.getenv()` function and would appear like the below example:

tweets <- tweet_acquire(
  twitter_app = "twitter_app",
  consumer_api_key = Sys.getenv('consumer_api_key'),
  consumer_api_secret_key = Sys.getenv('consumer_api_secret_key'),
  access_token = Sys.getenv('access_token'),
  access_token_secret = Sys.getenv('access_token_secret'),
  query = "#icecream",
  num_tweets = 100,
  distinct = TRUE)


## End(Not run)
```

`tweet_box` *Twitter Data Box Plot*

Description

Displays the distribution scores of either hashtag or topic Twitter data.

Usage

```
tweet_box(DataFrameTidyScores, HT_Topic)
```

Arguments

DataFrameTidyScores	DataFrame of Twitter Data that has been tidy'd and scored.
HT_Topic	If using hashtag data select: "hashtag". If using topic data select: "topic".

Value

A ggplot box plot.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
ht_box <- tweet_box(DataFrameTidyScores = score_data,
                    HT_Topic = "hashtag")
ht_box

data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "topic")
topic_box <- tweet_box(DataFrameTidyScores = score_data,
                       HT_Topic = "topic")
topic_box

## End(Not run)
```

---

tweet_corpus_distribution

Twitter Corpus Distribution

Description

Determines the scores distribution for the entire Twitter data corpus.

Usage

```
tweet_corpus_distribution(
  DataFrameTidyScores,
  binwidth = 1,
  color = "black",
  fill = "grey"
)
```

Arguments

DataFrameTidyScores	DataFrame of Twitter Data that has been tidy'd and scored.
binwidth	The width of the bins. Default is 1.
color	The user selected color to highlight the bins.
fill	The interior color of the bins.

Value

A ggplot.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
Corp_Dist <- tweet_corpus_distribution(DataFrameTidyScores = score_data,
                                       binwidth = 1,
                                       color = "black",
                                       fill = "white")
Corp_Dist

## End(Not run)
```

---

tweet_distribution *Twitter Hashtag or Topic Distribution*

Description

Determines the scores distribution by hashtag or topic for Twitter data.

Usage

```
tweet_distribution(
  DataFrameTidyScores,
  HT_Topic,
  bin_width = 1,
  color = "black",
  fill = "black"
)
```

Arguments

DataFrameTidyScores	DataFrame of Twitter Data that has been tidy'd and scored.
HT_Topic	If using hashtag data select: "hashtag". If using topic data select: "topic".
bin_width	The width of the bins. Default is 1.
color	The user selected color to highlight the bins.
fill	The interior color of the bins.

Value

A facet wrap ggplot.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
Dist <- tweet_distribution(DataFrameTidyScores = score_data,
                     HT_Topic = "hashtag",
                     bin_width = 1,
                     color = "black",
                     fill = "white")
Dist

## End(Not run)
```

---

tweet_max_scores *Twitter Data Maximum Scores*

Description

Determines the Maximum scores for either the entire dataset or the Maximum scores associated with a hashtag or topic analysis.

Usage

```
tweet_max_scores(DataFrameTidyScores, HT_Topic, HT_Topic_Selection = NULL)
```

Arguments

DataFrameTidyScores	DataFrame of Twitter Data that has been tidy'd and scored.
HT_Topic	If using hashtag data select: "hashtag". If using topic data select: "topic".
HT_Topic_Selection	The hashtag or topic to be investigated. NULL will find min across entire data frame.

Value

A Tibble.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
min_scores <- tweet_max_scores(DataFrameTidyScores = score_data,
                               HT_Topic = "hashtag")

data <- twitter_data
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
min_scores <- tweet_max_scores(DataFrameTidyScores = score_data,
                               HT_Topic = "hashtag",
                               HT_Topic_Selection = "icecream")

## End(Not run)
```

---

tweet_min_scores *Twitter Data Minimum Scores*

Description

Determines the minimum scores for either the entire dataset or the minimum scores associated with a hashtag or topic analysis.

Usage

```
tweet_min_scores(DataFrameTidyScores, HT_Topic, HT_Topic_Selection = NULL)
```

Arguments

DataFrameTidyScores	DataFrame of Twitter Data that has been tidy'd and scored.
HT_Topic	If using hashtag data select: "hashtag". If using topic data select: "topic".
HT_Topic_Selection	The hashtag or topic to be investigated. NULL will find min across entire dataframe.

Value

A Tibble.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
min_scores <- tweet_min_scores(DataFrameTidyScores = score_data,
                               HT_Topic = "hashtag")

data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                     HT_Topic = "hashtag")
min_scores <- tweet_min_scores(DataFrameTidyScores = score_data,
                               HT_Topic = "hashtag",
                               HT_Topic_Selection = "icecream")

## End(Not run)
```

---

tweet_scores *Score Tidy Twitter Data*

Description

Function to Calculate Sentiment Scores that will account for sentiment by hashtag or topic.

Usage

```
tweet_scores(DataFrameTidy, HT_Topic)
```

Arguments

DataFrameTidy	Data Frame of Twitter Data that has been tidy'd.
HT_Topic	If using hashtag data select: "hashtag". If using topic data select: "topic"

Value

A Scored DataFrame.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
score_data
```

```
## End(Not run)
```

`tweet_tidy` *Tidy Twitter Data*

Description

Function to Tidy Twitter Data. This function will remove a significant amount of the original twitter metadata, as it is not needed to determine the sentiment of the tweets. This function will remove all emoticons, punctuation, weblinks while maintaining actual Tweet text.

Usage

```
tweet_tidy(DataFrame)
```

Arguments

`DataFrame`	Data Frame of Twitter Data.

Value

A Tidy tibble.

Examples

```
## Not run:
library(saotd)

data <- raw_tweets
tidy_data <- tweet_tidy(DataFrame = data)
tidy_data


## End(Not run)
```

`tweet_time` *Twitter Data Timeseries Plot.*

Description

Displays the Twitter data sentiment scores through time. The sentiment scores by hashtag or topic are summed per day and plotted to show the change in sentiment through time.

Usage

```
tweet_time(DataFrameTidyScores, HT_Topic)
```

Arguments

DataFrameTidyScores	DataFrame of Twitter Data that has been tidy'd and scored.
HT_Topic	If using hashtag data select: "hashtag". If using topic data select: "topic".

Value

A ggplot plot.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
ht_time <- tweet_time(DataFrameTidyScores = score_data,
                      HT_Topic = "hashtag")
ht_time

data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "topic")
topic_time <- tweet_time(DataFrameTidyScores = score_data,
                         HT_Topic = "topic")
topic_time

## End(Not run)
```

---

tweet_topics *Tweet Topics*

Description

Determines the Latent topics within a data frame by using Latent Dirichlet Allocation (LDA) model parameters. Uses the 'ldatuning' package and outputs an ldatuning plot. Prepares Tweet text, creates DTM, conducts LDA, display data terms associated with each topic.

Usage

```
tweet_topics(
  DataFrame,
  clusters,
  method = "Gibbs",
  num_terms = 10,
  set_seed = 1234
)
```

Arguments

DataFrame	Data Frame of Twitter Data.
clusters	The number of latent clusters.
method	method = "Gibbs"
num_terms	The desired number of terms to be returned for each topic.
set_seed	Seed for reproducible results.

Value

Returns LDA topics.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
LDA_data <- tweet_topics(DataFrame = data,
                         clusters = 8,
                         method = "Gibbs",
                         set_seed = 1234,
                         num_terms = 10)

LDA_data

## End(Not run)
```

---

tweet_violin *Twitter Data Violin Plot*

Description

Displays the distribution scores of either hashtag or topic Twitter data.

Usage

```
tweet_violin(DataFrameTidyScores, HT_Topic)
```

Arguments

DataFrameTidyScores	DataFrame of Twitter Data that has been tidy'd and scored.
HT_Topic	If using hashtag data select: "hashtag". If using topic data select: "topic".

Value

A ggplot violin plot.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
ht_violin <- tweet_violin(DataFrameTidyScores = score_data,
                          HT_Topic = "hashtag")
ht_violin

data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "topic")
topic_violin <- tweet_violin(DataFrameTidyScores = score_data,
                             HT_Topic = "topic")
topic_violin

## End(Not run)
```

---

unigram *Twitter Uni-Grams*

Description

Determines and displays the text Uni-Grams within the Twitter data in sequence from the most used to the least used. A Uni-Gram is a single word.

Usage

```
unigram(DataFrame)
```

Arguments

DataFrame	Data Frame of Twitter Data.

Value

A tibble.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
TD_Unigram <- unigram(DataFrame = data)
TD_Unigram

## End(Not run)
```

word_corr *Twitter Word Correlations*

Description

The word correlation displays the mutual relationship between words.

Usage

```
word_corr(DataFrameTidy, number, sort = TRUE)
```

Arguments

DataFrameTidy	Data Frame of Twitter Data that has been tidy'd.
number	The number of word instances to be included.
sort	Rank order the results from most to least correlated.

Value

A Tibble.

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
TD_Word_Corr <- word_corr(DataFrameTidy = tidy_data,
                          number = 500,
                          sort = TRUE)

TD_Word_Corr

## End(Not run)
```

word_corr_network *Twitter Word Correlations Plot*

Description

The word correlation network displays the mutual relationship between words. The correlation network shows higher correlations with a thicker and darker edge color.

Usage

```
word_corr_network(
  WordCorr,
  Correlation = 0.15,
  layout = "fr",
  edge_color = "royalblue",
  node_color = "black",
  node_size = 2,
  set_seed = 1234
)
```

Arguments

WordCorr	Data Frame of Word Correlations.
Correlation	Minimum level of correlation to be displayed.
layout	Desired layout from the 'ggraph' package. Acceptable layouts: "star", "circle", "gem", "dh", "graphopt", "grid", "mds", "randomly", "fr", "kk", "drl", "lgl"
edge_color	User desired edge color.
node_color	User desired node color.
node_size	User desired node size.
set_seed	Seed for reproducible results.

Value

An igraph plot

Examples

```
## Not run:
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
TD_Word_Corr <- word_corr(DataFrameTidy = tidy_data,
                          number = 500,
                          sort = TRUE)
TD_Word_Corr_Network <- word_corr_network(WordCorr = TD_Word_Corr,
                                          Correlation = 0.15,
                                          layout = "fr",
                                          edge_color = "royalblue",
                                          node_color = "black",
                                          node_size = 2,
                                          set_seed = 1234)

TD_Word_Corr_Network

## End(Not run)
```

Index