Package 'saotd'

April 4, 2019

Type Package

Title Sentiment Analysis of Twitter Data

Version 0.2.0

Date 2019-04-02

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 x bing Lexicon, and output visualizations.

License GPL (>= 2)

Imports plyr, dplyr, widyr, stringr, tidytext, twitteR, purrr, tidyr, igraph, maps, ggplot2, ggraph, scales, reshape2, lubridate, utils, stats, magrittr, ldatuning, topicmodels

RoxygenNote 6.1.1

Suggests testthat, knitr, rmarkdown, httr, base64enc

Depends R (>= 3.3.0)

VignetteBuilder knitr

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

Encoding UTF-8

LazyLoad true

NeedsCompilation no

Author Evan Munson [aut, cre] (<https://orcid.org/0000-0002-9958-6800>), Christopher Smith [aut] (<https://orcid.org/0000-0002-8288-270X>), Bradley Boehmke [aut] (<https://orcid.org/0000-0002-3611-8516>), Jason Freels [aut] (<https://orcid.org/0000-0002-2415-0340>)

Repository CRAN

Date/Publication 2019-04-04 16:30:03 UTC

R topics documented:

`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`	DataFrame of Twitter Data.

Value

A tribble.

Examples

```
library(saotd)
data <- raw_tweets
TD_Bigram <- bigram(DataFrame = data)
TD_Bigram
```

`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 = 300, layout = "fr",
  edge_color = "royalblue", node_color = "black", node_size = 3,
  set_seed = 1234)
```

Arguments

`BiGramDataFrame`	DataFrame of Bi-Grams.
`number`	The minimum desired number of Bi-Gram occurances 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 reproducable results.

Value

A ggraph plot.

Examples

```
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
```

---

merge_terms *Merge Terms*

Description

Function to merge terms within a dataframe 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)
```

Arguments

DataFrame	DataFrame of Twitter Data.
term	Term selected for merging.
term_replacement	Desired replacement term.

Value

A Tidy DataFrame.

Examples

```
library(saotd)
data <- raw_tweets
data <- merge_terms(DataFrame = data,
                    term = "ice cream",
                    term_replacement = "ice_cream")
data
```

`number_topics` *Number Topics*

Description

Determines the optimal number of Latent topics within a dataframe 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 dataframe.__

Usage

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

Arguments

`DataFrame`	DataFrame 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 reproducable results.

Value

A Tidy DataFrame.

Examples

```
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
```

---

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

```
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

---

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.

hashtag 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 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	DataFrame of Twitter Data.

Value

A tribble.

Examples

```
library(saotd)
data <- raw_tweets
TD_Trigram <- trigram(DataFrame = data)
TD_Trigram
```

`tweet_acquire` *Acquire Twitter Tweets*

Description

Function will enable a user to access the Twitter API throught the [Twitter Developers Account](https://dev.twitter.com/) site. Once a user has a Twitter developers account and has recieved 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 entires per hashtag.

Usage

```
tweet_acquire(consumer_key, consumer_secret, access_token, access_secret,
  HT, num_tweets, file_name, distinct = TRUE)
```

Arguments

`consumer_key`	Twitter Application management consumer key.
`consumer_secret`	Twitter Application management consumer secret key.
`access_token`	Twitter Application management access token.
`access_secret`	Twitter Application management access secret key.
`HT`	A single hashtag or a list of hashtags the user has specified.
`num_tweets`	Number of Tweets to be acquired per each hashtag.
`file_name`	User desired output .RData file name.
`distinct`	Logical. If distinct = TRUE, the function removes multiple Tweets that originate from the same Twitter id at the exact same time.

Value

A DataFrame.

Examples

```
consumer_key <- "XXXXXXXXXXXXXXXXXXXXXXXXX"
consumer_secret <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
access_token <- "XXXXXXXXXXXXXXXXXX-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
access_secret <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

test_tweets <- file.path(tempdir(), "test_tweets.RData")

hashtags <- c("#job", "#Friday", "#fail", "#icecream", "#random", "#kitten", "#airline")

tweet_acquire(consumer_key = consumer_key,
              consumer_secret = consumer_secret,
              access_token = access_token,
              access_secret = access_secret,
              HT = hashtags,
              num_tweets = 10,
              file_name = test_tweets,
              distinct = TRUE)

load(test_tweets)
```

---

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

```
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
```

---

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 = "white")
```

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

```
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
```

---

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, binwidth = 1,
  color = "black", fill = "white")
```

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".
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 facet wrap ggplot.

Examples

```
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",
                    binwidth = 1,
                    color = "black",
                    fill = "white")
Dist
```

`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 dataframe.

Value

A Tidy DataFrame.

Examples

```
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")
```

`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 Tidy DataFrame.

Examples

```
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")
```

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	DataFrame 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

```
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
score_data
```

tweet_tidy *Tidy Twitter Data*

Description

Function to Tidy Twitter Data and remove all emoticons, punctuation, weblinks while maintaiing actual Tweet.

Usage

```
tweet_tidy(DataFrame)
```

Arguments

DataFrame	DataFrame of Twitter Data.

Value

A Tidy DataFrame.

Examples

```
library(saotd)
data <- raw_tweets
tidy_data <- tweet_tidy(DataFrame = data)
tidy_data
```

`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

```
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
```

---

tweet_topics *Tweet Topics*

Description

Determines the Latent topics within a dataframe 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", set_seed = 1234,
  num_terms = 10)
```

Arguments

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

Value

Returns LDA topics.

Examples

```
library(saotd)
data <- raw_tweets
LDA_data <- tweet_topics(DataFrame = data,
                         clusters = 8,
                         method = "Gibbs",
                         set_seed = 1234,
                         num_terms = 10)

LDA_data
```

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

```
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
```

tweet_worldmap *Twitter Data Worldmap Plot.*

Description

Displays the location of a Tweet across the globe by hashtag or topic.

Usage

```
tweet_worldmap(DataFrame, HT_Topic)
```

Arguments

DataFrame	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

```
library(saotd)
data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "hashtag")
ht_map <- tweet_worldmap(DataFrameTidyScores = score_data,
                         HT_Topic = "hashtag")
ht_map

data <- raw_tweets
tidy_data <- Tidy(DataFrame = data)
score_data <- tweet_scores(DataFrameTidy = tidy_data,
                           HT_Topic = "topic")
topic_map <- tweet_worldmap(DataFrameTidyScores = score_data,
                      HT_Topic = "topic")
topic_map
```

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	DataFrame of Twitter Data.

Value

A tribble.

Examples

```
library(saotd)
data <- raw_tweets
TD_Unigram <- unigram(DataFrame = data)
TD_Unigram
```

word_corr *Twitter Word Correlations*

Description

The word correlation displays the mutual relationship between words.

Usage

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

Arguments

DataFrameTidy	DataFrame 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 tribble

Examples

```
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
```

---

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	DataFrame 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 reproducable results.

Value

An igraph plot

Examples

```
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
```

Index