Package 'vosonSML'

October 12, 2022

Version 0.32.7

Title Collecting Social Media Data and Generating Networks for Analysis

Description A suite of easy to use functions for collecting social media data and generating networks for analysis. Supports Twitter, YouTube, Reddit and web site data sources.

Type Package

Imports data.table, dplyr (>= 1.0), httr (>= 1.3.0), jsonlite, lubridate, methods, purrr, rlang (>= 1.0), stringr, textutils, tidyr, tibble

Suggests httpuv, igraph (>= 1.2.2), knitr, readr, rmarkdown, robotstxt, rtweet (>= 1.0.2), rvest, stopwords, testthat, tidytext, urltools, xml2

Depends R (>= 4.1)

Encoding UTF-8

Maintainer Bryan Gertzel <bryan.gertzel@anu.edu.au>

License GPL (>= 3)

RoxygenNote 7.2.1

NeedsCompilation no

VignetteBuilder knitr

URL https://github.com/vosonlab/vosonSML

BugReports https://github.com/vosonlab/vosonSML/issues

Author Bryan Gertzel [aut, cre],
Robert Ackland [aut] (<https://orcid.org/0000-0002-0008-1766>),
Timothy Graham [aut] (<https://orcid.org/0000-0002-4053-9313>),
Francisca Borquez [ctb]

Repository CRAN

Date/Publication 2022-08-16 13:00:01 UTC

R topics documented:

AddText . . . 3
AddText.activity.reddit . . . 4
AddText.activity.twitter . . . 5
AddText.actor.reddit . . . 5
AddText.actor.twitter . . . 6
AddText.actor.youtube . . . 7
AddText.semantic.twitter . . . 8
AddText.twomode.twitter . . . 9
AddUserData . . . 9
AddUserData.actor.twitter . . . 10
AddUserData.twomode.twitter . . . 11
AddVideoData . . . 13
AddVideoData.actor.youtube . . . 13
Authenticate . . . 14
Authenticate.reddit . . . 15
Authenticate.twitter . . . 16
Authenticate.web . . . 18
Authenticate.youtube . . . 18
auth_twitter_app . . . 19
auth_twitter_dev . . . 20
auth_twitter_user . . . 21
Collect . . . 22
Collect.reddit . . . 23
Collect.search.twitter . . . 24
Collect.timeline.twitter . . . 26
Collect.web . . . 27
Collect.youtube . . . 28
Create . . . 30
Create.activity.reddit . . . 31
Create.activity.twitter . . . 32
Create.activity.web . . . 33
Create.activity.youtube . . . 34
Create.actor.reddit . . . 35
Create.actor.twitter . . . 35
Create.actor.web . . . 37
Create.actor.youtube . . . 38
Create.semantic.twitter . . . 39
Create.twomode.twitter . . . 41
Graph . . . 42
ImportRtweet . . . 43
Merge . . . 44
MergeFiles . . . 44

Index **46**

AddText *Add columns containing text data to network dataframes*

Description

Network is supplemented with additional social media text data applied as node or edge attributes.

Usage

```
AddText(net, data, ...)

add_text(net, data, ...)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
...	Additional parameters passed to function.

Value

Network as a named list of two dataframes containing $nodes and $edges including columns containing text data.

Note

Supports social media activity and actor networks. Refer to AddText.activity.reddit and AddText.actor.reddit for additional reddit parameters. Refer to AddText.actor.youtube for additional YouTube actor network parameters.

Examples

```
## Not run:
# add text to an activity network
net_activity <- collect_data |>
  Create("activity") |> AddText(collect_data)

# network
net_activity$nodes
net_activity$edges

## End(Not run)
```

`AddText.activity.reddit` *Add columns containing text data to reddit activity network dataframes*

Description

Add columns containing text data to reddit activity network dataframes

Usage

```
## S3 method for class 'activity.reddit'
AddText(net, data, cleanText = FALSE, ...)
```

Arguments

`net`	A named list of dataframes `nodes` and `edges` generated by `Create`.
`data`	A dataframe generated by `Collect`.
`cleanText`	Logical. Simple removal of problematic characters for XML 1.0 standard. Implemented to prevent reddit specific XML control character errors when generating graphml files. Default is `FALSE`.
`...`	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing `$nodes` and `$edges` including columns containing text data.

Examples

```
## Not run:
# add text to an activity network
net_activity <- collect_rd |>
  Create("activity") |>
  AddText(collect_rd)

# network
net_activity$nodes
net_activity$edges

## End(Not run)
```

AddText.activity.twitter *Add columns containing text data to twitter activity network dataframes*

Description

Add columns containing text data to twitter activity network dataframes

Usage

```
## S3 method for class 'activity.twitter'
AddText(net, data, hashtags = FALSE, ...)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
hashtags	Logical. Add tweet hashtags to dataframes. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges including columns containing text data.

AddText.actor.reddit *Add columns containing text data to reddit actor network dataframes*

Description

Add columns containing text data to reddit actor network dataframes

Usage

```
## S3 method for class 'actor.reddit'
AddText(net, data, cleanText = FALSE, ...)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
cleanText	Logical. Simple removal of problematic characters for XML 1.0 standard. Implemented to prevent reddit specific XML control character errors when generating graphml files. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges including columns containing text data.

Examples

```
## Not run:
# add text to an actor network ignoring references to actors at the beginning of
# comment text
net_actor <- collect_rd |>
  Create("actor") |>
  AddText(collect_rd)

# network
net_actor$nodes
net_actor$edges

## End(Not run)
```

AddText.actor.twitter *Add columns containing text data to twitter actor network dataframes*

Description

Add columns containing text data to twitter actor network dataframes

Usage

```
## S3 method for class 'actor.twitter'
AddText(net, data, hashtags = FALSE, ...)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
hashtags	Logical. Add tweet hashtags to dataframes. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges including columns containing text data.

AddText.actor.youtube *Add columns containing text data to YouTube activity network dataframes*

Description

Text comments are added to the network as node attributes.

Text comments are added to the network as edge attributes. References to actors are detected at the beginning of comments and edges redirected to that actor instead if they differ from the top-level comment author.

Usage

```
## S3 method for class 'activity.youtube'
AddText(net, data, ...)

## S3 method for class 'actor.youtube'
AddText(net, data, repliesFromText = FALSE, atRepliesOnly = TRUE, ...)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
...	Additional parameters passed to function. Not used in this method.
repliesFromText	Logical. If comment text for an edge begins with screen_name change the edge to be directed to screen_name - if different from the top level comment author that the reply comment was posted to. Default is FALSE.
atRepliesOnly	Logical. Comment screen_names must begin with an '@' symbol to be redirected. Default is TRUE.

Value

Network as a named list of two dataframes containing $nodes and $edges including columns containing text data.

Network as a named list of two dataframes containing $nodes and $edges including columns containing text data.

Examples

```
## Not run:
# add text to an activity network
net_activity <- collect_yt |>
  Create("activity") |> AddText(collect_yt)

# network
```

```
net_activity$nodes
net_activity$edges

## End(Not run)

## Not run:
# add text to an actor network ignoring references to actors at
# the beginning of comment text
net_actor <- collect_yt |>
  Create("actor") |>
  AddText(collect_yt, repliesFromText = FALSE)

# network
net_actor$nodes
net_actor$edges

## End(Not run)
```

`AddText.semantic.twitter` *Add columns containing text data to twitter semantic network dataframes*

Description

Add columns containing text data to twitter semantic network dataframes

Usage

```
## S3 method for class 'semantic.twitter'
AddText(net, data, hashtags = FALSE, ...)
```

Arguments

`net`	A named list of dataframes `nodes` and `edges` generated by `Create`.
`data`	A dataframe generated by `Collect`.
`hashtags`	Logical. Add tweet hashtags to dataframes. Default is `FALSE`.
`...`	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing `$nodes` and `$edges` including columns containing text data.

AddText.twomode.twitter

Add columns containing text data to twitter 2mode network dataframes

Description

Add columns containing text data to twitter 2mode network dataframes

Usage

```
## S3 method for class 'twomode.twitter'
AddText(net, data, hashtags = FALSE, ...)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
hashtags	Logical. Add tweet hashtags to dataframes. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges including columns containing text data.

AddUserData

Add columns of user information as node attributes to network dataframes

Description

Network is supplemented with additional downloaded social media user information applied as node attributes.

Usage

```
AddUserData(net, data, ...)

add_users(net, data, ...)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
...	Additional parameters passed to function.

Value

Network as a named list of three dataframes containing `$nodes`, `$edges`. Nodes include columns for additional user profile data and metrics. Referenced users for which no data was found are returned in `missing_users`.

Note

Only supports twitter actor networks. Refer to `AddUserData.actor.twitter`.

`AddUserData.actor.twitter`

Supplement twitter actor network by adding user profile attributes to nodes

---

Description

Network is supplemented with additional downloaded user information applied as 2mode node attributes.

Usage

```
## S3 method for class 'actor.twitter'
AddUserData(
  net,
  data,
  lookupUsers = FALSE,
  twitterAuth = NULL,
  retryOnRateLimit = TRUE,
  refresh = FALSE,
  rmMisc = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`net`	A named list of dataframes `nodes` and `edges` generated by `Create`.
`data`	A dataframe generated by `Collect`.
`lookupUsers`	Logical. Lookup user profile metadata using the twitter API for any users data missing from the collect data set. For example fetches profile information for users that became nodes during network creation because they were mentioned in a tweet but did not author any tweets themselves. Default is `FALSE`.
`twitterAuth`	A twitter authentication object from `Authenticate`.
`retryOnRateLimit`	Logical. When the API rate-limit is reached should the collection wait and resume when it resets. Default is `TRUE`.

refresh	Logical. Lookup and replace all available user metadata. Default is FALSE.
rmMisc	Logical. Remove miscellaneous user data columns such as user profile colors and other visual elements. Default is TRUE.
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of three dataframes containing $nodes, $edges. Nodes include columns for additional user profile data and metrics. Referenced users for which no data was found are returned in missing_users.

Note

Using the standard twitter API this function is limited to collecting profiles of 90000 users per 15 mins before hitting the rate limit.

Examples

```
## Not run:
# add user info to a twitter actor network
net_actor <- data_collect |>
  Create("actor") |>
  AddUserData(data_collect)

## End(Not run)
```

AddUserData.twomode.twitter

Supplement twitter 2mode network by adding user profile attributes to nodes

---

Description

Network is supplemented with additional downloaded user information applied as 2mode node attributes.

Usage

```
## S3 method for class 'twomode.twitter'
AddUserData(
  net,
  data,
  lookupUsers = FALSE,
  twitterAuth = NULL,
  retryOnRateLimit = TRUE,
```

```
  refresh = FALSE,
  rmMisc = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
data	A dataframe generated by Collect.
lookupUsers	Logical. Lookup user profile metadata using the twitter API for any users data missing from the collect data set. For example fetches profile information for users that became nodes during network creation because they were mentioned in a tweet but did not author any tweets themselves. Default is FALSE.
twitterAuth	A twitter authentication object from Authenticate.
retryOnRateLimit	Logical. When the API rate-limit is reached should the collection wait and resume when it resets. Default is TRUE.
refresh	Logical. Lookup and replace all available user metadata. Default is FALSE.
rmMisc	Logical. Remove miscellaneous user data columns such as user profile colors and other visual elements. Default is TRUE.
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of three dataframes containing $nodes, $edges. Nodes include columns for additional user profile data and metrics. Referenced users for which no data was found are returned in missing_users.

Note

Using the standard twitter API this function is limited to collecting profiles of 90000 users per 15 mins before hitting the rate limit.

Examples

```
## Not run:
# add user info to a twitter 2mode network
net_2mode <- data_collect |>
  Create("twomode") |>
  AddUserData(data_collect)

## End(Not run)
```

`AddVideoData` *Add columns of video information to network dataframes*

Description

Network is supplemented with additional downloaded video information.

Usage

```
AddVideoData(net, youtubeAuth = NULL, ...)

add_videos(net, youtubeAuth = NULL, ...)
```

Arguments

`net`	A named list of dataframes `nodes` and `edges` generated by `Create`.
`youtubeAuth`	YouTube Authenticate object.
`...`	Additional parameters passed to function.

Value

Network as a named list of three dataframes containing `$nodes`, `$edges` and `$videos` nodes and edges include columns for additional video data.

Note

Only supports YouTube actor networks. Refer to `AddVideoData.actor.youtube`.

`AddVideoData.actor.youtube`
Add video information to youtube actor network dataframes

Description

YouTube actor network is supplemented with additional downloaded video information. Adds video id, title, description and publish time as edge attributes. Nodes or actor references to video id's in the network are substituted with the actor id (video channel id) retrieved from the video details.

Usage

```
## S3 method for class 'actor.youtube'
AddVideoData(
  net,
  youtubeAuth = NULL,
  videoIds = NULL,
  actorSubOnly = FALSE,
  ...
)
```

Arguments

net	A named list of dataframes nodes and edges generated by Create.
youtubeAuth	YouTube Authenticate object.
videoIds	List. Video id's for which to download video information.
actorSubOnly	Logical. Only substitute video id's for their publishers channel id. Don't add additional video data to edge list.
...	Additional parameters passed to function.

Value

Network as a named list of three dataframes containing $nodes, $edges and $videos nodes and edges include columns for additional video data.

Examples

```
## Not run:
# replace video id references with actors and add video id, title, description and plublish time
# to an actor network
actorNetwork <- collectData |> Create("actor") |> AddVideoData(youtubeAuth)

# only replace video id references with actors that published videos in network
actorNetwork <- collectData |> Create("actor") |> AddVideoData(youtubeAuth, actorSubOnly = TRUE)

# network
# actorNetwork$nodes
# actorNetwork$edges

# dataframe of downloaded video data
# actorNetwork$videos

## End(Not run)
```

Authenticate *Create a credential object to access social media APIs*

Description

Authenticate creates a credential object that enables R to make authenticated calls to social media APIs. A credential object is a S3 object containing authentication related information such as an access token or key, and a class name identifying the social media that grants authentication. Authenticate is the first step of the Authenticate, Collect and Create workflow.

Refer to Authenticate.twitter, Authenticate.youtube and Authenticate.reddit, Authenticate.web for parameters and usage.

Usage

```
Authenticate(socialmedia, ...)
```

Arguments

socialmedia	Character string. Identifier for social media API to authenticate with. Supported social media are "twitter", "youtube", "reddit" and "web".
...	Optional parameters to pass to functions providied by supporting R packages that are used for social media API access.

Authenticate.reddit *Reddit API authentication*

Description

Reddit does not require authentication in this version of vosonSML.

Usage

```
## S3 method for class 'reddit'
Authenticate(socialmedia, ...)
```

Arguments

socialmedia	Character string. Identifier for social media API to authenticate, set to "reddit".
...	Additional parameters passed to function. Not used in this method.

Value

A credential object containing a $auth = NULL value and social media type descriptor $socialmedia set to "reddit". Object has the class names "credential" and "reddit".

Note

Even though reddit does not require authentication in this version of vosonSML the Authenticate function must still be called to set the socialmedia identifier. This is used to route to the appropriate social media Collect function.

Examples

```
## Not run:
# reddit authentication
redditAuth <- Authenticate("reddit")

## End(Not run)
```

`Authenticate.twitter` *Twitter API authentication*

Description

Twitter authentication uses OAuth and typically requires four developer API keys generated when you create a twitter app via the twitter developer web site. These keys and the authentication object produced are referenced as developer keys and developer access in this package. Software using developer access to the API can operate with full access to the API including within the user-context but with the understanding that it is an app. The **rtweet** package refers to this as bot authentication.

There is another method available commonly used by third-party apps in which an app can be authorized by a user to use the twitter API on their behalf. The implementation of this method in **vosonSML** does not require a developer account but does still require the user to have access to an apps two consumer API keys (generated by the app developer). The authentication object with token produced from this method allows the user to access the API within their own user-context and rate-limits.

If an individual has applied for and been granted Twitter API access they will also have a bearer token associated with their app. This token allows read-only access to the API but higher rate-limits so it is the most suited method for this package and data collection.

The twitter OAuth process is described here: https://developer.twitter.com/en/docs/authentication/overview.

Usage

```
## S3 method for class 'twitter'
Authenticate(
  socialmedia,
  appName,
  apiKey,
  apiSecret,
  accessToken,
  accessTokenSecret,
  bearerToken = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`socialmedia`	Character string. Identifier for social media API to authenticate, set to `"twitter"`.
`appName`	Character string. Registered twitter app name associated with the API keys.
`apiKey`	Character string. API consumer key to authenticate (also called API key).
`apiSecret`	Character string. API consumer secret to authenticate (also called API secret).
`accessToken`	Character string. API access token to authenticate.

accessTokenSecret	Character string. API access token secret to authenticate.
bearerToken	Character string. Twitter app bearer token. Default is NULL.
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

A credential object containing an access token $auth and social media type descriptor $socialmedia set to "twitter". Object has the class names "credential" and "twitter".

Note

vosonSML uses the **rtweet** package for twitter data collection and also gratefully acknowledges the techniques and code written by its authors reproduced in this package for creating twitter API access tokens.

Examples

```
## Not run:
# twitter API access using an app bearer token
app_auth <- Authenticate(
  "twitter",
  bearerToken = "xxxxxxxxxxxx"
)

# twitter authentication using developer app API keys
dev_auth <- Authenticate(
  "twitter",
  appName = "My App",
  apiKey = "xxxxxxxxxxxx",
  apiSecret = "xxxxxxxxxxxx",
  accessToken = "xxxxxxxxxxxx",
  accessTokenSecret = "xxxxxxxxxxxx"
)

# twitter user authentication via authorization of an app
# requires the apps consumer API keys apiKey and apiSecret parameters are
# equivalent to the apps consumer key and secret will open a web browser
# to twitter prompting the user to log in and authorize the app
user_auth <- Authenticate(
  "twitter",
  appName = "An App",
  apiKey = "xxxxxxxxxxxx",
  apiSecret = "xxxxxxxxxxxx"
)

## End(Not run)
```

Authenticate.web *Web crawler authentication*

Description

Web crawler does not require authentication in this version of vosonSML.

Usage

```
## S3 method for class 'web'
Authenticate(socialmedia, ...)
```

Arguments

socialmedia	Character string. Identifier for social media API to authenticate, set to "web".
...	Additional parameters passed to function. Not used in this method.

Value

A credential object containing a $auth = NULL value and social media type descriptor $socialmedia set to "web". Object has the class names "credential" and "web".

Note

Even though the web crawler does not require authentication in this version of vosonSML the Authenticate function must still be called to set the socialmedia identifier. This is used to route to the appropriate social media Collect function.

Examples

```
## Not run:
# web authentication
webAuth <- Authenticate("web")

## End(Not run)
```

Authenticate.youtube *YouTube API authentication*

Description

YouTube authentication uses OAuth2 and requires a Google Developer API key as described here: https://developers.google.com/youtube/v3/docs/.

Usage

```
## S3 method for class 'youtube'
Authenticate(socialmedia, apiKey, ...)
```

Arguments

socialmedia	Character string. Identifier for social media API to authenticate, set to "youtube".
apiKey	Character string. Google developer API key to authenticate.
...	Additional parameters passed to function. Not used in this method.

Value

A credential object containing an api key $auth and social media type descriptor $socialmedia set to "youtube". Object has the class names "credential" and "youtube".

Examples

```
## Not run:
# youtube authentication with google developer api key
myAPIKey <- "xxxxxxxxxxx"

youtubeAuth <- Authenticate("youtube", apiKey = myAPIKey)

## End(Not run)
```

---

auth_twitter_app *Twitter App API authentication*

Description

If an individual has applied for and been granted Twitter API access they will also have a bearer token associated with their app. This token allows read-only access to the API but higher rate-limits so it is the most suited method for this package and data collection.

Usage

```
auth_twitter_app(bearer, verbose = FALSE, ...)
```

Arguments

bearer	Character string. Twitter app bearer token.
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

A `credential` object containing an access token `$auth` and social media type descriptor `$socialmedia` set to `"twitter"`. Object has the class names `"credential"` and `"twitter"`.

Note

vosonSML uses the **rtweet** package for twitter data collection and also gratefully acknowledges the techniques and code written by its authors reproduced in this package for creating twitter API access tokens.

Examples

```
## Not run:
# twitter API access using an app bearer token
auth <- auth_twitter_app("xxxxxxxxxxxx")

## End(Not run)
```

---

auth_twitter_dev *Twitter Dev API authentication*

Description

Twitter authentication uses OAuth and typically requires four developer API keys generated when you create a twitter app via the twitter developer web site. These keys and the authentication object produced are referenced as developer keys and developer access in this package. Software using developer access to the API can operate with full access to the API including within the user-context but with the understanding that it is an app. The **rtweet** package refers to this as 'bot' authentication.

Usage

```
auth_twitter_dev(
  app_name = "r twitter app",
  api_key,
  api_secret,
  access_token,
  access_token_secret,
  verbose = FALSE,
  ...
)
```

Arguments

`app_name`	Character string. Registered twitter app name associated with the API keys.
`api_key`	Character string. API consumer key to authenticate (also called API key).
`api_secret`	Character string. API consumer secret to authenticate (also called API secret).

access_token	Character string. API access token to authenticate.
access_token_secret	Character string. API access token secret to authenticate.
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

A credential object containing an access token $auth and social media type descriptor $socialmedia set to "twitter". Object has the class names "credential" and "twitter".

Note

vosonSML uses the **rtweet** package for twitter data collection and also gratefully acknowledges the techniques and code written by its authors reproduced in this package for creating twitter API access tokens.

Examples

```
## Not run:
# twitter authentication using developer app API keys
auth <- auth_twitter_dev(
  app_name = "My App",
  api_key = "xxxxxxxxxxxx",
  api_secret = "xxxxxxxxxxxx",
  access_token = "xxxxxxxxxxxx",
  access_token_secret = "xxxxxxxxxxxx"
)

## End(Not run)
```

---

auth_twitter_user *Twitter User API authentication*

Description

A method commonly used by third-party apps in which an app can be authorized by a user to use the twitter API on their behalf. The implementation of this method in **vosonSML** does not require a developer account but does still require the user to have access to an apps two consumer API keys (generated by the app developer). The authentication object with token produced from this method allows the user to access the API within their own user-context and rate-limits.

Usage

```
auth_twitter_user(api_key, api_secret, verbose = FALSE, ...)
```

Arguments

api_key	Character string. API consumer key to authenticate (also called API key).
api_secret	Character string. API consumer secret to authenticate (also called API secret).
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

A credential object containing an access token $auth and social media type descriptor $socialmedia set to "twitter". Object has the class names "credential" and "twitter".

Note

vosonSML uses the **rtweet** package for twitter data collection and also gratefully acknowledges the techniques and code written by its authors reproduced in this package for creating twitter API access tokens.

Examples

```
## Not run:
# twitter user authentication via authorization of an app requires the
# apps consumer API keys (api_key and api_secret)
# this method will open a web browser to twitter prompting the user to
# log in and authorize the app
auth <- auth_twitter_user(
  api_key = "xxxxxxxxxxxx", api_secret = "xxxxxxxxxxxx"
)

## End(Not run)
```

Collect *Collect data from social media for generating networks*

Description

This function collects data from social media and structures it into a dataframe that can be used for creating networks for further analysis. Collect is the second step of the Authenticate, Collect, and Create workflow.

Refer to Collect.search.twitter, Collect.timeline.twitter, Collect.youtube, Collect.reddit and Collect.web for parameters and usage.

Usage

```
Collect(credential, ...)
```

Arguments

credential	A credential object generated from Authenticate.
...	Optional parameters to pass to functions providied by supporting R packages that are used for social media API collection.

Collect.reddit *Collect comments data from reddit threads*

Description

Collects comments made by users on one or more specified subreddit conversation threads and structures the data into a dataframe with the class names "datasource" and "reddit".

Usage

```
## S3 method for class 'reddit'
Collect(
  credential,
  threadUrls,
  waitTime = c(3, 5),
  ua = getOption("HTTPUserAgent"),
  writeToFile = FALSE,
  verbose = FALSE,
  ...
)

collect_reddit_threads(
  threadUrls,
  waitTime = c(3, 5),
  ua = getOption("HTTPUserAgent"),
  writeToFile = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

credential	A credential object generated from Authenticate with class name "reddit".
threadUrls	Character vector. Reddit thread urls to collect data from.
waitTime	Numeric vector. Time range in seconds to select random wait from in-between url collection requests. Minimum is 3 seconds. Default is c(3, 5) for a wait time chosen from between 3 and 5 seconds.
ua	Character string. Override User-Agent string to use in Reddit thread requests. Default is option("HTTPUserAgent") value as set by vosonSML.
writeToFile	Logical. Write collected data to file. Default is FALSE.

verbose	Logical. Output additional information about the data collection. Default is TRUE.
...	Additional parameters passed to function. Not used in this method.

Value

A tibble object with class names "datasource" and "reddit".

Note

The reddit web endpoint used for collection has maximum limit of 500 comments per thread url.

Examples

```
## Not run:
# subreddit url to collect threads from
threadUrls <- c("https://www.reddit.com/r/xxxxxx/comments/xxxxxx/x_xxxx_xxxxxxxxx/")

redditData <- redditAuth |>
  Collect(threadUrls = threadUrls, writeToFile = TRUE)

## End(Not run)
```

Collect.search.twitter

Collect tweet data from twitter search

Description

This function collects tweet data based on search terms and structures the data into a dataframe with the class names "datasource" and "twitter".

The twitter Standard search API sets a rate limit of 180 requests every 15 minutes. A maximum of 100 tweets can be collected per search request meaning the maximum number of tweets per operation is 18000 / 15 minutes. More tweets can be collected by using retryOnRateLimit = TRUE parameter which will cause the collection to pause if the rate limit is reached and resume when the rate limit resets (in approximately 15 minutes). Alternatively the twitter API parameter since_id can be used in a later session to resume a twitter search collection from the last tweet previously collected as tweet status id's are sequential. The Standard API only returns tweets for the last 7 days.

All of the search query operators available through the twitter API can be used in the searchTerm field. For example, to search for tweets containing the term "love" or "hate" the "OR" operator can be used in the term field: searchTerm = "love OR hate". For more information refer to the twitter API documentation for query operators: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/guides/standard-operators.

Usage

```
## S3 method for class 'search.twitter'
Collect(
  credential,
  endpoint,
  searchTerm = "",
  searchType = "recent",
  numTweets = 100,
  includeRetweets = TRUE,
  retryOnRateLimit = TRUE,
  writeToFile = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

credential	A credential object generated from Authenticate with class name "twitter".
endpoint	API endpoint.
searchTerm	Character string. Specifies a twitter search term. For example, "Australian politics" or the hashtag "#auspol".
searchType	Character string. Returns filtered tweets as per search type recent, mixed or popular. Default type is recent.
numTweets	Numeric. Specifies how many tweets to be collected. Defaults is 100.
includeRetweets	Logical. Specifies if the search should filter out retweets. Defaults is TRUE.
retryOnRateLimit	Logical. When the API rate-limit is reached should the collection wait and resume when it resets. Default is TRUE.
writeToFile	Logical. Write collected data to file. Default is FALSE.
verbose	Logical. Output additional information. Default is FALSE.
...	Arguments passed on to rtweet::search_tweets geocode Geographical limiter of the template "latitude,longitude,radius" e.g., geocode = "37.78,-122.40,1mi". since_id Supply a vector of ids or a data frame of previous results to find tweets **newer** than since_id. max_id Supply a vector of ids or a data frame of previous results to find tweets **older** than max_id. parse If TRUE, the default, returns a tidy data frame. Use FALSE to return the "raw" list corresponding to the JSON returned from the Twitter API.

Value

A tibble object with class names "datasource" and "twitter".

Note

Additional parameters passed to this function in the ellipsis ... will also be passed to the Twitter search API request. Most parameters have been covered but a complete list can be found here: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets A useful additional parameter is language allowing the user can restrict tweets returned to a particular language using an ISO 639-1 code. For example, to restrict a search to tweets in English the value language = "en" can be passed to this function.

Examples

```
## Not run:
# search and collect 100 recent tweets for the hashtag #auspol
myTwitterData <- twitterAuth |>
  Collect(searchTerm = "#auspol", searchType = "recent", numTweets = 100, verbose = TRUE,
          includeRetweets = FALSE, retryOnRateLimit = TRUE, writeToFile = TRUE)

## End(Not run)
```

Collect.timeline.twitter

Collect tweet data from twitter timelines

Description

This function collects user timeline tweets and structures the data into a dataframe with the class names "datasource" and "twitter". The Twitter API limits collection to a maximum of 3,200 of the most recent timeline tweets per user.

Usage

```
## S3 method for class 'timeline.twitter'
Collect(
  credential,
  endpoint,
  users = c(),
  numTweets = 100,
  retryOnRateLimit = TRUE,
  writeToFile = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

credential	A credential object generated from Authenticate with class name "twitter".
endpoint	API endpoint.

`users` Character vector. Specifies one or more twitter users. Can be user names, user ids or a mixture.

`numTweets` Numeric vector. Specifies how many tweets to be collected per user. Defaults to single value of `100`.

`retryOnRateLimit`
Logical. When the API rate-limit is reached should the collection wait and resume when it resets. Default is `TRUE`.

`writeToFile` Logical. Write collected data to file. Default is `FALSE`.

`verbose` Logical. Output additional information about the data collection. Default is `FALSE`.

`...` Arguments passed on to `rtweet::get_timeline`

`since_id` Supply a vector of ids or a data frame of previous results to find tweets **newer** than `since_id`.

`parse` If `TRUE`, the default, returns a tidy data frame. Use `FALSE` to return the "raw" list corresponding to the JSON returned from the Twitter API.

`check` **[Deprecated]**

`retryonratelimit` If `TRUE`, and a rate limit is exhausted, will wait until it refreshes. Most Twitter rate limits refresh every 15 minutes. If `FALSE`, and the rate limit is exceeded, the function will terminate early with a warning; you'll still get back all results received up to that point. The default value, `NULL`, consults the option `rtweet.retryonratelimit` so that you can globally set it to `TRUE`, if desired.
If you expect a query to take hours or days to perform, you should not rely soley on `retryonratelimit` because it does not handle other common failure modes like temporarily losing your internet connection.

Value

A tibble object with class names `"datasource"` and `"twitter"`.

`Collect.web` *Collect hyperlinks from web pages*

Description

Collects hyperlinks from web pages and structures the data into a dataframe with the class names `"datasource"` and `"web"`.

Usage

```
## S3 method for class 'web'
Collect(credential, pages = NULL, writeToFile = FALSE, verbose = FALSE, ...)

collect_web_hyperlinks(pages = NULL, writeToFile = FALSE, verbose = FALSE, ...)
```

Arguments

credential	A credential object generated from Authenticate with class name "web".
pages	Dataframe. Dataframe of web pages to crawl. The dataframe must have the columns page (character), type (character) and max_depth (integer). Each row is a seed web page to crawl, with the page value being the page URL. The type value is type of crawl as either "int", "ext" or "all", directing the crawler to follow only internal links, follow only external links (different domain to the seed page) or follow all links. The max_depth value determines how many levels of hyperlinks to follow from the seed site.
writeToFile	Logical. Write collected data to file. Default is FALSE.
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

A tibble object with class names "datasource" and "web".

Examples

```
## Not run:
pages <- tibble::tibble(page = c("http://vosonlab.net",
                                 "https://rsss.cass.anu.edu.au"),
                        type = c("int", "all"),
                        max_depth = c(2, 2))

webData <- webAuth |>
  Collect(pages, writeToFile = TRUE)

## End(Not run)
```

Collect.youtube *Collect comments data for YouTube videos*

Description

This function collects public comments data for one or more YouTube videos using the YouTube Data API v3 and structures the data into a dataframe with the class names "datasource" and "youtube".

YouTube has a quota unit system as a rate limit with most developers having either 10,000 or 1,000,000 units per day. Many read operations cost a base of 1 unit such as retrieving individual comments, plus 1 or 2 units for text snippets. Retrieving threads or top-level comments with text costs 3 units per request (maximum 100 comments per request). Using this function a video with 250 top-level comments and 10 of those having reply comments of up to 100 each, should cost (9 + 20) 29 quota units and return between 260 and 1260 total comments. There is currently a limit of 100 reply comments collected per top-level comment.

More information about the YouTube Data API v3 can be found here: https://developers.google.com/youtube/v3/getting-started

Usage

```
## S3 method for class 'youtube'
Collect(
  credential,
  videoIDs = c(),
  verbose = FALSE,
  writeToFile = FALSE,
  maxComments = 1e+10,
  ...
)
```

Arguments

credential	A credential object generated from Authenticate with class name "youtube".
videoIDs	Character vector. Specifies YouTube video URLs or IDs. For example, if the video URL is https://www.youtube.com/watch?v=xxxxxxxxxxx then use URL or ID videoIDs = c("xxxxxxxxxxx").
verbose	Logical. Output additional information about the data collection. Default is FALSE.
writeToFile	Logical. Write collected data to file. Default is FALSE.
maxComments	Numeric integer. Specifies how many top-level comments to collect from each video. This value does not consider replies to top-level comments. The total number of comments returned for a video will usually be greater than maxComments depending on the number of reply comments present.
...	Additional parameters passed to function. Not used in this method.

Value

A tibble object with class names "datasource" and "youtube".

Note

Due to specifications of the YouTube Data API it is currently not efficient to specify the exact number of comments to return from the API using maxComments parameter. The maxComments parameter is applied to top-level comments only and not the replies to these comments. As such the number of comments collected is usually greater than expected. For example, if maxComments is set to 10 and one of the videos 10 top-level comments has 5 reply comments then the total number of comments collected will be 15 for that video. Comments data for multiple YouTube videos can be requested in a single operation, maxComments is applied to each individual video and not the combined total of comments.

Examples

```
## Not run:
# list of YouTube video urls or ids to collect
video_ids <- c("https://www.youtube.com/watch?v=xxxxxxxx",
               "https://youtu.be/xxxxxxxx",
               "xxxxxxx")
```

```
# collect approximately 200 threads/comments for each YouTube video
youtubeData <- youtubeAuth |>
  Collect(videoIDs = video_ids, writeToFile = TRUE, verbose = FALSE, maxComments = 200)

## End(Not run)
```

`Create` *Create networks from social media data*

Description

This function creates networks from social media data as produced from `Collect`. `Create` is the final step of the `Authenticate`, `Collect` and `Create` workflow.

There are four types of networks that can be created from collected data: `activity`, `actor`, `twomode` or `semantic`.

For `activity` networks refer to `Create.activity.twitter`, `Create.activity.youtube` and `Create.activity.reddit` for parameters and usage.

For `actor` networks refer to `Create.actor.twitter`, `Create.actor.youtube` and `Create.actor.reddit`.

For `twomode` and `semantic` networks refer to `Create.twomode.twitter` and `Create.semantic.twitter` functions for parameters and usage respectively.

Usage

```
Create(datasource, type, ...)
```

Arguments

`datasource`	Collected social media data of class `"datasource"` and `socialmedia`.
`type`	Character string. Type of network to be created, can be `"activity"`, `"actor"`, `"twomode"` or `"semantic"`.
`...`	Optional parameters to pass to functions providied by supporting R packages that are used for social media network creation.

`Create.activity.reddit` *Create reddit activity network*

Description

Creates a reddit activity network from subreddit thread comments. Nodes are comments and initial thread posts, edges form the discussion structure and signify to which comment or post a comment has been made to.

Usage

```
## S3 method for class 'activity.reddit'
Create(datasource, type, verbose = TRUE, ...)
```

Arguments

`datasource`	Collected social media data with `"datasource"` and `"reddit"` class names.
`type`	Character string. Type of network to be created, set to `"activity"`.
`verbose`	Logical. Output additional information about the network creation. Default is `TRUE`.
`...`	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing `$nodes` and `$edges`.

Examples

```
## Not run:
# create a reddit activity network graph
activityNetwork <- redditData |> Create("activity")

# network
# activityNetwork$nodes
# activityNetwork$edges

## End(Not run)
```

Create.activity.twitter *Create twitter activity network*

Description

Creates a twitter activity network from collected tweets. Nodes are tweets and directed edges represent the relationship of tweets to one another. For example, there is a directed edge from a quote tweet towards the tweet that was quoted. Stand-alone tweets that are not replies, retweets or quote tweets have no relation to others and will be isolates.

Usage

```
## S3 method for class 'activity.twitter'
Create(datasource, type, rmEdgeTypes = NULL, verbose = FALSE, ...)
```

Arguments

datasource	Collected social media data with "datasource" and "twitter" class names.
type	Character string. Type of network to be created, set to "activity".
rmEdgeTypes	Character vector. List of tweet edge types to remove from network. Options are "tweet", "retweet", "reply" and "quote". Default is NULL.
verbose	Logical. Output additional information. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges.

Examples

```
## Not run:
# create a twitter activity network with retweets removed
activity_net <- twitter_data |>
  Create("activity", rmEdgeTypes = c("retweet"))

# network nodes and edges
names(activity_net)
# "nodes", "edges"
names(activity_net$nodes)
# "status_id", "author_id", "author_screen_name", "created_at"
names(activity_net$edges)
# "from", "to", "user_id", "screen_name", "created_at", "edge_type"

## End(Not run)
```

`Create.activity.web` *Create web activity network*

Description

Creates a web page activity network from pages. Nodes are web pages.

Usage

```
## S3 method for class 'activity.web'
Create(datasource, type, lcase = TRUE, verbose = TRUE, ...)
```

Arguments

`datasource`	Collected social media data with `"datasource"` and `"web"` class names.
`type`	Character string. Type of network to be created, set to `"activity"`.
`lcase`	Logical. Convert urls and page names to lowercase.
`verbose`	Logical. Output additional information about the network creation. Default is `TRUE`.
`...`	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing `$nodes` and `$edges`.

Examples

```
## Not run:
# create a web activity network graph
net_activity <- data_collect |> Create("activity")

# network
# net_activity$nodes
# net_activity$edges

## End(Not run)
```

`Create.activity.youtube` *Create YouTube activity network*

Description

Creates an activity network from collected YouTube video comment threads. Nodes are top-level comments, reply comments and videos. Edges are directed between the nodes and represent commenting activity.

Usage

```
## S3 method for class 'activity.youtube'
Create(datasource, type, verbose = TRUE, ...)
```

Arguments

`datasource`	Collected social media data with `"datasource"` and `"youtube"` class names.
`type`	Character string. Type of network to be created, set to `"activity"`.
`verbose`	Logical. Output additional information about the network creation. Default is `TRUE`.
`...`	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing `$nodes` and `$edges`.

Examples

```
## Not run:
# create a YouTube activity network graph
activityNetwork <- youtubeData |> Create("activity")

# network
# activityNetwork$nodes
# activityNetwork$edges

## End(Not run)
```

Create.actor.reddit *Create reddit actor network*

Description

Creates a reddit actor network from thread comments on subreddits. Users who have commented on a thread are actor nodes and comment replies to each other are represented as directed edges.

Usage

```
## S3 method for class 'actor.reddit'
Create(datasource, type, ...)
```

Arguments

datasource	Collected social media data with "datasource" and "reddit" class names.
type	Character string. Type of network to be created, set to "actor".
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges.

Examples

```
## Not run:
# create a reddit actor network graph with comment text as edge attributes
actorNetwork <- redditData |> Create("actor")

# network
# actorNetwork$nodes
# actorNetwork$edges

## End(Not run)
```

Create.actor.twitter *Create twitter actor network*

Description

Creates a twitter actor network from tweets returned from the twitter search query. Twitter users who have tweeted, retweeted or been mentioned in a tweet are actor nodes. The created network is directed with edges of different types representing retweets, quote tweets, mentions and replies to other users. Users who have tweeted without ties to other users will appear in the network graph as nodes with self-loops.

Usage

```
## S3 method for class 'actor.twitter'
Create(
  datasource,
  type,
  rmEdgeTypes = NULL,
  inclMentions = TRUE,
  inclRtMentions = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

datasource	Collected social media data with "datasource" and "twitter" class names.
type	Character string. Type of network to be created, set to "actor".
rmEdgeTypes	Character vector. List of edge types to remove from network. Options are "tweet", "retweet", "reply" and "quote". Default is NULL.
inclMentions	Logical. Create edges for users mentioned or tagged in tweets. Default is TRUE.
inclRtMentions	Logical. Create edges for users mentioned or tagged in retweets. For tweet types other than retweets the collected tweet author has created the mention, for retweets the original tweet author has created the mention not the retweeter. Default is FALSE.
verbose	Logical. Output additional information about the network creation. Default is FALSE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges.

Note

When creating twitter actor networks, a network with additional user information can be generated using the AddUserData function. Additional calls can be made to the twitter API to get information about users that were identified as nodes during network creation but did not tweet (meaning no user profile information was initially collected for them).

Examples

```
## Not run:
# create a twitter actor network excluding retweet, quote tweets and mention edges
actor_net <- twitter_data |>
  Create("actor", rmEdgeTypes = c("retweet", "quote"))

# network nodes and edges
names(actor_net)
# "nodes", "edges"
```

```
names(actor_net$nodes)
# "user_id", "screen_name"
names(actor_net$edges)
# "from", "to", "status_id", "created_at", "edge_type"

## End(Not run)
```

`Create.actor.web` *Create web actor network*

Description

Creates a web page domain network from pages. Nodes are site domains.

Usage

```
## S3 method for class 'actor.web'
Create(datasource, type, verbose = TRUE, ...)
```

Arguments

`datasource`	Collected social media data with `"datasource"` and `"web"` class names.
`type`	Character string. Type of network to be created, set to `"activity"`.
`verbose`	Logical. Output additional information about the network creation. Default is `TRUE`.
`...`	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing `$nodes` and `$edges`.

Examples

```
## Not run:
# create a web actor network graph
net_activity <- data_collect |> Create("actor")

# network
# net_activity$nodes
# net_activity$edges

## End(Not run)
```

`Create.actor.youtube` *Create YouTube actor network*

Description

Creates a YouTube actor network from comment threads on YouTube videos. Users who have made comments to a video (top-level comments) and users who have replied to those comments are actor nodes. The comments are represented as directed edges between the actors. The video id is also included as an actor node, representative of the videos publisher with top-level comments as directed edges towards them.

Usage

```
## S3 method for class 'actor.youtube'
Create(datasource, type, ...)
```

Arguments

`datasource`	Collected social media data with `"datasource"` and `"youtube"` class names.
`type`	Character string. Type of network to be created, set to `"actor"`.
`...`	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing `$nodes` and `$edges`.

Examples

```
## Not run:
# create a YouTube actor network graph
actorNetwork <- youtubeData |> Create("actor")

# network
# actorNetwork$nodes
# actorNetwork$edges

## End(Not run)
```

`Create.semantic.twitter` *Create twitter semantic network*

Description

Creates a semantic network from tweets returned from the twitter search query. Semantic networks describe the semantic relationships between concepts. In this network the concepts are significant words and hashtags extracted from the tweet text. Network edges are weighted and represent occurrence of words and hashtags in the same tweets.

The creation of twitter semantic networks requires text processing and the tokenization of tweets. As such this function requires the additional installation of the **tidyr** and **tidytext** packages to achieve this.

Usage

```
## S3 method for class 'semantic.twitter'
Create(
  datasource,
  type,
  removeRetweets = TRUE,
  removeTermsOrHashtags = NULL,
  stopwords = TRUE,
  stopwordsLang = "en",
  stopwordsSrc = "smart",
  removeNumbers = TRUE,
  removeUrls = TRUE,
  termFreq = 5,
  hashtagFreq = 50,
  assoc = "limited",
  verbose = TRUE,
  ...
)
```

Arguments

`datasource`	Collected social media data with `"datasource"` and `"twitter"` class names.
`type`	Character string. Type of network to be created, set to `"semantic"`.
`removeRetweets`	Logical. Removes detected retweets from the tweet data. Default is `TRUE`.
`removeTermsOrHashtags`	Character vector. Words or hashtags to remove from the semantic network. For example, this parameter could be used to remove the search term or hashtag that was used to collect the data by removing any nodes with matching name. Default is `NULL` to remove none.
`stopwords`	Logical. Removes stopwords from the tweet data. Default is `TRUE`.

stopwordsLang	Character string. Language of stopwords to use. Refer to the **stopwords** package for further information on supported languages. Default is "en".
stopwordsSrc	Character string. Source of stopwords list. Refer to the **stopwords** package for further information on supported sources. Default is "smart".
removeNumbers	Logical. Removes whole numerical tokens from the tweet text. For example, a year value such as 2020 will be removed but not mixed values such as G20. Default is TRUE.
removeUrls	Logical. Removes twitter shortened URL tokens from the tweet text. Default is TRUE.
termFreq	Numeric integer. Specifies the percentage of most frequent words to include. For example, termFreq = 20 means that the 20 percent most frequently occurring words will be included in the semantic network as nodes. A larger percentage will increase the number of nodes and therefore the size of graph. The default value is 5, meaning the top 5 percent most frequent words are used.
hashtagFreq	Numeric integer. Specifies the percentage of most frequent hashtags to include. For example, hashtagFreq = 20 means that the 20 percent most frequently occurring hashtags will be included in the semantic network as nodes. The default value is 50.
assoc	Character string. Association of nodes. A value of "limited" includes only edges between most frequently occurring hashtags and terms. A value of "full" includes ties between most frequently occurring hashtags and terms, hashtags and hashtags, and terms and terms. Default is "limited".
verbose	Logical. Output additional information about the network creation. Default is TRUE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges.

Note

The words and hashtags passed to the function in the removeTermsOrHashtags parameter are removed before word frequencies are calculated and are therefore excluded from top percentage of most frequent terms completely rather than simply filtered out of the final network.

The top percentage of frequently occurring hashtags hashtagFreq and words termFreq are calculated to a minimum frequency and all terms that have an equal or greater frequency than the minimum are included in the network as nodes. For example, of unique hashtags of varying frequencies in a dataset the top 50% of total frequency or most common hashtags may calculate to being the first 20 hashtags. The frequency of the 20th hashtag is then used as the minimum and all hashtags of equal or greater frequency are included as part of the top 50% most frequently occurring hashtags. So the number of top hashtags may end up being greater than 20 if there is more than one hashtag that has frequency matching the minimum. The exception to this is if the minimum frequency is 1 and the hashtagFreq is set to less than 100, in this case only the first 20 hashtags will be included.

Hashtags and words in the top percentages are included in the network as isolates if there are no instances of them occurring in tweet text with other top percentage frequency terms.

Examples

```
## Not run:
# twitter semantic network creation additionally requires the tidytext
# and stopwords packages for working with text data
# install.packages(c("tidytext", "stopwords"))

# create a twitter semantic network graph removing the hashtag "#auspol"
# and using the top 2% frequently occurring words and 10% most frequently
# occurring hashtags as nodes
net_semantic <- collect_tw |>
  Create("semantic", removeTermsOrHashtags = c("#auspol"),
         termFreq = 2, hashtagFreq = 10, verbose = TRUE)

# network
# net_semantic$nodes
# net_semantic$edges

## End(Not run)
```

`Create.twomode.twitter`

Create twitter 2-mode network

Description

Creates a 2-mode network from tweets returned from the twitter search query. In this network there are two types of nodes, twitter users who authored or were mentioned in collected tweets and hashtags found within tweets. Network edges represent a users tweets that contain hashtags or mention users screen names.

The creation of twitter 2-mode networks requires text processing and the tokenization of tweets. As such this function requires the additional installation of the **tidytext** package to achieve this.

Usage

```
## S3 method for class 'twomode.twitter'
Create(
  datasource,
  type,
  removeTermsOrHashtags = NULL,
  rmRetweets = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

datasource	Collected social media data with "datasource" and "twitter" class names.
type	Character string. Type of network to be created, set to "twomode".
removeTermsOrHashtags	Character vector. Users or hashtags to remove from the twomode network. For example, this parameter could be used to remove the user or hashtag that was used to collect the data by removing any nodes with matching name. Default is NULL to remove none.
rmRetweets	Logical. Do not process retweets in the input data. Default is TRUE.
verbose	Logical. Output additional information about the network creation. Default is TRUE.
...	Additional parameters passed to function. Not used in this method.

Value

Network as a named list of two dataframes containing $nodes and $edges.

Examples

```
## Not run:
# twitter 2-mode network creation additionally requires the tidytext
# package for working with text data
# install.packages("tidytext")

# create a twitter 2-mode network graph with the hashtag "#auspol" removed
net_2mode <- collect_tw |>
  Create("twomode", removeTermsOrHashtags = c("#auspol"), verbose = TRUE)

# network
# net_2mode$nodes
# net_2mode$edges

## End(Not run)
```

Graph *Create an igraph graph from network*

Description

Create an igraph graph from network

Usage

```
Graph(net, directed = TRUE, writeToFile = FALSE, verbose = FALSE, ...)
```

Arguments

`net`	A named list of dataframes `nodes` and `edges` generated by `Create`.
`directed`	Logical. Create a directed graph. Default is `TRUE`.
`writeToFile`	Logical. Save graph to a file in the current working directory. Default is `FALSE`.
`verbose`	Logical. Output additional information. Default is `FALSE`.
`...`	Additional parameters passed to function. Not used in this method.

Value

An igraph object.

`ImportRtweet` *Import rtweet collected data*

Description

Imports **rtweet** collected data from `rda` or `rds` saved object file or from an rtweet dataframe. Ensures `datasource` and specified `socialmedia` type are set so data is usable by `Create` functions. Not required if collected data was collected by `vosonSML` and saved as an `rds` file, use `readRDS` instead.

Usage

```
ImportRtweet(data)
```

Arguments

`data`	Character string or dataframe. File path to or tibble of collected data from **rtweet**.

Value

A dataframe suitable for input into twitter network `Create` functions.

Note

Only supports **rtweet** data collected using the `search_tweets` or `get_timeline` functions.

Examples

```
## Not run:
# import rtweet collected data from dataframe
collect_tw <- ImportRtweet(rtweet_search_data)

# import rtweet collected data from file
collect_tw <- ImportRtweet("./rtweet_search_n100.rds")

## End(Not run)
```

Merge *Merge collected data*

Description

Merge collected data

Usage

```
Merge(..., unique = TRUE, rev = TRUE, writeToFile = FALSE, verbose = FALSE)

merge_data(
  ...,
  unique = TRUE,
  rev = TRUE,
  writeToFile = FALSE,
  verbose = FALSE
)
```

Arguments

...	Collect data to merge.
unique	Logical. Remove duplicates based on observation id. Default is TRUE.
rev	Logical. Reverses order of observations before removing duplicates. If collect data is provided chronologically then this should ensure the most recent copy of a duplicate is kept. Default is TRUE.
writeToFile	Logical. Save data to a file in the current working directory. Default is FALSE.
verbose	Logical. Output additional information. Default is FALSE.

Value

A merged Collect object.

MergeFiles *Merge collected data files*

Description

Merge collected data files

Usage

```
MergeFiles(
  path = ".",
  pattern = "(?-i).+?\\.rds$",
  unique = TRUE,
  rev = TRUE,
  writeToFile = FALSE,
  verbose = FALSE
)

merge_files(
  path = ".",
  pattern = "(?-i).+?\\.rds$",
  unique = TRUE,
  rev = TRUE,
  writeToFile = FALSE,
  verbose = FALSE
)
```

Arguments

path	Directory path of Collect data to merge. Default is the working directory.
pattern	Regular expression (regex) for matching file names to merge.
unique	Logical. Remove duplicates based on observation id. Default is TRUE.
rev	Logical. Reverses order of observations before removing duplicates. If collect data is provided chronologically then this should ensure the most recent copy of a duplicate is kept. Default is TRUE.
writeToFile	Logical. Save data to a file in the current working directory. Default is FALSE.
verbose	Logical. Output additional information. Default is FALSE.

Value

A merged Collect object.

Index

add_text *(AddText)*, 3
add_users *(AddUserData)*, 9
add_videos *(AddVideoData)*, 13
AddText, 3
AddText.activity.reddit, *3*, 4
AddText.activity.twitter, 5
AddText.activity.youtube *(AddText.actor.youtube)*, 7
AddText.actor.reddit, *3*, 5
AddText.actor.twitter, 6
AddText.actor.youtube, *3*, 7
AddText.semantic.twitter, 8
AddText.twomode.twitter, 9
AddUserData, 9, *36*
AddUserData.actor.twitter, *10*, 10
AddUserData.twomode.twitter, 11
AddVideoData, 13
AddVideoData.actor.youtube, *13*, 13
auth_twitter_app, 19
auth_twitter_dev, 20
auth_twitter_user, 21
Authenticate, 14, *22*, *30*
Authenticate.reddit, *14*, 15
Authenticate.twitter, *14*, 16
Authenticate.web, *14*, 18
Authenticate.youtube, *14*, 18

Collect, *14*, 22, *30*
Collect.reddit, *22*, 23
Collect.search.twitter, *22*, 24
Collect.timeline.twitter, *22*, 26
Collect.web, *22*, 27
Collect.youtube, *22*, 28
collect_reddit_threads *(Collect.reddit)*, 23
collect_web_hyperlinks *(Collect.web)*, 27
Create, *14*, *22*, 30, *43*
Create.activity.reddit, *30*, 31
Create.activity.twitter, *30*, 32
Create.activity.web, 33
Create.activity.youtube, *30*, 34
Create.actor.reddit, *30*, 35
Create.actor.twitter, *30*, 35
Create.actor.web, 37
Create.actor.youtube, *30*, 38
Create.semantic.twitter, *30*, 39
Create.twomode.twitter, *30*, 41

get_timeline, *43*
Graph, 42

ImportRtweet, 43

Merge, 44
merge_data *(Merge)*, 44
merge_files *(MergeFiles)*, 44
MergeFiles, 44

readRDS, *43*
rtweet::get_timeline, *27*
rtweet::search_tweets, *25*

search_tweets, *43*