Package: academictwitteR (via r-universe)

September 3, 2024

Title Access the Twitter Academic Research Product Track V2 API Endpoint

Version 0.3.1

Description Package to query the Twitter Academic Research Product Track, providing access to full-archive search and other v2 API endpoints. Functions are written with academic research in mind. They provide flexibility in how the user wishes to store collected data, and encourage regular storage of data to mitigate loss when collecting large volumes of tweets. They also provide workarounds to manage and reshape the format in which data is provided on the client side.

License MIT + file LICENSE

URL https://github.com/cjbarrie/academictwitteR

BugReports https://github.com/cjbarrie/academictwitteR/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Depends R (>= 3.4)

Imports dplyr (>= 1.0.0), httr, jsonlite, magrittr, lubridate, usethis, tibble, tidyr, tidyselect, purrr, rlang, utils, stringr, future, furrr

Suggests knitr, rmarkdown, devtools, testthat (>= 3.0.0), httptest, lifecycle, covr

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://cjbarrie.r-universe.dev

RemoteUrl https://github.com/cjbarrie/academictwitter

RemoteRef HEAD

RemoteSha 9b573fe799c56353362dddc94de442852da67266

Contents

`bind_tweets` *Bind information stored as JSON files*

Description

This function binds information stored as JSON files. The experimental function `convert_json` converts individual JSON files into either "raw" or "tidy" format.

Usage

```
bind_tweets(
  data_path,
  user = FALSE,
  verbose = TRUE,
  output_format = NA,
 vars = c("text", "user", "tweet_metrics", "user_metrics", "hashtags", "ext_urls",
    "mentions", "annotations", "context_annotations"),
  quoted_variables = FALSE
)

convert_json(
  data_file,
```

```
  output_format = "tidy",
 vars = c("text", "user", "tweet_metrics", "user_metrics", "hashtags", "ext_urls",
    "mentions", "annotations", "context_annotations"),
  quoted_variables = F
)
```

Arguments

data_path	string, file path to directory of stored tweets data saved as data_*id*.json and users_*id*.json
user	If FALSE, this function binds JSON files into a data frame containing tweets; data frame containing user information otherwise. Ignore if output_format is not NA
verbose	If FALSE, messages are suppressed
output_format	**[Experimental]** string, if it is not NA, this function return an unprocessed data.frame containing either tweets or user information. Currently, this function supports the following format(s) • "raw"List of data frames; Note: not all data frames are in Boyce-Codd 3rd Normal Form • "tidy"Tidy format; all essential columns are available • "tidy2"Tidy format; additional variables (see vars) are available. Untruncates retweet text and adds indicators for retweets, quotes and replies. Automatically drops duplicated tweets. Handling of quoted tweets can be specified (see quoted_variables)
vars	**[Experimental]** vector of strings, determining the variables provided by the tidy2 format. Can be any (or all) of the following: • "text"Text of the tweet, including language classification, indicator of sensitive content and (if applicable) sourcetweet text • "user"Information on the user in addition to their ID • "tweet_metrics"Tweet metrics, specifically the like, retweet and quote counts • "user_metrics"User metrics, specifically their tweet, list, follower and following counts • "hashtags"Hashtags contained in the tweet. Untrunctated for retweets • "ext_urls"Shortened and expanded URLs contained in the tweet, excluding those internal to Twitter (e.g. retweet URLs). Includes additional data provided by Twitter, such as the unwound URL, their title and description (if available). Untrunctated for retweets • "mentions"Mentioned usernames and their IDs, excluding retweeted users. Untrunctated for retweets. Note that quoted users are only mentioned here if explicitly named in the tweet text. This was usually the case with older versions of Twitter, but is no longer the standard behaviour. Extracting mentions allows the usernames of the RT authors (rather than only their ID) to be preserved • "annotations"Annotations provided by Twitter, including their probability and type. Basically Named Entities. See https://developer.twitter.com/en/docs/twitter-api/annotations/overview for details

- "context_annotations"Context annotations provided by Twitter, including additional data on their domains. See https://developer.twitter.com/en/docs/twitter-api/annotations/overview for details

`quoted_variables`
: **[Experimental]** Should additional vars be returned for the quoted tweet? Defaults to FALSE. TRUE returns additional "_quoted" var-columns containing the vars (mentions, hashtags, etc.) of the quoted tweet in addition to the actual tweet's data

`data_file`
: string, a single file path to a JSON file; or a vector of file paths to JSON files of stored tweets data saved as data_*id*.json

Details

By default, `bind_tweets` binds into a data frame containing tweets (from data_*id*.json files).

If users is TRUE, it binds into a data frame containing user information (from users_*id*.json).

For the "tidy" and "tidy2" format, parallel processing with furrr is supported. In order to enable parallel processing, workers need to be set manually through `future::plan()`. See examples

Note that output of the tidy2 vars returns results of the Twitter API, rather than from tweet text. Therefore, certain variables, especially context annotations and quoted_variables, may not be present in older data.

Value

a data.frame containing either tweets or user information

Examples

```
## Not run:
# bind json files in the directory "data" into a data frame containing tweets
bind_tweets(data_path = "data/")

# bind json files in the directory "data" into a data frame containing user information
bind_tweets(data_path = "data/", user = TRUE)

# bind json files in the directory "data" into a "tidy" data frame / tibble
bind_tweets(data_path = "data/", user = TRUE, output_format = "tidy")

# bind json files in the directory "data" into a "tidy2" data frame / tibble, get hashtags and
# URLs for both original and quoted tweets
bind_tweets(data_path = "data/", user = TRUE, output_format = "tidy2",
            vars = c("hashtags", "ext_urls"),
            quoted_variables = T)

# bind json files in the directory "data" into a "tidy2" data frame / tibble with parallel computing
## set up a multisession
future::plan("multisession")
## run the function - note that no additional arguments are required
bind_tweets(data_path = "data/", user = TRUE, output_format = "tidy2")
## Shut down parallel workers
future::plan("sequential")
```

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

`build_query` *Build tweet query*

Description

Build tweet query according to targeted parameters.

Usage

```
build_query(
  query = NULL,
  exact_phrase = NULL,
  users = NULL,
  reply_to = NULL,
  retweets_of = NULL,
  exclude = NULL,
  is_retweet = NULL,
  is_reply = NULL,
  is_quote = NULL,
  is_verified = NULL,
  remove_promoted = FALSE,
  has_hashtags = NULL,
  has_cashtags = NULL,
  has_links = NULL,
  has_mentions = NULL,
  has_media = NULL,
  has_images = NULL,
  has_videos = NULL,
  has_geo = NULL,
  place = NULL,
  country = NULL,
  point_radius = NULL,
  bbox = NULL,
  lang = NULL,
  conversation_id = NULL,
  url = NULL
)
```

Arguments

`query`	string or character vector, search query or queries
`exact_phrase`	If `TRUE`, only tweets will be returned matching the exact phrase
`users`	string or character vector, user handles to collect tweets from the specified users

reply_to	string or character vector, user handles to collect replies to the specified users
retweets_of	string or character vector, user handles to collects retweets of tweets by the specified users
exclude	string or character vector, tweets containing the keyword(s) will be excluded
is_retweet	If TRUE, only retweets will be returned; if FALSE, retweets will be excluded; if NULL, both retweets and other tweet types will be returned.
is_reply	If TRUE, only replies will be returned; if FALSE, replies will be excluded; if NULL, both replies and other tweet types will be returned.
is_quote	If TRUE, only quote tweets will be returned; if FALSE, quote tweets will be excluded; if NULL, both quote tweets and other tweet types will be returned.
is_verified	If TRUE, only tweets from verified accounts will be returned; if FALSE, tweets from verified accounts will be excluded; if NULL, both verified account tweets and tweets from non-verified accounts will be returned.
remove_promoted	If TRUE, tweets created for promotion only on ads.twitter.com are removed
has_hashtags	If TRUE, only tweets containing hashtags will be returned; if FALSE, tweets containing hashtags will be excluded; if NULL, both tweets containing hashtags and tweets without hashtags will be returned.
has_cashtags	If TRUE, only tweets containing cashtags will be returned; if FALSE, tweets containing cashtags will be excluded; if NULL, both tweets containing cashtags and tweets without cashtags will be returned.
has_links	If TRUE, only tweets containing links (and media) will be returned; if FALSE, tweets containing links (and media) will be excluded; if NULL, both tweets containing links (and media) and tweets without links (and media) will be returned.
has_mentions	If TRUE, only tweets containing mentions will be returned; if FALSE, tweets containing mentions will be excluded; if NULL, both tweets containing mentions and tweets without mentions will be returned.
has_media	If TRUE, only tweets containing media such as a photo, GIF, or video (as determined by Twitter) will be returned will be returned; if FALSE, tweets containing media will be excluded; if NULL, both tweets containing media and tweets without media will be returned.
has_images	If TRUE, only tweets containing (recognized URLs to) images will be returned will be returned will be returned; if FALSE, tweets containing images will be excluded; if NULL, both tweets containing images and tweets without images will be returned.
has_videos	If TRUE, only tweets containing contain videos (recognized as native videos uploaded directly to Twitter) will be returned will be returned; if FALSE, tweets containing videos will be excluded; if NULL, both tweets containing videos and tweets without videos will be returned.
has_geo	If TRUE, only tweets containing geo information (Tweet-specific geolocation data provided by the Twitter user) will be returned; if FALSE, tweets containing geo information will be excluded; if NULL, both tweets containing geo information and tweets without geo information will be returned.
place	string, name of place e.g. "London"

country	string, name of country as ISO alpha-2 code e.g. "GB"
point_radius	numeric, a vector of two point coordinates latitude, longitude, and point radius distance (in miles)
bbox	numeric, a vector of four bounding box coordinates from west longitude to north latitude
lang	string, a single BCP 47 language identifier e.g. "fr"
conversation_id	string, return tweets that share the specified conversation ID
url	string, url

Details

This function is already called within the main get_all_tweets function.

It may also be called separately and the output saved as a character object query string to be input as query parameter to get_all_tweets.

Value

a query string

Examples

```
## Not run:
query <- build_query(query = "happy", is_retweet = FALSE,
                     country = "US",
                     place = "seattle",
                     point_radius = c(-122.33795253639994, 47.60900846404393, 25),
                     lang = "en")

query <- build_query(query = "twitter",
                     point_radius = c(-122.33795253639994, 47.60900846404393, 25),
                     lang = "en")


## End(Not run)
```

---

count_all_tweets *Count tweets from full archive search*

Description

This function returns aggregate counts of tweets by query string or strings between specified date ranges.

Usage

```
count_all_tweets(
  query = NULL,
  start_tweets,
  end_tweets,
  bearer_token = get_bearer(),
  n = 100,
  file = NULL,
  data_path = NULL,
  export_query = TRUE,
  bind_tweets = TRUE,
  granularity = "day",
  verbose = TRUE,
  ...
)
```

Arguments

query	string or character vector, search query or queries
start_tweets	string, starting date; default to now - 30 days
end_tweets	string, ending date; default to now - 30 seconds
bearer_token	string, bearer token
n	integer, upper limit of tweet counts to be fetched (i.e., for 365 days n must be at least 365). Default is 100.
file	string, name of the resulting RDS file
data_path	string, if supplied, fetched data can be saved to the designated path as jsons
export_query	If TRUE, queries are exported to data_path
bind_tweets	If TRUE, tweets captured are bound into a data.frame for assignment
granularity	string, the granularity for the search counts results. Options are "day"; "hour"; "minute". Default is day.
verbose	If FALSE, query progress messages are suppressed
...	arguments will be passed to build_query() function. See ?build_query() for further information.

Value

a data.frame

Examples

```
## Not run:

count_all_tweets(query = "Hogmanay",
          start_tweets = "2019-12-2700:00:00Z",
          end_tweets = "2020-01-05T00:00:00Z",
          bearer_token = get_bearer())
```

```
count_all_tweets(query = "Hogmanay",
          start_tweets = "2019-12-27T00:00:00Z",
          end_tweets = "2020-01-05T00:00:00Z",
          bearer_token = get_bearer(),
          granularity = "hour",
          n = 500)

## End(Not run)
```

`create_compliance_job` *Create Compliance Job*

Description

This function creates a new compliance job and upload the Tweet IDs or user IDs. By default, the parameter `x` with the length of one is assumed to be a text file containing either Tweet IDs or user IDs. This default behavior can be bypassed using `force_ids` For example, if you want to check for just a single Tweet ID.

Usage

```
create_compliance_job(
  x,
  type = "tweets",
  bearer_token = get_bearer(),
  force_ids = FALSE,
  verbose = TRUE
)
```

Arguments

`x`	either a character vector of Tweet IDs or user IDs; or a plain text file that each line contains a Tweet ID or user ID.
`type`	the type of the job, whether "tweets" or "users".
`bearer_token`	string, bearer token
`force_ids`	logical, make sure `x` is treated as a character vector of Tweet IDs or user IDs.
`verbose`	If `FALSE`, query progress messages are suppressed

Value

the job ID (invisibly)

Examples

```
## Not run:
create_compliance_job(x = "tweetids.txt", type = "tweets")

## End(Not run)
```

`get_all_tweets` *Get tweets from full archive search*

Description

This function collects tweets by query string or strings between specified date ranges.

Usage

```
get_all_tweets(
  query = NULL,
  start_tweets,
  end_tweets,
  bearer_token = get_bearer(),
  n = 100,
  file = NULL,
  data_path = NULL,
  export_query = TRUE,
  bind_tweets = TRUE,
  page_n = 500,
  context_annotations = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`query`	string or character vector, search query or queries
`start_tweets`	string, starting date; default to now - 30 days
`end_tweets`	string, ending date; default to now - 30 seconds
`bearer_token`	string, bearer token
`n`	integer, upper limit of tweets to be fetched
`file`	string, name of the resulting RDS file
`data_path`	string, if supplied, fetched data can be saved to the designated path as jsons
`export_query`	If `TRUE`, queries are exported to data_path
`bind_tweets`	If `TRUE`, tweets captured are bound into a data.frame for assignment
`page_n`	integer, amount of tweets to be returned by per page
`context_annotations`	If `TRUE`, context_annotations will be fetched. Note it will limit the page_n to 100 due restrictions of Twitter API.
`verbose`	If `FALSE`, query progress messages are suppressed
`...`	arguments will be passed to `build_query()` function. See `?build_query()` for further information.

Details

The function can also collect tweets by users. These may be specified alongside a query string or without. When no query string is supplied, the function collects all tweets by that user.

If a filename is supplied, the function will save the result as a RDS file.

If a data path is supplied, the function will also return tweet-level data in a data/ path as a series of JSONs beginning "data_"; while user-level data will be returned as a series of JSONs beginning "users_".

Value

When bind_tweets is TRUE (default), the function returns a data frame. Nothing otherwise.

Examples

```
## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"

get_all_tweets(query = "BLM",
               start_tweets = "2020-01-01T00:00:00Z",
               end_tweets = "2020-01-05T00:00:00Z",
               bearer_token = get_bearer(),
               data_path = "data",
               n = 500)

get_all_tweets(users = c("cbarrie", "jack"),
               start_tweets = "2021-01-01T00:00:00Z",
               end_tweets = "2021-06-01T00:00:00Z",
               bearer_token = get_bearer(),
               n = 1000)

get_all_tweets(start_tweets = "2021-01-01T00:00:00Z",
               end_tweets = "2021-06-01T00:00:00Z",
               bearer_token = get_bearer(),
               n = 1500,
               conversation_id = "1392887366507970561")

## End(Not run)
```

---

get_bearer *Manage bearer token*

Description

This function attempts to retrieve your bearer token from the environmental variable "TWITTER_BEARER". The easiest way to setup this environmental variable is to use `set_bearer()` and insert your bearer token to `.Renviron` file following the format: TWITTER_BEARER=YOURTOKENHERE. Replace YOURTOKENHERE with your own token.

Usage

```
get_bearer()
```

Details

Note: for `get_bearer()` to retrieve your bearer token you will need to restart the R session after storing in `.Renviron`.

Value

string represents your bearer token, if it the environmental variable "TWITTER_BEARER" has been preset.

`get_compliance_result` *Get Compliance Result*

Description

This function retrieves the information for a single compliance job.

Usage

```
get_compliance_result(id, bearer_token = get_bearer(), verbose = TRUE)
```

Arguments

`id`	string, the job id
`bearer_token`	string, bearer token
`verbose`	If `FALSE`, query progress messages are suppressed

Value

a data frame

Examples

```
## Not run:
get_compliance_result("1460077048991555585")

## End(Not run)
```

get_liked_tweets *Get liked tweets*

Description

This function fetches returns tweets liked by a user or users.

Usage

```
get_liked_tweets(x, bearer_token = get_bearer(), ...)
```

Arguments

x	string containing one user id or a vector of user ids
bearer_token	string, bearer token
...	arguments passed to other backend functions

Value

a data frame

Examples

```
## Not run:
users <- c("2244994945", "95226101")
get_liked_tweets(users, bearer_token = get_bearer())

## End(Not run)
```

get_liking_users *Get liking users*

Description

This function fetches users who liked a tweet or tweets.

Usage

```
get_liking_users(x, bearer_token = get_bearer(), verbose = TRUE)
```

Arguments

x	string containing one tweet id or a vector of tweet ids
bearer_token	string, bearer token
verbose	If FALSE, query progress messages are suppressed

Value

a data frame

Examples

```
## Not run:
tweet <- "1387744422729748486"
get_liking_users(tweet, bearer_token = get_bearer())

## End(Not run)
```

`get_retweeted_by` *Get users who has retweeted a tweet*

Description

This function fetches users who retweeted a tweet

Usage

```
get_retweeted_by(
  x,
  bearer_token = get_bearer(),
  data_path = NULL,
  verbose = TRUE
)
```

Arguments

`x`	string containing one tweet id or a vector of tweet ids
`bearer_token`	string, bearer token
`data_path`	string, if supplied, fetched data can be saved to the designated path as jsons
`verbose`	If `FALSE`, query progress messages are suppressed

Value

a data frame

Examples

```
## Not run:
tweets <- c("1392887366507970561","1409931481552543749")
get_retweeted_by(tweets, bearer_token = get_bearer())

## End(Not run)
```

get_user_followers *Get user followers*

Description

This function fetches users who are followers of the specified user ID.

Usage

```
get_user_followers(x, bearer_token = get_bearer(), ...)
```

Arguments

x	string containing one user id or a vector of user ids
bearer_token	string, bearer token
...	arguments passed to other backend functions

Value

a data frame

Examples

```
## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- "2244994945"
get_user_followers(users, bearer_token = get_bearer())

## End(Not run)
```

get_user_following *Get user following*

Description

This function fetches a list of users the specified user ID is following.

Usage

```
get_user_following(x, bearer_token = get_bearer(), ...)
```

Arguments

x	string containing one user id or a vector of user ids
bearer_token	string, bearer token
...	arguments passed to other backend functions

Value

a data frame

Examples

```
## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- "2244994945"
get_user_following(users, bearer_token)

## End(Not run)
```

`get_user_id` *Get user id*

Description

This function get the user IDs (e.g. 1349149096909668363) of given usernames, e.g. "potus".

Usage

```
get_user_id(
  usernames,
  bearer_token = get_bearer(),
  all = FALSE,
  keep_na = TRUE
)
```

Arguments

`usernames`	character vector containing screen names to be queried
`bearer_token`	string, bearer token
`all`	logical, default FALSE to get a character vector of user IDs. Set it to TRUE to get a data frame, see below
`keep_na`	logical, default TRUE to keep usernames that cannot be queried. Set it to TRUE to exclude those usernames. Only useful when all is FALSE

Value

a string vector with the id of each of the users unless all = TRUE. If all = TRUE, a data.frame with ids, names (showed on the screen) and usernames is returned.

Examples

```
## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- c("Twitter", "TwitterDev")
get_user_id(users, bearer_token)

## End(Not run)
```

---

get_user_mentions *Get tweets mentioning a single user*

Description

This function collects tweets mentioning an user ID from the users endpoint.

Usage

```
get_user_mentions(
  x,
  start_tweets,
  end_tweets,
  bearer_token = get_bearer(),
  n = 100,
  file = NULL,
  data_path = NULL,
  export_query = TRUE,
  bind_tweets = TRUE,
  page_n = 100,
  verbose = TRUE,
  ...
)
```

Arguments

x	string containing one user id or a vector of user ids
start_tweets	string, starting date
end_tweets	string, ending date
bearer_token	string, bearer token
n	integer, upper limit of tweets to be fetched
file	string, name of the resulting RDS file
data_path	string, if supplied, fetched data can be saved to the designated path as jsons
export_query	If TRUE, queries are exported to data_path
bind_tweets	If TRUE, tweets captured are bound into a data.frame for assignment
page_n	integer, amount of tweets to be returned by per page
verbose	If FALSE, query progress messages are suppressed
...	arguments will be passed to build_query() function. See ?build_query() for further information.

Details

Only the most recent 800 Tweets can be retrieved.

This does not return retweets of the users' tweets.

If a filename is supplied, the function will save the result as a RDS file.

If a data path is supplied, the function will also return tweet-level data in a data/ path as a series of JSONs beginning "data_"; while user-level data will be returned as a series of JSONs beginning "users_".

When bind_tweets is TRUE, the function returns a data frame.

Value

a data.frame

Examples

```
## Not run:

get_user_mentions("881861582715850752",
                  start_tweets = "2022-01-01T00:00:00Z",
                  end_tweets = "2022-05-14T00:00:00Z",
                  bearer_token = get_bearer(),
                  n = 200)

## End(Not run)
```

---

get_user_profile *Get user profile*

Description

This function fetches user-level information for a vector of user IDs.

Usage

```
get_user_profile(x, bearer_token = get_bearer())
```

Arguments

x	string containing one user id or a vector of user ids
bearer_token	string, bearer token

Value

a data frame

Examples

```
## Not run:
bearer_token <- "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
users <- c("2244994945", "6253282")
get_user_profile(users, bearer_token)

## End(Not run)
```

`get_user_timeline` *Get tweets by a single user*

Description

This function collects tweets by an user ID from the users endpoint.

Usage

```
get_user_timeline(
  x,
  start_tweets,
  end_tweets,
  bearer_token = get_bearer(),
  n = 100,
  file = NULL,
  data_path = NULL,
  export_query = TRUE,
  bind_tweets = TRUE,
  page_n = 100,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	string containing one user id or a vector of user ids
`start_tweets`	string, starting date
`end_tweets`	string, ending date
`bearer_token`	string, bearer token
`n`	integer, upper limit of tweets to be fetched
`file`	string, name of the resulting RDS file
`data_path`	string, if supplied, fetched data can be saved to the designated path as jsons
`export_query`	If `TRUE`, queries are exported to data_path
`bind_tweets`	If `TRUE`, tweets captured are bound into a data.frame for assignment
`page_n`	integer, amount of tweets to be returned by per page
`verbose`	If `FALSE`, query progress messages are suppressed
`...`	arguments will be passed to `build_query()` function. See `?build_query()` for further information.

Details

Only the most recent 3,200 Tweets can be retrieved.

If a filename is supplied, the function will save the result as a RDS file.

If a data path is supplied, the function will also return tweet-level data in a data/ path as a series of JSONs beginning "data_"; while user-level data will be returned as a series of JSONs beginning "users_".

When bind_tweets is `TRUE`, the function returns a data frame.

Value

a data.frame

Examples

```
## Not run:

get_user_timeline("2244994945",
                  start_tweets = "2020-01-01T00:00:00Z",
                  end_tweets = "2021-05-14T00:00:00Z",
                  bearer_token = get_bearer(),
                  n = 200)

## End(Not run)
```

`hydrate_tweets` *Hydrate Tweets Based On Tweet IDs*

Description

This function is helpful for hydrating Tweet IDs (i.e. getting the full content of tweets from a list of Tweet IDs).

Usage

```
hydrate_tweets(
  ids,
  bearer_token = get_bearer(),
  data_path = NULL,
  context_annotations = FALSE,
  bind_tweets = TRUE,
  verbose = TRUE,
  errors = FALSE
)
```

Arguments

ids	a character vector of Tweet IDs
bearer_token	string, bearer token
data_path	string, if supplied, fetched data can be saved to the designated path as jsons
context_annotations	If TRUE, context_annotations will be fetched.
bind_tweets	If TRUE, tweets captured are bound into a data.frame for assignment
verbose	If FALSE, query progress messages are suppressed
errors	logical, if TRUE, the error capturing mechanism is enabled. See details below.

Details

When the error capturing mechanism is enabled, Tweets IDs that cannot be queried (e.g. with error) are stored as `errors_*.json` files. If `bind_tweets` is TRUE, those error Tweets IDs are retained in the returned data.frame with the column `error` indicating the error.

Value

When bind_tweets is TRUE, the function returns a data frame. The `data_path` (invisibly) if `bind_tweets` is FALSE

Examples

```
## Not run:
hydrate_tweets(c("1266876474440761346", "1266868259925737474", "1266867327079002121",
"1266866660713127936", "1266864490446012418", "1266860737244336129",
"1266859737615826944", "1266859455586676736", "1266858090143588352",
"1266857669157097473"))

## End(Not run)
```

---

list_compliance_jobs *List Compliance Jobs*

Description

This function lists all compliance jobs.

Usage

```
list_compliance_jobs(type = "tweets", bearer_token = get_bearer())
```

Arguments

type	the type of the job, whether "tweets" or "users".
bearer_token	string, bearer token

Value

a data frame

Examples

```
## Not run:
list_compliance_jobs()

## End(Not run)
```

`resume_collection` *Resume previous collection*

Description

This function resumes a previous interrupted collection session.

Usage

```
resume_collection(data_path, bearer_token = get_bearer(), verbose = TRUE, ...)
```

Arguments

`data_path`	string, name of an existing data_path
`bearer_token`	string, bearer token
`verbose`	If `FALSE`, query progress messages are suppressed
`...`	arguments will be passed to `get_all_tweets()` function. See `?get_all_tweets()` for further information.

Details

For this function to work, export_query must be set to "TRUE" during the original collection.

Value

a data.frame

Examples

```
## Not run:
resume_collection(data_path = "data", bearer_token = get_bearer())

## End(Not run)
```

set_bearer *Set bearer token*

Description

This function lets the user add their bearer token to the `.Renviron` file.

Usage

```
set_bearer()
```

Details

It is in general not safe to 1) hard code your bearer token in your R script or 2) have your bearer token in your command history.

`set_bearer` opens the .Renviron file for the user and provides instructions on how to add the bearer token, which requires the addition of just one line in the `.Renviron` file, following the format TWITTER_BEARER=YOURTOKENHERE.

Replace YOURTOKENHERE with your own token.

update_collection *Update previous collection session*

Description

This function continues a previous collection session with a new end date. For this function to work, export_query must be set to "TRUE" during the original collection.

Usage

```
update_collection(
  data_path,
  end_tweets,
  bearer_token = get_bearer(),
  verbose = TRUE,
  ...
)
```

Arguments

`data_path`	string, name of an existing data_path
`end_tweets`	string, ending date; default to now - 30 seconds
`bearer_token`	string, bearer token
`verbose`	If `FALSE`, query progress messages are suppressed
`...`	arguments will be passed to `get_all_tweets()` function. See `?get_all_tweets()` for further information.

Value

a data.frame

Examples

```
## Not run:
update_collection(data_path = "data", "2020-01-03T00:00:00Z", bearer_token = get_bearer())

## End(Not run)
```

Index