Package: episomer (via r-universe)

June 7, 2026

Title Early Detection of Public Health Threats from Social Media Data

Version 3.0.35

Description It allows you to automatically monitor trends of social media messages by time, place and topic aiming at detecting public health threats early through the detection of signals (i.e., an unusual increase in the number of messages per time, topic and location). It was designed to focus on infectious diseases, and it can be extended to all hazards or other fields of study by modifying the topics and keywords. More information on the original package 'epitweetr' is available in the peer-review publication Espinosa et al. (2022) <doi:10.2807/1560-7917.ES.2022.27.39.2200177>.

License EUPL

URL https://github.com/EU-ECDC/episomer

BugReports https://github.com/EU-ECDC/episomer/issues

Encoding UTF-8

Imports dplyr, curl, DT, emayili, future, httr, htmltools, jsonlite, keyring, ggplot2, janitor, magrittr, parallel, plotly, processx, readxl, rlang, rmarkdown, rnaturalearthdata, openxlsx, shiny, stringr, stats, tibble, tools, utils, xtable, httr2, lubridate, sf, cli

RoxygenNote 7.3.3

Suggests knitr, taskscheduleR, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libicu-dev libsecret-1-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://eu-ecdc.r-universe.dev

Date/Publication 2026-05-08 07:48:27 UTC

RemoteUrl https://github.com/eu-ecdc/episomer

RemoteRef HEAD

RemoteSha 98169519c8166fc5ef8e8604090e78ab36508d9d

RemoteSubdir episomer

Contents

`add_new_social_media` *Create new social media template files, so a developer can extend episomer and support a new social media. this function has to be called within episomer subfolder containing thr R subfolder after executing devtools::load_all()*

Description

Create new social media template files, so a developer can extend episomer and support a new social media. this function has to be called within episomer subfolder containing thr R subfolder after executing devtools::load_all()

Usage

```
add_new_social_media(social_media, social_media_ref = "bluesky")
```

Arguments

social_media	Name of the new social media
social_media_ref	Name of the reference social media

Value

TRUE, called for side effects (creation of new social media files).

---

bluesky_check_token_validity *check the current bluesky token validity*

Description

check the current bluesky token validity

Usage

```
bluesky_check_token_validity(
  access_jwt,
  search_url = "https://bsky.social/xrpc/app.bsky.feed.searchPosts"
)
```

Arguments

access_jwt	char, the jwt token provided by the bluesky api
search_url	char, the url to test if the current token is still valid

---

bluesky_create_session *Get bearer token*

Description

Get bearer token

Usage

```
bluesky_create_session(handle = NULL, password = NULL)
```

Arguments

handle	bluesky handle
password	bluesky password

Details

You can find more information about the bluesky API here: https://docs.bsky.app/docs/api/com-atproto-server-create-session

Value

List with access_jwt and did

Examples

```
## Not run:
bluesky_create_session()

## End(Not run)
```

---

bluesky_format_date *formats a date in the expected bluesky format*

Description

formats a date in the expected bluesky format

Usage

```
bluesky_format_date(datetime)
```

Arguments

datetime	the datetime to format

`bluesky_parse_date` *parse a date as provided by bluesky into a date_time object*

Description

parse a date as provided by bluesky into a date_time object

Usage

```
bluesky_parse_date(date_input)
```

Arguments

`date_input`	char, the text of a date

`bluesky_parse_features` *extracts features from the post structure returned by the bluesky api*

Description

extracts features from the post structure returned by the bluesky api

Usage

```
bluesky_parse_features(post)
```

Arguments

`post`	list, the post to extract features from in bluesky format

`bluesky_parse_quoted` *extracts the quotes information from a post in one of the many possible locations found in bluesky format*

Description

extracts the quotes information from a post in one of the many possible locations found in bluesky format

Usage

```
bluesky_parse_quoted(post)
```

Arguments

`post`	list, the post to to extract data from in bluesky format

`bluesky_parse_response` *parses the response of bluesky API into the standardised episomer format. For more details see the vignette.*

Description

parses the response of bluesky API into the standardised episomer format. For more details see the vignette.

Usage

```
bluesky_parse_response(response, current_min_date)
```

Arguments

`response`	list, the standardised response
`current_min_date`	datetime, the current min date obtained on the plan to detect if there are messages after this limit and discard them

`bluesky_rate_limited_check` *function to check if there has been a rate limit error returned by the bluesky API*

Description

function to check if there has been a rate limit error returned by the bluesky API

Usage

```
bluesky_rate_limited_check(resp)
```

Arguments

`resp`	the httr2 response object

bluesky_rerun_after_rate_limit

function to extract the number of seconds to wait after a rate limit has been reached

Description

function to extract the number of seconds to wait after a rate limit has been reached

Usage

```
bluesky_rerun_after_rate_limit(resp)
```

Arguments

resp	the httr2 response object

calculate_region_alerts

calculate alerts for for a particular region

Description

calculate alerts for a region and topic using provided parameters to the episomer signal detection function

Usage

```
calculate_region_alerts(
  sms,
  topic,
  country_codes = list(),
  country_code_cols = "geo_country_code",
  start = NA,
  end = NA,
  with_quotes = FALSE,
  alpha = 0.025,
  alpha_outlier = 0.05,
  k_decay = 4,
  no_historic = 7,
  bonferroni_m = 1,
  same_weekday_baseline = FALSE,
  logenv = NULL
)
```

```
calculate_regions_alerts(
  sms,
  topic,
  regions = c(1),
  date_type = c("created_date"),
  date_min = as.Date("1900-01-01"),
  date_max = as.Date("2100-01-01"),
  with_quotes = FALSE,
  alpha = 0.025,
  alpha_outlier = 0.05,
  k_decay = 4,
  no_historic = 7,
  bonferroni_correction = FALSE,
  same_weekday_baseline = FALSE,
  logenv = NULL
)
```

Arguments

sms	A given social media for which aggregations have been calculated
topic	Limit the alert detection to the given topic
country_codes	List with the codes of countries to be aggregated for this alerts
country_code_cols	the name of the column containing the country codes
start	the start date for the period in consideration
end	the end date for the period in consideration
with_quotes	wether to consider quotes in the time series to evaluate
alpha	The alpha is used to compute the upper limit of the prediction interval: (1-alpha) * 100%, default: 0.025
alpha_outlier	Residuals beyond 1-alpha_outlier quantile of the the t(n-k-1) distribution are downweighted, default: 0.05
k_decay	Power k in the expression (r_star/r_threshold)^k determining the weight, default: 4
no_historic	Number of previous values i.e -1, -2, ..., no_historic to include when computing baseline parameters, default: 7
bonferroni_m	Number the value to apply the Bonferroni correction; it specifies the global number of timeseries being observed
same_weekday_baseline	whether to calculate baseline using same weekdays or any day, default: FALSE
logenv	an environment variable to store the total number of posts concerned including all country_cols
regions	List with the codes of regions to be aggregated for this alerts
date_type	the type of date to be used for the period in consideration
date_min	the start date for the period in consideration

`date_max` the end date for the period in consideration

`bonferroni_correction`
whether to apply the bonferroni correction to the alerts, default: FALSE

Details

for algorithm details see package vignette.

Value

A dataframe containing the monitored time point for the given countries and topic, the upper limit and whether a signal is detected or not.

A dataframe containing the monitored time point for the given regions and topic, the upper limit and whether a signal is detected or not.

Examples

```
## Not run:
   library(episomer)
   #Running the alerts for France in topic covid-19
   df <-calculate_region_alerts(sms="bluesky",
                                topic="covid-19",
                                country_codes = "FR",
                                start=as.Date("2025-09-20"),
                                end=as.Date("2025-09-30"))
   show(df)

## End(Not run)
## Not run:
   library(episomer)
   #Running the alerts for World in topic covid-19
   df <-calculate_regions_alerts(
                                sms="bluesky",
                                topic="covid-19",
                                date_min=as.Date("2025-09-20"),
                                date_max=as.Date("2025-09-30")
)
   show(df)

## End(Not run)
```

---

check_all *Run automatic sanity checks*

Description

It runs a set of automated sanity checks for helping the user to troubleshot issues

Usage

```
check_all()
```

Details

This function executes a series of sanity checks, concerning, Java, bitness, task status, dependencies and social media authentication.

Value

Data frame containing the statuses of all realized checks

Examples

```
## Not run:
   #importing episomer
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   #running all tests
   check_all()

## End(Not run)
```

---

create_api_and_plan_files_for_new_social_media

internal function for producing the code template for a new social media

---

Description

internal function for producing the code template for a new social media

Usage

```
create_api_and_plan_files_for_new_social_media(
  new_media,
  ref_media = "bluesky"
)
```

Arguments

new_media	char, name of the social media to add
ref_media	char, name of the social media to use as a reference to produce template

`create_map` *Plot the map report on the episomer dashboard*

Description

Generates a bubble map plot of number of posts by country for a single topic

Usage

```
create_map(
  sms,
  topic = c(),
  countries = c(1),
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_quotes = FALSE,
  caption = "",
  proj = NULL,
  forplotly = FALSE
)
```

Arguments

`sms`	Social media
`topic`	Character(1) containing the topic to use for the report
`countries`	Character vector containing the name of the countries and regions to plot or their respective indexes on the Shiny app, default: c(1)
`date_min`	Date indicating start of the reporting period, default: "1900-01-01"
`date_max`	Date indicating end of the reporting period, default: "2100-01-01"
`with_quotes`	Logical value indicating whether to include reposts in the time series, default: FALSE
`caption`	Character(1) vector indicating a caption to print at the bottom of the chart, default: ""
`proj`	Parameter indicating the CRS (Coordinate Reference System) to use on PROJ4 format, default NULL If null and all countries are selected +proj=robin is used (Robinson projection) otherwise the Lambert azimuthal equal-area projection will be chosen, default: NULL
`forplotly`	Logical(1) parameter indicating whether some hacks are activated to improve plotly rendering, default: FALSE

Details

Produces a bubble chart map for a particular topic on number of posts collected based on the provided parameters. The map will display information at country level if more than one country is

selected, otherwise it will display bubbles at the smallest possible location identified for each post within the period which could be any administrative level or city level.

Posts associated with a country but with no finer granularity are omitted when displaying a single country.

When an aggregated zone is requested, all countries in that zone are included.

This functions requires that search_loop and detect_loop have already been run successfully to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the dataframe that was used to build the map.

See Also

trend_line create_topwords detect_loop search_loop spTransform,coordinates,is.projected,CRS-class fortify,geom_polygon,geom_point

Examples

```
## Not run:
   #Getting bubble chart for dengue for South America for last 30 days
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   create_map(
     topic = "dengue",
     countries = "South America",
     date_min = as.Date(Sys.time())-30,
     date_max=as.Date(Sys.time())
   )

## End(Not run)
```

---

create_snapshot *Snapshot of your episomer installation*

Description

Creates a snapshot file of your episomer installation folder. This can include all or a subset of the data files.

Usage

```
create_snapshot(
  destination_dir,
  types = c("settings", "dependencies", "machine-learning", "aggregations", "posts",
    "logs"),
  posts_period = get_aggregated_period(),
```

```
  aggregated_period = get_aggregated_period(),
  compress = TRUE,
  progress = function(v, m) message(paste(round(v * 100, 2), m))
)
```

Arguments

destination_dir	character(1) vector with the path of the destination folder to produce the snapshot
types	character vector indicating the types of data to include on a snapshot. Some of: "settings", "dependencies", "machine-learning", "aggregations", "posts", "logs"
posts_period	date(2) start and end dates to filter posts to include on snapshot (if selected)
aggregated_period	date(2) start and end dates to filter time series to include on snapshot (if selected)
compress	logical(1) whether to compress or not the output file
progress	function to report progress during execution.

Details

This function can be used to create a a portable file to move your episomer installation in a single file, to backup your data, to archive your old data or to send information to technical team in order to reproduce an observed issue. Different kinds of data can be included on the snapshot depending on the type of parameter. Possible values are: - 'settings': Including all setting files of your installation (excluding passwords) - 'dependencies': All jars and winutils.exe on windows installations - 'machine-learning': All trained models and vectors and training data (this can include post text which is personal data) - 'aggregations': Episomer aggregated time series - 'posts': Posts collected by episomer - 'logs': Log files produced automatically on windows task scheduler tasks.

Value

Nothing

Examples

```
## Not run:
   #importing episomer
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   message('Please choose a destination directory')
   dest <- file.choose()
   #creating a compressed snapshot for settings and logs
   create_snapshot(dest, c("settings","dependencies"), compress = TRUE)

## End(Not run)
```

`create_topchart` *Plot the top elements for a specific series on the episomer dashboard*

Description

Generates a bar plot of most popular elements in posts, for one topic. Top elements among ("topwords", "hashtags", "entities", "contexts", "urls")

Usage

```
create_topchart(
  sms,
  topic,
  serie,
  country_codes = c(),
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_quotes = FALSE,
  top = 25
)
```

Arguments

`sms`	Social media
`topic`	Character(1) containing the topic to use for the report
`serie`	Character(1) name of the series to be used for the report. It should be one of ("topwords", "hashtags", "entities", "contexts", "urls")
`country_codes`	Character vector containing the ISO 3166-1 alpha-2 countries to plot, default: c()
`date_min`	Date indicating start of the reporting period, default: "1900-01-01"
`date_max`	Date indicating end of the reporting period, default: "2100-01-01"
`with_quotes`	Logical value indicating whether to include reposts in the time series, default: FALSE
`top`	numeric(1) Parameter indicating the number of words to show, default: 25

Details

Produces a bar chart showing the occurrences of the most popular words in the collected posts based on the provided parameters.

This functions requires that `search_loop` and `detect_loop` have already been run successfully to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the data frame that was used to build the map.

See Also

trend_line create_map detect_loop search_loop

Examples

```
## Not run:
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   #Getting topword chart for dengue for France, Chile, Australia for last 30 days
   create_topchart(
     topic = "dengue",
     serie = "topwords",
     country_codes = c("FR", "CL", "AU"),
     date_min = as.Date(Sys.time())-30,
     date_max=as.Date(Sys.time())
   )

## End(Not run)
```

---

create_topwords *Plot the top words report on the episomer dashboard*

Description

Generates a bar plot of most popular words in posts, for one topic

Usage

```
create_topwords(
  sms,
  topic,
  country_codes = c(),
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_quotes = FALSE,
  top = 25
)
```

Arguments

sms	Social media
topic	Character(1) containing the topic to use for the report
country_codes	Character vector containing the ISO 3166-1 alpha-2 countries to plot, default: c()
date_min	Date indicating start of the reporting period, default: "1900-01-01"
date_max	Date indicating end of the reporting period, default: "2100-01-01"

with_quotes	Logical value indicating whether to include reposts in the time series, default: FALSE
top	numeric(1) Parameter indicating the number of words to show, default: 25

Details

Produces a bar chart showing the occurrences of the most popular words in the collected posts based on the provided parameters.

This report may be empty for combinations of countries and topics with very few posts since for performance reasons, the calculation of top words is an approximation using chunks of 10.000 posts.

This function requires that search_loop and detect_loop have already been run successfully to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the data frame that was used to build the map.

See Also

trend_line create_map detect_loop search_loop

Examples

```
## Not run:
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   #Getting topword chart for dengue for France, Chile, Australia for last 30 days
   create_topwords(
     topic = "dengue",
     country_codes = c("FR", "CL", "AU"),
     date_min = as.Date(Sys.time())-30,
     date_max=as.Date(Sys.time())
   )

## End(Not run)
```

---

detect_loop *Runs the detect loop*

Description

Infinite loop ensuring the daily signal detection and email alerts

Usage

```
detect_loop(data_dir = NA)
```

Arguments

data_dir	Path to the 'data directory' containing application settings, models and collected posts. If not provided the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, default: NA

Details

The detect loop is composed of three 'one shot tasks' download_dependencies, update_geonames, update_languages ensuring the system has all necessary components and data to run the three recurrent tasks generate_alerts

The loop report progress on the 'tasks.json' file which is read or created by this function.

The recurrent tasks are scheduled to be executed each 'detect span' minutes, which is a parameter set on the Shiny app.

If any of these tasks fails it will be retried three times before going to abort status. Aborted tasks can be relaunched from the Shiny app.

Value

nothing

See Also

download_dependencies

update_geonames

update_languages

detect_loop

generate_alerts

get_tasks

Examples

```
## Not run:
   #Running the detect loop
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   detect_loop()

## End(Not run)
```

`download_dependencies` *Updates Java dependencies*

Description

Download Java dependencies of the application mainly related to Apache SPARK, Apache Lucene, and Apache Pekko

Usage

```
download_dependencies(tasks = get_tasks())
```

Arguments

`tasks`	Task object for reporting progress and error messages, default: get_tasks()

Details

Run a one-shot task consisting of downloading Java and Scala dependencies, this is separated by the following subtasks

- Download jar dependencies from configuration maven repo to project data folder. This includes, scala, spark, lucene. Packages to be downloaded are defined in package file 'sbt-deps.txt'
- Download winutils from configuration URL to project data folder. For more details on winutils please see https://issues.apache.org/jira/browse/HADOOP-13223 and https://issues.apache.org/jira/browse/HADOOP-16816

The URLs to download the JAR dependencies (maven package manager) and Winutils are on the configuration tab of the Shiny app.

Normally this function is not called directly by the user but from the `detect_loop` function.

Value

The list of tasks updated with produced messages

See Also

`detect_loop`

`get_tasks`

Examples

```
## Not run:
   library(episomer)
   # setting up the data folder
   message('Please choose the episomer data directory')
   setup_config(file.choose())

   # geolocating last posts
   tasks <- download_dependencies()

## End(Not run)
```

`ears_t_reweighted` *algorithm for outbreak detection, extends the EARS algorithm*

Description

The simple 7 day running mean version of the Early Aberration Reporting System (EARS) algorithm is extended as follows:

- proper computation of the prediction interval
- downweighting of previous signals, similar to the approach by Farrington (1996)

Usage

```
ears_t_reweighted(
  ts,
  alpha = 0.025,
  alpha_outlier = 0.05,
  k_decay = 4,
  no_historic = 7L,
  same_weekday_baseline = FALSE
)
```

Arguments

`ts`	A numeric vector containing the counts of the univariate time series to monitor. The last time point in ts is investigated
`alpha`	The alpha is used to compute the upper limit of the prediction interval: (1-alpha) * 100%, default: 0.025
`alpha_outlier`	Residuals beyond 1-alpha_outlier quantile of the the t(n-k-1) distribution are downweighted, default: 0.05
`k_decay`	Power k in the expression (r_star/r_threshold)^k determining the weight, default: 4
`no_historic`	Number of previous values i.e -1, -2, ..., no_historic to include when computing baseline parameters, default: 7
`same_weekday_baseline`	whether to calculate baseline using same weekdays or any day, default: FALSE

Details

for algorithm details see package vignette.

Value

A dataframe containing the monitored time point, the upper limit and whether a signal is detected or not.

Author(s)

Michael Hoehle <https://www.math.su.se/~hoehle>

Examples

```
## Not run:
   library(episomer)
   #Running the modifies version of the ears algorithm for a particular data series
    ts <- c(150, 130, 122, 160, 155, 128, 144, 125, 300, 319, 289, 277, 500)
    show(ears_t_reweighted(ts))

## End(Not run)
```

---

episomer_app *Run the episomer Shiny app*

Description

Open the episomer Shiny app, used to setup the Data collection & processing pipeline, the Requirements & alerts pipeline and to visualise the outputs.

Usage

```
episomer_app(
  data_dir = NA,
  profile = c("dashboard", "admin"),
  host = NULL,
  port = NULL
)

dashboard_app(data_dir = NA, host = NULL, port = NULL)

admin_app(data_dir = NA, host = NULL, port = NULL)
```

Arguments

data_dir	Path to the 'data directory' containing application settings, models and collected posts. If not provided the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, default: NA
profile	The profileis to run the Shiny app on, default: "dashboard" (can be "dashboard" or "admin")
host	The IP if the network interface to run the Shiny app on, default: NULL (will run on 127.0.0.1)
port	The port to run the Shiny app on, default: NULL (will run on a random port)

Details

The episomer app is the user entry point to the episomer package. This application will help the user to setup the post collection process, manage all settings, see the interactive dashboard visualisations, export them to Markdown or PDF, and setup the alert emails.

All its functionality is described on the episomer vignette.

The dashboard app is the user entry point to the episomer package for regular users. This application shows the interactive dashboard with visualisations, allow export them to Markdown or PDF.

All its functionality is described on the episomer vignette.

The admin app is contain all functionality for an administrator. It help the user to setup the post collection process, manage all settings, and setup the alert emails.

All its functionality is described on the episomer vignette.

Value

The Shiny server object containing the launched application

The Shiny server object containing the launched application

The Shiny server object containing the launched application

See Also

search_loop

detect_loop

Examples

```
## Not run:
   #Running the episomer app
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   episomer_app()

## End(Not run)
```

fs_loop *Runs the episomer embedded database loop*

Description

Infinite loop ensuring that the episomer embedded database is running (Lucene + akka-http)

Usage

```
fs_loop(data_dir = NA)
```

Arguments

data_dir	Path to the 'data directory' containing application settings, models and collected posts. If not provided, the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, default: NA

Details

Launches the episomer embedded database which is accessed via a REST API located on localhost at port 8080. You can test that the database is running by accesssing the local URL and adding "/ping". the REST API provide episomer a way to send and retrieve data related with posts and time series and to trigger geolocation or aggregation The database is implemented using Apache Lucene indexes allowing episomer to access its data as a search engine but also as a tabular database. health_check called each 60 seconds on a background process to send alerts to the administrator if some episomer components fail.

Value

nothing

See Also

detect_loop

search_loop

health_check

Examples

```
## Not run:
   #Running the detect loop
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   fs_loop()

## End(Not run)
```

`generate_alerts` *Execute the alert task*

Description

Evaluate alerts for the last collected day for all topics and regions and send email alerts to subscribers

Usage

```
generate_alerts(tasks = get_tasks())
```

Arguments

`tasks`	Current tasks for reporting purposes, default: get_tasks()

Details

This function calculates alerts for the last aggregated day and then sends emails to subscribers.

The alert calculation is based on the country_counts time series which stores alerts by country, hour and topics.

For each country and region, the process starts by aggregating the last N days. A day is defined as a block of consecutive 24 hours ending before the hour of the last collected post. N is defined by the alert baseline parameter on the configuration tab of the Shiny application (the default is N=7).

An alert will be produced when the number of posts observed is above the threshold calculated by the modified version of the EARS algorithm (for more details see the package vignette). The behaviour of the alert detection algorithm is modified by the signal false positive rate (alpha), by downweighting previous alerts and by weekly or daily baseline parameters, as defined in the configuration tab of the Shiny application and the topics file.

A prerequisite for this function is that the `search_loop` must already have stored posts in the search folder and that the geotagging and aggregation tasks have been completed. Normally, this function is not called directly by the user, but from the `detect_loop` function.

Value

The list of tasks updated with produced messages

See Also

`detect_loop`

Examples

```
## Not run:
   library(episomer)
   # setting up the data folder
   message('Please choose the episomer data directory')
   setup_config(file.choose())

   # calculating alerts for last day posts and sending emails to subscribers
   generate_alerts()

## End(Not run)
```

`geolocate_text` *geolocate text in a data frame given a text column and optionally a language column*

Description

extracts geolocaion information on text on a column of the provided data frame and returns a new data frame with geolocation information

Usage

```
geolocate_text(df, text_col = "text", lang_col = NULL, min_score = NULL)
```

Arguments

`df`	A data frame containing at least character column with text, a column with the language name can be provided to improve geolocation quality
`text_col`	character, name of the column on the data frame containing the text to geolocalize, default:text
`lang_col`	character, name of the column on the data frame containing the language of texts, default: NULL
`min_score`	numeric, the minimum score obtained on the Lucene scoring function to accept matches on GeoNames. It has to be empirically set default: NULL

Details

This function perform a call to the episomer database which includes functionality for geolocating for languages activated and successfully processed on the shiny app.

The geolocation process tries to find the best match in GeoNames database https://www.geonames.org/ including all local aliases for words.

If no language is associated to the text, all tokens will be sent as a query to the indexed GeoNames database.

If a language code is associated to the text and this language is trained on episomer, entity recognition techniques will be used to identify the best candidate in text to contain a location and only these tokens will be sent to the GeoNames query.

A custom scoring function is implemented to grant more weight to cities increasing with population to try to perform disambiguation.

Rules for forcing the geolocation choices of the algorithms and for tuning performance with manual annotations can be performed on the geotag tab of the Shiny app.

A prerequisite to this function is that the tasks download_dependencies update_geonames and update_languages has been run successfully.

This function is called from the Shiny app on geolocation evaluation tab but can also be used for manually evaluating the episomer geolocation algorithm.

Value

A new data frame containing the following geolocation columns: geo_code, geo_country_code, geo_country, geo_name, tags

See Also

download_dependencies

update_geonames

detect_loop

Examples

```
## Not run:
   library(episomer)
   # setting up the data folder
   message('Please choose the episomer data directory')
   setup_config(file.choose())

   # creating a test dataframe
  df <- data.frame(text = c("Me gusta Santiago de Chile es una linda ciudad"), lang = c("es"))
   geo <- geolocate_text(df = df, text_col = "text", lang_col="lang")


## End(Not run)
```

---

get_aggregates *Getting already aggregated time series produced by* detect_loop

---

Description

Reads and returns the required aggregated dataset for the selected period and topics defined by the filter.

Usage

```
get_aggregates(
  dataset = "country_counts",
  cache = TRUE,
  filter = list(),
  top_field = NULL,
  top_freq = NULL
)
```

Arguments

dataset	A character(1) vector with the name of the series to request, it must be one of 'country_counts', 'geolocated', 'topwords', 'hashtags', 'entities', 'urls', 'contexts', default: 'country_counts'
cache	Whether to use the cache for lookup and storing the returned dataframe, default: TRUE
filter	A named list defining the filter to apply on the requested series, it should be in the form of a named list; e.g., list(post_geo_country_code=list('FR', 'DE')) default: list()
top_field	Name of the top field used with top_frequency to enable optimisation for getting only most frequent elements. It will only keep top 500 items after first 50k lines on reverse index order
top_freq	character, Name of the frequency fields used with top_field to enable optimisation for retrieving only most frequent elements. It will only keep top 500 items after first 50k rows on reverse index order

Details

This function returns data aggregated by episomer. The data is found on the 'series' folder, which contains Rds files per weekday and type of series. starting on v 1.0.x it will also look on Lucene indexes situated on fs folder. Names of files and folders are parsed to limit the files to be read. When using Lucene indexes, filters are directly applied on read. This is an improvement compared 'series' folder where filters are applied after read. All returned rows are joined into a single dataframe. If no filter is provided, the entire data series is returned, which can result in millions of rows depending on the time series. To limit by period, the filter list must include an element 'period' containing a date vector or list with two dates representing the start and end of the request.

To limit by topic, the filter list must include an element 'topic' containing a non-empty character vector or list with the names of the topics to return.

The available time series are:

- "country_counts" counting posts and reposts by posted date, hour and country
- "geolocated" counting posts and reposts by posted date and the smallest possible geolocated unit (city, administrative level or country)
- "topwords" counting posts and reposts by posted date, country and the most popular words, (this excludes words used in the topic search)

The returned dataset can be cached for further calls if requested. Only one dataset per series is cached.

Value

A data frame containing the requested series for the requested period

See Also

detect_loop fs_loop

Examples

```
## Not run:
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   # Getting all country posts between 2020-jan-10 and 2020-jan-31 for all topics
   df <- get_aggregates(
     dataset = "country_counts",
     filter = list(period = c("2020-01-10", "2020-01-31"))
   )

   # Getting all country posts for the topic dengue
   df <- get_aggregates(dataset = "country_counts", filter = list(topic = "dengue"))

   # Getting all country posts between 2020-jan-10 and 2020-jan-31 for the topic dengue
    df <- get_aggregates(
        dataset = "country_counts",
         filter = list(topic = "dengue", period = c("2020-01-10", "2020-01-31"))
    )

## End(Not run)
```

---

get_alerts *Getting signals produced by the task generate_alerts of detect_loop*

Description

Returns a data frame of signals produced by the detect_loop, which are stored on the signal folder.

Usage

```
get_alerts(
  topic = character(),
  countries = numeric(),
  from = "1900-01-01",
  until = "2100-01-01",
  topposts = 0,
  limit = 0,
  duplicates = "all",
  progress = function(a, b) {
 }
)
```

Arguments

topic	Character vector. When it is not empty it will limit the returned signals to the provided topics, default: character()
countries	Character vector containing the names of countries or regions or a numeric vector containing the indexes of countries as displayed at the Shiny App to filter the signals to return., default: numeric()
from	Date defining the beginning of the period of signals to return, default: '1900-01-01'
until	Date defining the end of the period of signals to return, default: '2100-01-01'
topposts	Integer number of top posts to be added to the alert. These are obtained from the post index based on topwords and Lucene score, default: 0
limit	Maximum number of alerts returned, default: 0
duplicates	Character, action to decide what to do with alerts generated on the same day. Options are "all" (keep all alerts), "first" get only first alert and "last" for getting only the last alert
progress	Function, function to report progress it should receive two parameter a progress between 0 and 1 and a message, default: empty function

Details

For more details see the package vignette.

Value

a data frame containing the calculated alerts for the period. If no alerts are found then NULL is returned

See Also

generate_alerts

detect_loop

Examples

```
## Not run:
   library(episomer)
   # setting up the data folder
   message('Please choose the episomer data directory')
   setup_config(file.choose())

   # Getting signals produced for last 30 days for a particular country
   get_alerts(
     countries = c("Chile", "Australia", "France"),
     from = as.Date(Sys.time())-30,
     until = as.Date(Sys.time())
   )

## End(Not run)
```

get_tasks *Get the* detect_loop *task status*

Description

Reads the status of the detect_loop tasks and update it with changes requested by the Shiny app

Usage

```
get_tasks(statuses = list())
```

Arguments

statuses	Character vector for limiting the status of the returned tasks, default: list()

Details

After reading the tasks.json file and parsing it with jsonlite, this function will update the necessary fields in the tasks for executing and monitoring them.

Value

A named list containing all necessary information to run and monitor the detect loop tasks.

See Also

download_dependencies

update_geonames

update_languages

detect_loop

generate_alerts

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   tasks <- get_tasks()

## End(Not run)
```

`health_check` *Send email to administrator if a failure of episomer is detected*

Description

It validates if episomer is not collecting posts, aggregating posts or not calculating alerts

Usage

```
health_check(send_mail = TRUE, one_per_day = TRUE)
```

Arguments

`send_mail`	Boolean. Whether an email should be sent to the defined administrator, default: TRUE
`one_per_day`	Boolean. Whether a limit of one email per day will be applied, default: TRUE

Details

This function sends an email to the defined administrator if episomer is not collecting posts, aggregating posts or not calculating alerts

Value

A list of health check errors found

Examples

```
## Not run:
   #importing episomer
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
  #sending the email to the administrator if episomer components are not properly working
   health_check()

## End(Not run)
```

is_detect_running *Check whether the alert detection task is running*

Description

gets the alert detection runner execution status

Usage

```
is_detect_running()
```

Details

returns a logical value being TRUE if the alert detection task is actually running

Value

logical Whether the alert detection task is running

Examples

```
## Not run:
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   is_detect_running()

## End(Not run)
```

is_fs_running *Check whether the database is running*

Description

gets the database runner execution status

Usage

```
is_fs_running()
```

Details

returns a logical value being TRUE if the database runner is actually running

Value

logical Whether the database is running

Examples

```
## Not run:
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   is_fs_running()

## End(Not run)
```

`is_r_folder_present` *check whether the R folder is present in the current directory*

Description

check whether the R folder is present in the current directory

Usage

```
is_r_folder_present()
```

`is_search_running` *Check whether the post collection task is running*

Description

gets the post collection execution status

Usage

```
is_search_running()
```

Details

returns a logical value being TRUE if the post collection is running

Value

logical Whether the post collection is running

Examples

```
## Not run:
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   is_search_running()

## End(Not run)
```

keep_roxygen_and_function_declarations

internal function for exporting code of necessary functions for creating templates for a new social media

Description

internal function for exporting code of necessary functions for creating templates for a new social media

Usage

```
keep_roxygen_and_function_declarations(file_content)
```

Arguments

file_content	char, lines of the script to extract content from

missing_search_jobs *Missing search jobs if any*

Description

check if search jobs for each activated social media are running and return the list names of social media with missing jobs

Usage

```
missing_search_jobs()
```

Details

returns a char vector with the name of missing social media

Value

char list of social media with no search job running

Examples

```
## Not run:
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   missing_search_jobs()

## End(Not run)
```

`register_detect_runner_task` *Registers the alert detection task*

Description

registers the alert detection task or stops if no configuration has been set or if it is already running

Usage

```
register_detect_runner_task()
```

Details

Registers the alert detection task or stops if no configuration has been set or if it is already running. To generate alerts, this task needs the post collection to had successfully run since the last time it ran. This function will use the task scheduler on windows and will fall back to launching the runner as a separate process (attached to this session) on Linux.

Value

Nothing

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   register_detect_runner_task()

## End(Not run)
```

`register_fs_monitor` *Registers the fs_monitor for the current process or exits*

Description

registers the fs_monitor (by writing fs.monitor.PID file) for the current process or stops if no configuration has been set or if it is already running

Usage

```
register_fs_monitor()
```

Details

Registers the fs_monitor (by writing fs.monitor.PID file) for the current process or stops if no configuration has been set or if it is already running This function has been exported so it can be properly called from the future package on the database runner, but it is not intended to be directly called by end users.

Value

Nothing

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   register_fs_monitor()

## End(Not run)
```

`register_fs_runner_task` *Registers the episomer database task*

Description

registers the episomer database task or stops if no configuration has been set or if it is already running

Usage

```
register_fs_runner_task()
```

Details

Registers the episomer database task or stops if no configuration has been set or if it is already running. This task need the dependencies, geonames and languages steps to have been successfully ran. This can be done on the shiny app configuration page or by manually running the detect_runner_task. This function will try to use the task scheduler on windows and will fall back to launching the runner as a separate process (attached to this session) on Linux.

Value

Nothing

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   register_fs_runner_task()

## End(Not run)
```

`register_search_runner_task` *Registers the social media message collection task*

Description

registers the social media message collection task or stops if no configuration has been set or if it is already running

Usage

```
register_search_runner_task()
```

Details

Registers the message collextion task or stops if no configuration has been set or if it is already running. Bluesky authentication needs to be previously set on the shiny app or by calling set_auth(). You can test if authentication is working on the shiny app troubleshot page or by calling (with dplyr): episomer::check_all() This function will use the task scheduler on windows and will fall back to launching the runner as a separate process (attached to this session) on Linux.

Value

Nothing

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   register_search_runner_task()

## End(Not run)
```

save_config *Save the configuration changes*

Description

Permanently saves configuration changes to the data folder (excluding social media credentials, but not SMTP credentials)

Usage

```
save_config(data_dir = conf$data_dir, properties = TRUE, sm_topics = NULL)
```

Arguments

data_dir	Path to a directory to save configuration settings, Default: conf$data_dir
properties	Whether to save the general properties to the properties.json file, default: TRUE
sm_topics	Whether to save topic download plans to the topics.json file, default: TRUE

Details

Permanently saves configuration changes to the data folder (excluding social media credentials, but not SMTP credentials) to save social media credentials please use sm_api_set_auth_bluesky

Value

Nothing

See Also

setup_config sm_api_set_auth_bluesky

Examples

```
## Not run:
   library(episomer)
   #load configuration
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   #make some changes
   #conf$collect_span = 90
   #saving changes
   save_config()

## End(Not run)
```

search_loop *Runs the search loop*

Description

Infinite loop ensuring the permanent collection of social media messages

Usage

```
search_loop(
  data_dir = NA,
  sandboxed = FALSE,
  log_to_file = TRUE,
  max_requests = 0
)

search_loop_worker(network, data_dir = NA, sandboxed = FALSE, max_requests = 0)
```

Arguments

data_dir	The data directory to use.
sandboxed	Whether to run in sandboxed mode.
log_to_file	logical indicating if the search loop should log to a file. Default: TRUE
max_requests	The maximum number of requests to perform.
network	The social media network to search on.

Details

The detect loop is a pure R function designed for downloading posts from the social media search API. It can handle several topics ensuring that all of them will be downloaded fairly using a round-robin philosophy and respecting social media API rate-limits.

The progress of this task is reported on the 'topics.*.json' files which are created by this function. This function will try to collect posts respecting a 'collect_span' window in minutes, which is defined on the Shiny app and defaults to 60 minutes.

To see more details about the collection algorithm please see episomer vignette.

In order to work, this task needs Bluesky credentials, which can be set on the Shiny app or using sm_api_set_auth_bluesky

The search loop worker is a function that runs the search loop for a given social media network. It is used to run the search loop in a separate process.

Value

Nothing

Nothing

See Also

sm_api_set_auth_bluesky

Examples

```
## Not run:
   #Running the search loop
   library(episomer)
   message('Please choose the episomer data directory')
   search_loop(file.choose())

## End(Not run)
```

---

search_posts *perform full text search on posts collected with episomer*

Description

perform full text search on posts collected with episomer (posts migrated from episomer v<1.0.x are also included)

Usage

```
search_posts(
  query = NULL,
  topic = NULL,
  from = NULL,
  to = NULL,
  countries = NULL,
  mentioning = NULL,
  users = NULL,
  hide_users = FALSE,
  action = NULL,
  max = 100,
  by_relevance = FALSE
)
```

Arguments

query	character. The query to be used if a text it will match the post text. To see how to match particular fields please see details, default:NULL
topic	character, Vector of topics to include on the search default:NULL
from	an object which can be converted to ‘"POSIXlt"’ only posts posted after or on this date will be included, default:NULL
to	an object which can be converted to ‘"POSIXlt"’ only posts posted before or on this date will be included, default:NULL

countries	character or numeric, the position or name of episomer regions to be included on the query, default:NULL
mentioning	character, limit the search to the posts mentioning the given users, default:NULL
users	character, limit the search to the posts created by the provided users, default:NULL
hide_users	logical, whether to hide user names on output replacing them by the USER keyword, default:FALSE
action	character, an action to be performed on the search results respecting the max parameter. Possible values are 'delete' or 'anonymise' , default:NULL
max	integer, maximum number of posts to be included on the search, default:100
by_relevance	logical, whether to sort the results by relevance of the matching query or by indexing order, default:FALSE If not provided the system will try to reuse the existing one from last session call of setup_config or use the EPI_HOME environment variable, default: NA

Details

episomer translate the query provided by all parameters into a single query that will be executed on post indexes which are weekly indexes. The q parameter should respect the syntax of the Lucene classic parser https://lucene.apache.org/core/8_5_0/queryparser/org/apache/lucene/queryparser/classic/QueryParser.html So other than the provided parameters, multi field queries are supported by using the syntax field_name:value1;value2 AND, OR and -(for excluding terms) are supported on q parameter. Order by week is always applied before relevance so even if you provide by_relevance = TRUE all of the matching posts of the first week will be returned first

Value

a data frame containing the posts matching the selected filters, the data frame contains the following columns: user_name, created_date, is_geo_located, is_quote, text, text_loc, user_id, hash, quote_lang, totalCount, created_at, topic_post_id, topic, lang, user_name, quoted_text, id, quopted_text_loc, tags

See Also

search_loop

detect_loop

Examples

```
## Not run:
   #Running the detect loop
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   df <- search_posts(
        q = "vaccination",
        topic="COVID-19",
        countries=c("Chile", "Australia", "France"),
        from = Sys.Date(),
```

```
      to = Sys.Date()
  )
  df$text

## End(Not run)
```

`search_topic` *performs a search following the given plan, query and topic.*

Description

performs a search following the given plan, query and topic.

Usage

```
search_topic(plan, query, topic, sandboxed = FALSE)
```

Arguments

`plan`	list, the plan that defines the timespan to search for
`query`	char, the query to use when searching for the topic
`topic`	char, the topic associated to this search
`sandboxed`	logical, whether to un in sandboxed mode which prevent sending data to the embedded database.

`search_touch` *updates the modificatin date of a flag file indicating that messages have been sent to the embedded database.*

Description

updates the modificatin date of a flag file indicating that messages have been sent to the embedded database.

Usage

```
search_touch()
```

setup_config *Load episomer application settings*

Description

Load episomer application settings from the designated data directory

Usage

```
setup_config(
  data_dir = if (exists("data_dir", where = conf)) conf$data_dir else if
   (Sys.getenv("EPI_HOME") != "") Sys.getenv("EPI_HOME") else file.path(tempdir(),
     "episomer"),
  ignore_keyring = FALSE,
  ignore_properties = FALSE,
  ignore_topics = FALSE,
  save_properties_first = FALSE,
  save_topics_first = list()
)
```

Arguments

data_dir	Path to the directory containing the application settings (it must exist). If not provided it takes the value of the latest call to setup_config in the current session, or the value of the EPI_HOME environment variable or episomer subdirectory in the working directory, default: if (exists("data_dir", where = conf)) conf$data_dir else if (Sys.getenv("EPI_HOME") != "") Sys.getenv("EPI_HOME") else file.path(tempdir(), "episomer")
ignore_keyring	Whether to skip loading settings from the keyring (social media and SMTP credentials), default: FALSE
ignore_properties	Whether to skip loading settings managed by the Shiny app in properties.json file, Default: FALSE
ignore_topics	Whether to skip loading settings defined in the topics.xlsx file and download plans from topics.json file, default: FALSE
save_properties_first	Whether to save current settings before loading new ones from disk, default: FALSE
save_topics_first	List of topics to save before loading new ones from disk, default: list()

Details

episomer relies on settings and data stored in a system folder, so before loading the dashboard, collecting posts or detecting alerts the user has to designate this folder. When a user wants to use episomer from the R console they will need to call this function for initialisation. The 'data_folder'

can also be given as a parameter for program launch functions episomer_app, search_loop or detect_loop, which will internally call this function.

This call will fill (or refresh) a package scoped environment 'conf' that will store the settings. Settings stored in conf are:

- General properties of the Shiny app (stored in properties.json)
- Download plans from the social media collection process (stored in topics.json merged with data from the topics.xlsx file
- Credentials for social media APIs and SMTP stored in the defined keyring

When calling this function and the keyring is locked, a password will be prompted to unlock the keyring. This behaviour can be changed by setting the environment variable 'ecdc_social_tool_kr_password' with the password.

Changes made to conf can be stored permanently (except for 'data_dir') using:

- save_config, or
- sm_api_set_auth_bluesky

Value

Nothing

See Also

save_config sm_api_set_auth_bluesky episomer_app search_loop detect_loop

Examples

```
## Not run:
   library(episomer)
   #loading system settings
   message('Please choose the episomer data directory')
   setup_config(file.choose())

## End(Not run)
```

---

sm_api_get_token *Get social media token*

Description

Get social media token

Usage

```
sm_api_get_token(network)
```

Arguments

network	char, network the name of the social media

Details

this function implements the authethication on the respective social media and stores it securely. It is called by the search loop before starting message collection

---

sm_api_get_token_bluesky

Get social media token

Description

Get social media token

Usage

```
sm_api_get_token_bluesky()
```

Details

this function implements the authethication on the respective social media and stores it securely. It is called by the search loop before starting message collection

Value

Token

---

sm_api_got_rows_bluesky

Receive the results provided by a request and return a logical value indicating whether the request returned results.

Description

Receive the results provided by a request and return a logical value indicating whether the request returned results.

Usage

```
sm_api_got_rows_bluesky(results)
```

Arguments

results	list, results of the request in the standardised episomer format

Details

This function needs to be implemented for each social media since it is interpreted as 'No more messages are available for this plan' When FALSE is returned the underlying plan will be ended

---

sm_api_search *Search posts for a given query and produce standardised results*

Description

Search posts for a given query and produce standardised results

Usage

```
sm_api_search(query, token, plan)
```

Arguments

query	char, the query to use in the search in nattive format
token	char, the token to authenticate against the underlying social media api
plan	list, the plan containing the targeted time frame

Details

This function impleents the search of messages for a given query and plan.

---

sm_api_search_bluesky *Search posts for a given query and produce standardised results*

Description

Search posts for a given query and produce standardised results

Usage

```
sm_api_search_bluesky(query, token, plan, max_retries = 10, verbose = TRUE)
```

Arguments

query	query to search for
token	Access token
plan	Plan object
max_retries	Maximum number of retries
verbose	Whether to print progress messages

Details

This function impleents the search of messages for a given query and plan. The query has been already transformed by sm_api_translate_query_xxx so it is expected to be already in the native social media format The token has been produced by the function sm_api_get_token_xxx and shoulb be used to autenticate the request The plan should be used to extract the precise time-span to request messages

- Response attributes - network: string - posts: [...] - count: number, - pagination:{...} - post in posts - id: string - uri: string - created_at: datetime - user_name: string - text: string - lang: string - quoted_text: string - quoted_lang: string - tags: [...] - urls: [...] - categories: [...] - tag in post.tags: string - url in post.urls: string - category in post.categories: string - pagination: attributes used for paginating requests

This is the implementation of the bluesky search. You can find more information about the bluesky API here: https://docs.bsky.app/docs/api/app-bsky-feed-search-posts Additionnaly more information about the "post" object is available here: https://atproto.blue/en/latest/atproto/atproto_client.models.app.bsky.feed.defs.html#atproto_client.models.app.bsky.feed.defs.PostView

Value

List with posts int the standard post format defined in episomer

---

sm_api_set_auth *Save credentials for a social media provider and store them securely*

Description

Update configuration object and underlying storage with given bluesky username and password. The password is encrypted.

Usage

```
sm_api_set_auth(network, shiny_input_list)
```

Arguments

network	char, the name of the social media
shiny_input_list	list, the shiny input object to extract the information from

`sm_api_set_auth_bluesky` *Save bluesky credentials (login and password) and store them securely*

Description

Update configuration object and underlying storage with given bluesky username and password. The password is encrypted.

Usage

```
sm_api_set_auth_bluesky(shiny_input_list)
```

Arguments

`shiny_input_list`	List of shiny input values containing the bluesky specific credentials

Details

Update authentication tokens in configuration object

Value

Nothing. Called for side effects (saving credentials)

See Also

`save_config`

Examples

```
## Not run:
 #Setting the configuration values
   sm_api_set_auth_bluesky(
    shiny_input_list = list(bluesky_user = "your user here", bluesky_password = "your password"),
   )

## End(Not run)
```

`sm_api_translate_query` *Translate a parsed query to the social media query format*

Description

Translate a parsed query to the social media query format

Usage

```
sm_api_translate_query(network, parsed)
```

Arguments

`network`	char, the name of the social media
`parsed`	char, parsed query to translate in one or many native equivalent queries

Details

This function receive as a parameter a parsed query from the a topic in the list of topics. It can return one or many queries that matches the expected results using the underlying social media API

`sm_api_translate_query_bluesky` *Translate a parsed query to the social media query format*

Description

Translate a parsed query to the social media query format

Usage

```
sm_api_translate_query_bluesky(parsed)
```

Arguments

`parsed`	list. Named list containing two attributes 'ors' and 'neg'. 'ors' is a list containing subqueries that should be contatenated with an OR logic - each subquery in 'ors' is a list containing terms that should be concatecated with an AND logic - each term in 'ors subqueries' is a list of synonyms for a particular literal 'neg' is a list of terms that should be removed from any result of the query

Details

This function receive as a parameter a parsed query from the a topic in the list of topics. It can return one or many queries that matches the expected results using the underlying social media API

`sm_api_update_plan_after_request`

Takes a plan after a request has been performed and the standardised results provided as returned by the function sm_api_search_xxx it should return a plan with necessary information updated so next request will continue paginating as expected to obtain targeted messages

---

Description

Takes a plan after a request has been performed and the standardised results provided as returned by the function sm_api_search_xxx it should return a plan with necessary information updated so next request will continue paginating as expected to obtain targeted messages

Usage

```
sm_api_update_plan_after_request(plan, results)
```

Arguments

`plan`	list, the plan producing the resusts
`results`	list, the result of the query in episomer format

`sm_api_update_plan_after_request_bluesky`

Takes a plan after a request has been performed and the standardised results provided as returned by the function sm_api_search_xxx it should return a plan with necessary information updated so next request will continue paginating as expected to obtain targeted messages

---

Description

Takes a plan after a request has been performed and the standardised results provided as returned by the function sm_api_search_xxx it should return a plan with necessary information updated so next request will continue paginating as expected to obtain targeted messages

Usage

```
sm_api_update_plan_after_request_bluesky(plan, results)
```

Arguments

`plan`	list, the plan to update
`results`	list, the obtained results of the plan

`sm_plan_first_attributes` *returns a list with the custom social media plan attributes with the default values for a first plan in a topic.*

Description

returns a list with the custom social media plan attributes with the default values for a first plan in a topic.

Usage

```
sm_plan_first_attributes(network)
```

Arguments

`network`	char, the name of the social media

`sm_plan_first_attributes_bluesky` *returns a list with the custom social media plan attributes with the default values for a first plan in a topic.*

Description

returns a list with the custom social media plan attributes with the default values for a first plan in a topic.

Usage

```
sm_plan_first_attributes_bluesky()
```

`sm_plan_format` *prepare a plan for serialisation by transforming attributes with non standard serialisation types e.g. dates to string on the format expected by the parse functions*

Description

prepare a plan for serialisation by transforming attributes with non standard serialisation types e.g. dates to string on the format expected by the parse functions

Usage

```
sm_plan_format(plan)
```

Arguments

plan	the plan to format the attributes

---

sm_plan_format_bluesky

prepare a plan for serialisation by transforming attributes with non standard serialisation types e.g. dates to string on the format expected by the parse functions

---

Description

prepare a plan for serialisation by transforming attributes with non standard serialisation types e.g. dates to string on the format expected by the parse functions

Usage

```
sm_plan_format_bluesky(plan)
```

Arguments

plan	list, the plan with the attributes go format

---

sm_plan_get_progress

calculates and returns the estimated progress of a plan using custom attributes

---

Description

calculates and returns the estimated progress of a plan using custom attributes

Usage

```
sm_plan_get_progress(plan)
```

Arguments

plan	list, the plan to obtain the progress from

`sm_plan_get_progress_bluesky` *calculates and returns the estimated progress of a plan using custom attributes*

Description

calculates and returns the estimated progress of a plan using custom attributes

Usage

```
sm_plan_get_progress_bluesky(plan)
```

Arguments

`plan`	list, the plan to calculate the progress from

`sm_plan_next_attributes` *returns a list with the custom social media plan attributes with the values for the plan to be run after the plans provided as parameter*

Description

returns a list with the custom social media plan attributes with the values for the plan to be run after the plans provided as parameter

Usage

```
sm_plan_next_attributes(network, plans)
```

Arguments

`network`	char, network the name of the social media
`plans`	list, list of plans to use for calculate new plan attributes

`sm_plan_next_attributes_bluesky` *returns a list with the custom social media plan attributes with the values for the plan to be run after the plans provided as parameter*

Description

returns a list with the custom social media plan attributes with the values for the plan to be run after the plans provided as parameter

Usage

```
sm_plan_next_attributes_bluesky(plans)
```

Arguments

`plans`	list, the list with current plans of the given topic to calculate next plan attributes

`sm_plan_next_search_info` *Return a message to log what the next query is going to be for the current plan*

Description

Return a message to log what the next query is going to be for the current plan

Usage

```
sm_plan_next_search_info(plan)
```

Arguments

`plan`	list, the plan to obtain the message

`sm_plan_parse_attributes` *parse custom attributes for a plan in this specific social media and returns them as a named list with attributes in the proper type*

Description

parse custom attributes for a plan in this specific social media and returns them as a named list with attributes in the proper type

Usage

```
sm_plan_parse_attributes(network, ...)
```

Arguments

`network`	char, the name of the social media
`...`	list, triple dot arguments to be passed as a parameter tu underlying functions

`sm_plan_parse_attributes_bluesky` *parse custom attributes for a plan in this specific social media and returns them as a named list with attributes in the proper type*

Description

parse custom attributes for a plan in this specific social media and returns them as a named list with attributes in the proper type

Usage

```
sm_plan_parse_attributes_bluesky(...)
```

Arguments

`...`	list, triple dot arguments to be passed as a parameter tu underlying functions

---

`sm_plan_search_info_bluesky` *Return a message to log what the next query is going to be for the current plan*

Description

Return a message to log what the next query is going to be for the current plan

Usage

```
sm_plan_search_info_bluesky(plan)
```

Arguments

`plan`	list, the plan to print the information about

`stop_detect_runner_task` *Stops the alert detection task*

Description

stops the alert detection task

Usage

```
stop_detect_runner_task()
```

Details

Stops the alert detection task if it is already running This function will try also deactivate the respective scheduled task on Windows.

Value

Nothing

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   stop_detect_runner_task()

## End(Not run)
```

stop_fs_runner_task *Stops the episomer database task*

Description

stops the episomer database task

Usage

```
stop_fs_runner_task()
```

Details

Stops the episomer database task if it is already running This function will try also deactivate the respective scheduled task on Windows.

Value

Nothing

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   stop_fs_runner_task()

## End(Not run)
```

stop_search_runner_task *Stops the post collection task*

Description

stops the post collection task

Usage

```
stop_search_runner_task()
```

Details

Stops the post collection task if it is already running This function will try also deactivate the respective scheduled task on Windows.

Value

Nothing

Examples

```
## Not run:
   #getting tasks statuses
   library(episomer)
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   stop_search_runner_task()

## End(Not run)
```

`trend_line` *Plot the trendline report of episomer dashboard*

Description

Generates a trendline chart of number of posts by region for a single topic, including alerts detected using the reweighted version of the EARS algorithm

Usage

```
trend_line(
  sms,
  topic,
  countries = c(1),
  date_type = "created_date",
  date_min = "1900-01-01",
  date_max = "2100-01-01",
  with_quotes = FALSE,
  alpha = 0.025,
  alpha_outlier = 0.05,
  k_decay = 4,
  no_historic = 7,
  bonferroni_correction = FALSE,
  same_weekday_baseline = FALSE
)
```

Arguments

`sms`	Social media
`topic`	Character(1) containing the topic to use for the report
`countries`	Character vector containing the name of the countries and regions to plot or their respective indexes on the Shiny app select, default: c(1)

date_type	Character vector specifying the time granularity of the report either 'created_weeknum' or 'created_date', default: 'created_date'
date_min	Date indicating start of the reporting period, default: "1900-01-01"
date_max	Date indicating end of the reporting period, default: "2100-01-01"
with_quotes	Logical value indicating whether to include reposts in the time series, default: FALSE
alpha	Numeric(1) value indicating the alert detection confidence, default: 0.025
alpha_outlier	Numeric(1) value indicating the outliers detection confidence for downweighting, default: 0.05
k_decay	Strength of outliers downweighting, default: 4
no_historic	Number of observations to build the baseline for signal detection, default: 7
bonferroni_correction	Logical value indicating whether to apply the Bonferroni correction for signal detection, default: FALSE
same_weekday_baseline	Logical value indicating whether to use same day of weeks for building the baseline or consecutive days, default: FALSE

Details

Produces a multi-region line chart for a particular topic of number of posts collected based on the provided parameters. Alerts will be calculated using a modified version of the EARS algorithm that applies a Farrington inspired downweighting of previous outliers.

Days in this function are considered as contiguous blocks of 24 hours, starting from the previous hour preceding the last collected post.

This function requires that search_loop and detect_loop have already run successfully in order to show results.

Value

A named list containing two elements: 'chart' with the ggplot2 figure and 'data' containing the data frame that was used to build the chart.

See Also

create_map create_topwords generate_alerts detect_loop search_loop

Examples

```
## Not run:
   message('Please choose the episomer data directory')
   setup_config(file.choose())
   #Getting trendline for dengue for South America for the last 30 days
   trend_line(
     topic = "dengue",
     countries = "South America",
     date_min = as.Date(Sys.time())-30,
```

```
  date_max=as.Date(Sys.time())
)

## End(Not run)
```

`update_geonames` *Updates the local copy of the GeoNames database*

Description

Downloading and indexing a fresh version of the GeoNames database from the provided URL

Usage

```
update_geonames(tasks = get_tasks())
```

Arguments

`tasks`	Tasks object for reporting progress and error messages, default: get_tasks()

Details

Run a one shot task to download and index a local copy of the GeoNames database. The GeoNames geographical database covers all countries and contains over eleven million place names that are available; Creative Commons Attribution 4.0 License.

The URL to download the database from is set on the configuration tab of the Shiny app, in case it changes.

The indexing is developed in Spark and Lucene

A prerequisite to this function is that the `search_loop` must already have stored collected posts in the search folder and that the task `download_dependencies` has been successfully run.

Normally this function is not called directly by the user but from the `detect_loop` function.

Value

The list of tasks updated with produced messages

See Also

`download_dependencies`

`detect_loop`

`get_tasks`

Examples

```
## Not run:
   library(episomer)
   # setting up the data folder
   message('Please choose the episomer data directory')
   setup_config(file.choose())

   # geolocating last posts
   tasks <- update_geonames()

## End(Not run)
```

`update_languages` *Updates local copies of languages*

Description

Downloading and indexing a fresh version of language models tagged for update on the Shiny app configuration tab

Usage

```
update_languages(tasks = get_tasks(), reuse_downloads = FALSE)
```

Arguments

`tasks`	Tasks object for reporting progress and error messages, default: get_tasks()
`reuse_downloads`	logical indicating if the downloads should be reused if they already exist, default: FALSE

Details

Run a one shot task to download and index a local fasttext pretrained models. A fasttext model is a collection of vectors for a language automatically produced scrolling a big corpus of text that can be used to capture the semantic of a word.

The URL to download the vectors from are set on the configuration tab of the Shiny app.

This task will also update SVM models to predict whether a word is a location that will be used in the geolocation process.

The indexing is developed in SPARK and Lucene.

A prerequisite to this function is that the `search_loop` must already have stored collected posts in the search folder and that the tasks `download_dependencies` and `update_geonames` has been run successfully.

Normally this function is not called directly by the user but from the `detect_loop` function.

Value

The list of tasks updated with produced messages

See Also

download_dependencies

update_geonames

detect_loop

get_tasks

Examples

```
## Not run:
   library(episomer)
   # setting up the data folder
   message('Please choose the episomer data directory')
   setup_config(file.choose())

   # updating language tasks
   tasks <- update_languages()

## End(Not run)
```

Index