Package ‘DisasterAlert’

July 21, 2025

Type Package

Version 1.0.0

Title Disaster Alert and Sentiment Analysis

Description
By systematically aggregating and processing textual reports from earthquakes, floods, storms, wildfires, and other natural disasters, the framework enables a holistic assessment of crisis narratives.
Intelligent cleaning and normalization techniques transform raw commentary into structured data, ensuring
precise extraction of disaster-specific insights. Collective sentiments of affected communities are quantitatively scored and qualitatively categorized, providing a multifaceted view of societal responses
under duress. Interactive geographic maps and temporal charts illustrate the evolution and spatial dispersion
of emotional reactions and impact indicators.

License GPL-3

Depends R (>= 4.1.0)

Imports methods, tidyverse, ggplot2, leaflet, wordcloud,
textdata,tidytext, quanteda, tidyr, plotly, htmlwidgets,
RColorBrewer, dplyr, stringr, scales,DT

Suggests rmarkdown, testthat (>= 3.0.0)

Maintainer Leila Marvian Mashhad `<Leila.marveian@gmail.com>`

NeedsCompilation no

Author Hossein Hassani [aut],
Nadejda Komendantova [aut],
Leila Marvian Mashhad [aut, cre]

Encoding UTF-8

Repository CRAN

Date/Publication 2025-07-21 09:01:56 UTC

Contents

`analyze_disaster_sentiment` *Analyze Disaster Sentiment*

Description

It performs sentiment analysis on disaster-related text data using multiple methods.

Usage

```
analyze_disaster_sentiment(data, text_column = "User_Comment", method = "afinn")
```

Arguments

`data`	A data frame containing disaster data
`text_column`	Name of the column containing text to analyze (default: "User_Comment")
`method`	Sentiment analysis method: "afinn", "bing", "nrc", or "syuzhet" (default: "afinn")

Value

Data frame with added sentiment scores and categories.

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
tweets_df <- data.frame(
  User_Comment = c(
    "The earthquake was terrible and scary",
    "Rescue teams are doing a wonderful job, I feel hopeful",
    "No damage here, everything feels normal"
  ),
  stringsAsFactors = FALSE
)
result_df <- analyze_disaster_sentiment(tweets_df, text_column = "User_Comment", method = "afinn")
print(result_df)
```

`calculate_sentiment_stats` *Calculate Sentiment Statistics*

Description

This function calculates comprehensive statistics for sentiment analysis.

Usage

```
calculate_sentiment_stats(data)
```

Arguments

`data`	A data.frame with sentiment analysis results.

Value

List of statistical summaries.

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
result_df <- data.frame(
  User_Comment = c(
    "The earthquake was terrible and scary",
    "Rescue teams are doing a wonderful job, I feel hopeful",
    "No damage here, everything feels normal"
  ),
  sentiment_score = c(-2.5, 3.0, -0.5),
  sentiment_category = c("Negative", "Positive", "Neutral"),
  stringsAsFactors = FALSE
)

stats <- calculate_sentiment_stats(result_df)

str(stats)

print("=== Overall Sentiment ===")
print(stats$overall_sentiment)

print("=== Extreme Comments ===")
print(stats$extreme_comments)
print("=== Summary Counts ===")
print(stats$summary)
```

`clean_disaster_text` *Clean Disaster Text*

Description

It cleans and preprocesses text data for analysis.

Usage

```
clean_disaster_text(text)
```

Arguments

`text`	Vector of text strings to clean

Value

Vector of cleaned text strings.

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
raw_comments <- c(
  "The earthquake!!!  happened @ midnight...",
  NA,
  "Floods in 2025 were terrible?   Really scary.",
  "Support & rescue teams: amazing work!"
)


cleaned_comments <- clean_disaster_text(raw_comments)


print(cleaned_comments)
```

create_interactive_plots

Create Interactive Plots

Description

This function creates interactive plots using plotly for better user experience.

Usage

```
create_interactive_plots(data, plot_type = "scatter")
```

Arguments

`data`	A data.frame with sentiment analysis results
`plot_type`	Type of interactive plot: "scatter", "bar", "timeline"

Value

Plotly object.

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
sample_data <- data.frame(
  City              = c("CityA", "CityB", "CityA", "CityC", "CityB"),
  Longitude         = c(10.0, 11.5, 10.0, 12.2, 11.5),
  Latitude          = c(50.1, 49.9, 50.1, 50.5, 49.9),
  sentiment_score   = c( 2.5, -1.0,  0.0,  3.0, -2.0),
  sentiment_category= c("Positive","Negative","Neutral","Positive","Negative"),
  User_Comment      = c(
    "Amazing rescue efforts!",
    "Terrible flooding last night.",
    "All calm here.",
    "Hope everyone is safe.",
    "Worst disaster ever."
  ),
  Timestamp         = as.POSIXct(c(
    "2025-07-10 14:00", "2025-07-10 15:30",
    "2025-07-11 10:00", "2025-07-11 12:45",
    "2025-07-12 09:20"
  ))
)


scatter_plot <- create_interactive_plots(sample_data, plot_type = "scatter")
```

```
bar_plot <- create_interactive_plots(sample_data, plot_type = "bar")
bar_plot

timeline_plot <- create_interactive_plots(sample_data, plot_type = "timeline")
timeline_plot
```

`create_sentiment_map` *Create Interactive Sentiment Map*

Description

This function creates an interactive Leaflet map showing disaster locations colored by sentiment.

Usage

```
create_sentiment_map(data, lat_col = "Latitude", lon_col = "Longitude")
```

Arguments

`data`	A data.frame with sentiment analysis results
`lat_col`	Name of latitude column (default: "Latitude")
`lon_col`	Name of longitude column (default: "Longitude")

Value

Leaflet map object

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
sample_data <- data.frame(
  City               = c("CityA", "CityB", "CityC"),
  Longitude          = c(10.0, 11.5, 12.2),
  Latitude           = c(50.1, 49.9, 50.5),
  sentiment_score    = c( 2.5, -1.0,  0.0),
  sentiment_category = c("Positive", "Negative", "Neutral"),
  User_Comment       = c(
    "Amazing rescue efforts!",
    "Terrible flooding last night.",
    "All calm here."
  ),
  stringsAsFactors = FALSE
)
```

```
sentiment_map <- create_sentiment_map(sample_data,
                                      lat_col = "Latitude",
                                      lon_col = "Longitude")

sentiment_map
```

---

generate_tweets *Generate Random Tweets*

Description

This function Generates synthetic tweets with weather conditions and sentiment.

Usage

```
generate_tweets(n)
```

Arguments

n	The number of tweets to generate

Value

A data.frame containing two columns: Date: The date of the tweet T1: The text of the tweet

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
tweets <- generate_tweets(100)
head(tweets)
```

---

generate_word_clouds *Generate Word Clouds from Tweets*

Description

This function Creates and plots a word cloud based on the cleaned and stemmed words extracted from one or more tweets.

Usage

```
generate_word_clouds(tweet)
```

Arguments

`tweet`	A character vector of tweet texts, or a data frame/tibble whose first column contains tweet texts.

Value

The main side effect is the word cloud drawing.

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
# Generate word cloud from a single tweet
tweet_text <- "This is a sample tweet for word cloud generation!"
generate_word_clouds(tweet_text)

## This will generate a word cloud image where the most frequent words
## in the tweet will be displayed larger.
```

`plot_sentiment_distribution` *Plot Sentiment Distribution*

Description

This function Creates various plots showing sentiment distribution.

Usage

```
plot_sentiment_distribution(data, plot_type = "bar")
```

Arguments

`data`	A data frame with sentiment analysis results
`plot_type`	Type of plot: "pie", "bar", "histogram", or "geographic"

Value

ggplot object or plot.

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
sample_data <- data.frame(
  sentiment_score    = c( 2.5, -1.0,  0.0,  3.0, -2.0,  1.5, -0.7),
  sentiment_category = c("Positive", "Negative", "Neutral", "Positive",
                         "Negative", "Positive", "Neutral"),
  Longitude          = c(10.0, 11.5, 10.0, 12.2, 11.5, 10.8, 12.0),
  Latitude           = c(50.1, 49.9, 50.1, 50.5, 49.9, 50.3, 50.4),
  stringsAsFactors   = FALSE
)

bar_plot <- plot_sentiment_distribution(sample_data, plot_type = "bar")
print(bar_plot)

hist_plot <- plot_sentiment_distribution(sample_data, plot_type = "histogram")
print(hist_plot)

plot_sentiment_distribution(sample_data, plot_type = "pie")

geo_plot <- plot_sentiment_distribution(sample_data, plot_type = "geographic")
print(geo_plot)
```

`process_tweet` *Preprocess Tweets for Sentiment Analysis*

Description

This function takes a list of tweets as input and performs various preprocessing steps to prepare the data for sentiment analysis.

Usage

```
process_tweet(tweet)
```

Arguments

`tweet`	A vector of tweets

Value

A list including:

A vector containing preprocessed tweets.

A vector containing tokens of tweets.

Author(s)

Hossein Hassani and Leila Marvian Mashhad and Nadejda Komendantova.

Examples

```
tweets_data <- "I'm feeling really happy today! #goodvibes"

preprocessed_tweets <- process_tweet(tweets_data)
print(preprocessed_tweets)
```

Index