Package: MadanTextNetwork (via r-universe)

September 4, 2024

Type Package

Title Persian Textmining Tool for Co-Occurrence_Network

Version 0.1.0

Description MadanText_co-occurrence_network is an open-source software designed specifically for text mining in the Persian language. It adds co-occurrence network functionality to MadanText. The input file replaces the text format with an Excel format.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

Depends R (>= 4.0.0)

Imports xlsx, glue, lattice, stopwords,textmineR, tidytext, tidyr, udpipe, PersianStemmer, shiny (>= 1.8.0), shinythemes, tm, dplyr, hwordcloud, stringr, stringi, topicmodels, igraph, ngram, visNetwork

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

RemoteUrl https://github.com/kidoishi/madantextnetwork

RemoteRef HEAD

RemoteSha 97f79cc59a6156eadf77d20af12184ac5b508ddf

Contents

`ASDATA.FRAME` *Convert to Data Frame*

Description

This function converts the given object to a data frame.

Usage

```
ASDATA.FRAME(x)
```

Arguments

`x`	An object to be converted into a data frame.

Value

A data frame.

Examples

```
data <- ASDATA.FRAME(matrix(1:4, ncol = 2))
```

cluster.graph *Cluster a Graph and Extract Largest Component*

Description

This function applies clustering to a graph and extracts the largest connected component.

Usage

```
cluster.graph(network)
```

Arguments

network	A graph object.

Value

A list containing the largest connected component graph, node membership, and node importance data frame.

Examples

```
## Not run:
  # Assuming 'network' is a predefined graph object
  cluster.graph(network)

## End(Not run)
```

Community.Detection.Membership *Get Community Membership of a Graph*

Description

This function applies community detection to a graph and returns the membership information of each node.

Usage

```
Community.Detection.Membership(network)
```

Arguments

network	A graph object.

Value

A data frame of node names and their community membership.

Examples

```
## Not run:
  network <- make_graph("Zachary")
  membership_info <- Community.Detection.Membership(network)
  print(membership_info)

## End(Not run)
```

`Community.Detection.Plot`

Plot Community Detection in a Graph

Description

This function applies community detection to a graph and plots the result.

Usage

```
Community.Detection.Plot(network)
```

Arguments

`network`	A graph object.

Value

A plot of the graph with community detection.

Examples

```
## Not run:
  # Assuming 'network' is a predefined graph object
  # network <- make_graph("Zachary")
  Community.Detection.Plot(network)

## End(Not run)
```

f3 *Persian Text Normalization and Stemming*

Description

This function normalizes Persian text by replacing specific characters and applies stemming.

Usage

```
f3(x)
```

Arguments

x	A character vector of Persian text.

Value

A character vector of normalized and stemmed text.

Examples

```
## Not run:
  text <- c("Persian text here")
  normalized_text <- f3(text)

## End(Not run)
```

f5 *Filter Data Frame by Document ID*

Description

This function filters a data frame by the specified document ID. If the ID is 0, the entire data frame is returned.

Usage

```
f5(UPIP, I)
```

Arguments

UPIP	A data frame with a column named 'doc_id'.
I	An integer representing the document ID.

Value

A filtered data frame.

Examples

```
data <- data.frame(doc_id = 1:5, text = letters[1:5])
filtered_data <- f5(data, 2)
```

---

f6 *Extract Token Information from Data Frame*

Description

This function extracts token, lemma, and part-of-speech (POS) tag information from a given data frame and compiles them into a new data frame.

Usage

```
f6(UPIP)
```

Arguments

UPIP	A data frame containing columns 'token', 'lemma', and 'upos' for tokens, their lemmatized forms, and POS tags respectively.

Value

A data frame with columns 'TOKEN', 'LEMMA', and 'TYPE', each representing token, its lemma, and POS tag.

Examples

```
data <- data.frame(token = c("running", "jumps"),
                   lemma = c("run", "jump"),
                   upos = c("VERB", "VERB"))
token_info <- f6(data)
```

---

f7 *Extract and Count Specific Parts of Speech*

Description

This function extracts tokens of a specified part of speech (POS) from the given data frame and counts their frequency.

Usage

```
f7(UPIP, type)
```

Arguments

UPIP	A data frame with columns 'upos' (POS tags) and 'lemma' (lemmatized tokens).
type	A string representing the POS to filter (e.g., 'NOUN', 'VERB').

Value

A data frame with frequencies of each lemma for the specified POS.

Examples

```
data <- data.frame(upos = c('NOUN', 'VERB'), lemma = c('house', 'run'))
noun_freq <- f7(data, 'NOUN')
```

---

fun.all.sums *Apply Suffix Modifications to Persian Words*

Description

This function iteratively applies a series of suffix modifications to a vector of Persian words.

Usage

```
fun.all.sums(v, TYPE = TYPE.org)
```

Arguments

v	A character vector of Persian words.
TYPE	A vector of suffix types for modification.

Value

A modified character vector.

Examples

```
## Not run:
  words <- c("Persian text here")
  modified_words <- fun.all.sums(words, TYPE)

## End(Not run)
```

fun.one.sums *General Persian Suffix Modification*

Description

This function modifies Persian words based on a specified suffix type.

Usage

```
fun.one.sums(v, type)
```

Arguments

v	A character vector of Persian words.
type	A character string representing the suffix type.

Value

A modified character vector.

Examples

```
## Not run:
  words <- c("Persian text here")
  modified_words <- fun.one.sums(words, "Persian text here")

## End(Not run)
```

FUNbigrams *Extract Bigram Information and Count Frequency*

Description

This function processes a data frame containing bigrams and their frequency, and creates a new data frame with separated words and their frequencies.

Usage

```
FUNbigrams(tf.bigrams)
```

Arguments

tf.bigrams	A data frame with bigram terms and their frequency.

Value

A data frame with columns for each word in the bigram and their frequency.

Examples

```
tf_bigrams <- data.frame(term = c("hello_world", "shiny_app"),
                         term_freq = c(3, 2))
bigram_info <- FUNbigrams(tf_bigrams)
```

---

fungan *Persian Suffix Modification for 'Persian text here' Suffix*

Description

This function modifies Persian words ending with 'Persian text here' suffix.

Usage

```
fungan(v)
```

Arguments

v	A character vector of Persian words.

Value

A modified character vector.

Examples

```
## Not run:
  words <- c("Persian text here")
  modified_words <- fungan(words)

## End(Not run)
```

---

fungi *Persian Suffix Modification*

Description

This function modifies Persian words ending with 'Persian text here' suffix.

Usage

```
fungi(v)
```

Arguments

v	A character vector of Persian words.

Value

A modified character vector.

Examples

```
## Not run:
  words <- c("Persian text here")
  modified_words <- fungi(words)

## End(Not run)
```

`funmi` *Modify Persian Words Starting with 'Persian text here'*

Description

This function modifies Persian words starting with the prefix 'Persian text here'.

Usage

```
funmi(v)
```

Arguments

`v`	A character vector of Persian words.

Value

A modified character vector.

Examples

```
 ## Not run:
  words <- c("Persian text here")
  modified_words <- funmi(words)

## End(Not run)
```

LEMMA *Persian Lemmatization*

Description

This function performs lemmatization on a vector of Persian words.

Usage

```
LEMMA(Y, TYPE = TYPE.org)
```

Arguments

Y	A character vector of Persian words.
TYPE	A vector of suffix types for modification.

Value

A vector of lemmatized Persian words.

Examples

```
## Not run:
  words <- c("Persian text here")
  lemmatized_words <- LEMMA(words, TYPE)

## End(Not run)
```

network.cor *Create and Plot a Correlation Network*

Description

This function creates a correlation network based on specified terms and a threshold, and optionally plots it.

Usage

```
network.cor(dt, Terms, threshold = 0.4, pl = TRUE)
```

Arguments

dt	A document-term matrix.
Terms	A vector of terms to check for correlation.
threshold	A numeric threshold for correlation.
pl	A logical value to plot the network or not.

Value

If ‘pl‘ is TRUE, a plot of the network; otherwise, a data frame of correlations.

`PMI` *Calculate Pointwise Mutual Information (PMI)*

Description

This function calculates the PMI for collocations in a given text data.

Usage

```
PMI(x)
```

Arguments

`x`	A data frame with columns ’token’ and ’doc_id’.

Value

A data frame of keywords with their PMI scores.

Examples

```
data <- data.frame(token = c("word1", "word2"), doc_id = c(1, 1))
pmi_scores <- PMI(data)
```

`ScaleWeight` *Scale a Numeric Vector*

Description

This function scales a numeric vector by a specified lambda value.

Usage

```
ScaleWeight(x, lambda)
```

Arguments

`x`	A numeric vector.
`lambda`	A numeric scaling factor.

Value

A scaled numeric vector.

Examples

```
scaled_vector <- ScaleWeight(1:10, 2)
```

---

server *Server Logic for MadanText Shiny Application*

Description

This function contains the server-side logic for the MadanText application. It handles user inputs, processes data, and creates outputs to be displayed in the UI.

Usage

```
server(input, output)
```

Arguments

input	List of Shiny inputs.
output	List of Shiny outputs.

---

set.graph *Set Graph Attributes*

Description

This function sets various attributes for a given graph object, including vertex degree and edge width.

Usage

```
set.graph(network)
```

Arguments

network	A graph object.

Value

The graph object with updated attributes.

ui *User Interface for MadanText*

Description

This function creates a user interface for the MadanText Shiny application. It includes various input and output widgets for file uploads, text input, and visualization.

Usage

```
ui
```

Format

An object of class `shiny.tag.list` (inherits from `list`) of length 4.

Value

A Shiny UI object.

Index