Package 'MadanText'

December 6, 2023

Type Package

Title Persian Text Mining Tool for Frequency Analysis, Statistical Analysis, and Word Clouds

Version 0.1.0

Description This is an open-source software designed specifically for text mining in the Persian language. It allows users to examine word frequencies, download data for analysis, and generate word clouds. This tool is particularly useful for researchers and analysts working with Persian language data.
This package mainly makes use of the 'PersianStemmer' (Safshekan, R., et al. (2019). <https://CRAN.R-project.org/package=PersianStemmer>),
'udpipe' (Wijffels, J., et al. (2023). <https://CRAN.R-project.org/package=udpipe>),
and 'shiny' (Chang, W., et al. (2023). <https://CRAN.R-project.org/package=shiny>) packages.

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.2

Depends R (>= 4.0.0)

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

NeedsCompilation no

Author Kido Ishikawa [aut, cre], Hasan Khosravi [aut]

Maintainer Kido Ishikawa <kido.ishikawa6@gmail.com>

Repository CRAN

Date/Publication 2023-12-06 10:10:09 UTC

R topics documented:

`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

Returns a data frame with rows and columns corresponding to the original object's structure. If 'x' is a matrix, each column in the matrix becomes a column in the data frame. If 'x' is a list where all elements are of the same length, each element of the list becomes a column in the data frame. Attributes such as rownames, colnames, and dimnames (if any) are preserved in the conversion.

Examples

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

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

Returns a character vector where each element is the normalized and stemmed version of the corresponding element in the input vector. Specifically, it performs character replacement and stemming on each element of the input, thereby returning a vector of the same length but with processed text. If an element cannot be processed, it will be returned as NA in the output vector.

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

Returns a subset of the input data frame (‘UPIP‘) containing only the rows where the ’doc_id’ column matches the specified document ID ‘I‘. If ‘I‘ is 0, the function returns the entire data frame unmodified. The output is a data frame with the same structure as the input but potentially fewer rows, depending on the presence and frequency of the specified ID.

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

Returns a new data frame with three columns: ’TOKEN’, ’LEMMA’, and ’TYPE’. ’TOKEN’ contains the original tokens from the ’token’ column of the input data frame. ’LEMMA’ contains the lemmatized forms of these tokens, as provided in the ’lemma’ column. ’TYPE’ contains POS tags corresponding to each token, as provided in the ’upos’ column. The returned data frame has the same number of rows as the input data frame, with each row representing the token, its lemma, and its POS tag from the corresponding row of the input.

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

Returns a data frame where each row corresponds to a unique lemma of the specified POS type. The data frame has two columns: 'key', which contains the lemma, and 'freq', which contains the frequency count of that lemma in the data. The rows are ordered in decreasing frequency of occurrence. This format is useful for quickly identifying the most common terms of a particular POS in the data.

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)
```

Arguments

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

Value

Returns a character vector where each element corresponds to a word from the input vector ‘v‘ with all specified suffix modifications applied. This results in a transformed vector where each word has been modified according to the series of suffix types provided in ‘TYPE‘. The length of the returned vector matches the length of the input 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

Returns a character vector where each element corresponds to a word from the input vector ‘v‘ with the specified suffix type modified. This results in a transformed vector where each word has been modified to remove or alter the specified suffix. The length of the returned vector matches the length of the input vector, and each word is modified independently based on the specified suffix type.

Examples

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

## End(Not run)
```

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

Returns a character vector where each element corresponds to a word from the input vector 'v' with the 'Persian text here' suffix modified. This results in a transformed vector where each word ending with the specified suffix is altered. The length of the returned vector matches the length of the input vector, and each word is modified independently based on the presence of the specified suffix.

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

Returns a character vector where each element corresponds to a word from the input vector ‘v‘ with the specified suffix modified. This results in a transformed vector where each word ending with the specified suffix is altered. The length of the returned vector matches the length of the input vector, and each word is modified independently based on the presence of the specified suffix.

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

Returns a character vector where each element corresponds to a word from the input vector ‘v‘ with the specified suffix modified. This results in a transformed vector where each word ending with the specified suffix is altered. The length of the returned vector matches the length of the input vector, and each word is modified independently based on the presence of the specified suffix.

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)
```

Arguments

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

Value

Returns a character vector where each element is the lemmatized form of the corresponding element in the input vector 'Y'. Lemmatization involves removing inflectional endings and returning the word to its base or dictionary form. The length of the returned vector matches the length of the input vector, and each word is lemmatized independently based on the specified suffix types in 'TYPE'.

Examples

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

## End(Not run)
```

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

Returns a data frame where each row represents a unique keyword (collocation) in the input data. The data frame contains columns such as 'keyword', representing the keyword, and 'pmi', representing the PMI score of that keyword. Higher PMI scores indicate a stronger association between the components of the collocation within the corpus.

Examples

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

---

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.

Value

This function sets up the reactive environment and output elements in the Shiny application. It does not return any value but modifies the Shiny app's UI based on user inputs and reactive expressions. It returns a Shiny Server object.

TYPE *Persian Suffixes*

Description

A vector of common Persian suffixes used for text processing.

Usage

```
TYPE
```

Format

An object of class `character` of length 39. Each element in this vector is a string representing a common Persian suffix.

`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