Package 'oRus'

October 14, 2022

Title Operational Research User Stories

Version 1.0.0

Maintainer Melina Vidoni <melina.vidoni@rmit.edu.au>

Description A first implementation of automated parsing of user stories, when used to defined functional requirements for operational research mathematical models. It allows reading user stories, splitting them on the who-what-why template, and classifying them according to the parts of the mathematical model that they represent. Also provides semantic grouping of stories, for project management purposes.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 3.6.0)

Imports dplyr, stringr, tm, tibble, tidytext, topicmodels, rmarkdown, xlsx, knitr

RoxygenNote 7.1.0

URL https://github.com/melvidoni/oRus

BugReports https://github.com/melvidoni/oRus/issues

VignetteBuilder knitr

Suggests reshape2, qpdf

NeedsCompilation no

Author Melina Vidoni [aut, cre] (<https://orcid.org/0000-0002-4099-1430>), Laura Cunico [aut]

Repository CRAN

Date/Publication 2020-07-07 09:20:08 UTC

R topics documented:

`analyseStories` *Analysing Stories*

Description

Main function that fully automates the process of analysing a dataset of user stories. It can also write results as an Excel sheet in a given directory, and generate an advanced report with highlights of missing features.

Usage

```
analyseStories(
  storiesFile,
  groupsNumber,
  topGroups = 1,
  sheetFilePath = NULL,
  reportFilePath = NULL,
  outputType = "pdf_document",
  ignoreWordsList = NULL
)
```

Arguments

`storiesFile`	The path an name to a text file containing one user story per line. They need to be written in English.
`groupsNumber`	The number of groups you want to generate between user stories.
`topGroups`	How many groups per stories you want to keep. The default is 1.
`sheetFilePath`	The path and filename of the Excel sheet that will be stored; must include the '*.xlsx' extension. If no value is passed, the file will not be written.
`reportFilePath`	The path where the extensive report will be stored. It must include the correct extension (according to the type selected in the following argument). If no value is passed, the report won't be generated.
`outputType`	The type of document to be generated (from an RMarkdown). By default it is a PDF file.
`ignoreWordsList`	The list of words that you want to avoid using during the grouping of user stories. If nothing is passed, a default list will be used.

Value

A list of two datasets: the first one contains the stories split up, classified in types, analysed and grouped. Second dataframe contains top words per group and the belonging value of the word.

See Also

Other Simplified Process: `createIgnoreWords()`, `reportStories()`

Examples

```
# Libraries for the example
library(reshape2)

# Transform the stories
fileUrl <- example_stories()
stories <- analyseStories(fileUrl, 7)

# Print some information
head(dplyr::as_tibble(stories[[2]]))

head(stories[[1]])
```

`createIgnoreWords` *List Ignored Words*

Description

List Ignored Words

Usage

```
createIgnoreWords(wordsList = c(), addToExisting = TRUE)
```

Arguments

`wordsList`	List of words to be ignored when grouping user stories semantically by similarities of word.
`addToExisting`	If this param is TRUE, passed words will be added to a predefined set of words. Otherwise, only those in the previous argument will be ignored.

Value

Returns an array of words that will be ignored when processing the semantic groups.

See Also

Other Simplified Process: `analyseStories()`, `reportStories()`

Examples

```
# Generating default words only
createIgnoreWords()

# Adding words
createIgnoreWords(c("given", "said"))

# Replacing words
createIgnoreWords(c("given", "said"), FALSE)
```

`example_stories` *Path to Example Data*

Description

Path to Example Data

Usage

```
example_stories()
```

Value

Local path to example text file, containing user stories for Operational Research mathmatical models.

Examples

```
example_stories()
```

`oRus` `oRus` *package*

Description

Analysis of User Stories for Operational Research

Details

See the README on GitHub

readStories *Initial User Story Parse*

Description

This function will help you parse a set of stories into a dataframe, where one row is each user story; The user story gets splitted into who, what and why sections, according to the use of keywords. The file must be a text file written in English, with one user story per row.

Usage

```
readStories(url)
```

Arguments

url	The URL of the text file to be parsed. Every user story must be in a single line, and written in English. Punctuation is irrelevant as it is processed out. For this to work, user stories should follow the who, what, why template, with keywords: "as a/an ", " I want to ", " so that ", respectively.

Value

A dataframe of three colums, representing sections who, what, why of the user stories. There is one row per user story, and they may not have the "why" part if it wasn't added. Using incorrect keywords means incorrect parsing, so be careful.

Examples

```
# Analyse without reports
dataPath <- example_stories()
stories <- readStories(dataPath)

# Print some information
head(dplyr::as_tibble(stories))
```

reportStories *Reporting Stories*

Description

This function allows you to write the reports for the user stories if you didn't write them before on the analysis function. The key input is the output of 'oRus::analyseStories()'.#'

Usage

```
reportStories(
  stories,
  sheetFilePath = NULL,
  reportFilePath = NULL,
  outputType = "html_document"
)
```

Arguments

stories	The stories produced by the analysis function.
sheetFilePath	The path and filename of the Excel sheet that will be stored; must include the '*.xlsx' extension. If no value is passed, the file will not be written.
reportFilePath	The path where the extensive report will be stored. It must include the correct extension (according to the type selected in the following argument). If no value is passed, the report won't be generated.
outputType	The type of document to be generated (from an RMarkdown). By default it is a PDF file. Options are: 'html_output' or 'pdf_output'.

See Also

Other Simplified Process: `analyseStories()`, `createIgnoreWords()`

Index