Package ‘scraEP’

June 23, 2021

Type Package

Title Scrape the Web with Extra Power

Version 1.2

Date 2021-06-21

Author Julien Boelaert <jubo.stats@gmail.com>

Maintainer Julien Boelaert <jubo.stats@gmail.com>

Description Tools for scraping information from webpages and other XML contents, using XPath or CSS selectors.

Depends R (>= 4.0.0)

License GPL (>= 3)

Imports XML, xml2, rvest, data.table, parallel

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2021-06-23 07:00:02 UTC

R topics documented:

`scraEP-package` *Scrape the Web with Extra Power.*

Description

A set of tools for scraping information from webpages and other XML contents.

Details

Package:	scraEP
Type:	Package
Version:	1.2
Date:	2021-06-21
License:	GPL (>=3)

Function `xscrape` is a general tool to extract information from html pages into data frames using XPath or CSS queries.

Function `unaccent` removes accents from character strings.

Function `strcomp` compares the elements of two character vectors.

Author(s)

Julien Boelaert <`jubo.stats@gmail.com`>

`strcomp` *Compare the contents of two vectors of character strings.*

Description

This function compares the contents of two vectors of character strings: how many elements are present in both vectors, how many are only present in each one, etc., and lists the elements that are only present in one of the vectors.

Usage

```
strcomp(text1, text2)
```

Arguments

`text1, text2`	two character vectors to be compared

Value

A list containing the following elements:

matchTable: a cross table of all the unique elements of the concatenation of text1 and text2, indicating whether they belong to text1 and/or to text2.

matchOneInTwo: a table of the elements of text1 according to how many times they are present in text2.

matchTwoInOne: a table of the elements of text2 according to how many times they are present in text1.

tabOneInTwo: a table of the elements of text1 according to whether they are present in text2.

tabTwoInOne: a table of the elements of text2 according to whether they are present in text1.

oneNotInTwo: a vector containing the elements of text1 not present in text2.

twoNotInOne: a vector containing the elements of text2 not present in text1.

Author(s)

Julien Boelaert <jubo.stats@gmail.com>

Examples

```
str1 <- c("Alice", "Alice", "Bob", "Carol")
str2 <- c("Bob", "Denise", "Emerson", "Foteini")
strcomp(str1, str2)
```

---

unaccent *Remove all accents from character strings.*

Description

This function removes all diacritic accents from character strings (eg. transforms "àüî" into "aui").

Usage

```
unaccent(text)
```

Arguments

text	a character vector

Value

A character vector, containing the unaccentuated version of the given text.

Author(s)

Julien Boelaert <jubo.stats@gmail.com>

Examples

```
unaccent("âàéèïì")
```

wiki *Wikipedia page for R.*

Description

Toy example to extract data using `xscrape`.

Usage

```
data(wiki)
```

Format

Object `wiki` is a character vector.

Details

Object `wiki` is a raw webpage, containing the source of the ‘R (programming language)‘ article on English wikipedia (<https://en.wikipedia.org/wiki/R_(programming_language)>, retrieved on 15/11/2017).

Author(s)

Julien Boelaert <`jubo.stats@gmail.com`>

xscrape *Extract information from webpages to a data.frame, using XPath or CSS queries.*

Description

This function transforms an html/xml page (or list of pages) into a data.frame, extracting nodes specified by their XPath.

Usage

```
xscrape(pages,
        col.xpath = ".", row.xpath = "/html",
        col.css = NULL, row.css = NULL,
        collapse = " | ", encoding = NULL,
        page.name = TRUE, nice.text = TRUE,
        parallel = 0,
        engine = c("auto", "XML", "xml2"))
```

Arguments

`pages` an object of class `XMLInternalDocument` or `xml_document` (as returned by functions `XML::htmlParse` or `xml2::read_html` or `rvest::read_html`), or list of such objects. Alternatively, a character vector containing the URLs or local paths of webpages to be parsed. These are the webpages that information is to be extracted from. If the provided list or vector is named, its names will be used to indicate data provenance when `page.name` is TRUE.

`col.xpath` a character vector of XPath queries used for creating the result columns. If the vector is named, these names are given to the columns. The default "." takes the text from the whole of each page or intermediary node (specified by `row.xpath` or `row.css`).

`row.xpath` a character string, containing an XPath query for creating the result rows. The result of this query (on each page) becomes a row in the resulting data.frame. If not specified (default), the intermediary nodes are whole html pages, so that each page becomes a row in the result.

`col.css` same as `col.xpath`, but with CSS selectors instead of XPath queries. If `col.xpath` was also given, the XPath columns will be placed before the CSS columns in the result.

`row.css` same as `row.xpath`, but with a CSS selector instead of an XPath query. If given, this will be used instead of `row.xpath`.

`collapse` a character string, containing the separator that will be used in case a `col.xpath` query yields multiple results within a given intermediary node. The default is " | ".

`encoding` a character string (eg. "UTF-8" or "ISO-8859-1"), containing the encoding parameter that will be used by `htmlParse` or `read_html` if `pages` is a vector of URLs or local file names.

`page.name` a logical. If TRUE, the result will contain a column indicating the name of the page each row was extracted from. If `pages` has no names, they will be numbered from 1 to `length(pages)`

`nice.text` a logical. If TRUE (only possible with engine xml2), the rvest::html_text2 function is used to extract text into the result, often making the text much cleaner. If FALSE, the function runs faster, but the text might be less clean.

`parallel` a numeric, indicating the number of cores to use for parallel computation. The default 0 takes all available cores. The parallelization is done on the pages if their number is greater than the number of provided cores, otherwise it is done on the intermediary nodes. Note that parallelization relies on parallel::mclapply, and is thus not supported on Windows systems.

`engine` a character string, indicating the engine to use for data extraction: either "XML", "xml2", or "auto" (default). The default will adapt the engine to the type of `pages`, or will use "xml2" if `pages` are URLs or file names. Note: CSS selectors and `nice.text` are only available for "xml2", but XPath queries and "XML" engine tend to be much faster.

Details

If a col.xpath or col.css query designs a full node, only its text is extracted. If it designs an attribute (eg. ends with '/@href' for weblinks), only the attribute's value is extracted.

If a col.xpath or col.css query matches no elements in a page, returned value is NA. If it matches multiple elements, they are concatenated into a single character string, separated by collapse.

Value

A data.frame, where each row corresponds to an intermediary node (either a full page or an XML node within a page, specified by row.xpath or row.css), and each column corresponds to the text of a col.xpath or col.css query.

Author(s)

Julien Boelaert <jubo.stats@gmail.com>

Examples

```
## Extract all external links and their titles from a wikipedia page
data(wiki)
wiki.parse <- XML::htmlParse(wiki)
links <- xscrape(wiki.parse,
                 row.xpath= "//a[starts-with(./@href, 'http')]",
                 col.xpath= c(title= ".", link= "./@href"),
                 parallel = 1)

## Not run:
## Convert results from a search for 'R' on duckduckgo.com
## First download the search page
duck <- XML::htmlParse("http://duckduckgo.com/html/?q=R")
## Then run xscrape on the dowloaded and parsed page
results <- xscrape(duck,
                   row.xpath= "//div[contains(@class, 'result__body')]",
                   col.xpath= c(title= "./h2",
                                snippet= ".//*[@class='result__snippet']",
                                url= ".//a[@class='result__url']/@href"))

## End(Not run)

## Not run:
## Convert results from a search for 'R' and 'Julia' on duckduckgo.com
## Directly provide the URLs to xscrape
results <- xscrape(c("http://duckduckgo.com/html/?q=R",
                     "http://duckduckgo.com/html/?q=julia"),
                   row.xpath= "//div[contains(@class, 'result__body')]",
                   col.xpath= c(title= "./h2",
                                snippet= ".//*[@class='result__snippet']",
                                url= ".//a[@class='result__url']/@href"))

## End(Not run)
```

Index