Package ‘tm.plugin.webmining’

May 11, 2015

Version 1.3
Date 2015-05-07
Title Retrieve Structured, Textual Data from Various Web Sources
Depends R (>= 3.1.0)
Imports NLP (>= 0.1-2), tm (>= 0.6), boilerpipeR, RCurl, XML, RJSONIO
Suggests testthat
Description Facilitate text retrieval from feed formats like XML (RSS, ATOM) and JSON. Also direct retrieval from HTML is supported. As most (news) feeds only incorporate small fractions of the original text tm.plugin.webmining even retrieves and extracts the text of the original text source.
License GPL-3
URL https://github.com/mannau/tm.plugin.webmining
BugReports https://github.com/mannau/tm.plugin.webmining/issues
NeedsCompilation no
Author Mario Annau [aut, cre]
Maintainer Mario Annau <mario.annau@gmail.com>
Repository CRAN
Date/Publication 2015-05-11 00:20:43

R topics documented:

 tm.plugin.webmining-package ........................................ 2
corpus.update .......................................................... 3
ecloseHTML ............................................................... 3
extract ................................................................. 4
extractContentDOM ....................................................... 4
extractHTMLStrip ......................................................... 5
feedquery ............................................................... 6
getEmpty ............................................................... 6
tm.plugin.webmining-package

Retrieves structured, textual data from various web sources

**Description**

tm.plugin.webmining facilitates the retrieval of textual data through various web feed formats like XML and JSON. Also, direct retrieval from HTML is supported. As most (news) feeds only incorporate small fractions of the original text, tm.plugin.webmining goes a step further and even retrieves and extracts the text of the original text source. Generally, the retrieval procedure can be described as a two-step process:

**Meta Retrieval**  In a first step, all relevant meta feeds are retrieved. From these feeds all relevant meta data items are extracted.

**Content Retrieval**  In a second step, the relevant source content is retrieved. Using the `boilerpipeR` package even the main content of HTML pages can be extracted.

**Author(s)**

Mario Annau <mario.annau@gmail>

**See Also**


---

getLinkContent ....................................................  7
GoogleFinanceSource .................................................  8
GoogleNewsSource ...................................................  9
NYTimesSource ....................................................... 10
nytimes_appid ....................................................... 11
parse ................................................................. 11
readWeb ............................................................... 12
removeNonASCII ...................................................... 12
ReutersNewsSource ................................................... 13
source.update ......................................................... 14
trimWhiteSpaces ...................................................... 14
WebCorpus ............................................................ 15
WebSource ............................................................ 15
YahooFinanceSource .................................................. 16
YahooInplaySource ................................................... 17
yahoonews ............................................................. 18
YahooNewsSource ..................................................... 18

Index 20
Examples

## Not run:

```r

googlefinance <- WebCorpus(GoogleFinanceSource("NASDAQ:MSFT"))
googlenews <- WebCorpus(GoogleNewsSource("Microsoft"))
nytimes <- WebCorpus(NYTimesSource("Microsoft", appid = nytimes_appid))
reutersnews <- WebCorpus(ReutersNewsSource("businessNews"))
yahoofinance <- WebCorpus(YahooFinanceSource("MSFT"))
yahooinplay <- WebCorpus(YahooInplaySource())
yahoonews <- WebCorpus(YahooNewsSource("Microsoft"))
```

## End(Not run)

corpus.update

Update/Extend `WebCorpus` with new feed items.

Description

The `corpus.update` method ensures, that the original `WebCorpus` feed sources are downloaded and checked against already included `TextDocument`s. Based on the ID included in the `TextDocument`'s meta data, only new feed elements are downloaded and added to the `WebCorpus`. All relevant information regarding the original source feeds are stored in the `WebCorpus`' meta data (`meta`).

Usage

```r
corpus.update(x, ...)
```

Arguments

- `x` object of type `WebCorpus`
- `...`
  - `fieldname` name of `Corpus` field name to be used as ID, defaults to "ID"
  - `retryempty` specifies if empty corpus elements should be downloaded again, defaults to TRUE
  - ... additional parameters to `Corpus` function

encloseHTML

Enclose Text Content in HTML tags

Description

Simple helper function which encloses text content of character (or `TextDocument`) in HTML-tags. That way, HTML content can be easier parsed by `htmlTreeParse`.

Usage

```rencloseHTML(x)
```
**extractContentDOM**

**Arguments**

- x: object of PlainTextDocument class

**Description**

Use implemented extraction functions (through boilerpipeR) to extract main content from TextDocuments.

**Usage**

```r
extract(x, extractor, ...)
```

**extractContentDOM**

**Extract Main HTML Content from DOM**

**Description**

Function extracts main HTML Content using its Document Object Model. Idea comes basically from the fact, that main content of an HTML Document is in a subnode of the HTML DOM Tree with a high text-to-tag ratio. Internally, this function also calls assignValues, calcDensity, getMainText and removeTags.

**Usage**

```r
extractContentDOM(url, threshold, asText = TRUE, ...)
```

**Arguments**

- url: character, url or filename
- threshold: threshold for extraction, defaults to 0.5
- asText: boolean, specifies if url should be interpreted as character
- ...: Additional Parameters to htmlTreeParse

**Author(s)**

Mario Annau
extractHTMLStrip

References

http://www.elias.cn/En/ExtMainText http://ai-depot.com/articles/the-easy-way-to-extract-useful-text
Gupta et al., DOM-based Content Extraction of HTML Documents http://www2003.org/cdrom/
papers/refereed/p583/p583-gupta.html

See Also

xmlNode

extractHTMLStrip Simply strip HTML Tags from Document

Description

extractHTMLStrip parses an url, character or filename, reads the DOM tree, removes all HTML
tags in the tree and outputs the source text without markup.

Usage

extractHTMLStrip(url, asText = TRUE, encoding, ...)

Arguments

url character, url or filename
asText specifies if url parameter is a character, defaults to TRUE
encoding specifies local encoding to be used, depending on platform
... Additional parameters for htmlTreeParse

Note

Input text should be enclosed in <html>'TEXT'</html> tags to ensure correct DOM parsing (issue
especially under .Platform$os.type = 'windows')

Author(s)

Mario Annau

See Also

xmlNode
htmlTreeParse encloseHTML
**feedquery**

*Buildup string for feedquery.*

**Description**

Function has partly been taken from `getForm` function. Generally, a feed query is a string built up as follows:

```
<url>?<param1=value1>&<param2=value2>&...&<paramN=valueN>
```

By specifying a feed url and parameter–value pairs (as list) we can easily generate a feed query in R.

**Usage**

`feedquery(url, params)`

**Arguments**

- `url` character specifying feed url
- `params` list which contains feed parameters, e.g. `list(param1="value1", param2="value2")`

**Author(s)**

Mario Annau

**See Also**

`xmlNode getForm`

**Examples**

```r
## Not run:
feedquery(url = "http://dummy.com",
  params = list(param1 = "value1", param2 = "value2"))

## End(Not run)
```

---

**getEmpty**

*Retrieve Empty Corpus Elements through $postFUN.*

**Description**

Retrieve content of all empty (textlength equals zero) corpus elements. If corpus element is empty, $postFUN is called (specified in `meta`)

**Usage**

`getEmpty(x, ...)`
getLinkContent

Arguments

- `x` object of type `WebCorpus`
- ... additional parameters to PostFUN

See Also

WebCorpus

getLinkContent

Get main content for corpus items, specified by links.

Description

getLinkContent downloads and extracts content from weblinks for `Corpus` objects. Typically it is integrated and called as a post-processing function (field: `$postFUN`) for most `WebSource` objects.

getLinkContent implements content download in chunks which has been proven to be a stabler approach for large content requests.

Usage

getLinkContent(corpus, links = sapply(corpus, meta, "origin"),
    timeout = 30, chunksize = 20, verbose = getOption("verbose"),
    curl0pts = curl0ptions(verb0se = FALSE, followlocation = TRUE, maxconnects = 5,
        maxredirs = 20, timeout = timeout.request, connecttimeout =
        timeout.request, ssl.verifyhost = FALSE, ssl.verifypeer = FALSE, useragent =
        "R", cookiejar = tempfile()), retry.empty = 3, sleep.time = 3,
    extractor = ArticleExtractor, .encoding = integer(), ...)

Arguments

- `corpus` object of class `Corpus` for which link content should be downloaded
- `links` character vector specifying links to be used for download, defaults to `sapply(corpus, meta, "Origin")`
- `timeout.request` timeout (in seconds) to be used for connections/requests, defaults to 30
- `chunksize` Size of download chunks to be used for parallel retrieval, defaults to 20
- `verbose` Specifies if retrieval info should be printed, defaults to `getOption("verbose")`
- `curl0pts` curl options to be passed to `getURL`
- `retry.empty` Specifies number of times empty content sites should be retried, defaults to 3
- `sleep.time` Sleep time to be used between chunked download, defaults to 3 (seconds)
- `extractor` Extractor to be used for content extraction, defaults to `extractContentDOM`
- `.encoding` encoding to be used for `getURL`, defaults to `integer()` (=autodetect)
- ... additional parameters to `getURL`
GoogleFinanceSource  

*Value*

corpus including downloaded link content

*See Also*

`WebSource getURL Extractor`

---

**GoogleFinanceSource**  
*Get feed Meta Data from Google Finance.*

**Description**

Google Finance provides business and enterprise headlines for many companies. Coverage is particularly strong for US-Markets. However, only up to 20 feed items can be retrieved.

**Usage**

```r
GoogleFinanceSource(query, params = list(hl = "en", q = query, ie = "utf-8", start = 0, num = 20, output = "rss"), ...)
```

**Arguments**

- **query**: ticker symbols of companies to be searched for, see [http://www.google.com/finance](http://www.google.com/finance). Please note that Google ticker symbols need to be prefixed with the exchange name, e.g. NASDAQ:MSFT
- **params**: additional query parameters
- **...**: additional parameters to `WebSource`

**Value**

WebXMLSource

**Author(s)**

Mario Annau

**See Also**

`WebSource`

**Examples**

```r
## Not run:  
corpus <- Corpus(GoogleFinanceSource("NASDAQ:MSFT"))

## End(Not run)
```
GoogleNewsSource

Get feed data from Google News Search http://news.google.com/

Description

Google News Search is one of the most popular news aggregators on the web. News can be retrieved for any customized user query. Up to 100 can be retrieved per request.

Usage

GoogleNewsSource(query, params = list(hl = "en", q = query, ie = "utf-8", num = 100, output = "rss"), ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query</td>
<td>Google News Search query</td>
</tr>
<tr>
<td>params,</td>
<td>additional query parameters</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters to WebSource</td>
</tr>
</tbody>
</table>

Value

WebXMLSource

Author(s)

Mario Annau

See Also

WebSource

Examples

```
## Not run:
corpus <- Corpus(GoogleNewsSource("Microsoft"))
## End(Not run)
```
NYTimesSource Get feed data from NYTimes Article Search (http://developer.nytimes.com/docs/read/article_search_api_v2).

Description

Excerpt from the website: "With the NYTimes Article Search API, you can search New York Times articles from 1981 to today, retrieving headlines, abstracts, lead paragraphs, links to associated multimedia and other article metadata. Along with standard keyword searching, the API also offers faceted searching. The available facets include Times-specific fields such as sections, taxonomic classifiers and controlled vocabulary terms (names of people, organizations and geographic locations)." Feed retrieval is limited to 1000 items (or 100 pages).

Usage

NYTimesSource(query, n = 100, appid, count = 10, sleep = 1,
params = list(format = "json", q = query, page = 1:ceiling(n/count),
"api-key" = appid), curlopts = curlOptions(followlocation = TRUE,
maxconnects = 10, maxredirects = 10, timeout = 30, connecttimeout = 30), ...)

Arguments

query character specifying query to be used to search NYTimes articles
n number of items, defaults to 100
appid Developer App id to be used, obtained from http://developer.nytimes.com/
count number of results per page, defaults to 10
sleep integer; Seconds to sleep between feed retrieval.
params additional query parameters, specified as list, see http://developer.nytimes.com/docs/read/article_search_api
curlopts CURLOptions; RCurl options used for feed retrieval.
... additional parameters to WebSource

Author(s)

Mario Annau

See Also

WebSource, readNYTimes
Examples

```r
## Not run:
nytimes_appid needs to be specified
corpus <- WebCorpus(NYTimesSource("Microsoft", appid = nytimes_appid))

## End(Not run)
```

nytimes_appid  

*AppID for the NYtimes-API.*

Description


Author(s)

Mario Annau

parse  

*Wrapper/Convenience function to ensure right encoding for different Platforms*

Description

Depending on specified type one of the following parser functions is called:

- **XML** xmlInternalTreeParse
- **HTML** htmlTreeParse
- **JSON** fromJSON

Usage

```r
parse(..., asText = TRUE, type = c("XML", "HTML", "JSON"))
```

Arguments

- `...` arguments to be passed to specified parser function
- `asText` defines if input should be treated as text/character, default to TRUE
- `type` either "XML", "HTML" or "JSON". Defaults to "XML"
**Description**

readWeb is a FunctionGenerator which specifies content retrieval from a WebSource content elements. Currently, it is defined for XML, HTML and JSON feeds through readWebXML, readWebHTML and readWebJSON. Also content parsers (xml_content, json_content) need to be defined.

**Usage**

readWeb(spec, doc, parser, contentparser, freeFUN = NULL)

**Arguments**

- `spec`: specification of content reader
- `doc`: document to be parsed
- `parser`: parser function to be used
- `contentparser`: content parser function to be used, see also `tm::xml_content` or `json_content`
- `freeFUN`: function to free memory from parsed object (actually only relevant for XML and HTML trees)

**Value**

FunctionGenerator

---

**Description**

This is a helper function to generate package data without non-ASCII character and omit the warning at R CMD check.

**Usage**

removeNonASCII(x, fields = c("Content", "Heading", "Description"),
from = "UTF-8", to = "ASCII//TRANSLIT")

**Arguments**

- `x`: object of PlainTextDocument class
- `fields`: specifies fields to be converted, defaults to `fields = c("Content", "Heading", "Description")`
- `from`: specifies encoding from which conversion should be done, defaults to "UTF-8"
- `to`: specifies target encoding, defaults to "ASCII//TRANSLIT"
**Description**

Get feed data from Reuters News RSS feed channels. Reuters provides numerous feed channels ([http://www.reuters.com/tools/rss](http://www.reuters.com/tools/rss)) which can be retrieved through RSS feeds. Only up to 25 items can be retrieved—therefore an alternative retrieval through the Google Reader API ([link](link)GoogleReaderSource) could be considered.

**Usage**

```
ReutersNewsSource(query = "businessNews", ...)
```

**Arguments**

- **query**: Reuters News RSS Feed, see [http://www.reuters.com/tools/rss](http://www.reuters.com/tools/rss) for a list of all feeds provided. Note that only string after `http://feeds.reuters.com/reuters/` must be given. Defaults to `businessNews`.
- **...**: additional parameters to `WebSource`

**Value**

WebXMLSource

**Author(s)**

Mario Annau

**See Also**

`WebSource`

**Examples**

```r
## Not run:
corpus <- Corpus(ReutersNewsSource("businessNews"))

## End(Not run)
```
source.update

Update WebXMLSource/WebHTMLSource/WebJSONSource

Description

Typically, update is called from link{corpus.update} and refreshes $Content in Source object.

Usage

source.update(x)

Arguments

x Source object to be updated

trimWhiteSpaces

Trim White Spaces from Text Document.

Description

Transformation function, actually equal to stripWhiteSpace applicable for simple strings using Perl parser

Usage

trimWhiteSpaces(txt)

Arguments

txt character

Author(s)

Mario Annau

See Also

stripWhitespace
WebCorpus

WebCorpus constructor function.

Description

WebCorpus adds further methods and meta data to Corpus and therefore constructs a derived class of Corpus. Most importantly, WebCorpus calls $PostFUN on the generated WebCorpus, which retrieves the main content for most implemented WebSources. Thus it enables an efficient retrieval of new feed items (corpus.update). All additional WebCorpus fields are added to tm$meta like $source, $readerControl and $postFUN.

Usage

WebCorpus(x, readerControl = list(reader = reader(x), language = "en"),
postFUN = x$postFUN, retryEmpty = TRUE, ...)

Arguments

x object of type Source, see also Corpus
readerControl specifies reader to be used for Source, defaults to list(reader = x$DefaultReader, language = "en"
postFUN function to be applied to WebCorpus after web retrieval has been completed, defaults to x$PostFUN
retryEmpty specifies if retrieval for empty content elements should be repeated, defaults to TRUE
... additional parameters for Corpus function (actually Corpus reader)

WebSource

Read Web Content and respective Link Content from feedurls.

Description

WebSource is derived from Source. In addition to calling the base Source constructor function it also retrieves the specified feedurls and pre–parses the content with the parser function. The fields $Content, $Feedurls $Parser and $Curlopts are finally added to the Source object.

Usage

WebSource(feedurls, class = "WebXMLSource", reader, parser,
encoding = "UTF-8", curlopts = curlOptions(followlocation = TRUE,
maxconnects = 20, maxredirs = 10, timeout = 30, connecttimeout = 30),
potFUN = NULL, retrieveFeedURL = TRUE, ...)
YahooFinanceSource

Arguments

- **feedurls**: urls from feeds to be retrieved
- **class**: class label to be assigned to Source object, defaults to "WebXMLSource"
- **reader**: function to be used to read content, see also readWeb
- **parser**: function to be used to split feed content into chunks, returns list of content elements
- **encoding**: specifies default encoding, defaults to 'UTF-8'
- **curlOpts**: a named list or CURLOptions object identifying the curl options for the handle. Type listCurlOptions() for all Curl options available.
- **postFUN**: function saved in WebSource object and called to retrieve full text content from feed urls
- **retrieveFeedURL**: logical; Specify if feedurls should be downloaded first.
- ... additional parameters passed to WebSource object/structure

Value

WebSource

Author(s)

Mario Annau

---

YahooFinanceSource  
*Get feed data from Yahoo! Finance.*

Description

Yahoo! Finance is a popular site which provides financial news and information. It is a large source for historical price data as well as financial news. Using the typical Yahoo! Finance ticker news items can easily be retrieved. However, the maximum number of items is 20.

Usage

YahooFinanceSource(query, params = list(s = query, region = "US", lang = "en-US"), ...)

Arguments

- **query**: ticker symbols of companies to be searched for, see [http://finance.yahoo.com/lookup](http://finance.yahoo.com/lookup).
- **params**, ... additional query parameters, see [http://developer.yahoo.com/rss/](http://developer.yahoo.com/rss/)
- ... additional parameters to WebSource
YahooInplaySource

Value
  WebXMLSource

Author(s)
  Mario Annau

See Also
  WebSource

Examples
  ## Not run:
corpus <- Corpus(YahooFinanceSource("MSFT"))

## End(Not run)

---

YahooInplaySource  Get News from Yahoo Inplay.

Description
  Yahoo Inplay lists a range of company news provided by Briefing.com. Since Yahoo Inplay does not
  provide a structured XML news feed, content is parsed directly from the HTML page. Therefore,
  no further Source parameters can be specified. The number of feed items per request can vary
  substantially.

Usage
  YahooInplaySource(...)  

Arguments
  ...  additional parameters to WebSource

Value
  WebHTMLSource

Author(s)
  Mario Annau

Examples
  ## Not run:
corpus <- Corpus(YahooInplaySource())

## End(Not run)
YahooNewsSource

YahooNewsSource

WebCorpus retrieved from Yahoo! News for the search term ”Microsoft” through the YahooNewsSource. Length of retrieved corpus is 20.

Description

WebCorpus retrieved from Yahoo! News for the search term ”Microsoft” through the YahooNewsSource. Length of retrieved corpus is 20.

Author(s)

Mario Annau

Examples

# Data set has been generated as follows:
```r
# Not run:
yahooews <- WebCorpus(YahooNewsSource("Microsoft"))
# End(Not run)
```

YahooNewsSource

Get news data from Yahoo! News (https://news.search.yahoo.com/search/).

Description

Currently, only a maximum of 10 items can be retrieved.

Usage

YahooNewsSource(query, params = list(p = query), ...)

Arguments

- **query**: words to be searched in Yahoo News, multiple words must be separated by '+'
- **params**: additional query parameters, see http://developer.yahoo.com/rss/
- **...**: additional parameters to WebSource

Value

WebXMLSource

Author(s)

Mario Annau
See Also

WebSource

Examples

```r
## Not run:
corpus <- Corpus(YahooNewsSource("Microsoft"))

## End(Not run)
```
Index

Topic **data**
- `nytimes_appid`, 11
- `yahoonews`, 18

Topic **package**
- `tm.plugin.webmining-package`, 2

- `assignValues (extractContentDOM)`, 4
- `calcDensity (extractContentDOM)`, 4
- `Corpus`, 3, 7, 15
- `corpus.update`, 3, 15
- `encloseHTML`, 3, 5
- `extract`, 4
- `extractContentDOM`, 4, 4
- `extractHTMLStrip`, 5
- `Extractor`, 8
- `feedquery`, 6
- `fromJson`, 11
- `getEmpty`, 6
- `getForm`, 6
- `getLinkContent`, 7
- `getMainText (extractContentDOM)`, 4
- `getURL`, 7, 8
- `GoogleFinanceSource`, 2, 8
- `GoogleNewsSource`, 2, 9
- `htmlTreeParse`, 3–5, 11
- `json_content (readWeb)`, 12
- `meta`, 3, 6
- `nytimes_appid`, 11
- `NYTimesSource`, 2, 10
- `parse`, 11
- `readGoogle (GoogleFinanceSource)`, 8
- `readNYTimes`, 10

- `readReutersNews (ReutersNewsSource)`, 10
- `readReutersNews (ReutersNewsSource)`, 13
- `readWeb`, 12, 16
- `readWebHTML (readWeb)`, 12
- `readWebJSON (readWeb)`, 12
- `readWebXML (readWeb)`, 12
- `readYahoo (YahooFinanceSource)`, 16
- `readYahooHTML (YahooNewsSource)`, 18
- `readYahooInplay (YahooInplaySource)`, 17
- `removeNonASCII`, 12
- `removeTags (extractContentDOM)`, 4
- `ReutersNewsSource`, 2, 13
- `Source`, 15
- `source.update`, 14
- `stripWhitespace`, 14

- `TextDocument`, 3
- `tm.plugin.webmining`
  - `(tm.plugin.webmining-package)`, 2
- `tm.plugin.webmining-package`, 2
- `trimWhiteSpaces`, 14

- `WebCorpus`, 2, 3, 7, 15
- `webmining`
  - `(tm.plugin.webmining-package)`, 2
- `WebSource`, 7–10, 12, 13, 15, 16–19

- `xmlInternalTreeParse`, 11
- `xmlNode`, 5, 6

- `YahooFinanceSource`, 2, 16
- `YahooInplaySource`, 2, 17
- `yahoonews`, 18
- `YahooNewsSource`, 2, 18