Package ‘mediacloudr’

October 13, 2022

Type Package
Title Wrapper for the ‘mediacloud.org’ API
Version 0.1.0
Depends R (>= 3.2.0)
Description API wrapper to gather news stories, media information and tags from the ‘mediacloud.org’ API, based on a multilevel query <https://mediacloud.org/>. A personal API key is required.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Imports httr, jsonlite, rvest, xml2
Suggests testthat, covr, knitr, rmarkdown
RoxygenNote 6.1.1
VignetteBuilder knitr
NeedsCompilation no
Author Dix Jan [cre, aut]
Maintainer Dix Jan <jan.dix@uni-konstanz.de>
Repository CRAN
Date/Publication 2019-07-24 07:50:02 UTC

R topics documented:

<table>
<thead>
<tr>
<th>R Topic</th>
<th>Documentation Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>extract_meta_data</td>
<td>2</td>
</tr>
<tr>
<td>get_media_source</td>
<td>2</td>
</tr>
<tr>
<td>get_story</td>
<td>3</td>
</tr>
<tr>
<td>get_story_list</td>
<td>4</td>
</tr>
<tr>
<td>meta_data_html</td>
<td>5</td>
</tr>
</tbody>
</table>

Index 6
extract_meta_data  

*Extract meta data*

**Description**

`extract_meta_data` extracts native, open graph and twitter meta data from html documents. The meta data include url, title, description and image. The html document is parsed within the function.

**Usage**

```r
extract_meta_data(html_doc)
```

**Arguments**

html_doc  
Character string including the html document.

**Value**

List with three sublists for native, open graph and twitter.

**Examples**

```r
## Not run:
library(httr)
response <- GET(url)
html_document <- content(response, type = "text", encoding = "UTF-8")
meta_data <- extract_meta_data(html_doc = html_document)
## End(Not run)
```

get_media_source  

*Get media by id*

**Description**

`get_media` returns media source by their id. A media source is one publisher. Every story that can be collected via `get_story` or `get_story_list` belongs to one media source.

**Usage**

```r
get_media_source(media_id, api_key = Sys.getenv("MEDIACLOUD_API_KEY"))
```
get_story

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>media_id</td>
<td>Positive integer that contains a valid media“ id.</td>
</tr>
<tr>
<td>api_key</td>
<td>Character string with the API key you get from mediacloud.org. Passing it is compulsory. Alternatively, function can be provided from the global environment.</td>
</tr>
</tbody>
</table>

Value

Data frame with results. See https://github.com/berkmancenter/mediacloud/blob/master/doc/api_2_0_spec/api_2_0_spec.md#media for field descriptions.

Examples

```r
## Not run:
media_source <- get_media_source(media_id = 604L)

## End(Not run)
```

get_story

Description

get_story returns news stories by their id. One story represents one online publication. Each story refers to a single URL from any feed within a single media source.

Usage

```r
get_story(story_id, api_key = Sys.getenv("MEDIACLOUD_API_KEY"))
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>story_id</td>
<td>Positive numeric that contains a valid story id.</td>
</tr>
<tr>
<td>api_key</td>
<td>Character string with the API key you get from mediacloud.org. Passing it is compulsory. Alternatively, function can be provided from the global environment.</td>
</tr>
</tbody>
</table>

Value

Data frame with results. See https://github.com/berkmancenter/mediacloud/blob/master/doc/api_2_0_spec/api_2_0_spec.md#stories for field descriptions.
### Examples

```r
## Not run:
story <- get_story(story_id = 604L)

## End(Not run)
```

---

### Description

**get_story** returns a list of stories based on a multifaceted query. One story represents one online publication. Each story refers to a single URL from any feed within a single media source.

### Usage

```r
get_story_list(last_process_stories_id = 0L, rows = 100,
feeds_id = NULL, q = NULL, fq = NULL,
sort = "processed_stories_id", wc = FALSE, show_feeds = FALSE,
api_key = Sys.getenv("MEDIACLOUD_API_KEY"))
```

### Arguments

- **last_process_stories_id**: Return stories in which the processed_stories_id is greater than this value.
- **rows**: Number of stories to return, max 1000.
- **feeds_id**: Return only stories that match the given feeds_id, sorted by descending publish date.
- **q**: If specified, return only results that match the given Solr query. Only one q parameter may be included.
- **fq**: If specified, file results by the given Solr query. More than one fq parameter may be included.
- **sort**: Returned results sort order. Supported values: processed_stories_id, random.
- **wc**: If set to TRUE, include a 'word_count' field with each story that includes a count of the most common words in the story.
- **show_feeds**: If set to TRUE, include a 'feeds' field with a list of the feeds associated with this story.
- **api_key**: Character string with the API key you get from mediacloud.org. Passing it is compulsory. Alternatively, function can be provided from the global environment.

### Value

Data frame with results. See [https://github.com/berkmancenter/mediacloud/blob/master/doc/api_2_0_spec/api_2_0_spec.md#stories](https://github.com/berkmancenter/mediacloud/blob/master/doc/api_2_0_spec/api_2_0_spec.md#stories) for field descriptions.
Examples

```r
## Not run:
stories <- get_story_list()
stories <- get_story_list(q = "Trump")
## End(Not run)
```

Description

A HTML document with basic meta tags for open-graph, twitter and native meta data.

Usage

```r
meta_data_html
```

Format

An object of class character of length 1.
Index

* datasets
  meta_data_html, 5

extract_meta_data, 2
get_media_source, 2
gt_story, 3
gt_story_list, 4
meta_data_html, 5