Package ‘crminer’

June 27, 2020

Type Package

Title Fetch 'Scholary' Full Text from 'Crossref'

Description Text mining client for 'Crossref' (<https://crossref.org>). Includes functions for getting getting links to full text of articles, fetching full text articles from those links or Digital Object Identifiers ('DOIs'), and text extraction from 'PDFs'.

Version 0.4.0

License MIT + file LICENSE

URL `https://github.com/ropensci/crminer` (devel)
`https://docs.ropensci.org/crminer` (docs)

BugReports `https://github.com/ropensci/crminer/issues`

LazyData true

Encoding UTF-8

Language en-US

Imports crul, jsonlite, xml2, pdftools, hoardr

Suggests roxygen2 (>= 7.1.0), rcrossref, testthat, vcr (>= 0.5.4)

RoxygenNote 7.1.0

X-schema.org-applicationCategory Literature

X-schema.org-keywords text-mining, literature, publications, crossref, pdf, xml

X-schema.org-isPartOf https://ropensci.org

NeedsCompilation no

Author Scott Chamberlain [aut, cre] (<https://orcid.org/0000-0003-1444-9135>), rOpenSci [fnd] (https://ropensci.org/)

Maintainer Scott Chamberlain <`myrmecocystus+r@gmail.com`>

Repository CRAN

Date/Publication 2020-06-27 04:50:12 UTC

R topics documented:

`crminer-package` *crminer*

Description

Crossref text miner

Package API (functions)

- `crm_links()` - get full text links from DOIs
- `as_tdmurl()` - coerce a URL to a tdmurl
- `crm_text()` - general purpose interface to request full text
- `crm_pdf()` - request pdf full text
- `crm_plain()` - request plain text full text
- `crm_xml()` - request xml full text
- `crm_extract()` - extract text from a pdf

Authentication

You should first start reading up on authentication (`auth()`) since you are probably here to do text mining, and most of the full text links available via Crossref are behind authentication.

Author(s)

Scott Chamberlain <`myrmecocystus+r@gmail.com`>

`as_tdmurl` *Coerce a url to a tdmurl with a specific type*

Description

A tmd url is just a URL with some attributes to make it easier to handle within other functions in this package.

Usage

```
as_tdmurl(url, type, doi = NULL, member = NULL, intended_application = NULL)

## S3 method for class 'tdmurl'
as_tdmurl(url, type, doi = NULL, member = NULL, intended_application = NULL)

## S3 method for class 'character'
as_tdmurl(url, type, doi = NULL, member = NULL, intended_application = NULL)
```

Arguments

`url`	(character) A URL.
`type`	(character) One of 'xml' (default), 'html', 'plain', 'pdf', 'unspecified', or 'all'
`doi`	(character) A DOI, optional, default: `NULL`
`member`	(character) Crossref member id. optional
`intended_application`	(character) intended application string, optional

Examples

```
as_tdmurl("http://downloads.hindawi.com/journals/bmri/2014/201717.xml",
   "xml")
as_tdmurl("http://downloads.hindawi.com/journals/bmri/2014/201717.pdf",
   "pdf")
out <-
 as_tdmurl("http://downloads.hindawi.com/journals/bmri/2014/201717.pdf",
   "pdf", "10.1155/2014/201717")
attributes(out)
identical(attr(out, "type"), "pdf")
```

auth *Crossref TDM authentication*

Description

Crossref TDM authentication

Authentication

There's a set of publishers that are involved in the Crossref Text and Data Mining (TDM) program (http://tdmsupport.crossref.org/), which means essentially the publishers deposit URLs for fulltext in Crossref metadata.

Authentication is applicable only when the publisher you want to get fulltext from requires it. OA publishers shouldn't need it as you'd expect. There's many publishers that don't share links at all, so they are irrelevant here.

For publishers that required authentication, the Crossref TDM program allows for a single token to authenticate across publishers (to make it easier for text miners). The publishers involved with the authentication scheme are really only Elsevier and Wiley.

There's a how to guide for Crossref TDM at `http://tdmsupport.crossref.org/researchers/`. Get your Crossref TDM token by registering at `https://apps.crossref.org/clickthrough/researchers`. Save the token in your `.Renviron` file with a new row like CROSSREF_TDM=your key. We will read that key in for you - it's best this way rather than passing in a key via a parameter - since you might put that code up on the web somewhere and someone could use your key.

IP addresses

If you don't know what IP addresses are, check out `https://en.wikipedia.org/wiki/IP_address`. At least Elsevier and I think Wiley also check your IP address in addition to requiring the authentication token. Usually your're good if you're physically at the institution that has access to the publishers content OR on a VPN (i.e., pretending you're there).

If you forget about this, you'll get errors about not being authorized. So check and make sure you're on a VPN if you're not physically located at your institution.

Fences

There's yet another issue to worry about. At least with Elsevier, they have a so-called "fence" - that is, even if an institution has access to Elsevier content, Elsevier doesn't necessarily have the fence turned off - if its not off, you can't get through - if it's off, you can. If you have the right token and you are sure you're on the right IP address, this could be the problem for your lack of access.

HELP!

If you're having trouble with any of this, get in touch with the package maintainer.

`crm_cache` *Caching*

Description

Manage cached `crminer` files with **hoardr**

Details

The dafault cache directory is `paste0(rappdirs::user_cache_dir(),"/R/crminer")`, but you can set your own path using `cache_path_set()`

`cache_delete` only accepts 1 file name, while `cache_delete_all` doesn't accept any names, but deletes all files. For deleting many specific files, use `cache_delete` in a `lapply()` type call

Useful user functions

- `crm_cache$cache_path_get()` get cache path
- `crm_cache$cache_path_set()` set cache path. You can set the entire path directly via the `full_path` arg like `crm_cache$cache_path_set(full_path = "your/path")`
- `crm_cache$list()` returns a character vector of full path file names
- `crm_cache$files()` returns file objects with metadata
- `crm_cache$details()` returns files with details
- `crm_cache$delete()` delete specific files
- `crm_cache$delete_all()` delete all files, returns nothing

Examples

```
## Not run:
crm_cache

# list files in cache
crm_cache$list()

# delete certain database files
# crm_cache$delete("file path")
# crm_cache$list()

# delete all files in cache
# crm_cache$delete_all()
# crm_cache$list()

## End(Not run)
```

`crm_extract` *Extract text from a single pdf document*

Description

Extract text from a single pdf document

Usage

```
crm_extract(path = NULL, raw = NULL, try_ocr = FALSE, ...)
```

Arguments

`path`	(character) path to a file, file must exist
`raw`	(raw) raw bytes
`try_ocr`	(logical) whether to try extracting OCRed pages with `pdftools::pdf_ocr_text()`. default: `FALSE`. if `FALSE`, we use `pdftools::pdf_text()`
`...`	args passed on to `pdftools::pdf_info()` and `pdftools::pdf_text()` (or `pdftools::pdf_ocr_text(` if `try_ocr=TRUE`) - any args are passed to both of those function calls, which makes sense

Details

We use **pdftools** under the hood to do pdf text extraction.

You have to supply either `path` or `raw` - not both.

Value

An object of class `crm_pdf` with a slot for `info` (pdf metadata essentially), and `text` (the extracted text) - with an attribute (`path`) with the path to the pdf on disk

Examples

```
path <- system.file("examples", "MairChamberlain2014RJournal.pdf",
   package = "crminer")
(res <- crm_extract(path))
res$info
res$text
# with newlines, pretty print
cat(res$text)

# another example
path <- system.file("examples", "ChamberlainEtal2013Ecosphere.pdf",
   package = "crminer")
(res <- crm_extract(path))
res$info
cat(res$text)
```

```
# with raw pdf bytes
path <- system.file("examples", "raw-example.rds", package = "crminer")
rds <- readRDS(path)
class(rds)
crm_extract(raw = rds)
```

`crm_html` *Get full plain text*

Description

Get full plain text

Usage

```
crm_html(url, overwrite_unspecified = FALSE, ...)
```

Arguments

`url`	A URL (character) or an object of class `tdmurl` from a call to `crm_links()`. If you'll be getting text from the publishers are use Crossref TDM (which requires authentication), we strongly recommend using `crm_links()` first and passing output of that here, as `crm_links()` grabs the publisher Crossref member ID, which we use to do authentication and other publisher specific fixes to URLs
`overwrite_unspecified`	(logical) Sometimes the crossref API returns mime type 'unspecified' for the full text links (for some Wiley dois for example). This parameter overrides the mime type to be `type`.
`...`	Named curl options passed on to crul::verb-GET, see `curl::curl_options()` for available curl options. See especially the User-agent section below

Details

Note that this function is not vectorized. To do many requests use a for/while loop or lapply family calls, or similar.

Note that some links returned will not in fact lead you to full text content as you would understandably think and expect. That is, if you use the `filter` parameter with e.g., `rcrossref::cr_works()` and filter to only full text content, some links may actually give back only metadata for an article. Elsevier is perhaps the worst offender, for one because they have a lot of entries in Crossref TDM, but most of the links that are apparently full text are not in facct full text, but only metadata.

Check out auth for details on authentication.

User-agent

You can optionally set a user agent string with the curl option `useragent`, like `crm_text("some doi","pdf",useragent = "foo bar")`. user agent strings are sometimes used by servers to decide whether to provide a response (in this case, the full text article). sometimes, a browser like user agent string will make the server happy. by default all requests in this package have a user agent string like libcurl/7.64.1 r-curl/4.3 crul/0.9.0, which is a string with the names and versions of the http clients used under the hood. If you supply a user agent string using the `useragent` curl option, we'll use it instead. For more information on user agent's, and exmaples of user agent strings you can use here, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

Examples

```
## Not run:
link <- crm_links("10.7717/peerj.1545", "html")
crm_html(link)

link <- crm_links("10.7717/peerj.1545")
crm_html(link)

crm_html("https://peerj.com/articles/1545.html")

## End(Not run)
```

`crm_links` *Get full text links from a DOI*

Description

Get full text links from a DOI

Usage

```
crm_links(doi, type = "all", ...)
```

Arguments

`doi`	(character) A Digital Object Identifier (DOI). required.
`type`	(character) One of 'xml', 'html', 'plain', 'pdf', 'unspecified', or 'all' (default). required.
`...`	Named parameters passed on to `crul::HttpClient()`

Details

Note that this function is not vectorized.

Some links returned will not in fact lead you to full text content as you would understandbly think and expect. That is, if you use the `filter` parameter with e.g., `rcrossref::cr_works()` and filter to only full text content, some links may actually give back only metadata for an article. Elsevier is perhaps the worst offender, for one because they have a lot of entries in Crossref TDM, but most of the links that are apparently full text are not in fact full text, but only metadata. You can get full text if you are part of a subscribing institution to that specific Elsever content, but otherwise, you're SOL.

Note that there are still some bugs in the data returned form CrossRef. For example, for the publisher eLife, they return a single URL with content-type application/pdf, but the URL is not for a PDF, but for both XML and PDF, and content-type can be set with that URL as either XML or PDF to get that type.

In another example, all Elsevier URLs at time of writing are have `http` scheme, while those don't actually work, so we have a custom fix in this function for that publisher. Anyway, expect changes...

Value

`NULL` if no full text links given; a list of tdmurl objects if links found. a tdmurl object is an S3 class wrapped around a simple list, with attributes for:

- type: type, matchin type passed to the function
- doi: DOI
- member: Crossref member ID
- intended_application: intended application, e.g., text-mining

Register for the Polite Pool

The `crm_links()` uses the Crossref API You should send your email address with your `crm_links()` requests. This has the advantage that queries are placed in the polite pool of servers. In addition, even if the non-polite pool is having server problems, the polite pool is often okay. Including your email address is good practice as described in the Crossref documentation under Good manners. To pass your email address to Crossref, simply store it as an environment variable in .Renviron file like `crossref_email=name@example.com`, or `CROSSREF_EMAIL=name@example.com`. Save the file and restart your R session. To stop sharing your email when using rcrossref simply delete it from your `.Renviron` file OR to temporarily not use your email unset it for the session like `Sys.unsetenv('crossref_email')`. To be sure your in the polite pool use curl verbose by e.g., `crm_links(doi = "10.5555/515151",verbose = TRUE)`

Examples

```
## Not run:
data(dois_crminer)

# pdf link
crm_links(doi = "10.5555/515151", "pdf")

# xml and plain text links
crm_links(dois_crminer[1], "pdf")
crm_links(dois_crminer[6], "xml")
crm_links(dois_crminer[7], "plain")
crm_links(dois_crminer[1]) # all is the default

# pdf link
```

```
crm_links(doi = "10.5555/515151", "pdf")
crm_links(doi = "10.3897/phytokeys.52.5250", "pdf")

# many calls, use e.g., lapply
lapply(dois_crminer[1:3], crm_links)

# elsevier
## DOI that is open acccess
crm_links('10.1016/j.physletb.2010.10.049')
## DOI that is not open acccess
crm_links('10.1006/jeth.1993.1066')

## End(Not run)
```

crm_pdf *Get full text PDFs*

Description

Get full text PDFs

Usage

```
crm_pdf(url, overwrite = TRUE, read = TRUE, overwrite_unspecified = FALSE, ...)
```

Arguments

`url`	A URL (character) or an object of class `tdmurl` from a call to `crm_links()`. If you'll be getting text from the publishers are use Crossref TDM (which requires authentication), we strongly recommend using `crm_links()` first and passing output of that here, as `crm_links()` grabs the publisher Crossref member ID, which we use to do authentication and other publisher specific fixes to URLs
`overwrite`	(logical) Overwrite file if it exists already? Default: `TRUE`
`read`	(logical) If reading a pdf, this toggles whether we extract text from the pdf or simply download. If `TRUE`, you get the text from the pdf back. If `FALSE`, you only get back the metadata. Default: `TRUE`
`overwrite_unspecified`	(logical) Sometimes the crossref API returns mime type 'unspecified' for the full text links (for some Wiley dois for example). This parameter overrides the mime type to be `type`.
`...`	Named curl options passed on to crul::verb-GET, see `curl::curl_options()` for available curl options. See especially the User-agent section below

Notes

Note that this function is not vectorized. To do many requests use a for/while loop or lapply family calls, or similar.

Note that some links returned will not in fact lead you to full text content as you would understandably think and expect. That is, if you use the `filter` parameter with e.g., `rcrossref::cr_works()` and filter to only full text content, some links may actually give back only metadata for an article. Elsevier is perhaps the worst offender, for one because they have a lot of entries in Crossref TDM, but most of the links that are apparently full text are not in facct full text, but only metadata.

Check out auth for details on authentication.

User-agent

You can optionally set a user agent string with the curl option `useragent`, like `crm_text("some doi","pdf",useragent = "foo bar")`. user agent strings are sometimes used by servers to decide whether to provide a response (in this case, the full text article). sometimes, a browser like user agent string will make the server happy. by default all requests in this package have a user agent string like libcurl/7.64.1 r-curl/4.3 crul/0.9.0, which is a string with the names and versions of the http clients used under the hood. If you supply a user agent string using the `useragent` curl option, we'll use it instead. For more information on user agent's, and exmaples of user agent strings you can use here, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

Elsevier-partial

For at least some PDFs from Elsevier, most likely when you do not have full access to the full text, they will return a successful response, but only return the first page of the PDF. They do however include a warning message in the response headers, which we look for and pass on to the user AND delete the pdf because we assume if you are using this package you don't want just the first page but the whole article. This behavior as far as we know does not occur with other article types (xml, plain), but let us know if you see it.

Caching

By default we use `paste0(rappdirs::user_cache_dir(),"/crminer")`, but you can set this directory to something different. Paths are setup under "/crminer" for each of the file types: "/crminer/pdf", "/crminer/xml", "/crminer/txt", and "/crminer/html". See crm_cache for caching details.

We cache all file types, as well as the extracted text from the pdf. The text is saved in a text file with the same file name as the pdf, but with the file extension ".txt". On subsequent requests of the same DOI, we first look for a cached .txt file matching the DOI, and return it if it exists. If it does not exist, but the the PDF does exist, we skip the PDF download step and move on to reading the PDF to text; we cache that text in to .txt file. If there's no .txt or .pdf file, we download the PDF and read the pdf to text, and both are cached.

Examples

```
## Not run:
# set a temp dir. cache path
crm_cache$cache_path_set(path = "crminer", type = "tempdir")
## you can set the entire path directly via the `full_path` arg
## like crm_cache$cache_path_set(full_path = "your/path")

## peerj
x <- crm_pdf("https://peerj.com/articles/6840.pdf")

## pensoft
data(dois_pensoft)
(links <- crm_links(dois_pensoft[10], "all"))
crm_pdf(links)

## hindawi
data(dois_pensoft)
(links <- crm_links(dois_pensoft[12], "all"))
### pdf
crm_pdf(links, read=FALSE)
crm_pdf(links)

## End(Not run)
```

`crm_plain` *Get full plain text*

Description

Get full plain text

Usage

```
crm_plain(url, overwrite_unspecified = FALSE, ...)
```

Arguments

`url`	A URL (character) or an object of class `tdmurl` from a call to `crm_links()`. If you'll be getting text from the publishers are use Crossref TDM (which requires authentication), we strongly recommend using `crm_links()` first and passing output of that here, as `crm_links()` grabs the publisher Crossref member ID, which we use to do authentication and other publisher specific fixes to URLs
`overwrite_unspecified`	(logical) Sometimes the crossref API returns mime type 'unspecified' for the full text links (for some Wiley dois for example). This parameter overrides the mime type to be `type`.
`...`	Named curl options passed on to crul::verb-GET, see `curl::curl_options()` for available curl options. See especially the User-agent section below

Details

Note that this function is not vectorized. To do many requests use a for/while loop or lapply family calls, or similar.

Note that some links returned will not in fact lead you to full text content as you would understandbly think and expect. That is, if you use the `filter` parameter with e.g., `rcrossref::cr_works()` and filter to only full text content, some links may actually give back only metadata for an article. Elsevier is perhaps the worst offender, for one because they have a lot of entries in Crossref TDM, but most of the links that are apparently full text are not in facct full text, but only metadata.

Check out auth for details on authentication.

User-agent

You can optionally set a user agent string with the curl option `useragent`, like `crm_text("some doi","pdf",useragent = "foo bar")`. user agent strings are sometimes used by servers to decide whether to provide a response (in this case, the full text article). sometimes, a browser like user agent string will make the server happy. by default all requests in this package have a user agent string like libcurl/7.64.1 r-curl/4.3 crul/0.9.0, which is a string with the names and versions of the http clients used under the hood. If you supply a user agent string using the `useragent` curl option, we'll use it instead. For more information on user agent's, and exmaples of user agent strings you can use here, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

Examples

```
## Not run:
link <- crm_links("10.1016/j.physletb.2010.10.049", "plain")
crm_plain(link)

# another eg, which requires Crossref TDM authentication, see ?auth
link <- crm_links(dois_elsevier[3], "plain")
# crm_plain(link)

## End(Not run)
```

`crm_text` *Get full text*

Description

Get full text

Usage

```
crm_text(
  url,
  type = "xml",
  overwrite = TRUE,
  read = TRUE,
  overwrite_unspecified = FALSE,
  try_ocr = FALSE,
  ...
)
```

Arguments

`url`	A URL (character) or an object of class `tdmurl` from a call to `crm_links()`. If you'll be getting text from the publishers are use Crossref TDM (which requires authentication), we strongly recommend using `crm_links()` first and passing output of that here, as `crm_links()` grabs the publisher Crossref member ID, which we use to do authentication and other publisher specific fixes to URLs
`type`	(character) One of 'xml' (default), 'html', 'plain', 'pdf', 'unspecified'
`overwrite`	(logical) Overwrite file if it exists already? Default: `TRUE`
`read`	(logical) If reading a pdf, this toggles whether we extract text from the pdf or simply download. If `TRUE`, you get the text from the pdf back. If `FALSE`, you only get back the metadata. Default: `TRUE`
`overwrite_unspecified`	(logical) Sometimes the crossref API returns mime type 'unspecified' for the full text links (for some Wiley dois for example). This parameter overrides the mime type to be `type`.
`try_ocr`	(logical) whether to try extracting OCRed pages with `pdftools::pdf_ocr_text()`. default: `FALSE`. if `FALSE`, we use `pdftools::pdf_text()`
`...`	Named curl options passed on to crul::verb-GET, see `curl::curl_options()` for available curl options. See especially the User-agent section below

Notes

Note that this function is not vectorized. To do many requests use a for/while loop or lapply family calls, or similar.

Note that some links returned will not in fact lead you to full text content as you would understandbly think and expect. That is, if you use the `filter` parameter with e.g., `rcrossref::cr_works()` and filter to only full text content, some links may actually give back only metadata for an article. Elsevier is perhaps the worst offender, for one because they have a lot of entries in Crossref TDM, but most of the links that are apparently full text are not in facct full text, but only metadata.

Check out auth for details on authentication.

Caching

By default we use `paste0(rappdirs::user_cache_dir(),"/crminer")`, but you can set this directory to something different. Paths are setup under "/crminer" for each of the file types: "/crminer/pdf", "/crminer/xml", "/crminer/txt", and "/crminer/html". See crm_cache for caching details.

We cache all file types, as well as the extracted text from the pdf. The text is saved in a text file with the same file name as the pdf, but with the file extension ".txt". On subsequent requests of the same DOI, we first look for a cached .txt file matching the DOI, and return it if it exists. If it does not exist, but the the PDF does exist, we skip the PDF download step and move on to reading the PDF to text; we cache that text in to .txt file. If there's no .txt or .pdf file, we download the PDF and read the pdf to text, and both are cached.

User-agent

You can optionally set a user agent string with the curl option `useragent`, like `crm_text("some doi","pdf",useragent = "foo bar")`. user agent strings are sometimes used by servers to decide whether to provide a response (in this case, the full text article). sometimes, a browser like user agent string will make the server happy. by default all requests in this package have a user agent string like libcurl/7.64.1 r-curl/4.3 crul/0.9.0, which is a string with the names and versions of the http clients used under the hood. If you supply a user agent string using the `useragent` curl option, we'll use it instead. For more information on user agent's, and exmaples of user agent strings you can use here, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

Elsevier-partial

For at least some PDFs from Elsevier, most likely when you do not have full access to the full text, they will return a successful response, but only return the first page of the PDF. They do however include a warning message in the response headers, which we look for and pass on to the user AND delete the pdf because we assume if you are using this package you don't want just the first page but the whole article. This behavior as far as we know does not occur with other article types (xml, plain), but let us know if you see it.

Examples

```
## Not run:
# set a temp dir. cache path
crm_cache$cache_path_set(path = "crminer", type = "tempdir")
## you can set the entire path directly via the `full_path` arg
## like crm_cache$cache_path_set(full_path = "your/path")

## pensoft
data(dois_pensoft)
(links <- crm_links(dois_pensoft[1], "all"))
### xml
crm_text(url=links, type='xml')
### pdf
crm_text(url=links, type="pdf", read = FALSE)
crm_text(links, "pdf")

## hindawi
data(dois_pensoft)
(links <- crm_links(dois_pensoft[1], "all"))
### xml
crm_text(links, 'xml')
### pdf
crm_text(links, "pdf", read=FALSE)
crm_text(links, "pdf")

## DOIs w/ full text, and with CC-BY 3.0 license
data(dois_crminer_ccby3)
(links <- crm_links(dois_crminer_ccby3[40], "all"))
# crm_text(links, 'pdf')

## You can use crm_xml, crm_plain, and crm_pdf to go directly to
## that format
(links <- crm_links(dois_crminer_ccby3[5], "all"))
crm_xml(links)

### Caching, for PDFs
if (requireNamespace("rcrossref")) {
  library("rcrossref")
  out <- cr_members(2258, filter=c(has_full_text = TRUE), works = TRUE)
  # (links <- crm_links(out$data$DOI[10], "all"))
  # crm_text(links, type = "pdf")
  # system.time( first <- crm_text(links, type = "pdf") )
  ### second time should be faster
  # system.time( second <- crm_text(links, type = "pdf") )
  # identical(first, second)
}

## elsevier
## requires authentication
### load some Elsevier DOIs
data(dois_elsevier)

## set key first - OR set globally - See ?auth
# Sys.setenv(CROSSREF_TDM_ELSEVIER = "your-key")
## XML
link <- crm_links("10.1016/j.funeco.2010.11.003", "xml")
# res <- crm_text(url = link, type = "xml")
## plain text
link <- crm_links("10.1016/j.funeco.2010.11.003", "plain")
# res <- crm_text(url = link, "plain")

# try_ocr
x <- crm_links('10.1006/jeth.1993.1066')
# (out <- crm_text(x, "pdf", try_ocr = TRUE))
x <- crm_links('10.1006/jeth.1997.2332')
# (out <- crm_text(x, "pdf", try_ocr = TRUE))

## Wiley
## requires authentication
### load some Wiley DOIs
data(dois_wiley)

## set key first - OR set globally - See ?auth
# Sys.setenv(CROSSREF_TDM = "your-key")

### all wiley
tmp <- crm_links("10.1111/apt.13556", "all")
# crm_text(url = tmp, type = "pdf",
#   overwrite_unspecified = TRUE)

#### older dates for Wiley
# tmp <- crm_links(dois_wiley$set2[1], "all")
# crm_text(tmp, type = "pdf",
#    overwrite_unspecified=TRUE)
```

```
### Wiley paper with CC By 4.0 license
# tmp <- crm_links("10.1113/jp272944", "all")
# crm_text(tmp, type = "pdf")

## End(Not run)
```

`crm_xml` *Get full text XML*

Description

Get full text XML

Usage

```
crm_xml(url, overwrite_unspecified = FALSE, ...)
```

Arguments

`url`	A URL (character) or an object of class `tdmurl` from a call to `crm_links()`. If you'll be getting text from the publishers are use Crossref TDM (which requires authentication), we strongly recommend using `crm_links()` first and passing output of that here, as `crm_links()` grabs the publisher Crossref member ID, which we use to do authentication and other publisher specific fixes to URLs
`overwrite_unspecified`	(logical) Sometimes the crossref API returns mime type 'unspecified' for the full text links (for some Wiley dois for example). This parameter overrides the mime type to be `type`.
`...`	Named curl options passed on to crul::verb-GET, see `curl::curl_options()` for available curl options. See especially the User-agent section below

Details

Note that this function is not vectorized. To do many requests use a for/while loop or lapply family calls, or similar.

Note that some links returned will not in fact lead you to full text content as you would understandbly think and expect. That is, if you use the `filter` parameter with e.g., `rcrossref::cr_works()` and filter to only full text content, some links may actually give back only metadata for an article. Elsevier is perhaps the worst offender, for one because they have a lot of entries in Crossref TDM, but most of the links that are apparently full text are not in facct full text, but only metadata.

Check out auth for details on authentication.

User-agent

You can optionally set a user agent string with the curl option `useragent`, like `crm_text("some doi","pdf",useragent = "foo bar")`. user agent strings are sometimes used by servers to decide whether to provide a response (in this case, the full text article). sometimes, a browser like user agent string will make the server happy. by default all requests in this package have a user agent string like libcurl/7.64.1 r-curl/4.3 crul/0.9.0, which is a string with the names and versions of the http clients used under the hood. If you supply a user agent string using the `useragent` curl option, we'll use it instead. For more information on user agent's, and exmaples of user agent strings you can use here, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent

Examples

```
## Not run:
## peerj
x <- crm_xml("https://peerj.com/articles/2356.xml")

## pensoft
data(dois_pensoft)
(links <- crm_links(dois_pensoft[1], "all"))
### xml
crm_xml(url=links)

## End(Not run)
```

`dois_crminer` *A character vector of 500 DOIs from Crossref*

Description

Obtained via `rcrossref::cr_works(filter = c(has_full_text = TRUE),limit = 500)`

Format

A character vector of length 500

`dois_crminer_ccby3` *A character vector of 100 DOIs from Crossref with license CC-BY 3.0*

Description

Obtained via `rcrossref::cr_works(filter = list(has_full_text = TRUE,license_url="http://creativecommons.`
`= 100)`

Format

A character vector of length 100

`dois_elsevier` *A character vector of 100 Elsevier DOIs from Crossref*

Description

Obtained via `rcrossref::cr_members(78,filter = c(has_full_text = TRUE),works = TRUE,limit = 100)`

Format

A character vector of length 100

`dois_hindawi` *A character vector of 50 Hindawi DOIs from Crossref*

Description

Obtained via `rcrossref::cr_members(98,filter = c(has_full_text = TRUE),works = TRUE,limit = 50)`

Format

A character vector of length 50

`dois_pensoft` *A character vector of 100 Pensoft DOIs from Crossref*

Description

Obtained via `rcrossref::cr_members(2258,filter = c(has_full_text = TRUE),works = TRUE,limit = 100)`

Format

A character vector of length 100

`dois_wiley` *A list of 3 character vectors totaling 250 Wiley DOIs from Crossref*

Description

- set1: Obtained via `rcrossref::cr_members(311,filter = c(has_full_text = TRUE),works = TRUE,limit = 100)`
- set2 (a set with older dates): Obtained via `rcrossref::cr_members(311,filter=c(has_full_text = TRUE,type = 'journal-article',until_created_date = "2013-12-31"),works = TRUE,limit = 100)`
- set3 (with CC By 4.0 license): Obtained via `rcrossref::cr_members(311,filter=c(has_full_text = TRUE,license.url = "http://creativecommons.org/licenses/by/4.0/"),works = TRUE,limit = 100)`

Format

A list of length 3, `set1` with a character vector of length 100, `set2` with a character vector of length 100, and `set3` with a character vector of length 50.

Index