The Crawler

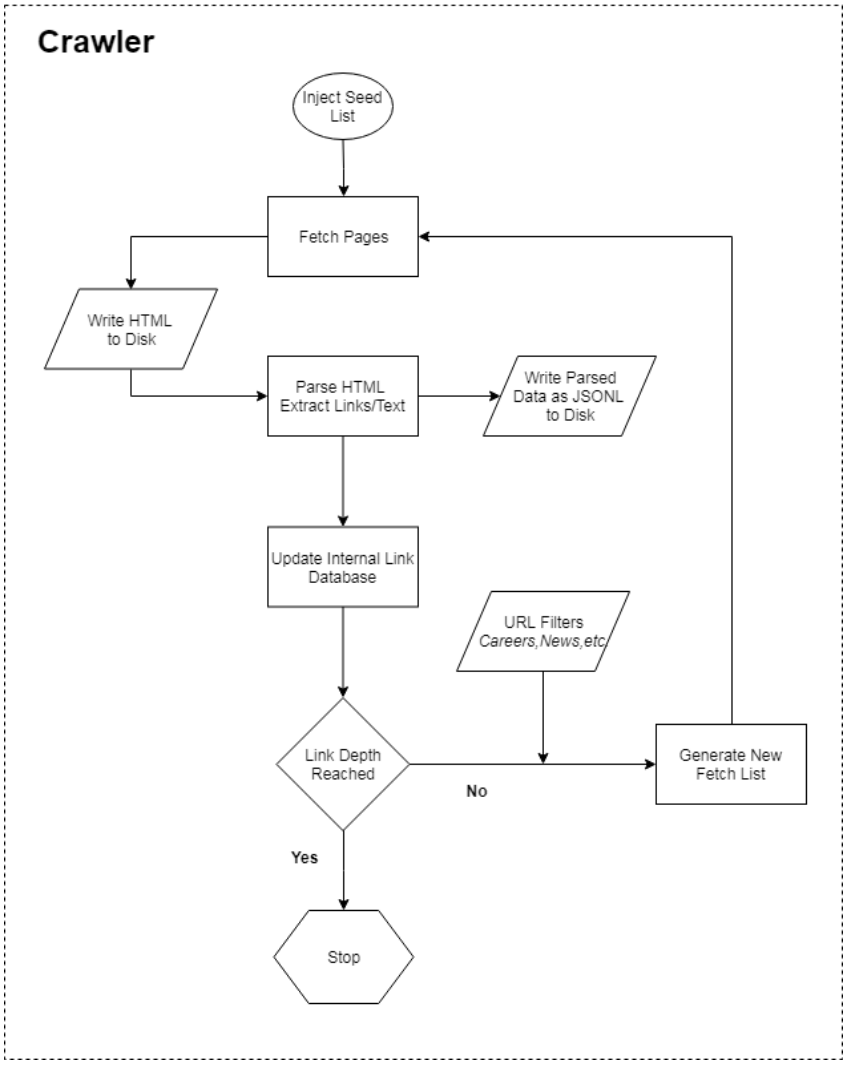
CrawlR

Program responsible for crawling and parsing HTML. Output from this program is fed into the next stage - the pipeline.

Web crawlers Apache Nutch and Scrapy were considered, but this crawler was eventually written in order to have more control/customization over the crawling process.

- Basic process:
  - inject seeds into linkDB
  - generate a fetch list from linkDB
  - fetch created fetch_list
  - parse fetched data
  - update pageDB and linkDB
  - if link_depth not yet reached, repeat

The crawler scheduling is currently being fleshed out, but the current plan is to assign a random 'Re-Crawl Date' , sometime between 2-4 weeks after each crawl – This is to stagger the number of sites being crawled at any given time and not kill the server.



crawlR: Crawl and Parse Web Pages

Description

Utilizes the curl package to crawl through a list of user supplied websites to given depth. Unlesss overriden, crawlR observes robots.txt.

Usage

```
crawlR(seeds = NULL, work_dir = NULL, out_dir = NULL,
  max_concurr = 2, max_host = 1, timeout = Inf, external_site = F,
  sitemaps = F, crawl_delay = 10, max_size = 4e+06, regExIn = NULL,
  regExOut = NULL, depth = 4, queue_scl = 1, topN = NULL,
  max_urls_per_host = 10, n_threads = 1, parser = parseR)
```

Arguments

Argument	Description
seeds	Seed url's. If null, then the work_dir must containg a linkDB. If additional seeds are provided after inital seeding, the new seed url's will be added to linkDB and fetched - keeping the original seeds/fetched pages.
work_dir	(Required) Main working directory.
out_dir	Directory to store crawled and parsed html If null defaults to work directory.
max_concurr	Max. total concurrent connections open at any given time.
max_host	Max. total concurrent connections per host at any given time.
timeout	Total (as in all url's in seed list) time per each iteration (for each depth).
external_site	If true, crawler will follow external links.
crawl_delay	time (in seconds) for calls to the same host. Only applies if the time is not specified by the host's robots.txt.
max_size	Max size of file or webpage to download and parse.
regExIn	url's matching this regular expression will be used.
regExOut	url's matching this reg-ex will be filtered out, including url's that match regExIn.
depth	Crawl depth - A value of 1 only crawls the seed pages, 2 crawls links found on seeds, etc..
queue_scl	(Deprecated) max_concur * queue_scl gives que.
topN	Top num links to fetch per per link depth iteration.
max_urls_per_host	Maximum url's from each host when creating fetch list for each link depth.
n_threads	Only applies to parsing.
parser	Parsing function to use.
score_func	Link Scoring Func

Details

After each iteration of crawling, the crawled pages are read from disk, parsed, and writen back to disk. The read/parse phase is done in parrallel using the future package

Current Crawl Configuration

```
## SETUP ----------------------------------------------------------------------

library(crawlR)

work_dir <- '/usr/local/solr/crawl/'
max_concurr = 150        # max concurrent connections - total
max_concurr_invest = 100 # invest pages (10-k) are text heavy
max_host = 1             # max concurrent connections - per host
crawl_delay = 30         # delay in seconds between sucessvie requests same host
timeout = Inf            # total time for crawling ALL urls

## CRAWL SEEDS/CONTACTS -------------------------------------------------------

if(file.exists(paste0(work_dir,'/seeds.txt'))){

## Get seeds
depth <- 1
topN <- NULL
out_dir <- paste0(work_dir)

fh <- file(paste0(work_dir,'/seeds.txt'))

seeds <- readLines(fh)
close(fh)

## Get initial pages
crawlR::crawlR(
seeds = seeds,
work_dir = work_dir,
out_dir =  out_dir,
max_concurr = max_concurr,
max_host = max_host,
timeout = timeout,
external_site = F,
crawl_delay=crawl_delay,
max_size = 4e6,
regExOut = NULL,
regExIn = NULL,
depth = 1,
queue_scl = 1,
topN=NULL,
max_urls_per_host = 1,
n_threads = 2,
parser = crawlR:::parse_content)
```

```
file.remove(paste0(work_dir,'/seeds.txt'))

if(F){

## CRAWL CONTACTS ----------------------------------------------------

depth <- 1
topN <- NULL
out_dir <- paste0(work_dir,'contact/')

contactFiltIn <- list()
contactFiltIn[paste(1:depth)]<-"contact"

contactFiltOut <- list()
contactFiltOut[paste0(1:depth)]<-paste0(
"report|quarterly-report|annual-report|finance|invest|asset|",
"holding|10k|10-k|10q|10-q|quarterly|earning|annual|",
"study|studies|portfolio|project|press|announce|event|news|",
"completed|future|past|complete|construction|current");

crawlR(
seeds = NULL,
work_dir = work_dir,
out_dir =  out_dir,
max_concurr = max_concurr,
max_host = max_host,
timeout = timeout,
external_site = F,
crawl_delay=crawl_delay,
max_size = 4e6,
regExOut = contactFiltOut,
regExIn = contactFiltIn,
depth = depth,
queue_scl = 1,
topN=NULL,
max_urls_per_host = 15,
n_threads = 2,
parser = crawlR:::parse_contact)
}
}

## CRAWL NEWS ----------------------------------------------------------

depth <- 4
topN <- 50000
out_dir <- paste0(work_dir,'projects/')

projFiltIn <- list()
projFiltIn[paste0(1:depth)]<-paste0(
"study|studies|portfolio|project|press|announce|",
"event|news|completed|future|past|complete|construction|current");

projFiltOut <- list()
```

```
projFiltOut[paste0(1:depth)]<-paste0(
"\\.xml|facebook|linkedin|instagram|career|job|finance|invest|",
"asset|holding|10k|10-k|10q|10-q|earnings|contact|team|quarter|annual");

crawlR(
seeds = NULL,
work_dir = work_dir,
out_dir =  out_dir,
max_concurr = max_concurr,
max_host = max_host,
timeout = timeout,
external_site = F,
crawl_delay=crawl_delay,
max_size = 4e6,
regExOut = projFiltOut,
regExIn = projFiltIn,
depth = depth,
queue_scl = 1,
topN=topN,
max_urls_per_host = 15,
n_threads = 2,
parser = crawlR:::parse_content)

## CRAWL INVESTOR ------------------------------------------------------------

depth<-3
topN <- 50000
out_dir <- paste0(work_dir,'invest/')

investFiltIn <- list()
investFiltIn[paste(1:depth)]<-paste0(
"report|quarterly-report|annual-report|finance|",
"invest|asset|holding|10k|10-k|10q|10-q|quarterly|earning|annual");

investFiltOut <- list()
investFiltOut[paste(1:depth)]<-paste0(
"facebook|linkedin|instagram|career|job|contact|about|openings|team");

crawlR(
seeds = NULL,
work_dir = work_dir,
out_dir =  out_dir,
max_concurr = max_concurr,
max_host = max_host,
timeout = timeout,
external_site = F,
crawl_delay=crawl_delay,
max_size = 4e6,
regExOut = investFiltOut,
regExIn = investFiltIn,
depth = depth,
queue_scl = 1,
topN=topN,
```

```
max_urls_per_host = 15,
n_threads = 2,
parser = crawlR:::parse_content)

## CRAWL CAREERS ------------------------------------------------------------

depth <- 2
topN <- 50000
out_dir <- paste0(work_dir,'career/')

careerFiltIn <- list()
careerFiltIn[paste(1:depth)]<-"career|job|opening|hiring|opening|work|team|join"

careerFiltOut <- list()
careerFiltOut[paste(1:depth)]<-paste0(
"facebook|linkedin|instagram|project|press|announce|",
"event|news|completed|future|past|complete|finance|invest|",
"asset|holding|10k|10-k|10q|10-q|earnings|contact");

crawlR(
seeds = NULL,
work_dir = work_dir,
out_dir =  out_dir,
max_concurr = max_concurr,
max_host = max_host,
timeout = timeout,
external_site = F,
crawl_delay=crawl_delay,
max_size = 4e6,
regExOut = careerFiltOut,
regExIn = careerFiltIn,
depth = depth,
queue_scl = 1,
topN=topN,
max_urls_per_host = 15,
n_threads = 2,
parser = crawlR:::parse_content)
```

injectR: Inject Seed List into CrawlDB.

Description

Inject Seed List into CrawlDB.

Usage

```
injectR(out_dir = NULL, work_dir = NULL, seeds = NULL,
  sitemaps = sitemaps, urlRegExFilterOut = NULL, max_concurr = 100,
  crawl_delay = 10, queue_scl = 5, timeout = Inf, return = F)
```

Arguments

Argument	Description
`out_dir`	Output Directory directory for this crawl.
`work_dir`	(Required) Working Directory directory for this crawl.
`seeds`	Url's to inject.
`sitemaps`	If true, the sitemaps are added to the hostDB.
`urlRegExFilterOut`	regex filter.
`max_concurr`	max concurrent conne
`crawl_delay`	crawl_delay
`queue_scl`	que
`timeout`	Inf
`return`	If true, a list contating linkDB and hostDB is returned.

Value

Returns character vector of links, or error message.

generateR: Generate Fetch List of Url's from linkDB

Description

Generate Fetch List of Url's from linkDB

Usage

```
generateR(out_dir = NULL, work_dir = NULL, regExOut = NULL,
  regExIn = NULL, topN = NULL, external_site = F,
  max_urls_per_host = NULL, crawl_delay = NULL, return = F)
```

Arguments

Argument	Description
out_dir	Output directory for this crawl.
work_dir	(Required) Working directory for this crawl.
regExOut	RegEx URL filter - omit links with these keywords.
regExIn	RegEx URL filter - keep links with these keywords.
topN	Choose these top links.
external_site	Logical. If False, host outside the seed list will NOT be crawled.
max_urls_per_host	Max URL's to generate per host.
return	return data.

Value

Returns character vector of links, or error message.

fetchR: Fetch a List of Url's.

Description

Based on the curl package (a wrapper for libcurl). The fetch list of urls is organized into batches, with each batch containing one url from one host. Provides a convienent way to avoid hitting a server too often. A delay also kicks in if a host is being queried too quickly.

Usage

```
fetchR(out_dir = NULL, work_dir = NULL, fetch_list = NULL,
  crawl_delay = NULL, max_concurr = NULL, max_host = NULL,
  timeout = Inf, queue_scl = 1, comments = "", save_to_disk = T,
  return = F)
```

Arguments

Argument	Description
`out_dir`	(Required) Current output directory.
`work_dir`	(Required) Current working directory.
`fetch_list`	(Required) Created by generateR.R.
`crawl_delay`	time (in seconds) for calls to the same host.
`max_concurr`	Max. total concurrent connections open at any given time.
`max_host`	Max. total concurrent connections per host at any given time.
`timeout`	Timeout time
`queue_scl`	Scaler
`comments`	Some comments to print while running.
`save_to_disk`	Save output to disk or not.
`return`	Return output or not.

Value

None.

parse_content: General Parser

Description

Extracts the title, headers, span, and p tags from a page.

Usage

```
parse_content(res)
```

Arguments

Argument	Description
`res`	Return value from curl.

Value

Returns character vector of links, or error message.

parse_contact: Parse Contact

Description

Taken from crawl example given in the curl package.

Usage

```
parse_contact(res)
```

Arguments

Argument	Description
`res`	Return value from curl.

Value

Returns character vector of links, or error message.

parseExt: Get File Extension from Content-Type

Description

Uses lookup provided by rTika to get file extension. content-type: text/html would return ".html" file extension, and content-type: application/pdf would return ".pdf".

Usage

```
parseExt(type, extLookUp)
```

Arguments

Argument	Description
`type`	content-type of crawled page.

parseR: Parse Processor

Description

Parse Processor

Usage

```
parseR(this_dir = NULL, parser = parse_content, n_threads = 4)
```

Arguments

Argument	Description
`this_dir`	Fetch directory to parse.
`n_threads`	Number of threads to use for parseing.
`parseR`	Function that will be used for parseing.

Value

URLS beginning with (http/https), followed by ://www..

get_links: Grab Links Found on Webpage.

Description

Parses HTML for URLs and stores then in the linkDB.

Usage

```
get_links(res)
```

Arguments

Argument	Description
`res`	Return value from curl.

Value

Returns character vector of links.

updateR: Update linkDB and pgDB After Crawl

Description

Updates the various DB's by taking the last fetched_pg file.

Usage

```
updateR(work_dir = NULL, out_dir = NULL, max_concurr = 75,
  crawl_delay = 5, queue_scl = 1, timeout = Inf, sitemaps = F,
  return = F)
```

Arguments

Argument	Description
`out_dir`	(Required) Working directory for this crawl.
`max_concurr`	max conn.
`crawl_delay`	crawl_delay
`queue_scl`	queue scaler.
`timeout`	Inf
`sitemaps`	If true, the sitemaps are added to the hostDB.
`return`	If true, a list contating linkDB and hostDB is returned.

Value

Returns character vector of links, or error message.

writeR: Helper Function to Write Out Web Pages

Description

Write web page to disk. Return file name of output.

Usage

```
writeR(out_dir = NULL, res = NULL, compress = T, extLookUp = NULL)
```

Arguments

Argument	Description
`out_dir`	Directory where output is written.
`res`	Crawled page data.
`compress`	If true, text/html is compressed.
`type`	content-type of crawled page.
`out_dir`	Directory where output is written.
`type`	content-type of crawled page.
`res`	Crawled page data.

Value

Returns filename of output.

Returns filename of output.