Wayback Textmining

Martin Ratzmann

2026-02-16

Überblick

`crawl_wayback_site_to_files(domain, year, ...)` lädt einen **Wayback-Machine-Snapshot** (Internet Archive) der **Homepage** einer Domain für ein bestimmtes Jahr (möglichst nahe **01.07.**) und crawlt von dort aus **interne Inhalte** per **Breadth-First Search (BFS)** bis zu einer maximalen Tiefe.

Dabei werden:

- nur **interne Links** (optional inkl. Subdomains) verfolgt,
- nur **„Content-URLs“** akzeptiert (kein Login/Suche/Tag/Newsletter, keine Medien-/Download-Dateien wie PDF/JPG/ZIP),
- URLs **kanonisiert** (Tracking-Parameter und Fragmente entfernt), um Duplikate und Schleifen zu vermeiden,
- pro Seite **bereinigter Text** extrahiert (Navigation/Footer/Overlays/Cookie-Banner werden entfernt),
- Ergebnisse in **zwei Dateien** geschrieben:
  - `ID_YYYY.txt` → reiner Text, Seite an Seite (getrennt durch Leerzeilen), **ohne URL-Header**
  - `ID_YYYY.csv` → Request-Log (`original_url`, `canonical_url`, `wayback_id_url`, `depth`, `http_status`, `error`)

Voraussetzungen

Benötigt wird eine lokal installierte Version von R (getestet mit Version 4.4.3). Die erforderlichen Pakete werden bei Bedarf per `install_if_missing()` automatisch installiert: `httr2`, `jsonlite`, `xml2`, `rvest`, `stringi`, `readr`, `tcltk`.

> Hinweis: In diesem Dokument werden Codebeispiele gezeigt. Damit sie laufen, muss der vollständige Funktionscode zuvor in der Session geladen sein (z. B. durch `source("wayback_crawler.R")` oder Copy & Paste).

Basisnutzung

Die Funktion erzeugt pro Abfrage zwei Dateien, die in einem lokalen Ordner gespeichert werden. Die Position dieses Ordners wird zuvor festgelegt, z. B.:

```
out_dir <- "/Users/DEINNAME/Downloads/Mining/"
```

Der Aufruf `crawl_wayback_site_to_files("www.example.com", 2020, id = "EX01", out_dir = out_dir)` erzeugt die Dateien `EX01_2020.txt` und `EX01_2020.csv` im Verzeichnis: `/Users/DEINNAME/Downloads/Mining/`. Wenn `id` nicht gesetzt ist, wird stattdessen die `Domain` als Basisname verwendet.

Abfrage-Variationen

Eine umfangreichere Abfrage mit bis zu vier Ebenen der Homepage und bis zu 800 Seiten erfolgt mit der Anweisung:

```
crawl_wayback_site_to_files("example.com", 2020, id="EX01", max_depth=4, max_pages=800,
out_dir=out_dir)
```

In der Regel schneller, aber weniger vollständig, ist die Abfrage ohne Unterseiten:

```
crawl_wayback_site_to_files("example.com", 2020, include_subdomains=FALSE)
```

Es kann auch festgelegt werden, dass die Abfrage abbricht, sobald eine angegebene Anzahl von extrahierten Zeichen erreicht ist:

```
crawl_wayback_site_to_files("example.com", 2020, max_text_chars=100000)
```

Weil häufige Anfragen innerhalb kurzer Zeit zu Sperrungen oder Zurückweisung führen, können Pausen zwischen einzelnen Anfragen eingerichtet werden.

```
crawl_wayback_site_to_files("example.com", 2020,  polite_sleep=1.5, long_break_every=15,
long_break_range=c(10, 30))
```

Parameter von `crawl_wayback_site_to_files()`

```
crawl_wayback_site_to_files(domain, year,

    id = NULL,
    max_depth = 3,
    max_pages = 250,
    max_text_chars = 500000,
    include_subdomains = TRUE,
    timeout_s = 120,
    out_dir = ".",
    polite_sleep = 0.8,
    long_break_every = 25,
    long_break_range = c(5, 15)

)
```

`domain`

Bedeutung: Die zu crawlende Domain (Startpunkt ist die Homepage).

Beispiele: `"www.bosch.de"`, `"bosch.de"`, `"https://bosch.de/"`

Hinweis: Im Memento-Finder werden Varianten mit/ ohne `www` sowie `http/https` ausprobiert.

`year`

Bedeutung: Jahr, für das ein Wayback-Snapshot gesucht wird.

Wirkung: Es wird via CDX-API ein Snapshot im Zeitfenster `YYYY-01-01` bis `YYYY-12-31` gesucht, bevorzugt nahe 01.07. (`prefer_date = YYYY0701`).

Typisch: Ein Jahr, in dem die Seite aktiv war.

`id = NULL`

Bedeutung: Kennung für die Ausgabedateien.

Wirkung: Wenn gesetzt (nicht leer), wird sie im Dateinamen genutzt: `ID_YEAR.txt/csv` Wenn nicht gesetzt, wird `domain` als Ersatz verwendet.

Praktisch: Für stabile, kurze Dateinamen bei vielen Läufen (z. B. Unternehmens-ID).

`max_depth = 3`

Bedeutung: Maximale Link-Tiefe ab Startseite (BFS-Ebenen).

Wirkung:

- Tiefe 0 = Startseite
- Tiefe 1 = Links direkt von Startseite
- Tiefe 2/3 entsprechend weiter

Trade-off: Höhere Tiefe → mehr Abdeckung, aber deutlich mehr Seiten/Requests.

`max_pages = 250`

Bedeutung: Maximale Anzahl erfolgreich/versucht verarbeiteter Seiten (Requests), unabhängig von Tiefe.

Wirkung: Stoppt den Crawl, sobald `pages_used` diesen Wert erreicht.

Nutzen: Harte Obergrenze gegen „explodierende“ Websites.

`max_text_chars = 500000`

Bedeutung: Obergrenze für die gesammelte Textmenge (Zeichen) im Output-Textfile.

Wirkung: Sobald das Limit überschritten würde:

- wird der letzte Text ggf. passend abgeschnitten,
- dann wird abgebrochen (`[STOP] max_text_chars reached`).

Setzen auf `NULL/Inf`: würde praktisch „kein Textlimit“ bedeuten (im Code wird auf `is.finite()` geprüft).

`include_subdomains = TRUE`

Bedeutung: Ob Subdomains als „intern“ gelten.

Wirkung: `TRUE`: `blog.example.com` zählt als intern, wenn Seed-Host `example.com` ist

`FALSE`: nur exakt derselbe Host wird akzeptiert

Wichtig: Gerade bei großen Websites liegen Inhalte oft auf Subdomains.

`timeout_s = 120`

Bedeutung: Netzwerk-Timeout pro Request (Sekunden).

Wirkung: Wird sowohl bei der CDX-Abfrage (Snapshot finden) als auch beim Seiten-Fetch genutzt (`req_timeout(timeout_s)`).

Tipp: Bei langsamen Antworten/Retry-Ketten ggf. erhöhen.

`out_dir = "."`

Bedeutung: Ausgabeordner für Text- und Logdatei.

Wirkung:

- `"."` = aktuelles Working Directory (getwd())
- sollte existieren; andernfalls schlägt das Schreiben fehl.

Tipp: Pfad explizit setzen, wenn du reproduzierbare Ablagen willst.

`polite_sleep = 0.8`

Bedeutung: „Höfliche“ Wartezeit zwischen Requests (Sekunden, mit Zufallsstreuung).

Wirkung: Nach jeder Seite:

- normal: `runif(1, polite_sleep*0.7, polite_sleep*1.3)`
- bei HTTP 429/503/504: stattdessen `runif(1, 15, 45)` (deutlich längere Pause)

Ziel: Rate-Limiting vermeiden und das Archiv nicht überlasten.

`long_break_every = 25`

Bedeutung: Nach wie vielen Seiten zusätzlich eine längere Pause gemacht wird.

Wirkung: Wenn pages_used %% long_break_every == 0, dann Long-Break.

`long_break_range = c(5, 15)`

Bedeutung: Intervall (Sekunden) für die „Long Break“-Pause.

Wirkung: `runif(1, long_break_range[1], long_break_range[2])`

Tuning: Größere Werte → weniger Risiko für 429/503, aber langsamer.

Output-Dateien

Die Textdatei `ID_YYYY.txt` enthält ausschließlich den extrahierten Text, Seite für Seite. Die Seiten sind durch Leerzeilen getrennt und URLs werden nicht in den Text geschrieben. Wenn hier weiterhin unerwünschte Textfragmente auftreten (z.B. Browser- oder Cookie-Hinweise, Banner von `wayback machine`, `javascript`-Anweisungen), können solche Fragmente in zukünftigen Abfragen gezielt herausgefiltert werden, indem sie im Code in `drop_patterns` ergänzt werden:

```
drop_patterns <- c(

"diese seite teilen",
"mehr Links",
"Close share layer",
"downloads",
"bitte benutzen sie einen anderen browser",
"browser, der nicht vollst[aä]ndig unterst[uü]tzt wird",
"darstellung und bedienbarkeit.*eingeschr[aä]nkt",
"zur optimalen nutzbarkeit empfehlen wir",
"download eines unterst[uü]tzten browsers",
"\\b(internet explorer|chrome|firefox|safari)\\b",
"\\bmit dem aktuellen browser fortfahren\\b",
"^kontakt\\b",
"\\bdeutschland\\s*\\|\\s*deutsch\\b",
"^suchen\\b",
"region wechseln"

)
```

Die Logdatei `ID_YYYY.csv` enthält pro Anfrage eine Zeile mit:

- die `original_url` (die gecrawlte URL)
- die `canonical_url` (nach Tracking-Bereinigung/Normalisierung)
- die `wayback_id_url` (konkrete Wayback-ID-URL mit Timestamp)
- die `depth` (BFS-Tiefe)
- der `http_status` (die Antwort der Abfrage)
- und `error` (Fehlermeldung, falls Fetch/Parse scheitert)

Funktionscode

```
install_if_missing <- function(pkgs) {
  missing <- pkgs[!vapply(pkgs, requireNamespace, logical(1), quietly = TRUE)]
  if (length(missing)) install.packages(missing)
}

install_if_missing(c("httr2", "jsonlite", "xml2", "rvest", "stringi","readr", "tcltk"))
library(httr2)
library(jsonlite)
library(xml2)
library(rvest)
library(stringi)
library(readr)
library(tcltk)

UA <- "R wayback text mining (polite; contact=you@example)"

#### Version ohne Übersetzung

# Funktion für den Wayback-Crawler (1 Domain + 1 Jahr):
# - 1 Snapshot (nahe 01.07.) holen
# - interne Links bis Tiefe 3 (BFS)
# - nur "Content-URLs" (kein Search/Tag/Login, keine Medien/PDF etc.)
# - URL-Kanonisierung (Tracking-Parameter raus) => keine Kreise
# - doppelte Seiten nur einmal
# - Textfile OHNE URL/Depth-Header (nur Text, Seite für Seite getrennt)
# - zweites File: Request-Log mit original_url, canonical_url, wayback_id_url, depth,
#   status, error
# ----------------------------
# Snapshot (memento) pro Jahr finden (Homepage exakt)
# ----------------------------
get_wayback_memento_for_year <- function(domain, year,
                                         timeout_s = 120,
                                         max_tries = 6,
                                         prefer_date = sprintf("%d0701", year)) {
  domain <- sub("^https?://", "", domain)
  domain <- sub("/+$", "", domain)
  base <- sub("^www\\.", "", domain)

  variants <- unique(c(
    paste0("https://", base, "/"),
    paste0("http://",  base, "/"),
    paste0("https://www.", base, "/"),
    paste0("http://www.",  base, "/")
  ))

  from <- sprintf("%d0101", year)
  to   <- sprintf("%d1231", year)
  closest_ts <- paste0(prefer_date, "000000")

  fetch_one <- function(u, filter_200 = TRUE) {
    q <- list(
```

```
    url = u,
    matchType = "exact",
    from = from,
    to = to,
    output = "json",
    fl = "timestamp,original",
    closest = closest_ts,
    limit = "1"
  )
  if (isTRUE(filter_200)) q$filter <- "statuscode:200"

  message(sprintf("[CDX] query url=%s  filter200=%s", u, filter_200))

  r <- request("https://web.archive.org/cdx/search/cdx") |>
    req_url_query(!!!q) |>
    req_user_agent(UA) |>
    req_timeout(timeout_s) |>
    req_retry(max_tries = max_tries, backoff = function(i) min(2^(i - 1), 30)) |>
    req_perform()

  st <- resp_status(r)
  message(sprintf("[CDX] http_status=%s", st))

  body <- resp_body_string(r)

  dat <- jsonlite::fromJSON(
    body,
    simplifyVector = TRUE,
    simplifyDataFrame = TRUE,
    simplifyMatrix = TRUE
  )

  # Robust gegen "list of lists"
  if (is.list(dat) && !is.data.frame(dat) && is.null(dim(dat))) {
    dat <- do.call(rbind, dat)
  }

  if (is.null(dim(dat)) || nrow(dat) <= 1) {
    message("[CDX] no rows")
    return(NULL)
  }

  dat <- dat[-1, , drop = FALSE]
  ts   <- as.character(dat[1, 1])
  orig <- as.character(dat[1, 2])

  message(sprintf("[CDX] found ts=%s  original=%s", ts, orig))

  list(
    ts = ts,
    original = orig,
    wayback = paste0("https://web.archive.org/web/", ts, "/", orig)
  )
```

```
  }

  message(sprintf("[CDX] searching memento for domain=%s year=%d (closest=%s)",
                  domain, year, closest_ts))

  for (u in variants) {
    m <- tryCatch(fetch_one(u, TRUE),  error = function(e) {
      message(sprintf("[CDX] error for %s: %s", u, conditionMessage(e)))
      NULL
    })
    if (!is.null(m)) return(m)
  }
  for (u in variants) {
    m <- tryCatch(fetch_one(u, FALSE), error = function(e) {
      message(sprintf("[CDX] error for %s: %s", u, conditionMessage(e)))
      NULL
    })
    if (!is.null(m)) return(m)
  }

  message("[CDX] no memento found for any variant")
  NULL
}

# ----------------------------
# URL canonicalization + Content filter
# ----------------------------
strip_fragment <- function(u) sub("#.*$", "", u)

host_of <- function(u) {
  m <- stringi::stri_match_first_regex(u, "^https?://([^/]+)")
  if (is.na(m[1,2])) NA_character_ else tolower(m[1,2])
}

is_same_site <- function(u, seed_host, include_subdomains = TRUE) {
  h <- host_of(u)
  if (is.na(h) || is.na(seed_host)) return(FALSE)
  if (h == seed_host) return(TRUE)
  if (include_subdomains && endsWith(h, paste0(".", seed_host))) return(TRUE)
  FALSE
}

# Entfernt typische Tracking-Parameter + sortiert Query deterministisch
canonicalize_url <- function(u) {
  u <- strip_fragment(u)
  if (!is.character(u) || length(u) != 1L || is.na(u) || !nzchar(u)) return("")
  if (!stringi::stri_detect_regex(u, "^https?://")) return(u)  # nur absolute URLs

  # Query string roh aus URL ziehen (ohne httr2 query-build)
  q_raw <- ""
  if (grepl("\\?", u)) q_raw <- sub("^[^?]*\\?", "", u)

  # scheme/host/path robust über httr2 parsen (ohne später url_build zu nutzen)
```

```r
  p <- httr2::url_parse(u)
  scheme <- tolower(p$scheme %||% "https")
  host   <- tolower(p$hostname %||% "")
  port   <- p$port %||% ""

  path <- p$path %||% "/"
  path <- stringi::stri_replace_all_regex(path, "/{2,}", "/")
  if (nzchar(path) && path != "/") path <- sub("/+$", "", path)
  if (!nzchar(path)) path <- "/"

  # Tracking-Parameter entfernen + deterministisch sortieren
  q_out <- ""
  if (nzchar(q_raw)) {
    kv <- strsplit(q_raw, "&", fixed = TRUE)[[1]]
    kv <- kv[nzchar(kv)]

    if (length(kv)) {
      keys <- sub("=.*$", "", kv)
      has_eq <- grepl("=", kv, fixed = TRUE)
      vals <- ifelse(has_eq, sub("^[^=]*=", "", kv), "")

      drop_keys <- c(
        "utm_source","utm_medium","utm_campaign","utm_term","utm_content",
        "gclid","dclid","fbclid","yclid","msclkid",
        "ref","referrer","cmp","campaign","src","source","mkt_tok",
        "session","sid","phpsessid","jsessionid"
      )

      keep <- !(tolower(keys) %in% drop_keys) & !stringi::stri_detect_regex(tolower(keys),
                                                                            "^utm_")
      keys <- keys[keep]
      vals <- vals[keep]

      if (length(keys)) {
        ord <- order(tolower(keys), vals)
        keys <- keys[ord]; vals <- vals[ord]
        q_new <- paste0(keys, ifelse(vals == "", "", paste0("=", vals)))
        q_out <- paste(q_new, collapse = "&")
      }
    }
  }

  # Optional: scheme vereinheitlichen (reduziert Duplikate http/https)
  # scheme <- "https"

  base <- paste0(scheme, "://", host)
  if (nzchar(port)) base <- paste0(base, ":", port)
  out <- paste0(base, path)
  if (nzchar(q_out)) out <- paste0(out, "?", q_out)
  out
}

`%||%` <- function(a, b) if (is.null(a) || length(a) == 0) b else a
```

```r
# Nur "Content-URLs": keine Suche/Tags/Login/Consent, keine Medien/Downloads
is_content_url <- function(u) {
  if (!nzchar(u)) return(FALSE)

  low <- tolower(u)

  # harte Ausschlüsse (Pfad/Keywords)
  bad_path_patterns <- c(
    "/search", "/suche", "/tag", "/tags", "/topic", "/thema", "/topics",
    "/login", "/logout", "/signin", "/signup", "/register",
    "/account", "/profil", "/profile", "/my-", "/user",
    "/cookie", "/consent", "/privacy", "/datenschutz",
    "/impressum", "/kontakt", "/contact",
    "/newsletter", "/subscribe", "/abo",
    "/sitemap", "/rss", "/feed"
  )

  if (any(vapply(bad_path_patterns, function(p) grepl(p, low, fixed = TRUE),
                 logical(1)))) return(FALSE)

  # Medien/Dateien ausschließen
  if (stringi::stri_detect_regex(
    low,
    "\\.(pdf|doc|docx|xls|xlsx|ppt|pptx|zip|rar|7z|mp3|
        wav|mp4|webm|mov|avi|png|jpe?g|gif|svg)(\\?|$)",
    opts_regex = stringi::stri_opts_regex(case_insensitive = TRUE)
  )) return(FALSE)

  TRUE
}

as_wayback_id_url <- function(original_url, ts) {
  paste0("https://web.archive.org/web/", ts, "id_/", original_url)
}

# ----------------------------
# Text extraction helpers
# ----------------------------
fix_missing_spaces <- function(x) {
  x <- stringi::stri_replace_all_regex(x, "([\\p{Ll}])([\\p{Lu}])", "$1 $2")
  x <- stringi::stri_replace_all_regex(x, "([0-9])([\\p{L}])", "$1 $2")
  x <- stringi::stri_replace_all_regex(x, "([\\p{L}])([0-9])", "$1 $2")
  x <- stringi::stri_replace_all_regex(x, "\\s+", " ")
  trimws(x)
}

# ----------------------------
# Fetch + Parse: Text + interne Links
# ----------------------------
fetch_text_and_links <- function(original_url, ts,
timeout_s = 120,
seed_host = NULL,
include_subdomains = TRUE) {
```

```
original_url <- canonicalize_url(original_url)
fetch_url <- as_wayback_id_url(original_url, ts)

message(sprintf("[FETCH] %s", fetch_url))

resp <- request(fetch_url) |>
req_user_agent(UA) |>
req_timeout(timeout_s) |>
req_retry(max_tries = 5, backoff = function(i) min(2^(i - 1), 30)) |>
req_perform()

status <- resp_status(resp)
message(sprintf("[FETCH] status=%s", status))

doc <- xml2::read_html(resp_body_string(resp))

  # Entferner
  kill_selectors <- paste(c(
    "#wm-ipp", "#wm-ipp-base",
    "script", "style", "noscript", "iframe",
    "header", "nav", "footer", "aside",
    "[role='navigation']", "[role='banner']", "[role='contentinfo']",
    "#onetrust-banner-sdk", ".onetrust-pc-dark-filter", ".ot-sdk-container",
    "[id*='cookie']", "[class*='cookie']",
    "[id*='consent']", "[class*='consent']",
    "[class*='overlay']", "[class*='modal']", "[class*='popup']",
    "[aria-modal='true']",
    "[class*='search']", "[id*='search']",
    "[class*='region']", "[id*='region']"
  ), collapse = ", ")
  nodes <- rvest::html_elements(doc, kill_selectors)
  if (length(nodes) > 0) xml2::xml_remove(nodes)

  # Hauptinhalt
  main_selectors <- c("main", "article", "[role='main']", "#content", "#main", "#page",
                      "#app", ".content", ".main", ".page")
  main_nodes <- rvest::html_elements(doc, paste(main_selectors, collapse = ", "))

  pick_text_richest <- function(nodes) {
    if (length(nodes) == 0) return(NULL)
    txts <- vapply(nodes, function(n) paste(rvest::html_text(n), collapse = " "),
                   character(1))
    nodes[[which.max(stringi::stri_length(txts))]]
  }
  main_node <- pick_text_richest(main_nodes)

  text_from_node <- function(node) {
    tn <- xml2::xml_find_all(node, ".//text()")
    if (length(tn) == 0) return("")
    x <- paste(xml2::xml_text(tn), collapse = " ")
    x <- stringi::stri_replace_all_regex(x, "\\s+", " ")
    trimws(x)
  }
```

```
raw <- if (!is.null(main_node)) {
  text_from_node(main_node)
} else {
  body <- rvest::html_elements(doc, "body")
  if (length(body)) text_from_node(body[[1]]) else ""
}

# Zeilenweise filtern
txt <- stringi::stri_replace_all_regex(raw, "\\r\\n|\\r", "\n")
txt <- stringi::stri_replace_all_regex(txt, "[ \t]+", " ")
lines <- unlist(strsplit(txt, "\n", fixed = TRUE), use.names = FALSE)
lines <- trimws(lines)
lines <- lines[nzchar(lines)]

drop_patterns <- c(
  "diese seite teilen",
  "mehr Links",
  "Close share layer",
  "downloads",
  "bitte benutzen sie einen anderen browser",
  "browser, der nicht vollst[aä]ndig unterst[uü]tzt wird",
  "darstellung und bedienbarkeit.*eingeschr[aä]nkt",
  "zur optimalen nutzbarkeit empfehlen wir",
  "download eines unterst[uü]tzten browsers",
  "\\b(internet explorer|chrome|firefox|safari)\\b",
  "\\bmit dem aktuellen browser fortfahren\\b",
  "^kontakt\\b",
  "\\bdeutschland\\s*\\|\\s*deutsch\\b",
  "^suchen\\b",
  "region wechseln"
)

keep <- !vapply(lines, function(z) {
  any(stringi::stri_detect_regex(
    z, drop_patterns,
    opts_regex = stringi::stri_opts_regex(case_insensitive = TRUE)
  ))
}, logical(1))

lines <- lines[keep]
lines <- lines[nchar(lines) >= 4]
lines <- vapply(lines, fix_missing_spaces, character(1))

out_text <- paste(lines, collapse = "\n")
out_text <- stringi::stri_replace_all_regex(out_text, "\\n{3,}", "\n\n")
out_text <- trimws(out_text)

# Links extrahieren (absolut auf Original-URL, dann kanonisieren + filtern)
hrefs <- rvest::html_elements(doc, "a") |> rvest::html_attr("href")
hrefs <- hrefs[!is.na(hrefs)]
hrefs <- hrefs[!stringi::stri_detect_regex(hrefs, "^(mailto:|javascript:|tel:)")]

abs <- xml2::url_absolute(hrefs, original_url)
```

```
  abs <- strip_fragment(abs)
  abs <- unique(abs)

  if (is.null(seed_host)) seed_host <- host_of(original_url)

  # intern + content-only + canonicalize
  abs <- abs[vapply(abs, is_same_site, logical(1), seed_host = seed_host,
                    include_subdomains = include_subdomains)]
  abs <- abs[vapply(abs, is_content_url, logical(1))]
  abs <- vapply(abs, canonicalize_url, character(1))
  abs <- unique(abs)

message(sprintf("[PARSE] text_chars=%d links=%d",
nchar(out_text, type = "chars"),
length(abs)))

list(
canonical_url = original_url,
text = out_text,
links = abs,
fetch_url = fetch_url,
status = status
)
}

# ----------------------------
# Crawl BFS + Dateien
# ----------------------------
crawl_wayback_site_to_files <- function(domain, year,
                                        id = NULL,
                                        max_depth = 3,
                                        max_pages = 250,
                                        max_text_chars = 500000,
                                        include_subdomains = TRUE,
                                        timeout_s = 120,
                                        out_dir = ".",
                                        polite_sleep = 0.8,
                                        long_break_every = 25,
                                        long_break_range = c(5, 15)) {

  m <- get_wayback_memento_for_year(domain, year, timeout_s = timeout_s)
  if (is.null(m)) stop("Kein Wayback-Snapshot für die Homepage in ", year, " gefunden: ",
                       domain)

  ts <- m$ts
  start_url <- canonicalize_url(m$original)
  seed_host <- host_of(start_url)

  # BFS Queue (canonical URLs)
  q_url <- c(start_url)
  q_depth <- c(0L)

  visited <- new.env(parent = emptyenv())  # key = canonical_url
```

```
pages_used <- 0L

texts_out <- character()
total_chars <- 0L
log_rows <- list()

while (length(q_url) > 0 && pages_used < max_pages) {
  u <- q_url[[1]]
  d <- q_depth[[1]]
  q_url <- q_url[-1]
  q_depth <- q_depth[-1]

  u <- canonicalize_url(u)
  if (!nzchar(u)) next
  if (exists(u, envir = visited, inherits = FALSE)) next
  assign(u, TRUE, envir = visited)

  pages_used <- pages_used + 1L

  res <- tryCatch(
    fetch_text_and_links(u, ts, timeout_s = timeout_s, seed_host = seed_host,
                         include_subdomains = include_subdomains),
    error = function(e) list(
      canonical_url = u, text = "", links = character(),
      fetch_url = as_wayback_id_url(u, ts),
      status = NA_integer_, error = conditionMessage(e)
    )
  )

  # Text sammeln (OHNE URL/Depth im Textfile) - nur Trennlinie zwischen Seiten
    if (!is.null(res$text) && nzchar(res$text)) {

    sep <- "\n\n"
    add_chars <- nchar(res$text, type = "chars") + nchar(sep, type = "chars")

    # Wenn Limit erreicht/überschritten:
    # letzten Text passend abschneiden und dann abbrechen
    if (!is.null(max_text_chars) && is.finite(max_text_chars) &&
        (total_chars + add_chars) > max_text_chars) {

      remaining <- max_text_chars - total_chars
      if (remaining > 0) {
        # Platz für Separator berücksichtigen
        rem_for_text <- max(0, remaining - nchar(sep, type = "chars"))
        if (rem_for_text > 0) {
          texts_out <- c(texts_out, substr(res$text, 1, rem_for_text), sep)
          total_chars <- total_chars + rem_for_text + nchar(sep, type = "chars")
        }
      }

      message(sprintf("[STOP] max_text_chars reached: %d", total_chars))
      break
    }
```

```
      texts_out <- c(texts_out, res$text, sep)
      total_chars <- total_chars + add_chars
    }

    # Log (enthält URLs/Depth/Fetch)
    log_rows[[length(log_rows) + 1L]] <- data.frame(
      original_url = u,
      canonical_url = res$canonical_url %||% u,
      wayback_id_url = res$fetch_url %||% as_wayback_id_url(u, ts),
      depth = d,
      http_status = res$status %||% NA_integer_,
      error = res$error %||% "",
      stringsAsFactors = FALSE
    )

    # Links enqueue (nur neue canonical URLs)
    if (d < max_depth && !is.null(res$links) && length(res$links) > 0) {
      new_links <- vapply(res$links, canonicalize_url, character(1))
      new_links <- new_links[vapply(new_links, is_same_site, logical(1),
                                    seed_host = seed_host,
                                    include_subdomains = include_subdomains)]
      new_links <- new_links[vapply(new_links, is_content_url, logical(1))]
      new_links <- unique(new_links)

      # nur unvisited
      new_links <- new_links[!vapply(new_links, function(x) exists(x, envir = visited,
                                                  inherits = FALSE), logical(1))]

      if (length(new_links) > 0) {
        q_url <- c(q_url, new_links)
        q_depth <- c(q_depth, rep.int(d + 1L, length(new_links)))
      }
    }

    # Polite delays
    if (!is.na(res$status) && res$status %in% c(429, 503, 504)) {
      Sys.sleep(runif(1, 15, 45))
    } else {
      Sys.sleep(runif(1, polite_sleep * 0.7, polite_sleep * 1.3))
    }
    if (pages_used %% long_break_every == 0) {
      Sys.sleep(runif(1, long_break_range[1], long_break_range[2]))
    }
  }

# Dateinamen sollen ID enthalten (fallback: domain, falls keine ID übergeben wurde)
id_used <- if (!is.null(id) && !is.na(id) && nzchar(trimws(as.character(id)))) {
  trimws(format(id, scientific = FALSE, trim = TRUE))
} else {
  domain
}
safe_id <- gsub("[^A-Za-z0-9._-]+", "_", id_used)
```

```
#text_file <- file.path(out_dir, sprintf("%s_%d_%s_texts.txt", safe_id, year, ts))
#log_file  <- file.path(out_dir, sprintf("%s_%d_%s_requests.csv", safe_id, year, ts))

# kurze Bezeichnung der Dateien: ID_year.txt und ID_year.csv
text_file <- file.path(out_dir, sprintf("%s_%d.txt", safe_id, year))
log_file  <- file.path(out_dir, sprintf("%s_%d.csv", safe_id, year))

  writeLines(texts_out, text_file, useBytes = TRUE)
  log_df <- do.call(rbind, log_rows)
  write.csv(log_df, log_file, row.names = FALSE, fileEncoding = "UTF-8")

  message("Fertig.\nText: ", normalizePath(text_file, winslash = "/"),
          "\nLog:  ", normalizePath(log_file, winslash = "/"),
          "\nPages used: ", pages_used, " (max_depth=", max_depth, ")")

  invisible(list(text_file = text_file, log_file = log_file,
                 pages_used = pages_used, ts = ts, start_url = start_url))
}
```

Test für eine Seite

Die komplette Routine wird durch **crawl_wayback_site_to_files ()** aufgerufen. Die erforderlichen Parameter für die Abfrage sind: * die URL der Webseite (domain =), * das relevante Jahr (year =), * die ID für die spätere Zuordnung (id =).

als Abbruchkriterien können die folgenden Parameter festegelegt werden:

- die max. Tiefe der untergeordneten Seiten (max_depth = 3),
- die max. Anzahl der Seiten (max_pages = 250)
- die max. Menge des extrahierten Textes in Zeichen (max_text_chars = 500000)

Der Parameter `include_subdomains = TRUE` schließt Unterseiten der Homepage in die Abfrage mit ein und mit `timeout_s = 120` wird festgelegt, dass die Abfrage abgebrochen wird, wenn nach 120 Sekunden keine Antwort empfangen wurde.

Die resultierenden Datein werden im Verzeichnis `out_dir = "."` gespeichert und zwischen einzelnen Abfragen wird eine Pause von `polite_sleep = 0.8` eingelegt.

```
crawl_wayback_site_to_files <- function(domain, year,

id = NULL,

max_depth = 3,

max_pages = 250,

max_text_chars = 500000,

include_subdomains = TRUE,

timeout_s = 120,

out_dir = ".",

polite_sleep = 0.8,

long_break_every = 25,

long_break_range = c(5, 15))
```

Beispiel-Aufruf

```
out_dir = "~/Bookdown/Textmining/Wayback/Mining"
crawl_wayback_site_to_files("www.twitter.com", 2008, "ID666",
                            max_depth = 5, max_pages = 50, out_dir = out_dir)
```

Ausgabe in der Konsole

```
[CDX] searching memento for domain=www.twitter.com year=2008 (closest=20080701000000)
[CDX] query url=https://twitter.com/  filter200=TRUE
[CDX] http_status=200
[CDX] found ts=20080101061456  original=http://twitter.com:80/?
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/
[FETCH] status=200
[PARSE] text_chars=458 links=6
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/why
[FETCH] status=200
[PARSE] text_chars=458 links=6
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/how
[FETCH] status=200
[PARSE] text_chars=461 links=6
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/api
[FETCH] status=200
[PARSE] text_chars=564 links=5
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/jobs
[FETCH] status=200
[PARSE] text_chars=2284 links=4
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/tos
[FETCH] status=200
[PARSE] text_chars=176 links=7
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/celly
[FETCH] status=200
[PARSE] text_chars=2106 links=11
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/TOS/friends
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/TOS/favorites
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/TOS
[FETCH] status=200
[PARSE] text_chars=176 links=7
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly/friends
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly/
  favorites
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly
[FETCH] status=200
[PARSE] text_chars=2106 links=10
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml
[FETCH] status=200
[PARSE] text_chars=2654 links=14
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brian
[FETCH] status=200
[PARSE] text_chars=1209 links=7
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly?page=2
```

```
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml/
  friends
[FETCH] status=200
[PARSE] text_chars=153 links=5
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml/
  favorites
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula
[FETCH] status=200
[PARSE] text_chars=2866 links=15
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/blankbaby
[FETCH] status=200
[PARSE] text_chars=0 links=18
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brianoberkirch
[FETCH] status=200
[PARSE] text_chars=2278 links=10
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/factoryjoe
[FETCH] status=200
[PARSE] text_chars=2946 links=13
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/davespeers
[FETCH] status=200
[PARSE] text_chars=1868 links=12
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/joshualane
[FETCH] status=200
[PARSE] text_chars=2829 links=16
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml?
  page=2
[FETCH] status=200
[PARSE] text_chars=2749 links=20
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brian/
  friends
[FETCH] status=200
[PARSE] text_chars=153 links=5
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brian/
  favorites
[FETCH] status=200
[PARSE] text_chars=114 links=8
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula/
  friends
[FETCH] status=200
[PARSE] text_chars=153 links=5
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula/
  favorites
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/megnificent
[FETCH] status=200
[PARSE] text_chars=37 links=7
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/sarahgilbert
[FETCH] status=200
[PARSE] text_chars=3013 links=9
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/ringmaster
[FETCH] status=200
[PARSE] text_chars=2566 links=15
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/natmechanics
[FETCH] status=200
```

```
[PARSE] text_chars=2877 links=17
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/swirlspice
[FETCH] status=200
[PARSE] text_chars=2351 links=18
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/stellargirl
[FETCH] status=200
[PARSE] text_chars=37 links=8
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula?
  page=2
[FETCH] status=200
[PARSE] text_chars=2562 links=22
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/blankbaby/
  friends
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/blankbaby/
  favorites
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/friends/index/
  41693
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/zorn
[FETCH] status=200
[PARSE] text_chars=2593 links=11
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Fishbreakfast
[FETCH] status=200
[PARSE] text_chars=2267 links=10
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/chockenberry
[FETCH] status=200
[PARSE] text_chars=0 links=18
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/gruber
[FETCH] status=200
[PARSE] text_chars=2812 links=13
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/jsnell
[FETCH] status=200
[PARSE] text_chars=2233 links=18
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/sweetums
[FETCH] status=200
[PARSE] text_chars=2352 links=11
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/siracusa
[FETCH] status=200
[PARSE] text_chars=2723 links=17
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/ejacqui
[FETCH] status=200
[PARSE] text_chars=2198 links=14
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/blankbaby?
  page=2
[FETCH] status=200
[PARSE] text_chars=2371 links=19
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/
  brianoberkirch/friends
[FETCH] status=200
[PARSE] text_chars=153 links=5
[FETCH] https://web.archive.org/web/20080101061456id_/http://twitter.com:80/
  brianoberkirch/favorites
[FETCH] status=200
[PARSE] text_chars=2940 links=11
```

```
Fertig.
Text: /Users/bwl6-mr/Bookdown/Textmining/Wayback/Mining/ID666_2008.txt
Log:  /Users/bwl6-mr/Bookdown/Textmining/Wayback/Mining/ID666_2008.csv
Pages used: 50 (max_depth=5)
```

Ausgabe in der Text-Datei

What is Twitter? What? Why? How? Twitter is a service for friends, family, and co-workers to communicate and stay connected through the exchange of quick, frequent answers to one simple question: What are you doing? Get Started-Join! Oriana from ccs... wants you to join Twitter Once you join, you'll be connected to oriana on Twitter. Please Sign In username password Remember me Forgot password? Click here . Already using Twitter by SMS or IM? Click here.

Why use Twitter? What? Why? How? Why? Because even basic updates are meaningful to family members, friends, or colleagues-especially when they're timely. Eating soup? Research shows that moms want to know. Running late to a meeting? Your co-workers might find that useful. Partying? Your friends may want to join you. Get Started-Join! Please Sign In username password Remember me Forgot password? Click here . Already using Twitter by SMS or IM? Click here.

How does it work? What? Why? How? With Twitter, you can stay hyper-connected to your friends and always know what they're doing. Or, you can stop following them any time. You can even set quiet times on Twitter so you're not interrupted. Twitter puts you in control and becomes a modern antidote to information overload. Get Started-Join! Please Sign In username password Remember me Forgot password? Click here . Already using Twitter by SMS or IM? Click here.

Twitter API The official Twitter API documentation is part of the Twitter Development Talk Google Group. Join the group, read the docs, and create neat stuff like the apps, hacks, and mashups listed on the Twitter Fan Wiki ! Flash/Action Script Developers Want to make a Twitter Flash app like Celly ? Here are recently updated versions of the Actionscript libraries for both Actionscript 2 and Actionscript 3 . You might also be interested in the SWX Twitter API for Flash and Flash Lite. This is a third-party service not affiliated with the offical Twitter API.

What are you doing? How about: "Working at Twitter!" We're looking for bright minds to help us work on the ongoing challenge of creating a compelling experience while accommodating rapid growth. Twitter is a new way for people to communicate, express themselves, and connect with one another in a lightweight, ambient fashion. Our core technology is a large scale message routing system with social functionality. Interested? Please email us at . Positions Available Senior Engineers Do you like to write code that works elegantly and efficiently, then push it out to thousands of customers the same day it's finished? Do you enjoy thinking carefully about system design and then sitting down to write a river of code which passes its own tests the first time through? Do you want to help build a fast, reliable, complex messaging application that bridges the Internet and cell phones? If so, we're looking people with these skills... BS or MS in Computer Science or equivalent experience. 5+ years of real-world software development experience. Excellent and influential communication skills with engineers and non-engineers. Extensive experience programming in both scripting and application-specific languages. Deep familiarity with Unix environments, HTTP, TCP/IP. Experience with Jabber, Ruby on Rails, and My SQL a plus. Experience practicing agile development methodologies. Open source community participation is highly respected. Strong interest in Twitter and developing a world class Internet utility. Operations Engineers Focus on consistent availability and reliability (self and systems). Excellent and influential communication skills (team, vendors, community). Experience with: Linux, OS X, Solaris, HTTPD (Apache, lighttpd, nginx), mongrel, tomcat, My SQL, pgsql, storage systems (NFS, NAS, SAN), load balancing systems (hardware, software, dns), automated system tools (cfengine). Experience maintaining: large number of servers, multi-tiered environments, monitoring and notification tools. Experience executing: well planned and tested solutions to complex problems, goal oriented workdays with multiple context changes and interruption. Must have: excellent triage skills, mild manner, rockstar inside (ready to rise to any occasion), strong interest in Twitter.

TOS Kub 3 months ago from txt With Others Previous hmmmm 02:36 AM August 28, 2007 from web my name is tos i want to test the twitter too! 02:34 AM August 28, 2007 from web RSS

Ausgabe in der Log-Datei

	A	B	C	D	E	F
1	original_url	canonical_url	wayback_id_url	depth	http_status	error
2	http://twitter.com:80/	http://twitter.com:80/	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/	0	200	
3	http://twitter.com:80/help/why	http://twitter.com:80/help/why	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/why	1	200	
4	http://twitter.com:80/help/how	http://twitter.com:80/help/how	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/how	1	200	
5	http://twitter.com:80/help/api	http://twitter.com:80/help/api	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/api	1	200	
6	http://twitter.com:80/help/jobs	http://twitter.com:80/help/jobs	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/help/jobs	1	200	
7	http://twitter.com:80/tos	http://twitter.com:80/tos	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/tos	1	200	
8	http://twitter.com:80/celly	http://twitter.com:80/celly	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/celly	2	200	
9	http://twitter.com:80/TOS/friends	http://twitter.com:80/TOS/friends	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/TOS/friends	2	NA	HTTP 404 Not Found.
10	http://twitter.com:80/TOS/favorites	http://twitter.com:80/TOS/favorites	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/TOS/favorites	2	NA	HTTP 404 Not Found.
11	http://twitter.com:80/TOS	http://twitter.com:80/TOS	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/TOS	2	200	
12	http://twitter.com:80/Celly/friends	http://twitter.com:80/Celly/friends	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly/friends	3	NA	HTTP 404 Not Found.
13	http://twitter.com:80/Celly/favorites	http://twitter.com:80/Celly/favorites	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly/favorites	3	NA	HTTP 404 Not Found.
14	http://twitter.com:80/Celly	http://twitter.com:80/Celly	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly	3	200	
15	http://twitter.com:80/alexknowshtml	http://twitter.com:80/alexknowshtml	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml	3	200	
16	http://twitter.com:80/brian	http://twitter.com:80/brian	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brian	3	200	
17	http://twitter.com:80/Celly?page=2	http://twitter.com:80/Celly?page=2	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/Celly?page=2	3	NA	HTTP 404 Not Found.
18	http://twitter.com:80/alexknowshtml/friends	http://twitter.com:80/alexknowshtml/friends	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml/friends	4	200	
19	http://twitter.com:80/alexknowshtml/favorites	http://twitter.com:80/alexknowshtml/favorites	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml/favorites	4	NA	HTTP 404 Not Found.
20	http://twitter.com:80/marusula	http://twitter.com:80/marusula	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula	4	200	
21	http://twitter.com:80/blankbaby	http://twitter.com:80/blankbaby	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/blankbaby	4	200	
22	http://twitter.com:80/brianoberkirch	http://twitter.com:80/brianoberkirch	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brianoberkirch	4	200	
23	http://twitter.com:80/factoryjoe	http://twitter.com:80/factoryjoe	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/factoryjoe	4	200	
24	http://twitter.com:80/davespeers	http://twitter.com:80/davespeers	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/davespeers	4	200	
25	http://twitter.com:80/joshualane	http://twitter.com:80/joshualane	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/joshualane	4	200	
26	http://twitter.com:80/alexknowshtml?page=2	http://twitter.com:80/alexknowshtml?page=2	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/alexknowshtml?page=2	4	200	
27	http://twitter.com:80/brian/friends	http://twitter.com:80/brian/friends	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brian/friends	4	200	
28	http://twitter.com:80/brian/favorites	http://twitter.com:80/brian/favorites	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/brian/favorites	4	200	
29	http://twitter.com:80/marusula/friends	http://twitter.com:80/marusula/friends	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula/friends	5	200	
30	http://twitter.com:80/marusula/favorites	http://twitter.com:80/marusula/favorites	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula/favorites	5	NA	HTTP 404 Not Found.
31	http://twitter.com:80/megnificent	http://twitter.com:80/megnificent	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/megnificent	5	200	
32	http://twitter.com:80/sarahgilbert	http://twitter.com:80/sarahgilbert	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/sarahgilbert	5	200	
33	http://twitter.com:80/ringmaster	http://twitter.com:80/ringmaster	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/ringmaster	5	200	
34	http://twitter.com:80/natmechanics	http://twitter.com:80/natmechanics	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/natmechanics	5	200	
35	http://twitter.com:80/swirlspice	http://twitter.com:80/swirlspice	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/swirlspice	5	200	
36	http://twitter.com:80/stellargirl	http://twitter.com:80/stellargirl	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/stellargirl	5	200	
37	http://twitter.com:80/marusula?page=2	http://twitter.com:80/marusula?page=2	https://web.archive.org/web/20080101061456id_/http://twitter.com:80/marusula?page=2	5	200	