Computational Sociology

Scraping the web

Dr. Thomas Davidson

Rutgers University

February 15, 2024

Plan

1. Ethics and computational research
2. Introduction to webscraping
3. When to use it

Ethics and computational research

New ethical questions

- Salganik discusses some examples of recent studies that raise new ethical questions
  - Emotional contagion experiment on Facebook
  - Observational study of Facebook networks
  - Browser-based study of censorship

Ethics and computational research

Four ethical principles

- *Respect for persons*
    - Treating people as autonomous and honoring their wishes
- *Beneficence*
    - Understanding risks and benefits; finding the right balance
- *Justice*
    - Even distribution of the risks and benefits
- *Respect for law and public interest*
    - Extends beyond research participants to other stakeholders
    - Compliance and transparency-based accountability

Ethics and computational research

Two ways of thinking about research ethics

- *Consequentialism*
  - Focus on the consequences of research
  - Ends
- *Deontology*
  - Consideration of ethical duties, irrespective of consequences
  - Means

Ethics and computational research

Four challenges in digital research

- *Informed consent*
  - When is it practical to get consent to participate?
  - When is it acceptable to proceed without consent?
- *Managing informational risk*
  - Risks of disclosure of personal information
  - Anonymization is often imperfect
- *Privacy*
  - What information is public or private?
  - Context-relative informational norms
- *Ethical decisions and uncertainty*
  - Minimal risk standard
  - Power analysis

What is web-scraping?

Terminology

- Web-scraping is a method to collect data from websites
    - We use the code underlying a webpage to collect data (**scraping**)
    - The process is then repeated for other pages on the same website in an automated fashion (**crawling**)

What is web-scraping?

Challenges

- Different websites have different structures, so a script used to scrape one website will likely have to be changed to scrape another
- Websites can be internally inconsistent, making them difficult to scrape
- Some websites are easier to crawl than others
- Some websites limit or prohibit scraping

When should I use it?

Commercial use cases

- Search engines
  - Google scrapes websites to create a searchable index of the internet
- Price comparison
  - Kayak scrape airlines to compare flight prices, other websites do the same for hotels and rental cars
- Recruitment
  - Recruitment companies scrape LinkedIn to get data on workers

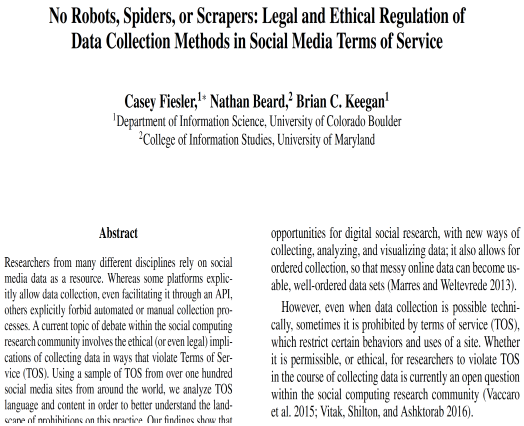
When should I use it?

Social scientific use cases

- Web-scraping is a useful tool to collect data from websites without APIs
  - Large social media platforms and other sites have APIs but smaller websites do not
    - Local newspapers, forums, small businesses, educational institutions, etc.
- Often we want to collect data from a single website
  - e.g. All posts written on a forum
- Sometimes we might want to collect data from many websites
  - e.g. All schools in a school district

When should I use it?

Ethical and legal considerations



No Robots, Spiders, or Scrapers: Legal and Ethical Regulation of Data Collection Methods in Social Media Terms of Service

Casey Fiesler,[1]* Nathan Beard,[2] Brian C. Keegan[1]
[1]Department of Information Science, University of Colorado Boulder
[2]College of Information Studies, University of Maryland

Abstract

Researchers from many different disciplines rely on social media data as a resource. Whereas some platforms explicitly allow data collection, even facilitating it through an API, others explicitly forbid automated or manual collection processes. A current topic of debate within the social computing research community involves the ethical (or even legal) implications of collecting data in ways that violate Terms of Service (TOS). Using a sample of TOS from over one hundred social media sites from around the world, we analyze TOS language and content in order to better understand the landscape of prohibitions on this practice. Our findings show that

opportunities for digital social research, with new ways of collecting, analyzing, and visualizing data; it also allows for ordered collection, so that messy online data can become usable, well-ordered data sets (Marres and Weltevrede 2013).

However, even when data collection is possible technically, sometimes it is prohibited by terms of service (TOS), which restrict certain behaviors and uses of a site. Whether it is permissible, or ethical, for researchers to violate TOS in the course of collecting data is currently an open question within the social computing research community (Vaccaro et al. 2015; Vitak, Shilton, and Ashktorab 2016).

When should I use it?

Ethical and legal considerations

- Fiesler, Beard, and Keegan (2020)s review the legal cases related to web-scraping and analyze website terms of service
    - "In short, it is an unsettled question as to whether it is explicitly illegal (or even a criminal act) to violate TOS."
    - No academic or journalist has ever been prosecuted for violating a website terms of service to collect data for research
- They analyze terms of service of over 100 social media websites
    - Terms of service are ambiguous, inconsistent, and lack context

When should I use it?

Ethical and legal considerations

- Recent legal cases have considered webscraping and ruled in favor of legality
  - *hiQ Labs, Inc. v. LinkedIn Corp.* (https://en.wikipedia.org/wiki/HiQ_Labs_v._LinkedIn)
    - Court ruled in favor of company sued for scraping LinkedIn
- Advocacy for public interest webscraping
  - https://themarkup.org/news/2020/12/03/why-web-scraping-is-vital-to-democracy

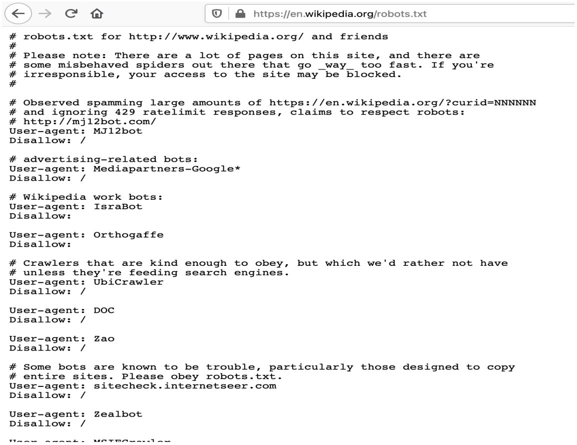
When should I use it?

Best-practices

- Only scrape publicly available data
  - i.e. You can access the page on the web without logging in
- Do not scrape copyright protected data
- Try not to violate website terms of service
- Do not burden the website
  - Limit the number of calls you make (similar to rate-limiting in APIs)
- Avoid using the data in a way that may interfere with business
  - i.e. Don't copy valuable data from a small business and share it on Github

How to scrape a web page

Start by looking up “robots.txt’ ’



```
# robots.txt for http://www.wikipedia.org/ and friends
#
# Please note: There are a lot of pages on this site, and there are
# some misbehaved spiders out there that go _way_ too fast. If you're
# irresponsible, your access to the site may be blocked.
#

# Observed spamming large amounts of https://en.wikipedia.org/?curid=NNNNNN
# and ignoring 429 ratelimit responses, claims to respect robots:
# http://mj12bot.com/
User-agent: MJ12bot
Disallow: /

# advertising-related bots:
User-agent: Mediapartners-Google*
Disallow: /

# Wikipedia work bots:
User-agent: IsraBot
Disallow:

User-agent: Orthogaffe
Disallow:

# Crawlers that are kind enough to obey, but which we'd rather not have
# unless they're feeding search engines.
User-agent: UbiCrawler
Disallow: /

User-agent: DOC
Disallow: /

User-agent: Zao
Disallow: /

# Some bots are known to be trouble, particularly those designed to copy
# entire sites. Please obey robots.txt.
User-agent: sitecheck.internetseer.com
Disallow: /

User-agent: Zealbot
Disallow: /

User-agent: MSIECrawler
```

How to scrape a web page

Decoding `robots.txt`

- **`User-agent`** = the name of the scraper
  - **`*`** = All scrapers
- **`Allow: /path/`** = OK to scrape
- **`Disallow: /path/`** = Not OK to scrape
  - **`Disallow: /`** = Not OK to scrape any pages
- **`Crawl-Delay: N`** = Wait `N` miliseconds between each call to the website

How to scrape a web page

Exercise

- Find a website of interest
- Locate the robots.txt file
  - Does the website allow webscraping?
  - Are there any restrictions on which pages can be accessed?

How to scrape a web page

Terminology

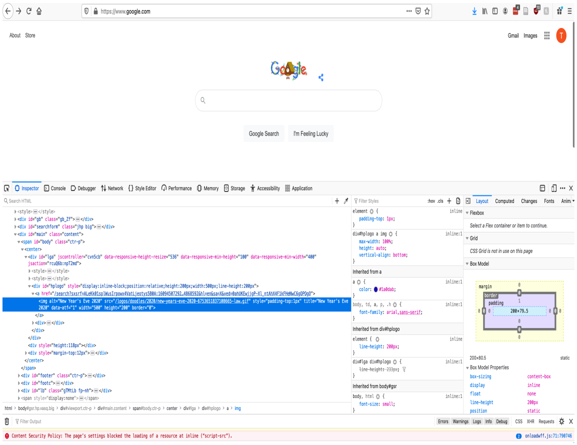
- A web-page is loaded using a **URL** (Uniform Resource Locator)
- The underlying code we are interested in is usually **HTML** (Hypertext Markup Language)
- Many websites use **CSS** (Cascading Style Sheets) to structure HTML
  - This will help us to find what we are interested in
    - See https://flukeout.github.io/ for an interactive tutorial on using CSS selectors
    - Chrome Plugin to help find CSS elements: https://selectorgadget.com/

How to scrape a web page

Inspecting HTML

- Open up a website and right click on any text or image on the screen
  - You should see an option `Inspect Element`
  - This will allow you to see the code used to generate the page

How to scrape a web page



```
<style>⋯</style>
<div id="gb" class="gb_Zf">⋯</div>
<div id="searchform" class="jhp big">⋯</div>
<div id="main" class="content">
  <span id="body" class="ctr-p">
    <center>
      <div id="lga" jscontroller="cvn5cb" data-responsive-height-resize="536" data-responsive-min-height="100" data-responsive-min-width="400"
      jsaction="rcuQ6b:npT2md">
        <style>⋯</style>
        <style>⋯</style>
        <div id="hplogo" style="display:inline-block;position:relative;height:200px;width:500px;line-height:200px">
          <a href="/search?sxsrf=ALeKk01splWusIrpowx4VatLjeotys5B0A:16094507292…4868592&hl=en&sa=X&ved=0ahUKEwjjgP-Xl_ntAhX4F1kFHeNwC6gQPQgO">
            <img alt="New Year's Eve 2020" src="/logos/doodles/2020/new-years-eve-2020-6753651837108665-law.gif" style="padding-top:1px" title="New Year's Eve
            2020" data-atf="1" width="500" height="200" border="0">
          </a>
          <div>⋯</div>
        </div>
      </div>
      <div style="height:118px"></div>
      <div style="margin-top:12px">⋯</div>
    </center>
  </span>
<div id="footer" class="ctr-p">⋯</div>
<div id="footc">⋯</div>
<div id="lb" class="gTMtLb fp-nh">⋯</div>
<span style="display:none">⋯</span>
```

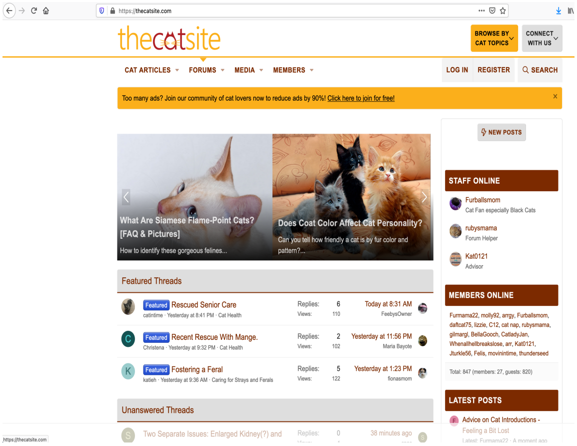
```
html › body#gsr.hp.vasq.big › div#viewport.ctr-p › div#main.content › span#body.ctr-p › center › div#lga › div#hplogo › a › img
```

```
element {                                   inline
  padding-top: 1px;
}
div#hplogo a img {                          inline:1
  max-width: 100%;
  height: auto;
  vertical-align: bottom;
}
Inherited from a
a {                                         inline:1
  color: #1a0dab;
}
body, td, a, p, .h {                        inline:1
  font-family: arial,sans-serif;
}
Inherited from div#hplogo
element {                                   inline
  line-height: 200px;
}
div#lga div#hplogo {                        inline:1
  line-height: 233px;
}
Inherited from body#gsr
body, html {                                inline:1
  font-size: small;
}
```

```
box-sizing      content-box
display         inline
float           none
line-height     200px
position        static
```

```
Content Security Policy: The page's settings blocked the loading of a resource at inline ("script-src").      onloadwff.js:71:790746
```

How to scrape a web page



https://thecatsite.com

thecatsite

BROWSE BY CAT TOPICS

CONNECT WITH US

CAT ARTICLES · FORUMS · MEDIA · MEMBERS

LOG IN · REGISTER · SEARCH

Too many ads? Join our community of cat lovers now to reduce ads by 90%! Click here to join for free!

NEW POSTS

What Are Siamese Flame-Point Cats? [FAQ & Pictures]

How to identify these gorgeous felines...

Does Coat Color Affect Cat Personality?

Can you tell how friendly a cat is by fur color and pattern?...

Featured Threads

Featured Rescued Senior Care
catintime · Yesterday at 8:41 PM · Cat Health
Replies: 6 Views: 110 Today at 8:31 AM FeebysOwner

Featured Recent Rescue With Mange.
Christena · Yesterday at 9:32 PM · Cat Health
Replies: 2 Views: 102 Yesterday at 11:56 PM Maria Bayote

Featured Fostering a Feral
katieh · Yesterday at 9:36 AM · Caring for Strays and Ferals
Replies: 5 Views: 122 Yesterday at 1:23 PM fionasmom

Unanswered Threads

Two Separate Issues: Enlarged Kidney(?) and
Replies: 0 38 minutes ago

STAFF ONLINE

Furballsmom
Cat Fan especially Black Cats

rubysmama
Forum Helper

Kat0121
Advisor

MEMBERS ONLINE

Furmama22, molly92, arrgy, Furballsmom, daftcat75, lizzie, C12, cat nap, rubysmama, gilmargl, BellaGooch, CatladyJan, Whenallhellbreakslose, arr, Kat0121, Jturkle56, Felis, movinintime, thunderseed

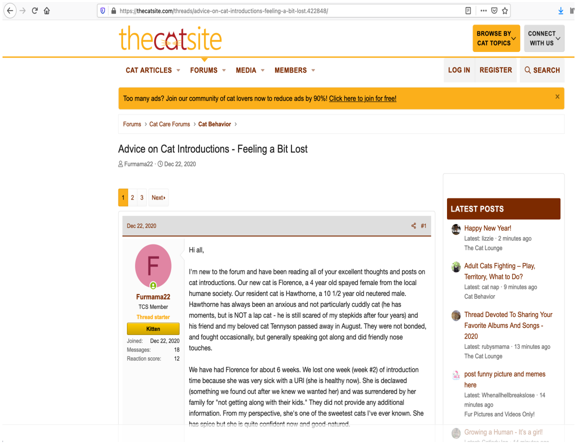
Total: 847 (members: 27, guests: 820)

LATEST POSTS

Advice on Cat Introductions - Feeling a Bit Lost
Latest: Furmama22 · A moment ago

https://thecatsite.com

How to scrape a web page


https://thecatsite.com/threads/advice-on-cat-introductions-feeling-a-bit-lost.422848/

Too many ads? Join our community of cat lovers now to reduce ads by 90%! Click here to join for free!

Forums › Cat Care Forums › Cat Behavior ›

Advice on Cat Introductions - Feeling a Bit Lost

Furmama22 · Dec 22, 2020

1 2 3 Next

Dec 22, 2020 #1

Furmama22
TCS Member
Thread starter
Kitten

Joined:	Dec 22, 2020
Messages:	18
Reaction score:	12

Hi all,

I'm new to the forum and have been reading all of your excellent thoughts and posts on cat introductions. Our new cat is Florence, a 4 year old spayed female from the local humane society. Our resident cat is Hawthorne, a 10 1/2 year old neutered male. Hawthorne has always been an anxious and not particularly cuddly cat (he has moments, but is NOT a lap cat - he is still scared of my stepkids after four years) and his friend and my beloved cat Tennyson passed away in August. They were not bonded, and fought occasionally, but generally speaking got along and did friendly nose touches.

We have had Florence for about 6 weeks. We lost one week (week #2) of introduction time because she was very sick with a URI (she is healthy now). She is declawed (something we found out after we knew we wanted her) and was surrendered by her family for "not getting along with their kids." They did not provide any additional information. From my perspective, she's one of the sweetest cats I've ever known. She has spice but she is quite confident now and good-natured.

LATEST POSTS

Happy New Year!
Latest: lizzie · 2 minutes ago
The Cat Lounge

Adult Cats Fighting – Play, Territory, What to Do?
Latest: cat nap · 9 minutes ago
Cat Behavior

Thread Devoted To Sharing Your Favorite Albums And Songs - 2020
Latest: rubysmama · 13 minutes ago
The Cat Lounge

post funny picture and memes here
Latest: Whenallhellbreakslose · 14 minutes ago
Fur Pictures and Videos Only!

Growing a Human - It's a girl!

How to scrape a web page

Using `rvest` to scrape HTML

```r
library(rvest)
library(tidyverse)
library(stringr)
```

How to scrape a web page

Using rvest to scrape HTML

```
url <- "https://thecatsite.com/threads/advice-on-cat-introductions-feel
thread <- rvest::read_html(url)
```

How to scrape a web page

Using rvest to scrape HTML

```
class(thread)
print(thread)
```

How to scrape a web page

Collecting messages

First, we parse the HTML to obtain the text of each message on the page. Here we use the CSS selector `.message-body`, which selects all elements with class `message-body`. The `html_nodes` function in `rvest` allows us to retrieve these nodes.

```
message.data <- thread %>% html_nodes(".message-body")
print(message.data[2])
```

How to scrape a web page

Collecting messages

Next we use `html_text()` to extract the text from the HTML.

```r
messages <- thread %>% html_nodes(".message-body") %>%
  html_text() %>% str_trim()
print(messages[1])
```

How to scrape a web page

Collecting messages

As expected, there are twenty messages.

```
print(length(messages))
print(substr(messages[20], 1, 250)) # print a substring
```

How to scrape a web page

Getting user names

Next we collect the name of each user using the same logic. User information is found by parsing the `.message-userDetails` node.

```
users <- thread %>% html_nodes(".message-userDetails") %>%
  html_text() %>% str_trim()
print(length(users))
class(users)
print(users[1])
```

How to scrape a web page

Getting user names

Let's add some more elements to the pipe to extract the user name from this string. Note how the elements in the string returned in the previous chunk are separated by the newline symbol (`\n`).

```
users <- thread %>% html_nodes(".message-userDetails") %>%
  html_text() %>% str_trim() %>% str_split('\n')
class(users)
print(users[1])
```

How to scrape a web page

Getting user names

The final step is to get the name from each list. This can be done by using the map command.

```
users <- thread %>% html_nodes(".message-userDetails") %>%
  html_text() %>% str_trim() %>% str_split('\n') %>% map(1)
class(users)
print(users[1:2])
```

How to scrape a web page

Collecting timestamps

Finally, we also want to get the time-stamp of each message. While the forum only displays dates, we can actually get the full timestamp. What's the problem here?

```
dates <- thread %>% html_nodes("time.u-dt")
print(dates[1])
length(dates)
```

How to scrape a web page

Collecting timestamps

I went back to the HTML and found this CSS selector `.u-concealed .u-dt` is selected instead. It returns the datetime for each post in the thread, along with the date time at the top indicating when the thread was created.

```
dates <- thread %>% html_nodes(".u-concealed .u-dt")
length(dates)
dates[1]
class(dates[1])
```

How to scrape a web page

Collecting timestamps

Each HTML node contains several different attributes related to the time. In this case we can select the `datetime` attribute using the `html_attr` function.

```
dates <- thread %>% html_nodes(".u-concealed .u-dt") %>% html_attr("dat
print(dates[1])
class(dates[1])
```

How to scrape a web page

Collecting timestamps

Finally, its often useful to clean up timestamps. We can do this using the `lubridate` package. In this case we extract the year, month, day, hour, minutes, and seconds, converted to EST. The result is a special type of object used to represent dates and times.

```r
library(lubridate)
dates <- dates %>% ymd_hms(tz = "EST")
print(dates[1])
class(dates)
```

How to scrape a web page

Putting it all together

```
length(users)
class(users)
length(messages)
class(messages)
length(dates)
class(dates)
```

How to scrape a web page

Putting it all together

```
data <- as_tibble(cbind(messages, unlist(users), dates[-1]))
colnames(data) <- c("message", "user", "timestamp")
head(data)
```

How to scrape a web page

Creating a function to collect and store data

```r
get.posts <- function(thread) {
  messages <- thread %>% html_nodes(".message-body") %>%
    html_text() %>% str_trim()
  users <- thread %>% html_nodes(".message-userDetails") %>%
    html_text() %>% str_trim() %>% str_split('\n') %>% map(1)
  timestamps <- thread %>% html_nodes(".u-concealed .u-dt") %>%
    html_attr("datetime") %>% ymd_hms(tz="EST")
  timestamps <- timestamps[-1] # remove first timestamp
  data <- as_tibble(cbind(messages, unlist(users), timestamps))
  colnames(data) <- c("message","user", "timestamp")
  return(data)
}
```

How to scrape a web page

Using the function

We can now easily run all the code to extract information using a single function call:

```r
results <- get.posts(thread)
head(results)
```

How to scrape a web page

Pagination

The next step is to figure out how we can navigate the different pages of the thread. Inspect the HTML to find the appropriate element to go to the next page.

```
thread %>% html_nodes("")
```

How to scrape a web page

Pagination

In this case I want both the links *and* the descriptions.

```r
links <- thread %>% html_nodes("") %>%
  html_attr("href")
desc <- thread %>% html_nodes("") %>%
  html_text()
pagination.info <- data.frame(links, desc) %>%
  filter(str_detect(desc, "Next")) %>% distinct()
head(pagination.info)
```

How to scrape a web page

Pagination

We can then use the base URL to get the link to the next page.

```
base <- "https://thecatsite.com"
next.page <- paste(base, pagination.info$links, sep = '')
print(next.page)
```

How to scrape a web page

Pagination

Let's verify this works by using the get.posts function.

```
results <- get.posts(read_html(next.page))
results[1:5,]
```

How to scrape a web page

Pagination function

```r
get.next.page <- function(thread){
  links <- thread %>% html_nodes(".pageNav-jump") %>%
    html_attr("href")
  desc <- thread %>% html_nodes(".pageNav-jump") %>%
    html_text()
  pagination.info <- data.frame(links, desc) %>%
    filter(str_detect(desc, "Next")) %>% distinct()
  base <- "https://thecatsite.com"
  next.page <- paste(base, pagination.info$links, sep = '')
  return(next.page)
}
get.next.page(thread)
```

How to scrape a web page

Testing the pagination function

We can easily use this function to paginate. In this case we use `get.next.page` to get the link to page 2, read the HTML for page 2, then use `get.next.page` to extract the link to page 3.

```r
thread.2 <- read_html(get.next.page(thread))
page.3 <- get.next.page(thread.2)
print(page.3)
```

How to scrape a web page

Testing the pagination function

What happens when we run out of pages? In this case there is no link to the next page. The `get.next.page` function does not produce an error, but only returns the base URL.

```
get.next.page(read_html("https://thecatsite.com/threads/advice-on-cat-i
```

How to scrape a web page

Improving the function

```
get.next.page <- function(thread){
  links <- thread %>% html_nodes(".pageNav-jump") %>%
    html_attr("href")
  desc <- thread %>% html_nodes(".pageNav-jump") %>%
    html_text()
  pagination.info <- data.frame(links, desc) %>%
    filter(str_detect(desc, "Next")) %>% distinct()
  if (dim(pagination.info)[1] == 1) {
    base <- "https://thecatsite.com"
    next.page <- paste(base, pagination.info$links, sep = '')
  return(next.page)
    } else {
    return("Final page")
  }
}
```

How to scrape a web page

Testing the pagination function

We now get this message when we try to paginate on the final page.

```
get.next.page(read_html("https://thecatsite.com/threads/advice-on-cat-i
```

How to scrape a web page

Paginate and scrape

```r
paginate.and.scrape <- function(url){
  thread <- read_html(url)
  posts <- get.posts(thread)
  next.page <- get.next.page(thread)

  while (!str_detect(next.page, "Final page"))
  {
    print(next.page)
    thread <- read_html(next.page)
    posts <- rbind(posts, get.posts(thread))
    next.page <- get.next.page(thread)
    Sys.sleep(0.5) # wait 0.5 seconds
  }
  return(posts)
}
```

How to scrape a web page

Paginate and scrape

```r
full.thread <- paginate.and.scrape(url)
dim(full.thread)
print(head(full.thread))
```

How to scrape a web page

Crawling a website

- Now we have a function we can use to paginate and scrape the data from threads on the website
- The next goal is to write a crawler to traverse the website and retrieve information from all of the threads we are interested in.
- Fortunately, these threads are organized in a similar way to posts
    - Each page contains 20 threads and links to the next page

How to scrape a web page

Crawling a website

```
get.threads <- function(url) {
  f <- read_html(url)
  title <- f %>% html_nodes(".structItem-title") %>%
    html_text() %>% str_trim()
  link <- f %>% html_nodes(".structItem-title a") %>%
    html_attr("href")  %>% str_trim()
  link <- data.frame(link)
  link <- link %>% filter(str_detect(link, "/threads/"))
  threads <- data.frame(title, link)
  return(threads)
}
```

How to scrape a web page

Crawling a website

```
forum.url <- "https://thecatsite.com/forums/cat-behavior.5/"

threads <- get.threads(forum.url)
```

How to scrape a web page

Crawling a website

```r
print(threads$title)
```

How to scrape a web page

Crawling a website

```
print(threads$link)
```

How to scrape a web page

Crawling a website

Iterating over the first 5 pages of threads and collecting all threads in each set.

```r
pagination.max <- 5
url <- forum.url
results <- tibble()

for (p in 1:pagination.max) {
  print(p)
  threads <- get.threads(url)
  for (t in 1:dim(threads)[1]) {
    page.url <- paste(base, threads$link[t], sep = '')
    new.results <- paginate.and.scrape(page.url)
    new.results$threads <- threads$title[t]
    results <- bind_rows(results, new.results)
  }
  url <- get.next.page(read_html(url))
}
```

How to scrape a web page

Storing the results

The results should consist of a few thousand messages and associated metadata. Save the results of this crawl to as a CSV file.

```
write_csv(results, "../data/cat_crawl.csv")
```

How to scrape a web page

Data storage

- If you try to collect all the data you need before saving it, you run the risk of data loss if you script crashes
  - This risk increases as you collect more data
    - More memory on your computer is being used
    - Increased likelihood of encountering anomalies that cause errors
- Reasonable solutions
  - Continuously save results to disk (e.g. concatenate each thread to a CSV)
  - Store results in chunks (e.g. each thread in a new CSV)

How to scrape a web page

Data storage

- A more robust solution
  - Write output to a relational database
    - This helps to organize the data and makes it easier to query and manage, particularly with large datasets
    - I recommend PostgreSQL, a free open-source, SQL-compatible relational database

How to scrape a web page

Data storage

- If collecting a lot of data, I recommend use a server to run the code and to store scraped data
- Requirements
  - Access to a server
    - But most universities have free computing clusters
  - Command line knowledge
  - Database knowledge

How to scrape a web page

Logging

- Log the progress of your webscraper
  - Simple option:
    - Print statements in code
  - Better option:
    - Use a log file
  - To keep yourself updated:
    - Use a Slack App to send yourself messages

Advanced webscraping

Javascript and browser automation

- Many websites use Javascript, which cause problems for web-scrapers as it cannot directly be parsed to HTML
- We can get around this by doing the following
    - Automatically to open a browser (e.g. Google Chrome)
    - Load a website in the browser
    - Read the HTML from the browser into R

Advanced webscraping

Selenium

- Selenium WebDriver and the package `RSelenium` (https://github.com/ropensci/RSelenium) is the most popular approach
- **However**, `RSelenium` requires a complicated set up using a Docker container
- It will be easier to use `selenium` in Python then read the data into R
  - https://python-bloggers.com/2020/07/rvspython-3-setting-up-selenium-limitations-with-the-rselenium-package-getting-past-them/

Advanced webscraping

Running Selenium using Python

This Python code uses selenium to open up a Chrome browser, visit a website, and collect the HTML. It then closes the browser.

```
from selenium import webdriver
from time import sleep
driver = webdriver.Chrome()
driver.get('https://www.sociology.rutgers.edu')
html = driver.page_source
sleep(15)
driver.close()
```

This will only work if the Chrome driver has been downloaded and is in your PATH. See https://chromedriver.chromium.org/getting-started

Advanced webscraping

Using reticulate to obtain results using R

We can use `reticulate` to pass objects from Python to R. Note the Python objects are shown in the `Environment` tab.

```r
library(reticulate)
html.text <- read_html(py$html) %>% html_text()
```

Advanced webscraping

Inspecting the results in R

Reticulate allowed us to run a Python script then pass the results to R. We can then use the same commands as above to process it.

```r
print(substr(html.text, 1, 35))
```

Advanced webscraping

Browser automation

- Using browser automation you can perform other tasks that simulate a human user
    - Clicking buttons
    - Filing forms
    - Logging in
    - Saving images

Concluding remarks

- Webscraping can be used to collect data from websites when APIs are not available
- Read terms of service and inspect robots.txt before starting
- Use HTML structure and CSS elements to navigate website
- Start small and test, test, test
- Log results and store data incrementally (ideally in a database)
- Some websites require more complex strategies using browser automation

Next week

- Online surveys and experiments
  - No coding, more discussion
- Homework 2 released