Data Collection from Web Sources

Session 2 @ CBA Batch 12
April 2019

sudhir_voleti@isb.edu

ISB

1

Session Outline

- A Motivating Example
  - Evolv-Xerox and Likert Scales
- Basic HTML primer
- DC from static web pages
  - Examples 1, 2, 3
- DC from dynamic webpages
  - Using webdrivers in py
- Session Wrap-up

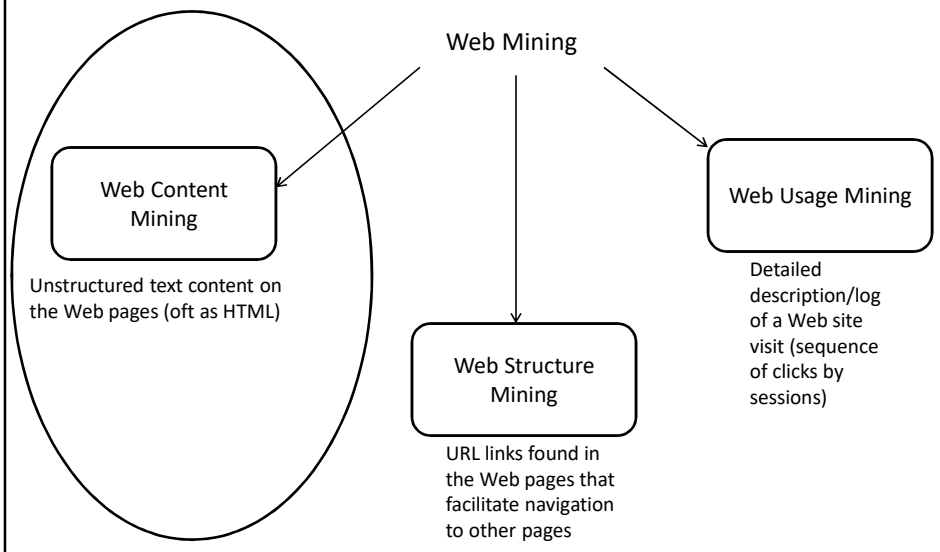
ISB

2

Some Preliminaries

3

Preliminaries: Web Mining Types



4

Preliminaries: HTML & the DOM

- What is *web-scraping*? What is its objective?
- What are webpages written in?
- What is HTML? What does it look like?
- What is a DOM? What does it look like?
- What converts HTML & CSS to human-readable form?
- How much of HTML code would typically be of interest in web-scraping?

5

Preliminaries: Static and Dynamic web pages

- Part of a web-page may have merely 'static' content in html.
- Part of the page may be dynamic - loading and updating constantly. (E.g., comments sections, twitter or FB timelines etc)
- Which of the 2 would be easier to scrape?
- For static pages, we'll use *rvest* in R and *beautiful soup* in Py.
- Sometimes we may need to evaluate conditions interactively and then only decide what to scrape and when. Enter *webdrivers*.
- Before we proceed, a basic primer on HTML …

6

A Basic HTML Primer

7

An *Elementary* HTML primer

- Aim of this primer is to demonstrate:

- Basic html structures
- Major html tag-types such as:
  - a. title tag
  - b. body tags
  - c. document structure tags
  - d. basic markup tags
  - e. links, image tags
  - f. table tags

- Idea is to enable easy scanning of the page-source of static webpages.

- Open file ‘html primer dc text-editor.txt' and *save-as* a '.html' file. Now open the .html file in browser.

8

Basic HTML Primer: Recap

- List some HTML tags we just saw.
- Consider the power and possibility to create variously designed and marked-up webpages simply remixing a handful of basic HTML tags.
- Can we now see page-sources on the web and estimate what the actual markups may look like?
- Ready to handle simple homework on building a static web-page?
- For scraping data off of web-pages, we'll need to identify particular patterns in html tags.
- Let's head there next, then.

ISB

9

Scraping Static Web-pages

rvest and beautifulSoup

ISB

10

A Roadmap for this Section

- Will need CSS selector gadget on chrome browser for this section.
- Plan is to cover 3 examples.
- 1. Scraping the IMDB - mini use-case as a walkthrough example.
- 2. Scraping a Telecom glossary @ https://glossary.atis.org/
- 3. Py scraping news-stories @ news.ycombinator.com
- Ready? Open 'Introducing webscraping with rvest.Rmd'

ISB

11

A Walk-through Example: IMDB

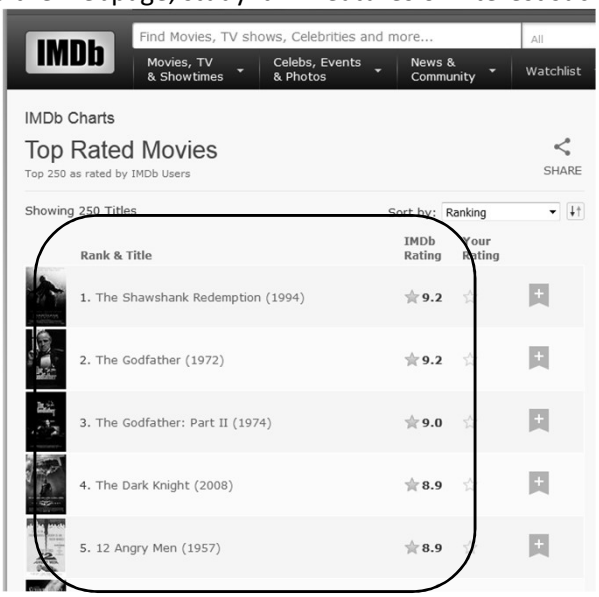
- Q: "What attributes characterize the *best* movies of all time?"
- How would you approach this Q? Where would you first look for data?
- Even before DC, what might "attributes" here mean? And what is 'best' anyway?

ISB

12

rvest for web scraping: Walkthrough Example

- Step 1: Go to the webpage, study it. ID features of interest such as _____.



ISB

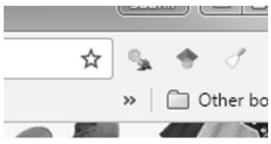
13

rvest for web scraping: Walkthrough Example

- Step 2: Read it into R using _____ function.

```
library("rvest")
library("XML")
# IMDB Top 250 Movies
url = "http://www.imdb.com/chart/top?ref_=nv_wl_img_3"
page = read_html(url)
```

- Step 3: Activate the SelectorGadget tool on Chrome browser.



ISB

14

rvest for web scraping: Walkthrough Example

- Step 4: ID the CSS tags needed and read them into _____ function.

15

rvest for web scraping: Walkthrough Example

- Step 5: Scan the structure of extracted nodes. ID that tiny % that is useful using the _____ function.

```
movie.nodes = html_nodes(page,'.titleColumn a')
# Check one node
xmlTreeParse(movie.nodes[[1]])
```

```
## $doc
## $file
## [1] "<buffer>"
##
## $version
## [1] "1.0"
##
## $children
## $children$a
## <a href="/title/tt0111161/?pf_rd_m=A2FGELUNOQJNL&amp;pf_rd_p=2398042102&amp;pf_rd_r=01E8JAZYM2837BHDB9Q5&amp;pf_rd_s=center-1&amp;pf_rd_t=15506&amp;pf_rd_i=top&amp;ref_=chttp_tt_1" title="Frank Darabont (dir.), Tim Robbins, Morgan Freeman">The Shawshank Redemption</a>
##
##
## attr(,"class")
## [1] "XMLDocumentContent"
##
```

Link URL to that movie on IMDB

Movie name as text

Star cast under 'title ='

16

rvest for web scraping: Walkthrough Example

- Step 6: Store fields of interest as separate vectors because ____.

```
movie.link = sapply(html_attrs(movie.nodes),`[[`,'href')
movie.link = paste0("http://www.imdb.com",movie.link)
movie.cast = sapply(html_attrs(movie.nodes),`[[`,'title')
movie.name = html_text(movie.nodes)
```

```
> head(cbind(movie.link, movie.cast, movie.name))
     movie.link

[1,] "http://www.imdb.com/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_1"
[2,] "http://www.imdb.com/title/tt0068646/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_2"
[3,] "http://www.imdb.com/title/tt0071562/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_3"
[4,] "http://www.imdb.com/title/tt0468569/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_4"
[5,] "http://www.imdb.com/title/tt0050083/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_5"
[6,] "http://www.imdb.com/title/tt0108052/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_6"
     movie.cast                                          movie.name
[1,] "Frank Darabont (dir.), Tim Robbins, Morgan Freeman"     "The Shawshank Redemption"
[2,] "Francis Ford Coppola (dir.), Marlon Brando, Al Pacino"  "The Godfather"
[3,] "Francis Ford Coppola (dir.), Al Pacino, Robert De Niro" "The Godfather: Part II"
[4,] "Christopher Nolan (dir.), Christian Bale, Heath Ledger" "The Dark Knight"
[5,] "Sidney Lumet (dir.), Henry Fonda, Lee J. Cobb"          "12 Angry Men"
[6,] "Steven Spielberg (dir.), Liam Neeson, Ralph Fiennes"    "Schindler's List"
```


rvest for web scraping: Walkthrough Example

- Step 7: Sometimes some basic _____ maybe needed to clean up the info before storing. Take movie release year ….

rvest for web scraping: Walkthrough Example

- Step 7: Sometimes some basic _____ maybe needed to clean up the info before storing. Take average movie rating that has CSS tags _____.

```
rating.nodes = html_nodes(page,'.imdbRating')
# Check One node
xmlTreeParse(rating.nodes[[1]])
```

```
## $children
## $children$td
## <td class="ratingColumn imdbRating">
##  <strong title="9.2 based on 1,707,259 user ratings">9.2</strong>
## </td>
##
```

```
# Correct the node
rating.nodes = html_nodes(page,'.imdbRating strong')
votes = as.numeric(gsub(',','',
                        gsub(' user ratings','',
                             gsub('.*?based on ','',
                                  sapply(html_attrs(rating.nodes),`[[`,'title')
                                  ))))
rating = as.numeric(html_text(rating.nodes))
```

19

rvest for web scraping: Walkthrough Example

- Step 8: Build a dataframe with all extracted info and write to disk.

```
top250 = data.frame(movie.name, movie.cast, movie.link,year,votes,rating)
head(top250)
```

```
##                movie.name
## 1 The Shawshank Redemption
## 2          The Godfather
## 3   The Godfather: Part II
## 4         The Dark Knight
## 5        Schindler's List
## 6            12 Angry Men
##                                          movie.cast
## 1     Frank Darabont (dir.), Tim Robbins, Morgan Freeman
## 2  Francis Ford Coppola (dir.), Marlon Brando, Al Pacino
## 3 Francis Ford Coppola (dir.), Al Pacino, Robert De Niro
## 4 Christopher Nolan (dir.), Christian Bale, Heath Ledger
## 5    Steven Spielberg (dir.), Liam Neeson, Ralph Fiennes
## 6          Sidney Lumet (dir.), Henry Fonda, Lee J. Cobb
##
movie.link
## 1 http://www.imdb.com/title/tt0111161/?pf_rd_m=A2FGELUUNOQJN
5&pf_rd_s=center-1&pf_rd_t=15506&pf_rd_i=top&ref_=chttp_tt_1
```

```
##   year   votes rating
## 1 1994 1707259    9.2
## 2 1972 1167246    9.2
## 3 1974  799349    9.0
## 4 2008 1694374    8.9
## 5 1993  874184    8.9
## 6 1957  453555    8.9
```

```
write.csv(top250,'IMDB Top 250.csv', row.names = F)
```

20

rvest for web-scraping: Recap

- We saw just how easy it can be to scrape data off (reasonably structured) web-pages.
- But hey, suppose you really needed to collect movie genre, then what?
- Turns out 'movie.link' links to the movie's IMDB page where such info is available.
- One can replicate the previous analysis this time on the movie's IMDB page --> ID & extract more features --> loop over the 250 movies.
- Heck, if there are more links of interest, then one can "chain" together web scraping ops on successive pages.

21

rvest : Recap and Ponder

- A good time to take a step back and review learnings from rvest
- What libraries did we call?
- What main inbuilt functions did we use?
- What user-defined functions did we use?
- What inputs and outputs to the web-scraping did we see?
- Any other comment on learnings? Applications? Assignments?

22

Another rvest Exercise

- So far we built vectors of features of interest (e.g., movie name, cast etc) by *applying* funcs over the entire webpage, then *column-binding* into DFs.
- Alternately, we could've taken each unit (e.g., movie) as an R object, extracted features of interest and *row-bound* the result into a DF.
- Open the HTML file 'Webscraping Basics with rvest'.
- The file contains 2 examples –
  - one for scraping a telecom industry glossary page-by-page, and
  - the other for scraping the ycombinator main news page.

23

Scraping a Glossary: Example

- Why scrape atis.org?
  - Well, this exercise is illustrative. For the tools & skills that’re more generally applicable.
- Note change in URLs as we click on each link.
- In structured web-pages, there is a **pattern** to that URL change that we can leverage to scrape the site easily.
- Each glossary term in turn is a link, with an id that leads to a new page (check its URL) with the text definition of that term.
- Our aim scrape the glossary’s terms alone.

24

Scraping a Glossary: Recap

- What did we just do?
- What libraries did we call?
- What functions did we use?
- What kind of structure did we rely on?
- P.S. What if we also needed to capture each glossary term's text definition?

ISB

25

Scraping news.ycombinator: Another Exercise

- Along the same lines, however there's a twist here.
- This one requires we study the view-source more deeply.
- From page-source, we can id the nodes of interest (e.g., 'a') and the attributes in that node (e.g., 'storylink', 'sitestr') that contain

```
="0" cellpadding="0" cellspacing="0" class="itemlist">

><span class="rank">1.</span></td>      <td valign="top" class="votelinks"
&amp;goto=news'><div class='votearrow' title='upvote'></div></a></center><
art/index.html" class="storylink">Depixelizing Pixel Art (2011)</a><span c
an class="sitestr">johanneskopf.de</span></a>)</span></td></tr><tr><td col

29 points</span> by <a href="user?id=doppp" class="hnuser">doppp</a> <span
an id="unv_16417389"></span> | <a href="hide?id=16417389&amp;goto=news">hi
        </td></tr>
```

26

Beautiful soup-ing the webpage in py

- Py up and running for everyone? Open spyder (or Jupyter).
- Copy paste the code, chunk by chunk.
- Examine the variables get created in the variable explorer pane.
- Recall that the hard work of navigating the page-source was done already.

27

27

A Few More Preliminaries

Data Display vs Data Storage formats

28

Some Well-known Data Storage Formats

- Unlike HTML & DOM which are more of data *markup and display* formats, JSON and XML are popular data *storage* formats.
- Consider an example of fields {Name, age, occupation} for two people A & B.
  - {Ravi, 38, Graphic Designer}
  - {Anu, Sales Executive, 27}
- Consider how a person vs how a machine would read & understand.
  - Why the difference? What can be done about *ensuring* such doesn't happen?
- Enter data storage formats like JSON and XML.
  - These contain both the field names and the field values for every data point.
  - Verbose, but accurate.
  - Sample this example

29

JSON & XML Data Storage Formats

- Here's a quick view of what JSON output looks like …
- Can you ID the field names (or 'keys') and values?
- And now a quick view of what XML output looks like …
- Can you ID the field names (or 'keys') and values?
- Note the ability to nest and build hierarchical data storage structures

30

Webdrivers for Web-scraping

31

Using Selenium in Py

- Open the HTML file 'Intro to webdrivers - Selenium in Py'
- Let's walk though it step by step.
- What I'll show next is fairly basic. However, ...
- If you're aware of alternatives, better ways to do the same thing etc., pls speak up and share with the class.

32

Using Webdrivers: Recap

- What are webdrivers and where are they most used?
- What modules did we invoke for using webdrivers?
- What main functions were called? What did they do?
- What further possibilities come to mind with webdriver use?
- Ready for some basic homework involving py and selenium?

33

Session Wrap-up

34

Session Wrap-up

- This was more of a 'workshop' session than the 2 preceding it.
- Expect the trend to continue here onwards for the rest of DC and for Text-analytics.
- Group formation issues remain.
- Any Qs or comments etc?

ISB

35

Thank You

Q & A

ISB

36