爬虫入门 (一)

林双全

2016 年 *12* 月 *8* 日

本文为本人在学习曾武雄学长 (fibears) 的 presentation 时做的笔记。

基础知识

学习爬虫前所需要的基础知识：

- URL:(uniform resource locator), 也就是所谓的网址, 通常由三部分构成：模式或协议、服务器或 IP 地址、路径和文件名
- HTTP：也就是超文本传输协议，是目前互联网上应用最为广泛的一种网络协议。HTTP 是基于请求响应模式，客户端向服务器发送一个请求，服务器则已一个状态作为响应，响应的内容通常为网页源码。HTTP 定义了操作资源的八种不同方法，其中最基本的 4 种：GET,POST,PUT,DELETE, 分别对应查询、修改、增加、删减 4 个操作。
- HTML：超文本标记语言，是一种建立在网页文件的语言，通过标记式的指令 (Tag), 将影像、图片、文件等信息显示出来。HTML 语言使用标记对的方法编写文件，通常使用 “内容” 来表示标志的开始和结束。
  - <html>…</html>: 网页的开始和结束
  - <body>…</body>: 网页的主体部分
  - … : 创建一个段落
  -
…</br>: 创建一个回车换行
  - … : 用于排版大块的 HTML 段落
  - <h1></h1>…<h6></h6>: 不同层级的标题
  - …: 创建一个新标签

CSS 是 Cascading Style Sheets 的缩写，即层叠样式表，是一种标记语言，不需要经过编译过程，可直接由浏览器执行，主要用于美化网页，如定义网页的背景颜色、字体的类型等。

JavaScript 是一种基于对象和事件驱动的客户端脚本语言，可以实现 Ajax 异步请求过程，实现与用户之间的交互过程等。

- Xpath：是一门在 XML 文档（节点树，其根节点也被称作文档节点）中查找信息的语言。Xpath 将节点树种的节点分为七类：元素、属性、文本、命名空间、处理指令、注释和文档节点。Xpath 使用路径表达式来选取 XML 文档中的节点或者节点集。简单来说，就是利用 Xpath 从网页中提取出目标数据。
- Cookie：是指网站为了辨别用户身份、进行 session 跟踪而存储在用户本地终端上的数据。Cookie 由服务器端生成，并发送给浏览器，浏览器随后会将 Cookie 的信息保存到某个目录下的文本文件内，下次请求同一网站时就发送该 Cookie 给服务器。

爬虫工具（R 中）

R 中用于网页抓取的库有 RCurl 和 rvest。
爬虫程序的运行逻辑是模拟访问 URL 地址，然后获取服务器返回的响应文件（HTML 源码或 JSON 格式的字符串）。因此必须明报网页的运行逻辑，然后再一步步利用程序进行实现。

rvest 简介

对于结构比较良好的网页，利用 rvest.CSS/Xpath 选择器和管道符号来处理效率比较高. 以下介绍一些常用的函数。

```
library(RCurl)
```

```
## Loading required package: bitops
```

```
library(rvest)
```

```
## Loading required package: xml2
```

```
str(read_html)
```

```
## function (x, encoding = "", ..., options = c("RECOVER", "NOERROR",
##     "NOBLANKS"))
```

```
# 既可以从网络中获取 html 文档，也可以从本地种载入 html 文档

str(html_nodes)
```

```
## function (x, css, xpath)
```

```
# 利用 CSS 和 Xpath 选择器从 html 文档中提取出节点信息

str(html_text)
```

```
## function (x, trim = FALSE)
```

```
# 提取所有满足条件的文本信息

str(html_attrs)
```

```
## function (x)
```

```
# 提取所有满足条件的属性信息

str(html_table)
```

```
## function (x, header = NA, trim = TRUE, fill = FALSE, dec = ".")
```

```
# 提取表格信息

str(html_session)
```

```
## function (url, ...)
```

```
str(jump_to)
```

```
## function (x, url, ...)
```

```
str(follow_link)
```

```
## function (x, i, css, xpath, ...)
```

```
str(back)
```

```
## function (x)
```

```
# 以上这些函数都是用于模拟浏览网站的
```

一个简单的小例子

```
library(rvest)
# 乐高电影信息
lego_movie <- read_html("http://www.imdb.com/title/tt1490017/")
str(lego_movie)
lego_movie # 分为网页的开始结尾和主体部分
str(html_nodes)
# 利用 CSS 和 Xpath 选择器从 html 文档中提取出节点信息

# 评分
rating <- lego_movie %>%
  html_nodes("strong span")
# 括号内为节点
rating

# 主演
cast <- lego_movie %>%
  html_nodes("#titleCast .itemprop span") %>%
    html_text()    # 提取名字
cast

# 提取图片
poster <- lego_movie %>%
  html_nodes(xpath="//div[@class='poster']/a/img") %>%
    html_attr("src")  # 提取其中的 str 属性
```

爬取豆瓣 top250 信息

```
library(RCurl)
library(rvest)
library(stringr)
library(plyr)
library(dplyr)


# 获取豆瓣电影首页 URL
DoubanUrl <- 'http://movie.douban.com/top250'

# 从首页中获取所有页面的 URL
PageUrlList <- read_html(DoubanUrl) %>%
  html_nodes(xpath = "//div[@class='paginator']/a") %>%  # 底下页数信息
  html_attr("href") %>%
  str_c(DoubanUrl, ., sep="") %>% c(DoubanUrl,.)

# 从每个 PageUrl 中提取出每部电影的链接
MovieUrl <-  NULL
for (url in PageUrlList) {
  item = read_html(url) %>%
    html_nodes(xpath="//div[@class='hd']/a/@href") %>%  # 各页电影的链接
    str_extract('https[\\S]+[\\d]{7}')
  MovieUrl = c(MovieUrl, item)
}
```

```
# 从每个 MovieUrl 中提取出最终的数据
## 定义函数 getdata，用于获取数据并输出 dataframe 格式
GetImdbScore <- function(url){
  ImdbScore = read_html(url) %>%
    html_nodes(xpath = "//span[@itemprop='ratingValue']/text()") %>%
    html_text()
  return(ImdbScore)
}

getdata <- function(url){
  Movie = url
  if(url.exists(url)){
    MovieHTML = read_html(url, encoding = 'UTF-8')
    Rank = html_nodes(MovieHTML, xpath = "//span[@class='top250-no']/text()") %>% html_text()
    MovieName = html_nodes(MovieHTML, xpath = "//span[@property='v:itemreviewed']/text()") %>% html_text()
    Director = html_nodes(MovieHTML, xpath = "//a[@rel='v:directedBy']/text()") %>%
      html_text() %>% paste(collapse = ";")
    Type = html_nodes(MovieHTML, xpath = "//span[@property='v:genre']/text()") %>%
      html_text() %>% paste(collapse = ";")
    Score = html_nodes(MovieHTML, xpath = "//strong[@property='v:average']/text()") %>% html_text()
    ImdbUrl = html_nodes(MovieHTML, xpath = "//a[contains(@href,'imdb')]/@href") %>% html_text()
    ImdbScore = GetImdbScore(ImdbUrl)
    Description = html_nodes(MovieHTML, xpath = "//span[@property='v:summary']/text()") %>%
      html_text() %>% str_replace("\n[\\s]+", "") %>% paste(collapse = ";")
    data.frame(Rank, Movie, MovieName, Director, Type, Score, ImdbScore, Description)
  }
}

## 抓取数据

Douban250 <- data.frame()
for (i in 1:length(MovieUrl)) {
  Douban250 = rbind(Douban250, getdata(MovieUrl[i]))
  print(paste("Movie",i,sep = "-"))
  Sys.sleep(round(runif(1,1,3)))
}
```

上面代码这里就不跑了。

```
# 豆瓣 API
url <- "https://api.douban.com/v2/movie/1292052"
library(rvest)
result <- read_html(url)
result <- html_nodes(result, "p") %>% html_text()
class(result)
```

```
## [1] "character"
```

```
library(rjson)
a = rjson::fromJSON(result)
a
```

```
## $rating
## $rating$max
## [1] 10
```

##
$rating$average
[1] ”9.6”
##
$rating$numRaters
[1] 752307
##
$rating$min
[1] 0
##
##
$author
$author[[1]]
$author[[1]]$name
[1] ”弗兰克·德拉邦特 Frank Darabont”
##
##
##
$alt_title
[1] ”肖申克的救赎 / 月黑高飞(港)”
##
$image
[1] ”https://img3.doubanio.com/view/movie_poster_cover/ipst/public/p480747492.jpg”
##
$title
[1] ”The Shawshank Redemption”
##
$summary
[1] ”20世纪40年代末， 小有成就的青年银行家安迪 （蒂姆·罗宾斯 Tim Robbins 饰） 因涉嫌杀害妻子及她的情人而锒铛入狱。 在这座名为肖申克的监狱内， 希望似乎虚无缥缈， 终身监禁的惩罚无疑注定了安迪接下来灰暗绝望的人生。未过多久， 安迪尝试接近囚犯中颇有声望的瑞德 （摩根·弗里曼 Morgan Freeman 饰）， 请求对方帮自己搞来小锤子。 以此为契机， 二人逐渐熟稔， 安迪也仿佛在鱼龙混杂、 罪恶横生、 黑白混淆的牢狱中找到属于自己的求生之道。 他利用自身的专业知识， 帮助监狱管理层逃税、 洗黑钱， 同时凭借与瑞德的交往在犯人中间也渐渐受到礼遇。 表面看来， 他已如瑞德那样对那堵高墙从憎恨转变为处之泰然， 但是对自由的渴望仍促使他朝着心中的希望和目标前进。 而关于其罪行的真相， 似乎更使这一切朝前推进了一步…… \n本片根据著名作家斯蒂芬·金（Stephen Edwin King）的原著改编。”
##
$attrs
$attrs$pubdate
[1] ”1994-09-10(多伦多电影节)” ”1994-10-14(美国)”
##
$attrs$language
[1] ”英语”
##
$attrs$title
[1] ”The Shawshank Redemption”
##
$attrs$country
[1] ”美国”
##
$attrs$writer
[1] ”弗兰克·德拉邦特 Frank Darabont” ”斯蒂芬·金 Stephen King”
##
$attrs$director

```
## [1] "弗兰克·德拉邦特 Frank Darabont"
##
## $attrs$cast
##  [1] "蒂姆·罗宾斯 Tim Robbins"
##  [2] "摩根·弗里曼 Morgan Freeman"
##  [3] "鲍勃·冈顿 Bob Gunton"
##  [4] "威廉姆·赛德勒 William Sadler"
##  [5] "克兰西·布朗 Clancy Brown"
##  [6] "吉尔·贝罗斯 Gil Bellows"
##  [7] "马克·罗斯顿 Mark Rolston"
##  [8] "詹姆斯·惠特摩 James Whitmore"
##  [9] "杰弗里·德曼 Jeffrey DeMunn"
## [10] "拉里·布兰登伯格 Larry Brandenburg"
## [11] "尼尔·吉恩托利 Neil Giuntoli"
## [12] "布赖恩·利比 Brian Libby"
## [13] "大卫·普罗瓦尔 David Proval"
## [14] "约瑟夫·劳格诺 Joseph Ragno"
## [15] "祖德·塞克利拉 Jude Ciccolella"
##
## $attrs$movie_duration
## [1] "142 分钟"
##
## $attrs$year
## [1] "1994"
##
## $attrs$movie_type
## [1] "犯罪" "剧情"
##
##
## $id
## [1] "https://api.douban.com/movie/1292052"
##
## $mobile_link
## [1] "https://m.douban.com/movie/subject/1292052/"
##
## $alt
## [1] "https://movie.douban.com/movie/1292052"
##
## $tags
## $tags[[1]]
## $tags[[1]]$count
## [1] 156246
##
## $tags[[1]]$name
## [1] "经典"
##
##
## $tags[[2]]
## $tags[[2]]$count
## [1] 128985
##
## $tags[[2]]$name
## [1] "励志"
##
```

```
##
## $tags[[3]]
## $tags[[3]]$count
## [1] 110460
##
## $tags[[3]]$name
## [1] "信念"
##
##
## $tags[[4]]
## $tags[[4]]$count
## [1] 95809
##
## $tags[[4]]$name
## [1] "自由"
##
##
## $tags[[5]]
## $tags[[5]]$count
## [1] 78664
##
## $tags[[5]]$name
## [1] "美国"
##
##
## $tags[[6]]
## $tags[[6]]$count
## [1] 66549
##
## $tags[[6]]$name
## [1] "人性"
##
##
## $tags[[7]]
## $tags[[7]]$count
## [1] 45197
##
## $tags[[7]]$name
## [1] "人生"
##
##
## $tags[[8]]
## $tags[[8]]$count
## [1] 43261
##
## $tags[[8]]$name
## [1] "剧情"
```

API 什么的现在还不懂，以后再慢慢学习。