Web Scraping with R

An introduction for beginners

Feng Li
feng.li@cufe.edu.cn

Outline

- Why web scraping?
- What researchers do with web data?
- How to scrape web data?

Why web scraping?

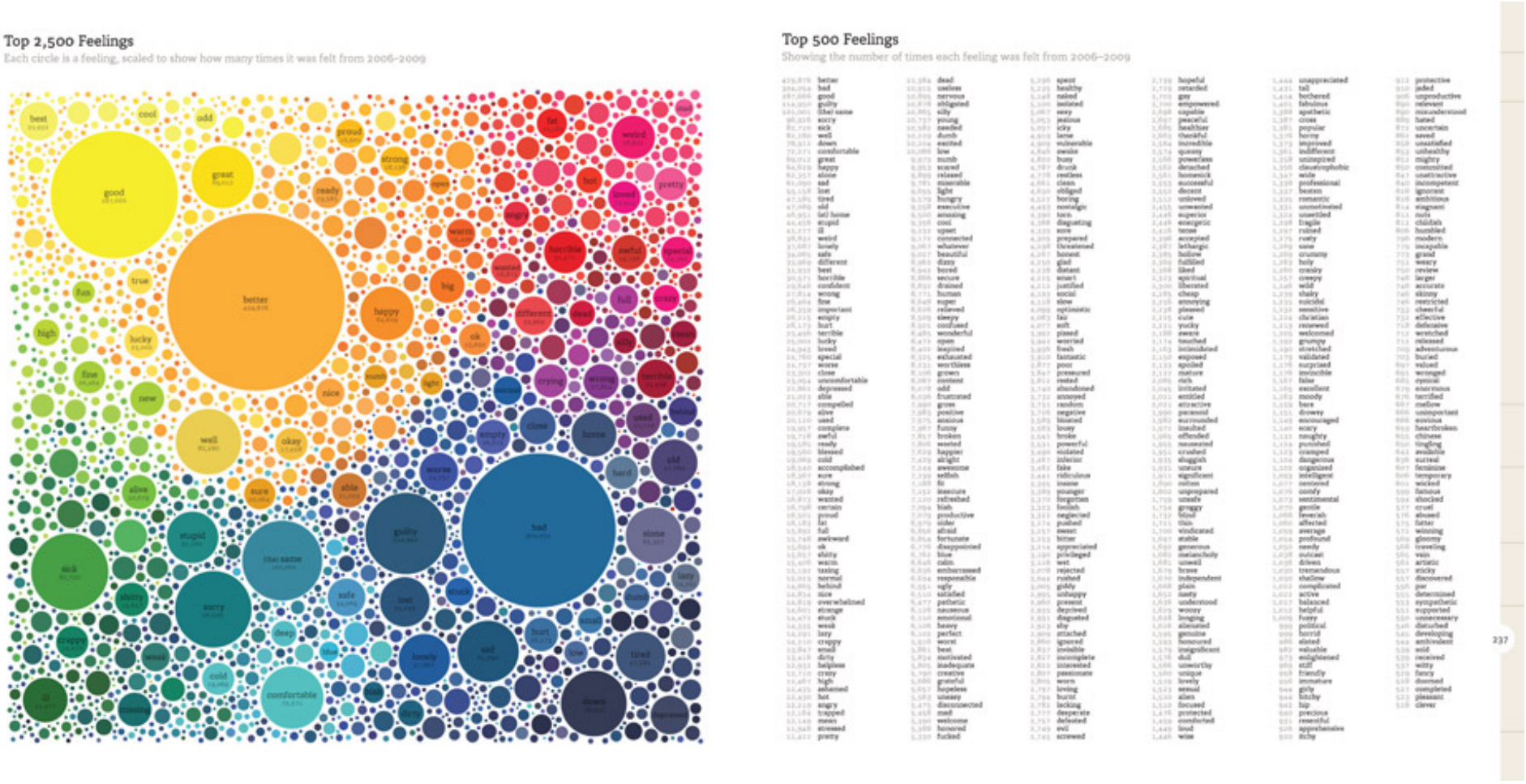
Why do we need Web Scraping?

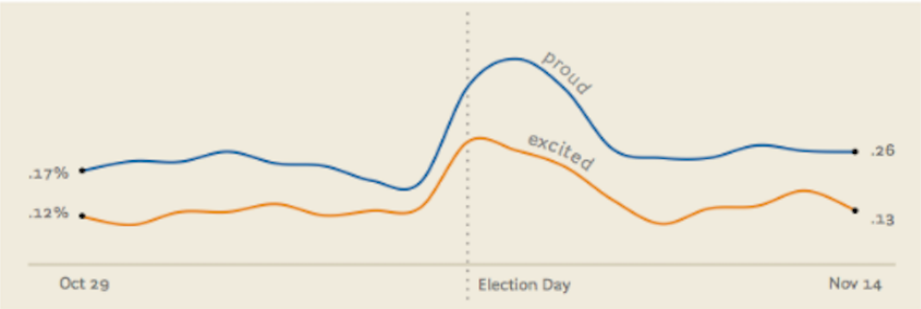
- Data on the web is growing exponentially.
- If the only way you access the Internet is through a browser, you're missing out on a huge range of possibilities.
- Rather than viewing one page at a time, you can access thousands or even millions of pages at once.
- If you can view it in your browser $\Rightarrow$ you can access it via a script $\Rightarrow$ you can store it in a database $\Rightarrow$ you can do virtually anything with that data.

A practical application: "We Feel Fine"

- Project by Jonathan Harris and Sep Kamvar.
- Scraped a variety of English-language blog sites for phrases starting with "I feel" or "I am feeling".
- Describe how the world was feeling day by day and minute by minute.

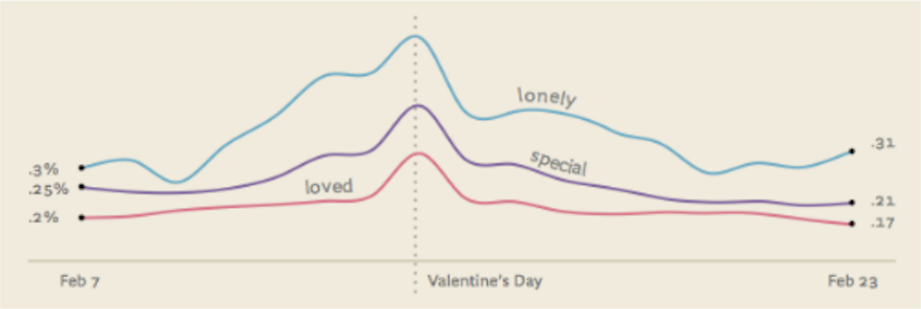
"We Feel Fine"


Top 2,500 Feelings
Each circle is a feeling, scaled to show how many times it was felt from 2006–2009
Top 500 Feelings
Showing the number of times each feeling was felt from 2006–2009

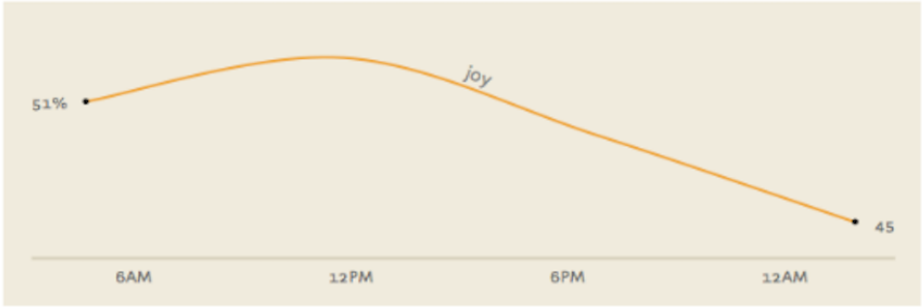


Obama's Election Day / Nov. 4, 2008
There was a dramatic spike in excitement and a swelling sense of pride, which lingered for several days after the election, during the palpable state of national euphoria.

Stressful Weeks
Stress is high throughout the work week, but begins to decline on Friday, just as relaxation starts to rise, climbing to its Sunday high. Stress begins again on Sunday.



Valentine's Day / 2006–2008 average
Loneliness sees the biggest rise, starting a few days before Valentine's Day and remaining high for a few days afterwards. Feeling special and loved is also typical of the holiday.



Joyful Mornings
Joy is high in the morning, and peaks just before lunchtime, before beginning its steady decline through the rest of the day as food coma and fatigue set in.

What researchers do with web data?

RESEARCH ARTICLE

The Effects of Twitter Sentiment on Stock Price Returns

Gabriele Ranco[1], Darko Aleksovski[2]*, Guido Caldarelli[1,3,4], Miha Grčar[2], Igor Mozetič[2]

1 IMT Institute for Advanced Studies, Piazza San Francesco 19, 55100 Lucca, Italy, **2** Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia, **3** Istituto dei Sistemi Complessi (ISC), Via dei Taurini 19, 00185 Rome, Italy, **4** London Institute for Mathematical Sciences, 35a South St. Mayfair, London W1K 2XF, United Kingdom

* darko.aleksovski@ijs.si

OPEN ACCESS

Citation: Ranco G, Aleksovski D, Caldarelli G, Grčar M, Mozetič I (2015) The Effects of Twitter Sentiment on Stock Price Returns. PLoS ONE 10(9): e0138441. doi:10.1371/journal.pone.0138441

Editor: Tobias Preis, University of Warwick, UNITED KINGDOM

Received: June 3, 2015

Accepted: August 31, 2015

Published: September 21, 2015

Abstract

Social media are increasingly reflecting and influencing behavior of other complex systems. In this paper we investigate the relations between a well-known micro-blogging platform Twitter and financial markets. In particular, we consider, in a period of 15 months, the Twitter volume and sentiment about the 30 stock companies that form the Dow Jones Industrial Average (DJIA) index. We find a relatively low Pearson correlation and Granger causality between the corresponding time series over the entire time period. However, we find a significant dependence between the Twitter sentiment and abnormal returns during the peaks of Twitter volume. This is valid not only for the expected Twitter volume peaks (e.g., quarterly announcements), but also for peaks corresponding to less obvious events. We formalize the procedure by adapting the well-known "event study" from economics and finance to the analysis of Twitter data. The procedure allows to automatically identify events as Twitter volume peaks, to compute the prevailing sentiment (positive or negative) expressed in tweets at these peaks, and finally to apply the "event study" methodology to relate them to stock returns. We show that sentiment polarity of Twitter peaks implies the direction of cumulative abnormal returns. The amount of cumulative abnormal returns is relatively low (about 1–2%), but the dependence is statistically significant for several days after the events.

Journal of Computational Science 2 (2011) 1–8

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Twitter mood predicts the stock market

Johan Bollen[a,*,1], Huina Mao[a,1], Xiaojun Zeng[b]

[a] School of Informatics and Computing, Indiana University, 919 E. 10th Street, Bloomington, IN 47408, United States
[b] School of Computer Science, University of Manchester, Kilburn Building, Oxford Road, Manchester M13 9PL, United Kingdom

ARTICLE INFO

Article history:
Received 15 October 2010
Received in revised form 2 December 2010
Accepted 5 December 2010
Available online 2 February 2011

Keywords:
Social networks
Sentiment tracking
Stock market
Collective mood

ABSTRACT

Behavioral economics tells us that emotions can profoundly affect individual behavior and decision-making. Does this also apply to societies at large, i.e. can societies experience mood states that affect their collective decision making? By extension is the public mood correlated or even predictive of economic indicators? Here we investigate whether measurements of collective mood states derived from large-scale Twitter feeds are correlated to the value of the Dow Jones Industrial Average (DJIA) over time. We analyze the text content of daily Twitter feeds by two mood tracking tools, namely OpinionFinder that measures positive vs. negative mood and Google-Profile of Mood States (GPOMS) that measures mood in terms of 6 dimensions (Calm, Alert, Sure, Vital, Kind, and Happy). We cross-validate the resulting mood time series by comparing their ability to detect the public's response to the presidential election and Thanksgiving day in 2008. A Granger causality analysis and a Self-Organizing Fuzzy Neural Network are then used to investigate the hypothesis that public mood states, as measured by the OpinionFinder and GPOMS mood time series, are predictive of changes in DJIA closing values. Our results indicate that the accuracy of DJIA predictions can be significantly improved by the inclusion of specific public mood dimensions but not others. We find an accuracy of 86.7% in predicting the daily up and down changes in the closing values of the DJIA and a reduction of the Mean Average Percentage Error (MAPE) by more than 6%.

OPEN ACCESS Freely available online

PLOS ONE

Early Prediction of Movie Box Office Success Based on Wikipedia Activity Big Data

Márton Mestyán[1], **Taha Yasseri**[1,2,3]*, **János Kertész**[1,3,4]

1 Institute of Physics, Budapest University of Technology and Economics, Budapest, Hungary, **2** Oxford Internet Institute, University of Oxford, Oxford, United Kingdom, **3** Department of Biomedical Engineering and Computational Science, Aalto University, Aalto, Finland, **4** Center for Network Science, Central European University, Budapest, Hungary

Abstract

Use of socially generated "big data" to access information about collective states of the minds in human societies has become a new paradigm in the emerging field of computational social science. A natural application of this would be the prediction of the society's reaction to a new product in the sense of popularity and adoption rate. However, bridging the gap between "real time monitoring" and "early predicting" remains a big challenge. Here we report on an endeavor to build a minimalistic predictive model for the financial success of movies based on collective activity data of online users. We show that the popularity of a movie can be predicted much before its release by measuring and analyzing the activity level of editors and viewers of the corresponding entry to the movie in Wikipedia, the well-known online encyclopedia.

SUBJECT AREAS:

STATISTICAL PHYSICS, THERMODYNAMICS AND NONLINEAR DYNAMICS

APPLIED PHYSICS

COMPUTATIONAL SCIENCE

INFORMATION THEORY AND COMPUTATION

Received
25 February 2013

Accepted
3 April 2013

Quantifying Trading Behavior in Financial Markets Using *Google Trends*

Tobias Preis[1]*, Helen Susannah Moat[2,3]* & H. Eugene Stanley[2]*

[1]Warwick Business School, University of Warwick, Scarman Road, Coventry, CV4 7AL, UK, [2]Department of Physics, Boston University, 590 Commonwealth Avenue, Boston, Massachusetts 02215, USA, [3]Department of Civil, Environmental and Geomatic Engineering, UCL, Gower Street, London, WC1E 6BT, UK.

Crises in financial markets affect humans worldwide. Detailed market data on trading decisions reflect some of the complex human behavior that has led to these crises. We suggest that massive new data sources resulting from human interaction with the Internet may offer a new perspective on the behavior of market participants in periods of large market movements. By analyzing changes in *Google* query volumes for search terms related to finance, we find patterns that may be interpreted as "early warning signs" of stock market moves. Our results illustrate the potential that combining extensive behavioral data sets offers for a better understanding of collective human behavior.

Tourism Management 61 (2017) 43–54

Contents lists available at ScienceDirect

Tourism Management

journal homepage: www.elsevier.com/locate/tourman

Relationship between customer sentiment and online customer ratings for hotels - An empirical analysis

M. Geetha [a, *], Pratap Singha [b], Sumedha Sinha [b]

[a] Department of Management Studies, IIT Madras, India
[b] IIT Madras, India

HIGHLIGHTS

- Sentiment analysis of online hotel reviews for explaining customer ratings.
- Premium and budget segment hotels in Goa considered for study.
- Statistically significant variation in ratings is explained by sentiment polarity.
- Sentiments are less positive for premium hotels than budget hotels in Goa.
- Premium hotels fair better in terms of staff performance and service.

ARTICLE INFO

Article history:
Received 29 March 2016
Received in revised form
29 December 2016
Accepted 30 December 2016

Keywords:
Hotel categories
Customer ratings
Customer sentiments
Sentiment analysis

ABSTRACT

This study aims to establish a relationship between customer sentiments in online reviews and customer ratings for hotels. Customer sentiment refers to the emotions expressed by customers through the text reviews. These sentiments can be positive, negative or neutral. The study explores customer sentiments and expresses them in terms of customer sentiment polarity. Our results find consistency between customer ratings and actual customer feelings across hotels belonging to the two categories of premium and budget. Customer sentiment polarity explains significant variation in customer ratings across both the hotel categories. With regard to managerial implications, the study finds that, when compared with premium hotels, managers of budget hotels should improve their staff performance and hotel services. The present study is not exhaustive and other factors like customer review length and review title sentiment can be analyzed for their effects on customer ratings.

International Journal of Forecasting 32 (2016) 243–256

Contents lists available at ScienceDirect

International Journal of Forecasting

journal homepage: www.elsevier.com/locate/ijforecast

Forecasting sales of new and existing products using consumer reviews: A random projections approach

Matthew J. Schneider [a,1], Sachin Gupta [b,*]

[a] Northwestern University, Medill School of Journalism, Media, Integrated Marketing Communications, 1845 Sheridan Road, Evanston, IL 60208-2101, United States

[b] Cornell University, Samuel Curtis Johnson Graduate School of Management, 452 Sage Hall, 14853 Ithaca, NY, United States

ARTICLE INFO

Keywords:
Big data
Forecasting
Consumer reviews
Textual data
Random projections

ABSTRACT

We consider the problem of predicting sales of new and existing products using both the numeric and textual data contained in consumer reviews. Many of the extant approaches require considerable manual pre-processing of the textual data, making the methods prohibitively expensive to implement and difficult to scale. In contrast, our approach uses a bag-of-words method that requires minimal pre-processing and parsing, making it efficient and scalable. However, a key implementation challenge with the bag-of-words approach is that the number of predictors can quickly outstrip the number of degrees of freedom available. Furthermore, the method can require impracticably large computational resources. We propose a random projections approach for dealing with the curse-of-dimensionality issue that afflicts bag-of-words models. The random projections approach is computationally simple, flexible and fast, and has desirable statistical properties. We apply the proposed approach to the forecasting of sales at Amazon.com using consumer reviews with an attributes-based regression model. The model is applied to produce of one-week-ahead rolling horizon sales forecasts for existing and newly-introduced tablet computers. The results show that the predictive performance of the proposed approach for both tasks is strong and significantly better than those of either models that ignore the textual content of consumer reviews, or a support vector regression machine with the textual content. Furthermore, the approach is easy to repeat across product categories, and readily scalable to much larger datasets.

But web data is messy!

Most of the web data is not readily available. It is present in an unstructured format (HTML format).

爱乐之城 8.4

血战钢锯岭 8.7

新 看不见的客人 8.7

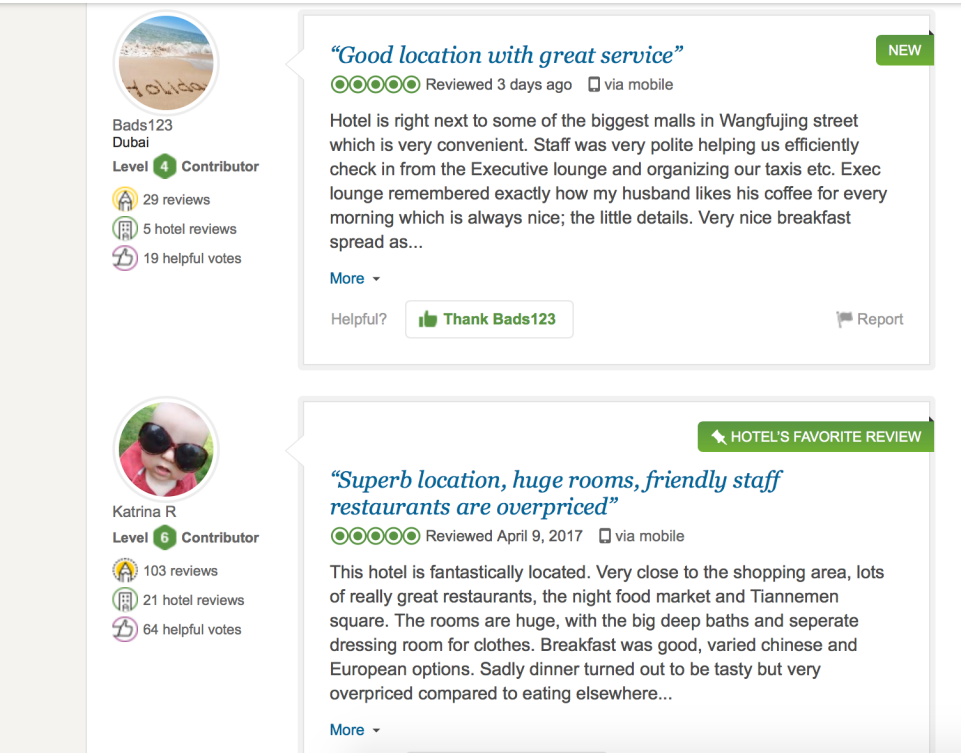
新 游戏王：次元的黑暗面 8.1

驴得水 8.3

你的名字。 8.5

欢乐好声音 8.3

海边的曼彻斯特 8.6



Data we really want

But we want a tidy format of data!

- rows == observations
- columns == attributes

MOVIE	SCORE	LENGTH (MINS)	LANGUAGE
爱乐之城	8.4	128	English
看不见的客人	8.7	106	Spanish
...	...	...	...

Web scraping expertise required!

How to scrape web data?

Get familiar with the structure of a html (tags)

- When we do web scraping, we deal with html tags to find the path of the information we want to extract.

- A simple html source code: tree structure of html tags. HTML tags normally come in pairs.

```
<!DOCTYPE html>
<html>
  <title> My title
  </title>
  <body>
    <h1> My first heading </h1>
      <p> My first paragraph </p>
  </body>
</html>
```

Work with other useful tags

- HTML links are defined with the `<a>` tag `<a href="http://www.test.com">This is a link for test.com</a>`
- HTML tables are defined with `<table>`, row as `<tr>` and rows are divided into data as `<td>`
- HTML list starts with `<ul>` (unordered) and `<ol>` (ordered). Each item of list starts with `<li>`
- Try http://www.tryiteditor.com to learn more about html.

XPath

- When we do web scraping, we use XPath to locate the piece of data we want.
- Path used to select nodes and info in html.
- One of the most crucial step to do web scraping.

```
<!DOCTYPE html>
<html>
  <title> My title
  </title>
  <body>
    <h1> My first heading </h1>
      <p> My first paragraph </p>
  </body>
</html>
```

- `/html/title`: selects the `<title>` element of an HTML document
- `//p`: selects all the `<p>` elements

XPath

```
<html>
  <head>
    <title>Example website</title>
  </head>
  <body>
    <div id='images', class='img'>
      <a href='image1.html'>Name: Image 1<img src='image1_thumb.jpg'/></a>
      <a href='image2.html'>Name: Image 2<img src='image2_thumb.jpg'/></a>
    </div>
  </body>
</html>
```

- `//div[@id="images"]`: selects all the `<div>` elements which contain an attribute `id="images"`.
- `//div[@id="images"]/a/`: selects all the `<a>` elements inside the aforementioned element.

XPath

```
<td class="zwmc" style="width: 250px;">
  <div style="width: 224px;*width: 218px; _width:200px; float: left">
    <a style="font-weight: bold">金融分析师</a>
  </div>
</td>
```

- `//td[@class="zwmc"]/div/a`
- `//td[@class="zwmc"]//a`

Where to go?

scrape job information from http://sou.zhaopin.com of jobs related to '阿里巴巴'.

- Inspect a web page (easily found in Chrome).
- Find the xpath for the elements you want to extract
  - E.g., xpath for job titles: `//td[@class="zwmc"]/div/a`.
  - You can also find xpath from viewing the whole page source

Scrapping a webpage using rvest package in R

- Parse the entire website: `read_html()`.
- Find and extract the pieces of the website you need using XPath: `html_nodes()`. It pull out the entire node.
- The following are done after using html_nodes() to extract content we need.
  - `html_table()`: extract all data inside a html table.
  - `html_text()`: extract all text within the node.
  - `html_attr()`: extract contents of a single attribute.
  - `html_attrs()`: extract all attributes.
- Cleanup

Go back to previous examples

```
web <- read_html('<!DOCTYPE html>
<html>
  <title> My title
  </title>
  <body>
    <h1> My first heading </h1>
      <p> My first paragraph </p>
  </body>
</html>')
title_node <- html_nodes(web, xpath = '//title')
# html_text(title_node)
str_trim(html_text(title_node))
```

```
## [1] "My title"
```

Now we want to scrape data from a html table

```
url <- "https://en.wikipedia.org/wiki/Provinces_of_China"
web <- read_html(url)
provinces_nodes <-
  html_nodes(web, xpath = '//*[@class="wikitable sortable"]')
provinces <- html_table(provinces_nodes)
```

GB[2]	ISO[3]	Province	Chinese Hanyu Pinyin	Capital	Population1	Density2	Area3	Abbreviation4
BJ	CN-11	Beijing Municipality	北京市Běijīng Shì	Beijing	19,612,368	1,167.40	16,800	京Jīng
TJ	CN-12	Tianjin Municipality	天津市Tiānjīn Shì	Tianjin	12,938,224	1,144.46	11,305	津Jīn
HE	CN-13	Hebei Province	河北省Héběi Shěng	Shijiazhuang	71,854,202	382.81	187,700	冀Jì
SX	CN-14	Shanxi Province	山西省Shānxī Shěng	Taiyuan	35,712,111	228.48	156,300	晋Jìn
NM	CN-15	Inner Mongolia Autonomous Region	内蒙古自治区Nèi Měnggǔ Zìzhìqū	Hohhot	24,706,321	20.88	1,183,000	内蒙古 (蒙)Nèi Měnggǔ (Měng)

Now scrape some employment data

```
library(rvest)
url <- 'http://sou.zhaopin.com/jobs/searchresult.ashx?jl=北京&kw=阿里巴巴'
web <- read_html(url)
job_title_nodes <- html_nodes(web, xpath = '//td[@class="zwmc"]/div/a')
job_title <- html_text(job_title_nodes)
job_title[1:2]
```

```
## [1] "阿里妈妈-java研发专家-北京"    "大文娱-APP推广-PP助手&豌豆荚"
```

```
link <- html_attr(job_title_nodes, 'href')
link[1:2]
```

```
## [1] "http://jobs.zhaopin.com/000127917285693.htm"
## [2] "http://jobs.zhaopin.com/0001279179028459200O.htm"
```

Pipeable!

```r
job_title_nodes <- html_nodes(web, xpath = '//td[@class="zwmc"]/div/a')
job_title <- html_text(job_title_nodes)
```

⇓

```r
job_title <- web %>%
  html_nodes(xpath = '//td[@class="zwmc"]/div/a') %>%
  html_text()
```

Let's extract more data

```
url <- 'http://sou.zhaopin.com/jobs/searchresult.ashx?jl=北京&kw=阿里巴巴'
web <- read_html(url, encoding = "utf-8")
job_title <- web %>%
  html_nodes(xpath = '//td[@class="zwmc"]/div/a') %>% html_text()
link <- web %>%
  html_nodes(xpath = '//td[@class="zwmc"]/div/a') %>% html_attr('href')
company <- web %>%
  html_nodes(xpath = '//td[@class="gsmc"]') %>% html_text()
salary <- web %>%
  html_nodes(xpath = '//td[@class="zwyx"]') %>% html_text()
location <- web %>%
  html_nodes(xpath = '//td[@class="gzdd"]') %>% html_text()
alibaba_jobs <- data.frame(job_title, company, salary, location, link)
```

JOB_TITLE	COMPANY	SALARY	LOCATION	LINK
阿里妈妈-java研发专家-北京	阿里巴巴集团	面议	北京	http://jobs.zhaopin.com/000127917285693.htm
JOB_TITLE	COMPANY	SALARY	LOCATION	LINK
大文娱-APP推广-PP助手&豌豆荚	阿里巴巴集团	面议	北京	http://jobs.zhaopin.com/000127917902845920000.htm
大文娱-SEM优化师-UC	阿里巴巴集团	面议	北京	http://jobs.zhaopin.com/000127917902845910000.htm
大文娱-广告平台优化师-UC	阿里巴巴集团	面议	北京	http://jobs.zhaopin.com/000127917902849670000.htm
大文娱-app推广-UC	阿里巴巴集团	面议	北京	http://jobs.zhaopin.com/000127917902849680000.htm
大文娱-运营优化师-终端	阿里巴巴集团	面议	北京	http://jobs.zhaopin.com/000127917902855700000.htm

How to turn pages?

Think about how to turn pages.

Extract more job details via its link

```
get_job_detail <- function(link){
  link = as.character(link)
  web = read_html(link)
  experience = web %>%
    html_nodes(xpath ='//ul[@class="terminal-ul clearfix"]/li[5]/strong') %>% html_text()
  degree = web %>%
    html_nodes(xpath ='//ul[@class="terminal-ul clearfix"]/li[6]/strong') %>% html_text()
  number = web %>%
    html_nodes(xpath ='//ul[@class="terminal-ul clearfix"]/li[7]/strong') %>% html_text()
  description = web %>%
    html_nodes(xpath ='//div[@class="terminalpage-main clearfix"]/div/div[1]')%>% html_text()
  description = str_trim(sub('查看职位地图', '', description))
  link_details = data.frame(experience, degree, number, description)
  return(link_details)
}
```

Extract more job details via its link

```
job_details <- data.frame()
for (i in 1:nrow(alibaba_jobs)){
  job_details = rbind(job_details, get_job_detail(alibaba_jobs$link[i]))
}
alibaba_job_details <- cbind(alibaba_jobs, job_details)
kable(head(subset(alibaba_job_details, select = -c(link,description))), format = "html")
```

JOB_TITLE	COMPANY	SALARY	LOCATION	EXPERIENCE	DEGREE	NUMBER
阿里妈妈-java研发专家-北京	阿里巴巴集团	面议	北京	3-5年	本科	若干
大文娱-APP推广-PP助手&豌豆荚	阿里巴巴集团	面议	北京	3-5年	本科	若干
大文娱-SEM优化师-UC	阿里巴巴集团	面议	北京	3-5年	本科	若干
大文娱-广告平台优化师-UC	阿里巴巴集团	面议	北京	3-5年	本科	若干
大文娱-app推广-UC	阿里巴巴集团	面议	北京	3-5年	本科	若干
大文娱-运营优化师-终端	阿里巴巴集团	面议	北京	5-10年	本科	若干

Practice if you like

- Extract at least 5 attributes of the movies listed on Douban top 250 (https://movie.douban.com/top250)
- Extract the top 5 pages of hotel information including the newest reviews from TripAdvisor (https://www.tripadvisor.com/Hotels-g294212-Beijing-Hotels.html)
- Extract the top 5 pages of book information from Amazon (https://www.amazon.cn/s/ref=nb_sb_noss?__mk_zh_CN=亚马逊网站&field-keywords=大数据)

Some notes

- When you scrape a website too frequently, the server may reject your request. One possible solution is to stop for several seconds irregularly.
- Not every website is scrappable! Some websites go with really high technoloy to protect their data from being extracted. For example, they use javascript, or really complex captcha codes.
- Python has more functionality for web scraping. It is more flexible to deal with the problems mentioned above. If you are interested in that, please refer to this book. Basics of web scraping with Python are similar.

References

- `rvest`
- Wikibooks on text processing

Further readings

- Other packages like `XML`, `RCurl` and `scrapR` are also used for web scraping
- Web Scraping with Python

Questions?