# IS5126 : Hands on with Business Analytics

Group 10

Yu Jianmin A0137787L

15-Feb-2016

# Part 1 - Web-Scraping

**1. Starting URLs**

For players related information:

http://www.basketball-reference.com/players/

For teams related information:

http://www.basketball-reference.com/teams/

**Remarks**: we keep all source code inside the ***source*** folder and csv data inside ***source/data*** folder.

**2. Identify Links and Compare Regex and BeautifulSoup Methods**

Players are grouped by their last names in A to Z categories. We need to open each web page of the categories to get the URL of each player. All categories for players are in http://www.basketball-reference.com/players/[a-z]/. So for all **active players** with last name in category A in page http://www.basketball-reference.com/players/a/, we need to grab all the URLs of them which are listed in **bold**. The syntax of their URLs looks like /players/[a-z]/.\*\.html (e.g. /players/a/abdelal01.html). Besides, we grabbed all players since year 2000. We checked this condition by looking at player last career year no earlier than 2000.

For teams, we need to grab all the URLs of teams in the **Active Franchises** table of the page in http://www.basketball-reference.com/teams/. The syntax of their URLs looks like /teams/[a-z]/ (e.g. /teams/ATL/).

All active player URLs are stored in ***active\_players\_urls.csv*** and all active team URLs are stored in ***team\_urls.csv***. All player since 2000 URLs are stored in ***player\_since\_2000\_urls.csv***.

**a. Regex to get player index A-Z URLs:**

|  |
| --- |
| urls = re.findall('href="(/players/[a-z]/)"',webpage)  hrefs = urls[0:25] if len(urls) > 0 else []  return hrefs; |

**Regex to get all active player URLs:**

|  |
| --- |
| webpage = urlopen(playerurl).read()  strongtags = re.findall('<strong><a(.\*?)  </strong>', webpage)  playerurls = []  if strongtags:  for i in strongtags:  href = re.findall('/players/[a-z]/.\*\.html', i)  if href:  playerurls.append(''.join(href).strip())  return playerurls; |

**Regex to get all active team URLs:**

|  |
| --- |
| urls = re.findall('<td align="left" >  <a href="(.\*?)">',webpage)  #only active teams  hrefs = urls[0:30]  return hrefs; |

**b. BS4 to get player index A-Z URLs:**

|  |
| --- |
| soup = BeautifulSoup(webpage, "html.parser")  index = soup.find("div",{"id":"page\_content"}).find("p")  hrefs = []  for i in index.findAll('a'):  href = i.get('href')  hrefs.append(href)  return hrefs; |

**BS4 to get all active player URLs:**

|  |
| --- |
| webpage = urlopen(playerurl).read()  soup = BeautifulSoup(webpage, "html.parser")  players = soup.find("table",{"id":"players"}).findAll("  strong")  playerurls = []  for i in players:  href = i.find("a").get("href")  playerurls.append(href)  return playerurls; |

**BS4 to get all active team URLs:**

|  |
| --- |
| soup = BeautifulSoup(webpage, "html.parser")  table = soup.find("table",{"id":"active"}).findAll("  tr",{"class":"full\_table"})  hrefs = []  for i in table:  for k in i.find\_all('a'):  team = k.get('href')  hrefs.append(team)  return hrefs; |

**c. Comparing Regular Expression (Regex) and BeautifulSoup (BS4)**

We use the Python **timeit** module to calculate the running time for each method to grab all the necessary data. Reading one web page delays 1 second. Source code is in ***CompareMethods.py***. Running results are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | **Running Time (s)** | **Function Name** | **Remarks** |
| A. Regex to get player index A-Z URLs | 1.79 | getPlayerIndexHrefByRegex | Include time to read a single web page |
| B. BS4 to get player index A-Z URLs | 1.86 | getPlayerIndexHrefByBS | Include time to read a single web page |
| C. Regex to get all active player URLs | 89.95 | getPlayerUrlsByRegex | Include method A and read many web pages |
| D. BS4 to get all active player URLs | 123.44 | getPlayerUrlsByBS | Include method B and read many web pages |
| E. Regex to get all active team URLs | 2.21 | getTeamUrlsByRegex | Include time to read a single web page |
| F. BS4 to get all active team URLs | 3.89 | getTeamUrlsByBS | Include time to read a single web page |

We chose Regex to process all active player and team URLs data. It's relatively easier to write the code using Regex method to grab these data. Besides, using Regex is much faster to process the data if the data size is big.

**3. Player's Page**

a. Active player's basic profile information is stored in ***active\_players\_profile.csv***, and all players since 2000 profile information is stored in ***player\_since\_2000\_profile.csv***.

b. Active players' statistics information is stored in ***active\_player\_stats.csv***, and all players since 2000 statistics information is stored in ***player\_since\_2000\_stats.csv***.

c. Active players' salary information is stored in ***active\_players\_salary.csv***, and all players since 2000 salary information is stored in ***player\_since\_2000\_salary.csv***.

d. We can repeat above part a using Regex, but it's very difficult to navigate the table. See logic in function ***getPlayerInfoByRegex*** from ***PlayerInfo.py.*** Although it's faster to run this method, writing such Regex logic is time consuming, and hence we prefer to use BS4. For example, to match the table and all the eight fields, we need to write Regex like:

|  |  |
| --- | --- |
| **Match Field** | **Regex Match Pattern** |
| Table tbody | tbody = re.findall('(?is)<tbody>.\*?</tbody>', webpage) |
| All trs | trs = re.findall('(?is)<tr.\*?</tr>', ''.join(tbody)) |
| All tds in tr | tds = re.findall('(?is)<td.\*?</td>',tr) |
| Player | re.findall('">(.\*?)</a>',td) |
| From,To,Pos,Ht,Wt | re.findall('>(.\*?)</td>',td) |
| Birth Date | t6 = re.findall('day=.\*?</a>',td)  dob = re.findall('>(.\*?)</a>',''.join(t6)) |
| College | re.findall('">(.\*?)</a>',td) |

**4. Team's page**

***TeamCrawler.py*** has the logic to crawl teams related information, ***TeamInfo.py*** is for getting basic team information and ***TeamStats.py*** is for getting team statistics by season. Run ***TeamCrawler.py*** to get both information.

a. Basic team information is stored in ***teams\_profile.csv***.

b. Team statistics by season information is stored in ***team\_season\_stats.csv***.

**5. Other information**

NBA Champion is always the target for most of NBA team owners. We chose to extract some awards list for NBA players, including Most Valuable Player, Rookie of the Year, Defensive Player of the Year, Most Improved Player of the Year. Team owners can choose some players from the awards winners. We also extracted NBA league index to get all the past Champion teams and top performers.

Secondly, NBA has the team salary cap which is a limit on the amount teams can spend on player contracts. This helps to maintain competitive balance in the league. By comparing with the Salary cap, team owner can understand how much money needs to be invested.

The source code is in ***AwardsList.py***. Corresponding data are stored as:

|  |  |
| --- | --- |
| **Data Name** | **File Name** |
| Most Valuable Player | ***mvp.csv*** |
| Rookie of the Year | **roy.csv** |
| Defensive Player of the Year | ***dpoy.csv*** |
| Most Improved Player of the Year | ***mip.csv*** |
| league index | ***league.csv*** |
| NBA salary cap history | ***salary\_cap.csv*** |

**Part 2 - ETL**