TopPages

December 20, 2016

1 Cleaning Your Data

Let's take a web access log, and figure out the most-viewed pages on a website from it! Sounds easy, right?

Let's set up a regex that lets us parse an Apache access log line:

```
In [1]: import re

        format_pat= re.compile(
            r"(?P<host>[\d\.]+)\s"
            r"(?P<identity>\S*)\s"
            r"(?P<user>\S*)\s"
            r"\[(?P<time>.*?)\]\s"
            r'"(?P<request>.*?)"\s'
            r"(?P<status>\d+)\s"
            r"(?P<bytes>\S*)\s"
            r'"(?P<referer>.*?)"\s'
            r'"(?P<user_agent>.*?)"\s*'
        )
```

Here's the full path to the log file I'm analyzing; change this if you want to run this stuff yourself:

```
In [2]: logPath = "E:\\sundog-consult\\Udemy\\DataScience\\access_log.txt"
```

Now we'll whip up a little script to extract the URL in each access, and use a dictionary to count up the number of times each one appears. Then we'll sort it and print out the top 20 pages. What could go wrong?

```
In [3]: URLCounts = {}

        with open(logPath, "r") as f:
            for line in (l.rstrip() for l in f):
                match= format_pat.match(line)
                if match:
                    access = match.groupdict()
                    request = access['request']
```

```
                    (action, URL, protocol) = request.split()
                    if URLCounts.has_key(URL):
                        URLCounts[URL] = URLCounts[URL] + 1
                    else:
                        URLCounts[URL] = 1

        results = sorted(URLCounts, key=lambda i: int(URLCounts[i]), reverse=True)

        for result in results[:20]:
            print(result + ": " + str(URLCounts[result]))


        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-3-8efc8eba3391> in <module>()
          7             access = match.groupdict()
          8             request = access['request']
    ----> 9             (action, URL, protocol) = request.split()
         10             if URLCounts.has_key(URL):
         11                 URLCounts[URL] = URLCounts[URL] + 1


        ValueError: need more than 1 value to unpack
```

Hm. The 'request' part of the line is supposed to look something like this:
GET /blog/ HTTP/1.1
There should be an HTTP action, the URL, and the protocol. But it seems that's not always happening. Let's print out requests that don't contain three items:

```
In [4]: URLCounts = {}

        with open(logPath, "r") as f:
            for line in (l.rstrip() for l in f):
                match= format_pat.match(line)
                if match:
                    access = match.groupdict()
                    request = access['request']
                    fields = request.split()
                    if (len(fields) != 3):
                        print(fields)
```

```
['_\\xb0ZP\\x07tR\\xe5']
[]
[]
[]
[]
```

```
[]
[]
[]
[]
[]
[]
```

Huh. In addition to empty fields, there's one that just contains garbage. Well, let's modify our script to check for that case:

```
In [5]: URLCounts = {}

        with open(logPath, "r") as f:
            for line in (l.rstrip() for l in f):
                match= format_pat.match(line)
                if match:
                    access = match.groupdict()
                    request = access['request']
                    fields = request.split()
                    if (len(fields) == 3):
                        URL = fields[1]
                        if URLCounts.has_key(URL):
                            URLCounts[URL] = URLCounts[URL] + 1
                        else:
                            URLCounts[URL] = 1

        results = sorted(URLCounts, key=lambda i: int(URLCounts[i]), reverse=True)

        for result in results[:20]:
            print(result + ": " + str(URLCounts[result]))
```

```
/xmlrpc.php: 68494
/wp-login.php: 1923
/: 440
/blog/: 138
/robots.txt: 123
/sitemap_index.xml: 118
/post-sitemap.xml: 118
/category-sitemap.xml: 117
/page-sitemap.xml: 117
/orlando-headlines/: 95
/san-jose-headlines/: 85
http://51.254.206.142/httptest.php: 81
/comics-2/: 76
/travel/: 74
/entertainment/: 72
/world/: 70
```

```
/business/: 70
/weather/: 70
/national/: 70
/national-headlines/: 70
```

It worked! But, the results don't really make sense. What we really want is pages accessed by real humans looking for news from our little news site. What the heck is xmlrpc.php? A look at the log itself turns up a lot of entries like this:

46.166.139.20 - - [05/Dec/2015:05:19:35 +0000] "POST /xmlrpc.php HTTP/1.0" 200 370 "-" "Mozilla/4.0 (compatible: MSIE 7.0; Windows NT 6.0)"

I'm not entirely sure what the script does, but it points out that we're not just processing GET actions. We don't want POSTS, so let's filter those out:

```
In [6]: URLCounts = {}

        with open(logPath, "r") as f:
            for line in (l.rstrip() for l in f):
                match= format_pat.match(line)
                if match:
                    access = match.groupdict()
                    request = access['request']
                    fields = request.split()
                    if (len(fields) == 3):
                        (action, URL, protocol) = fields
                        if (action == 'GET'):
                            if URLCounts.has_key(URL):
                                URLCounts[URL] = URLCounts[URL] + 1
                            else:
                                URLCounts[URL] = 1

        results = sorted(URLCounts, key=lambda i: int(URLCounts[i]), reverse=True)

        for result in results[:20]:
            print(result + ": " + str(URLCounts[result]))
```

```
/: 434
/blog/: 138
/robots.txt: 123
/sitemap_index.xml: 118
/post-sitemap.xml: 118
/category-sitemap.xml: 117
/page-sitemap.xml: 117
/orlando-headlines/: 95
/san-jose-headlines/: 85
http://51.254.206.142/httptest.php: 81
/comics-2/: 76
/travel/: 74
```

```
/entertainment/: 72
/world/: 70
/business/: 70
/weather/: 70
/national/: 70
/national-headlines/: 70
/defense-sticking-head-sand/: 69
/about/: 69
```

That's starting to look better. But, this is a news site - are people really reading the little blog on it instead of news pages? That doesn't make sense. Let's look at a typical /blog/ entry in the log:

54.165.199.171 - - [05/Dec/2015:09:32:05 +0000] "GET /blog/ HTTP/1.0" 200 31670 "-" "-"

Hm. Why is the user agent blank? Seems like some sort of malicious scraper or something. Let's figure out what user agents we are dealing with:

```
In [7]: UserAgents = {}

        with open(logPath, "r") as f:
            for line in (l.rstrip() for l in f):
                match= format_pat.match(line)
                if match:
                    access = match.groupdict()
                    agent = access['user_agent']
                    if UserAgents.has_key(agent):
                        UserAgents[agent] = UserAgents[agent] + 1
                    else:
                        UserAgents[agent] = 1

        results = sorted(UserAgents, key=lambda i: int(UserAgents[i]), reverse=True)

        for result in results:
            print(result + ": " + str(UserAgents[result]))
```

```
Mozilla/4.0 (compatible: MSIE 7.0; Windows NT 6.0): 68484
-: 4035
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0): 1724
W3 Total Cache/0.9.4.1: 468
Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html): 278
Mozilla/5.0 (compatible; Googlebot/2.1; +http://www.google.com/bot.html): 248
Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0: 144
Mozilla/5.0 (iPad; CPU OS 8_4 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/8.0
Mozilla/5.0 (Linux; Android 5.1.1; SM-G900T Build/LMY47X) AppleWebKit/537.36 (KHTML, like Gecko)
Mozilla/5.0 (compatible; bingbot/2.0; +http://www.bing.com/bingbot.htm): 43
Mozilla/5.0 (compatible; MJ12bot/v1.4.5; http://www.majestic12.co.uk/bot.php?+): 41
Opera/9.80 (Windows NT 6.0) Presto/2.12.388 Version/12.14: 40
```

```
Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots): 27
Ruby: 15
Mozilla/5.0 (Linux; Android 5.1.1; SM-G900T Build/LMY47X) AppleWebKit/537.36 (KHTML, like Gecko)
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 S
Mozilla/5.0 (compatible; AhrefsBot/5.0; +http://ahrefs.com/robot/): 11
Mozilla/5.0 (Windows NT 5.1; rv:6.0.2) Gecko/20100101 Firefox/6.0.2: 11
MobileSafari/600.1.4 CFNetwork/711.4.6 Darwin/14.0.0: 10
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47
Mozilla/5.0 (compatible; YandexImages/3.0; +http://yandex.com/bots): 9
Mozilla/5.0 (compatible; linkdexbot/2.0; +http://www.linkdex.com/bots/): 7
Mozilla/5.0 (iPhone; CPU iPhone OS 8_3 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Ve
Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ysearch/slurp): 6
Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.2.28) Gecko/20120306 Firefox/3.6.28 (.NET
Mozilla/5.0 (compatible; SeznamBot/3.2; +http://fulltext.sblog.cz/): 4
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1): 4
Mozilla/5.0 (X11; Linux i686) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/30.0.1599.66 Safari/
Mozilla/5.0 zgrab/0.x: 4
Mozilla/5.0 (compatible; spbot/4.4.2; +http://OpenLinkProfiler.org/bot ): 3
Mozilla/5.0 (Windows NT 6.1; rv:34.0) Gecko/20100101 Firefox/34.0: 3
Opera/9.80 (Windows NT 5.1; U; ru) Presto/2.9.168 Version/11.50: 3
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:31.0) Gecko/20100101 Firefox/31.0: 3
Mozilla/4.0 (compatible; MSIE 9.0; Windows NT 6.1; 2Pac; .NET CLR 2.0.50727; .NET CLR 3.0.04506.
Mozilla/5.0: 3
Mozilla/4.0 (compatible: FDSE robot): 3
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46
Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; WOW64; Trident/5.0; .NET CLR 2.0.50727; .NET
Mozilla/5.0 (Windows NT 6.0; rv:31.0) Gecko/20100101 Firefox/31.0: 2
netEstate NE Crawler (+http://www.website-datenbank.de/): 2
Opera/9.80 (Windows NT 6.1); U) Presto/2.10.289 Version/12.02: 2
Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 5.1; Trident/5.0; .NET CLR 2.0.50727; .NET CLR 3.5
Mozilla/5.0 (Windows NT 6.2; rv:24.0) Gecko/20100101 Firefox/24.0: 2
Mozilla/5.0 (Windows NT 6.1; WOW64; rv:21.0) Gecko/20100101 Firefox/21.0: 2
Mozilla/5.0 (X11; U; FreeBSD x86_64; en-US) AppleWebKit/534.16 (KHTML, like Gecko) Chrome/10.0.6
Mozilla/4.0 (compatible; Netcraft Web Server Survey): 2
Mozilla/5.0 (Windows NT 6.0; rv:29.0) Gecko/20120101 Firefox/29.0: 2
Googlebot-Image/1.0: 2
Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.86 S
Mozilla/5.0 (Windows NT 5.1; rv:28.0) Gecko/20100101 Firefox/28.0: 2
Mozilla/5.0 (Windows NT 6.2; WOW64; rv:36.0) Gecko/20100101 Firefox/36.0: 2
Mozilla/5.0 (Windows NT 5.1; rv:2.0.1) Gecko/20100101 Firefox/5.0: 2
Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.7.5) Gecko/20041107 Firefox/1.0: 2
Mozilla/5.0 (Windows NT 6.2; rv:31.0) Gecko/20100101 Firefox/31.0: 2
Mozilla/5.0 (Windows NT 6.1; rv:28.0) Gecko/20100101 Firefox/28.0: 2
Mozilla/5.0 (Windows NT 5.1; rv:36.0) Gecko/20100101 Firefox/36.0: 2
Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/53
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.3.1.1) Gecko/20101203 Firefox/3.6.12 (.NET
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41
Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:42.0) Gecko/20100101 Firefox/42.0: 1
```

```
Mozilla/5.0 (Windows NT 6.1; rv:22.0) Gecko/20130405 Firefox/22.0: 1
Mozilla/5.0 (Windows; U; Windows NT 5.0; fr-FR; rv:0.9.4) Gecko/20011019 Netscape6/6.2: 1
Mozilla/5.0 (Windows NT 6.3; WOW64; Trident/7.0; MAGWJS; rv:11.0) like Gecko: 1
Mozilla/3.0 (compatible; Indy Library): 1
Mozilla/5.0 (Windows NT 6.1; rv:33.0) Gecko/20100101 Firefox/33.0: 1
Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/37.0.2062.120 Safa
Scrapy/1.0.3 (+http://scrapy.org): 1
Mozilla/5.0 (Windows NT 6.1; rv:31.0) Gecko/20100101 Firefox/31.0: 1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46
Opera/9.80 (Windows NT 6.2; WOW64); U) Presto/2.12.388 Version/12.14: 1
facebookexternalhit/1.1 (+http://www.facebook.com/externalhit_uatext.php): 1
Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.59.10 (KHTML, like Gecko) Version
Telesphoreo: 1
Mozilla/5.0 (Windows NT 10.0; WOW64; rv:42.0) Gecko/20100101 Firefox/42.0: 1
Mozilla/4.0 (compatible; MSIE 6.0; MSIE 5.5; Windows 95) Opera 7.03 [de]: 1
Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.0.13) Gecko/2009073022 Firefox/3.0.13 (.NET
Mozilla/5.0 (X11; U; Linux i686; pl-PL; rv:1.9.0.2) Gecko/20121223 Ubuntu/9.25 (jaunty) Firefox/
Mozilla/5.0 (Windows NT 5.1; rv:32.0) Gecko/20100101 Firefox/31.0: 1
NokiaE5-00/SymbianOS/9.1 Series60/3.0 3gpp-gba: 1
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.73 S
Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/45.0.2454.101
Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; rv:11.0) like Gecko: 1
Mozilla/5.0 (Windows NT 5.1; rv:5.0) Gecko/20100101 Firefox/5.0: 1
```

Yikes! In addition to '-', there are also a million different web robots accessing the site and polluting my data. Filtering out all of them is really hard, but getting rid of the ones significantly polluting my data in this case should be a matter of getting rid of '-', anything containing "bot" or "spider", and W3 Total Cache.

```
In [8]: URLCounts = {}

        with open(logPath, "r") as f:
            for line in (l.rstrip() for l in f):
                match= format_pat.match(line)
                if match:
                    access = match.groupdict()
                    agent = access['user_agent']
                    if (not('bot' in agent or 'spider' in agent or
                            'Bot' in agent or 'Spider' in agent or
                            'W3 Total Cache' in agent or agent =='-')):
                        request = access['request']
                        fields = request.split()
                        if (len(fields) == 3):
                            (action, URL, protocol) = fields
                            if (action == 'GET'):
                                if URLCounts.has_key(URL):
```

```
                                    URLCounts[URL] = URLCounts[URL] + 1
                                else:
                                    URLCounts[URL] = 1

        results = sorted(URLCounts, key=lambda i: int(URLCounts[i]), reverse=True)

        for result in results[:20]:
            print(result + ": " + str(URLCounts[result]))
```

```
/: 77
/orlando-headlines/: 36
/?page_id=34248: 28
/wp-content/cache/minify/000000/M9bPKixNLarUy00szs8D0Zl5AA.js: 27
/wp-content/cache/minify/000000/lY7dDoIwDIVfiG0KxkfxfnbdKO4HuxICTy-it8Zw15PzfSftzPCckJem-x4qUWAr
/wp-content/cache/minify/000000/M9AvyUjVzUstLy7PLErVz8lMKkosqtTPKtYvTi7KLCgpBgA.js: 27
/wp-content/cache/minify/000000/fY45DoAwDAQ_FMvkRQgFA5ZyWLajiN9zNHR0O83MRkyt-pIctqYFJPedKyYzfHg2
/?author=1: 21
/wp-content/cache/minify/000000/hcrRCYAwDAXAhXyEjiQ1YKAh4SVSx3cE7_uG7ASr4M9qg3kGWyk1adklK84LHtRj
/wp-content/uploads/2014/11/nhn1.png: 19
/wp-includes/js/wp-emoji-release.min.js?ver=4.3.1: 17
/wp-content/cache/minify/000000/BcGBCQAgCATAiUSaKYSERPk3avzuht4SkBJnt4tHJdqgnPBqKldesTcN1R8.js:
/wp-login.php: 16
/comics-2/: 12
/world/: 12
/favicon.ico: 10
/wp-content/uploads/2014/11/babyblues.jpg: 6
/wp-content/uploads/2014/11/garfield.jpg: 6
/wp-content/uploads/2014/11/violentcrime.jpg: 6
/robots.txt: 6
```

Now, our new problem is that we're getting a bunch of hits on things that aren't web pages. We're not interested in those, so let's filter out any URL that doesn't end in / (all of the pages on my site are accessed in that manner - again this is applying knowledge about my data to the analysis!)

```
In [9]: URLCounts = {}

        with open(logPath, "r") as f:
            for line in (l.rstrip() for l in f):
                match= format_pat.match(line)
                if match:
                    access = match.groupdict()
                    agent = access['user_agent']
                    if (not('bot' in agent or 'spider' in agent or
                            'Bot' in agent or 'Spider' in agent or
                            'W3 Total Cache' in agent or agent =='-')):
                        request = access['request']
```

```
                        fields = request.split()
                        if (len(fields) == 3):
                            (action, URL, protocol) = fields
                            if (URL.endswith("/")):
                                if (action == 'GET'):
                                    if URLCounts.has_key(URL):
                                        URLCounts[URL] = URLCounts[URL] + 1
                                    else:
                                        URLCounts[URL] = 1

    results = sorted(URLCounts, key=lambda i: int(URLCounts[i]), reverse=True)

    for result in results[:20]:
        print(result + ": " + str(URLCounts[result]))
```

```
/: 77
/orlando-headlines/: 36
/world/: 12
/comics-2/: 12
/weather/: 4
/about/: 4
/australia/: 4
/national-headlines/: 3
/sample-page/feed/: 2
/feed/: 2
/technology/: 2
/science/: 2
/entertainment/: 1
/san-jose-headlines/: 1
/business/: 1
/travel/feed/: 1
```

This is starting to look more believable! But if you were to dig even deeper, you'd find that the /feed/ pages are suspect, and some robots are still slipping through. However, it is accurate to say that Orlando news, world news, and comics are the most popular pages accessed by a real human on this day.

The moral of the story is - know your data! And always question and scrutinize your results before making decisions based on them. If your business makes a bad decision because you provided an analysis of bad source data, you could get into real trouble.

Be sure the decisions you make while cleaning your data are justifiable too - don't strip out data just because it doesn't support the results you want!

1.1 Activity

These results still aren't perfect; URL's that include "feed" aren't actually pages viewed by humans. Modify this code further to strip out URL's that include "/feed". Even better, extract some log entries for these pages and understand where these views are coming from.

In [ ]: