SI 601 Homework 2 (100 points)

Regular expressions and log file filtering

Due Date: Tuesday Jan 27, 2014 5pm

This homework will apply your knowledge of Python regular expressions to an important real-world problem: filtering Web access logs. A Web access log records every HTTP request made to a Web server, along with features such as the date, time, IP address of the computer making the HTTP request, and the request's resulting status code (e.g. 200 == success). Log analysis and filtering is an important method for identifying valid vs. suspicious requests, as you will see from this assignment.

First, download the attached si601_w15_hw2.zip and unzip it. You should see these files:

`access_log.txt`

`invalid_log_desired_output.txt`

`valid_log_summary_desired_output.txt`

The file `access_log.txt` is from a security researcher who set up a 'honeypot' server to attract hackers, in order to record the HTTP requests used in hacking attempts, to learn about their methods and origins. See the original Web site at http://log-sharing.dreamhosters.com/. This access log was generated by a popular Web server program, Apache, and its format is explained here (http://httpd.apache.org/docs/2.2/logs.html), under the Access Log section. A couple of fields in the provided file are set to '-' because the data has been sanitized so that there won't be privacy issues. Note: the access_log.txt file is about 42Mb uncompressed.

Name your script 'log_analysis_***youruniquename***.py'. You should browse through 'access_log' to familiarize yourself with the format and think about how you might extract the data out of it (but note its size if you decide to open in a text editor. On my MBA, Sublime Text 3 opened it pretty fast for me).

Your goal is to create two files:

(a) A summary file that contains the counts of all valid daily visits to each top-level domain. The rows should be ordered in chronological order, and the columns should be sorted in alphabetical order of top-level domains. The file should be **tab-delimited.**

(b) A 'suspicious entries' file with the actual invalid access rows filtered from the original log, according to the criteria below.

For example, this line in the log counts as a valid visit to the 'hu' top-level domain.

```
195.82.31.125 - - [09/Mar/2004:22:03:09 -0500] "GET
http://www.goldengate.hu/cgi-bin/top/topsites.cgi?an12 HTTP/1.0" 200 558
"http://Afrique" "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)"
```

The `'hu'` part in the hostname `www.goldengate.hu` is the top-level domain (representing the country of Hungary). The status code of '200' tells us that the visit was successful. So the line should be counted as a valid visit for the `'hu'` domain on 09/Mar/2004. Note that the top-level domain for a hostname like www.cocegas.com.br is .br, not .com.

Use the following rules to determine if a line should be as a valid line. A line in the log file represents a valid visit only if these conditions are true:

1. The HTTP verb is GET or POST (uppercase).

2. AND the status code is 200

3. AND the URL being accessed starts with `http://` or `https://` (case-insensitive, it can be HTTP, Http, etc) followed by at least one alphabetic character (i.e. not a digit or a symbol). For example, the URL should NOT start with `'http:///'`, which is an error.

4. AND the top-level domain consists of only letters. This is to say, if the host name is actually a numerical IP address like `'202.96.254.200'`, we don't count it. If the whole domain name is just '.com' as in `http://.com/blah` or does not even contain a dot as in `http://c/blah`, we do not count it.

5. AND if the URL contains a query string (For more info, see http://en.wikipedia.org/wiki/Query_string#Structure), the value string for any field should not exceed 255 characters long. Note that you may want use the Python urlparse module (https://docs.python.org/2/library/urlparse.html) to parse the URLs instead of rolling your own code.

As an example, note that we do not count lines like the following as valid:

```
68.48.142.117 - - [09/Mar/2004:22:41:42 -0500] "GET
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir HTTP/1.0" 200 566 "-"
"-"
```

This is because /scripts/ obviously doesn't look like a web address since it doesn't start with `http://` or `https://`). This is a prime example of an attempt to get a file stored locally on that proxy server to exploit vulnerabilities.

Another example is on line 24877 in access_log.txt. In the query string on that line, the supplied value string for the field 'k' is VERY VERY long. This matches rule 5, so this line is not a valid visit. A normal user wouldn't enter that much gibberish into a web form. This case is an example of the "buffer overflow" attack attempt. Google "buffer overflow attack" if you are interested in knowing more about it.

Important Details (to check after getting your basic code working):

- Be aware that some rows use *multiple spaces* as a separator between fields, so make sure to use **`[\s]+`** if you split the line's fields using whitespace (and not just **`[\s]`**) .

- Be sure to handle the case where the hostname part of the URL contains a port number delimited by a colon, i.e. like `'abcde.com:8080'`, the 8080 part is called the port number. The port number is not part of the top-level domain and should be ignored: The top-level domain of `'abcde.com:8080'` should be counted as `'com'`.

- You should convert case-insensitive variants of top-level domain strings like ‘CoM’ or ‘cOm’ into the lower-case version (‘com’).

- A small number of valid lines are missing the ‘HTTP/1.x’ after the URL and just have the three-digit HTTP status code. So having the HTTP/1.x field after the URL is not a requirement to have a valid line.

Useful debugging tools:

To compare your output and desired output files, you can use a handy “file compare” utility to see which lines are different. On Windows, try **http://winmerge.org/** For the Mac, some people recommend **http://www.sourcegear.com/diffmerge/**

Your code should create two output files:

1. A tab-delimited log summary file named 'valid_log_summary_***youruniquename***.txt' which should be the same format as 'valid_log_summary_desired_output.txt'. That is, with the first column being the date, and remaining columns containing entries of the form *top-level-domain*:*count*. The top-level domain should be presented in sorted order (default string order).

2. A file named ‘invalid_access_log_***youruniquename***.txt’ containing the lines *rejected* by the conditions above, i.e. all the non-valid lines filtered out of the original log file. This file should be the same as ‘invalid_access_log_desired_output.txt’.

The assignment will be graded as follows.

5 points	You successfully open the input file and read log file entries.
5 points	You write a log summary output file (even if some entries are incorrect)
5 points	You write an “invalid log entry” output file (even if some entries are incorrect)
10 points	Log summary file is correctly grouped by date
5 points	Log summary file is correctly sorted by date
10 points	Log summary file has correct extraction of top-level domain string from URLs
5 points	Log summary file has correctly sorted top-level domains
5 points	Correct filtering by GET and POST
5 points	Correct filtering by URL starting with `http://` or `https://` followed by alphabetic character(s)
5 points	Correct filtering by status code != 200
5 points	Correct filtering by length of parameter value <= 255
20 points	The “invalid log entry” output file matches the desired output exactly.
15 points	Log summary file has correct counts for top-level domains that match the desired valid summary counts exactly.
100 points	

You'll get appropriately scaled points if your filtered output is almost like the reference output but doesn't match exactly. If you're close, or sort of close, or mildly close, you'll get credit for that. The points allocation is *not* "all or nothing".

What to submit:

A zip file named si601_hw2_***youruniquename***.zip that contains:

1. Your python program file: log_analysis_***youruniquename***.py

2. Your output files:

    valid_log_summary_***youruniquename***.txt

    invalid_access_log_***youruniquename***.txt

Acknowledgements: This is a re-revised version of an earlier log analysis assignment originally created by me and later revised by Prof. Kevyn Collins-Thompson.