S04_T01_Registre_de_logs

March 30, 2022

1 S04_T01_Registre de logs

1.0.1 Ex1:Estandaritza, identifica i enumera cada un dels atributs / variables de l'estructura de l'arxiu "Web_access_log-akumenius.com" que trobaràs al repositori de GitHub "Data-sources"

[2]:
```python
#Importem llibreries necessàries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# for regular expressions
import re
from datetime import datetime
import pytz #for time zones
```

https://mmas.github.io/read-apache-access-log-pandas

https://httpd.apache.org/docs/2.2/logs.html

M'he basat en aquests dos links, per entendre el format d'un arxiu log, i poder extreure la info de cada columna

Estudiem com és el nostre registre log per conèixer com estan exposades les nostres dades The above configuration will write log entries in a format known as the Common Log Format (CLF). This standard format can be produced by many different web servers and read by many log analysis programs. The log file entries produced in CLF will look something like this:

127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326

Each part of this log entry is described below.

`127.0.0.1 (%h)`

This is the IP address of the client (remote host) which made the request to the server. If HostnameLookups is set to On, then the server will try to determine the hostname and log it in place of the IP address. However, this configuration is not recommended since it can significantly slow the server. Instead, it is best to use a log post-processor such as logresolve to determine the hostnames. The IP address reported here is not necessarily the address of the machine at which the user is sitting. If a proxy server exists between the user and the server, this address will be the address of the proxy, rather than the originating machine.

`- (%l)`

The “hyphen” in the output indicates that the requested piece of information is not available. In this case, the information that is not available is the RFC 1413 identity of the client determined by identd on the clients machine. This information is highly unreliable and should almost never be used except on tightly controlled internal networks. Apache httpd will not even attempt to determine this information unless IdentityCheck is set to On.

`frank (%u)`

This is the userid of the person requesting the document as determined by HTTP authentication. The same value is typically provided to CGI scripts in the REMOTE_USER environment variable. If the status code for the request (see below) is 401, then this value should not be trusted because the user is not yet authenticated. If the document is not password protected, this part will be “-” just like the previous one.

```
[10/Oct/2000:13:55:36 -0700] (%t)
The time that the request was received. The format is:
  [day/month/year:hour:minute:second zone]
    day = 2*digit
    month = 3*letter
    year = 4*digit
    hour = 2*digit
    minute = 2*digit
    second = 2*digit
    zone = (`+' | `-') 4*digit

It is possible to have the time displayed in another format by specifying %{format}t in the lo
```

`"GET /apache_pb.gif HTTP/1.0" (\"%r\")`

The request line from the client is given in double quotes. The request line contains a great deal of useful information. First, the method used by the client is GET. Second, the client requested the resource /apache_pb.gif, and third, the client used the protocol HTTP/1.0. It is also possible to log one or more parts of the request line independently. For example, the format string “%m %U%q %H” will log the method, path, query-string, and protocol, resulting in exactly the same output as “%r”.

`200 (%>s)`

This is the status code that the server sends back to the client. This information is very valuable, because it reveals whether the request resulted in a successful response (codes beginning in 2), a redirection (codes beginning in 3), an error caused by the client (codes beginning in 4), or an error in the server (codes beginning in 5). The full list of possible status codes can be found in the HTTP specification (RFC2616 section 10).

`2326 (%b)`

The last part indicates the size of the object returned to the client, not including the response headers. If no content was returned to the client, this value will be “-”. To log “0” for no content, use %B instead.

Combined Log Format Another commonly used format string is called the Combined Log Format. It can be used as follows.

LogFormat "%h %l %u %t "%r" %>s %b "%{Referer}i" "%{User-agent}i"" combined CustomLog log/access_log combined

This format is exactly the same as the Common Log Format, with the addition of two more fields. Each of the additional fields uses the percent-directive %{header}i, where header can be any HTTP request header. The access log under this format will look like:

127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326 "http://www.example.com/start.html" "Mozilla/4.08 [en] (Win98; I ;Nav)"

```
The additional fields are:

"http://www.example.com/start.html" (\"%{Referer}i\")
```

The "Referer" (sic) HTTP request header. This gives the site that the client reports having been referred from. (This should be the page that links to or includes /apache_pb.gif).

```
"Mozilla/4.08 [en] (Win98; I ;Nav)" (\"%{User-agent}i\")
```

The User-Agent HTTP request header. This is the identifying information that the client browser reports about itself.

[13]:
```
# https://mmas.github.io/read-apache-access-log-pandas
# https://httpd.apache.org/docs/2.2/logs.html
# M'he basat en aquests dos links, per entendre el format d'un arxiu log, i␣
↪poder extreure cada columna

from datetime import datetime
import pytz

def parse_str(x):
    """
    Returns the string delimited by two characters.

    Example:
        `>>> parse_str('[my string]')`
        `'my string'`
    """
    return x[1:-1]

def parse_datetime(x):
    '''
    Parses datetime with timezone formatted as:
        `[day/month/year:hour:minute:second zone]`

    Example:
        `>>> parse_datetime('13/Nov/2015:11:45:42 +0000')`
        `datetime.datetime(2015, 11, 3, 11, 45, 4, tzinfo=<UTC>)`
    Due to problems parsing the timezone (`%z`) with `datetime.strptime`, the
    timezone will be obtained using the `pytz` library.
    '''
    dt = datetime.strptime(x[1:-7], '%d/%b/%Y:%H:%M:%S')
    dt_tz = int(x[-6:-3])*60+int(x[-3:-1])
    return dt.replace(tzinfo=pytz.FixedOffset(dt_tz))

# Next, we can read our access log file.
data = pd.read_csv(
    'Web_access_log-akumenius.com.txt',
    sep=r'\s(?=(?:[^"]*"[^"]*")*[^"]*$)(?![^\[]*\])',
    engine='python',
    na_values='-',
    header=None,
    usecols=[0, 1, 4, 5, 6, 7, 8, 9],
    names=['host','ip', 'Date_time', 'request', 'status', 'size', 'referer',␣
 ↪'user_agent'],
    converters={'Date_time': parse_datetime,
                'request': parse_str,
                'status': int,
                'size': int,
                'referer': parse_str,
                'user_agent': parse_str})

data.head(10)
```

[13]:

```
        host         ip                  Date_time           request  status  \
0  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
1  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
2  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
3  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
4  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
5  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
6  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
7  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
8  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200
9  localhost  127.0.0.1 2014-02-23 03:10:31+01:00  OPTIONS * HTTP/1.0     200

  size referer                            user_agent
0  NaN     NaN  Apache (internal dummy connection)
1  NaN     NaN  Apache (internal dummy connection)
2  NaN     NaN  Apache (internal dummy connection)
3  NaN     NaN  Apache (internal dummy connection)
4  NaN     NaN  Apache (internal dummy connection)
5  NaN     NaN  Apache (internal dummy connection)
6  NaN     NaN  Apache (internal dummy connection)
```

```
7  NaN     NaN  Apache (internal dummy connection)
8  NaN     NaN  Apache (internal dummy connection)
9  NaN     NaN  Apache (internal dummy connection)
```

[12]:
```
data.tail(10)
```

[12]:
```
                    host             ip                  Date_time  \
261863      akumenius.com   5.255.253.53 2014-03-02 03:05:32+01:00
261864       akumenius.es   5.255.253.53 2014-03-02 03:05:33+01:00
261865      akumenius.com   5.255.253.53 2014-03-02 03:05:35+01:00
261866  www.akumenius.com   5.255.253.53 2014-03-02 03:05:34+01:00
261867       akumenius.es   5.255.253.53 2014-03-02 03:05:35+01:00
261868  www.akumenius.com   5.255.253.53 2014-03-02 03:05:39+01:00
261869  www.akumenius.com  74.86.158.107 2014-03-02 03:09:52+01:00
261870          localhost      127.0.0.1 2014-03-02 03:10:18+01:00
261871          localhost      127.0.0.1 2014-03-02 03:10:18+01:00
261872          localhost      127.0.0.1 2014-03-02 03:10:18+01:00

                          request  status  size referer  \
261863  GET /robots.txt HTTP/1.1     301   301     NaN
261864  GET /robots.txt HTTP/1.1     301   304     NaN
261865            GET / HTTP/1.1     301   301     NaN
261866            GET / HTTP/1.1     200  7528     NaN
261867            GET / HTTP/1.1     301   304     NaN
261868            GET / HTTP/1.1     200  7528     NaN
261869           HEAD / HTTP/1.1     200   NaN     NaN
261870        OPTIONS * HTTP/1.0     200   NaN     NaN
261871        OPTIONS * HTTP/1.0     200   NaN     NaN
261872        OPTIONS * HTTP/1.0     200   NaN     NaN

                                               user_agent
261863  Mozilla/5.0 (compatible; YandexBot/3.0; +http:…
261864  Mozilla/5.0 (compatible; YandexBot/3.0; +http:…
261865  Mozilla/5.0 (compatible; YandexBot/3.0; +http:…
261866  Mozilla/5.0 (compatible; YandexBot/3.0; +http:…
261867  Mozilla/5.0 (compatible; YandexBot/3.0; +http:…
261868  Mozilla/5.0 (compatible; YandexBot/3.0; +http:…
261869  Mozilla/5.0+(compatible; UptimeRobot/2.0; http…
261870                Apache (internal dummy connection)
261871                Apache (internal dummy connection)
261872                Apache (internal dummy connection)
```

[14]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 261873 entries, 0 to 261872
Data columns (total 8 columns):
 #   Column     Non-Null Count   Dtype
```

```
---  ------      --------------   -----
 0   host        261873 non-null  object
 1   ip          261873 non-null  object
 2   Date_time   261873 non-null  datetime64[ns, pytz.FixedOffset(60)]
 3   request     261836 non-null  object
 4   status      261873 non-null  int64
 5   size        219538 non-null  object
 6   referer     162326 non-null  object
 7   user_agent  261654 non-null  object
dtypes: datetime64[ns, pytz.FixedOffset(60)](1), int64(1), object(6)
memory usage: 16.0+ MB
```

1.0.2 Ex2:Neteja, preprocesa, estructura i transforma (dataframe) les dades del registre d'Accés a la web

[ ]:

1.0.3 Ex3:Geolocalitza les IP's

[ ]:

[ ]:

1.0.4 Ex4:Mostra'm la teva creativitat, Sorprèn-me fes un pas més enllà amb l'anàlisi anterior.

[ ]: