Lucas Guichard

**Cyber-Physical System Security – ASSIGNMENT (week 4)**

*Data Preprocessing with Web Server Access Logs*

*(Github:* [*https://github.com/kounen/CPS\_A4*](https://github.com/kounen/CPS_A4)*)*

import pandas as pd

import re

import matplotlib.pyplot as plt

from tqdm.auto import tqdm

tqdm.pandas()

# Open log file and load its data into a list of string

with open('short\_access.txt', 'r') as log\_file:

raw\_data = [line for line in log\_file.readlines()]

# Convert raw\_data strings list into data frame

df = pd.DataFrame(raw\_data, columns = ['log\_line'])

## PRE-PROCESSING

# Use regex pattern to parse each log line

# We use axis=1 to loop on each line and not column

pattern = r'^(\S+) (\S+) (\S+) \[(.\*?)\] "(.\*?)" (\d+) (\d+) "(.\*?)" "(.\*?)" "(.\*?)"$'

df['client\_ip'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(1), axis = 1)

df['user'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(2), axis = 1)

df['http\_auth\_user'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(3), axis = 1)

df['timestamp'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(4), axis = 1)

df['request'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(5), axis = 1)

df['response\_code'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(6), axis = 1)

df['response\_size'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(7), axis = 1)

df['user\_agent'] = df.progress\_apply(lambda x: re.match(pattern, x['log\_line']).group(9), axis = 1)

# Drop columns containing always the same value

# Drop log\_line column (parsing done so its content is not useful now)

# Now, only : ['client\_ip', 'timestamp', 'request', 'response\_code', 'response\_size', 'user\_agent']

columns\_to\_drop = [column for column in df.columns if df[column].nunique() == 1]

columns\_to\_drop.append('log\_line')

df = df.drop(columns\_to\_drop, axis = 1)

## USER AGENT ANALYSIS

# Create new dataframe specific for this analysis

agent\_df = pd.DataFrame()

# Add the user agent raw date collected from previous parsing

agent\_df['raw\_data'] = df['user\_agent']

# Parse and store browser from this raw\_data

agent\_df['browser'] = agent\_df.progress\_apply(lambda x: x['raw\_data'].split('/')[0], axis = 1)

# Parse and store Operating System type

def get\_os(agent\_info: str):

match = re.search(r'\((.\*?)\)', agent\_info)

# Using only OS section

if match:

if 'Windows' in match.group(1):

return 'Windows'

elif 'iPhone' in match.group(1):

return 'iPhone'

elif 'Mac' in match.group(1):

return 'Mac'

elif 'Linux' in match.group(1):

return 'Linux'

elif 'Android' in match.group(1):

return 'Android'

# Barkrowler is a bot devoloped by eXenSa

elif 'bot' or 'Barkrowler' in match.group(1):

return 'bot'

else:

return '-'

# If no match, no OS section detected so we use all the agent information

else:

if 'bot' or 'python' in agent\_info:

return 'bot'

else:

return '-'

agent\_df['os'] = agent\_df.progress\_apply(lambda x: get\_os(x['raw\_data']), axis = 1)

# Count the number of occurrences of each OS

# Create a new dataframe from this computation

os\_counts = agent\_df['os'].value\_counts()

## DISPLAY ANALYSIS

# Create a bar chart of the OS counts

os\_counts.plot.bar()

# Set the chart title and axis labels

plt.title('User agent')

plt.xlabel('Operating System')

plt.ylabel('Number of users')

# Display the chart

plt.show()

## CSV WRITING

# Create a csv file for each dataframe

df.to\_csv('df.csv', index=False)

agent\_df.to\_csv('agent\_df.csv', index=False)

OS analysis figure created thanks to matplotlib:

Une image contenant graphique

Description générée automatiquement

CSV files screenshots:

* df.csv

Une image contenant texte

Description générée automatiquementUne image contenant table

Description générée automatiquement

* agent\_df.csv

Une image contenant texte, table

Description générée automatiquementUne image contenant table

Description générée automatiquement