LogRecords

January 7, 2020

[1]:
```
import datetime
import itertools
import os

import matplotlib.dates as mdates
import matplotlib.pyplot as plt
%matplotlib inline
```

[2]:
```
rootPath = '../../dataset/'
loggingLevels = ['INFO', 'DEBUG', 'TRACE', 'ERROR']

def plot_log_files_starting_with(file_prefix):
    aggregated = []
    for filename in os.listdir(rootPath):
        if filename.endswith('.log') and filename.startswith(file_prefix):
            filepath = rootPath + filename
            process_file(aggregated,filepath)

    grouped = itertools.groupby(aggregated, lambda x: x.date())
    count_per_day = [*map(lambda x: to_dates_count(*x), grouped)]
    count_per_day.sort(key=lambda x: x["log_date"])
    aggregated_dates = [x["log_date"] for x in count_per_day]

    mat_dates = mdates.date2num(aggregated_dates)
    mat_values = [x["number_of"] for x in count_per_day]
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-%d'))
    plt.gca().xaxis.set_major_locator(mdates.DayLocator(interval=7))
    plt.plot_date(mat_dates, mat_values, '-o')

def process_file(aggregated, filepath):
    f = open(filepath, "r+")
    allLines = f.readlines()
    filteredLines = []
    parsedDates = []
    for line in allLines:
        if any(line.startswith(x) for x in loggingLevels):
            filteredLines.append(line)
```

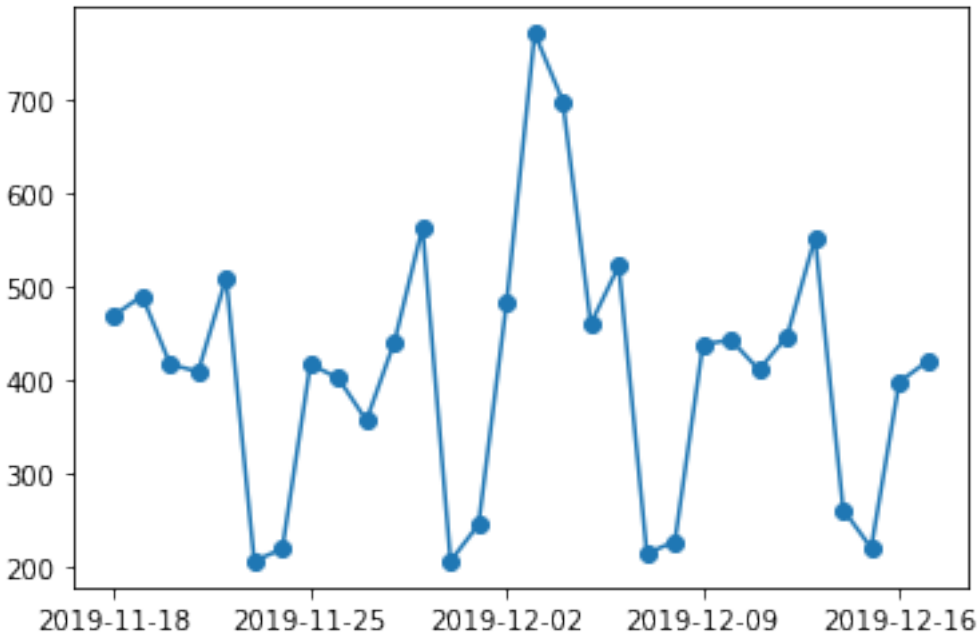
```
            splittedLine = line.split(None, 3)
            if splittedLine[1] != '' and splittedLine[2] != '':
                date = splittedLine[1] + splittedLine[2]
                parsedDate = datetime.datetime.strptime(date, '%Y-%m-%d%H:%M:%S.
↪%f')
                parsedDates.append(parsedDate)
    aggregated.extend(parsedDates)
    f.close()

def to_dates_count(x, y):
    return dict(log_date=x, number_of=len(list(y)))
```

0.1 Liczba wpisów logów timepot.*.log

Główne logi aplikacji Springowej, łącznie z wyjątkami oraz niektórymi zapytaniami SQL

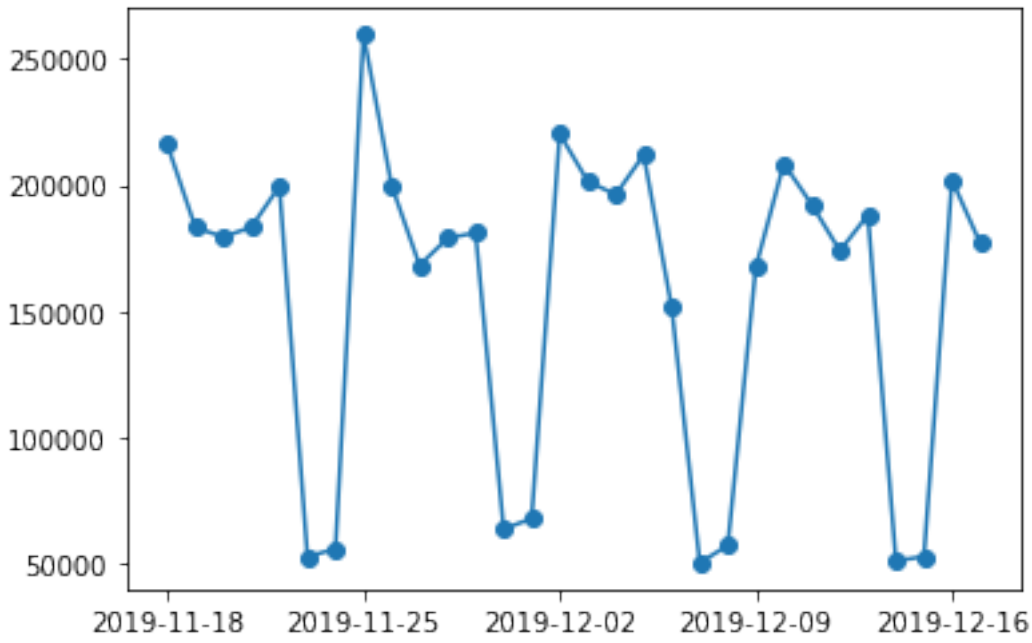
[3]:
```
plot_log_files_starting_with('timepot')
```



0.2 Liczba wpisów logów message.*.log

Komunikacja przychodząca HTTP po REST

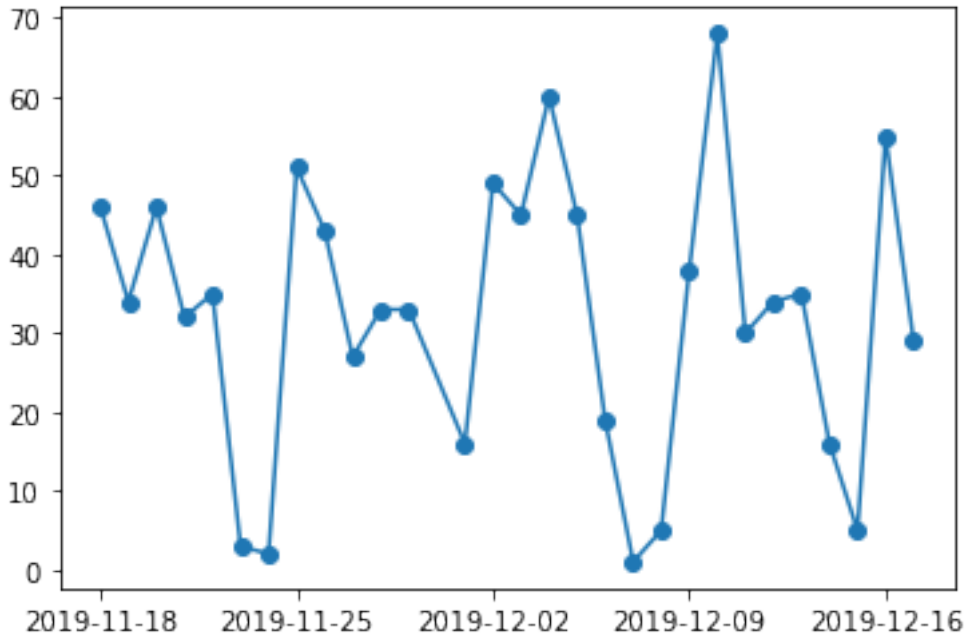
[4]:
```
plot_log_files_starting_with('message')
```



0.3 Liczba wpisów logów frontend.*.log

Dodatkowe logi frontendowe (najczęściej zawierające wystąpienia błędów) przesyłane poprzez dedykowaną usługę

[5]:
```
plot_log_files_starting_with('frontend')
```



0.4 Liczba wpisów logów braintree.*.log

Integracja z systemem płatności Braintree

[6]:
```
plot_log_files_starting_with('braintree')
```