Sensor Music Player

István Szőllősi

Faculty of Sciences and Letters, "Petru Maior" University of Târgu Mureș

August 25, 2018

Contents

1 About

The project is committed to the GitHub, you can find here.

The main structure of the repository is *a valid Android project* with several additionals folders, like the:

- **backend** folder where the *Python* and *JavaScript* codes are stored
- **docs** folder where the documents about the project are stored

2 Node.js

In Node.js is very simple to create a small web server for REST calls.

2.1 Installation

2.2 Usage

To start use the follow command in the project's folder:

```
npm run start
```

2.3 Configuration

Used tutorial: Build Node.js RESTful APIs in 10 Minutes

2.3.1 Mongoose

2.3.2 Express

2.3.3 Nodemon

3 MongoDB

MongoDB to store signal data from the *Y axis* of the accelerometer from the Android devices.

3.1 Installation

3.2 Runing

To start mongo service use the follow command:

```
sudo mongod --config /etc/mongodb.conf
```

To access the mongo shell enter the *mongod* command in terminal.

3.3 Drop collection

Code:

```
show dbs
use <db>
show collections
db.<collection>.drop()
```

Listing 1: MongoDB shell commands to drop a collection

4 Matplotlib

Matplotlib is a plotting library for the Python programming language. I used *matplotlib* to generate graphical view of my series.

4.1 Installation

Install module using this tutorial.

Optionally, you need to install the **python-tk** package also.

4.2 Examples

According to this official tutorial you can easily generate a plot about an array using this Python script:

```
import matplotlib.pyplot as plt
plt.plot([ 5.733, 1.704, -2.713, -1.343, 2.604, 3.922, 2.157, -0.910, -2.414,
    -2.943, -1.526, -0.823])
plt.ylabel('some numbers')
plt.show()
```

Result:

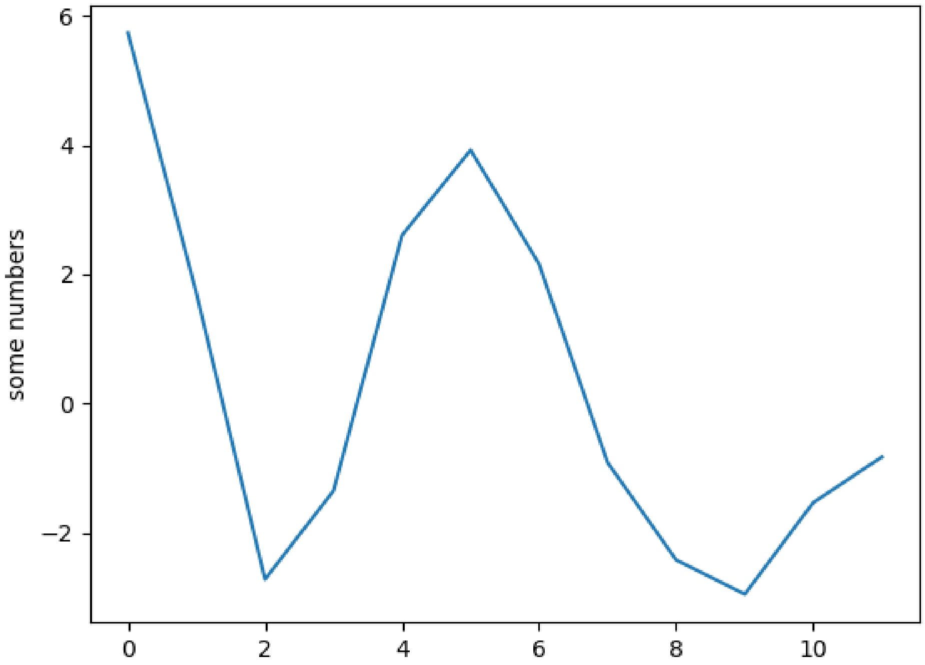

Figure 1: The result of the script

This example was very easy, so here is a *normal* signal from the accelerometer:

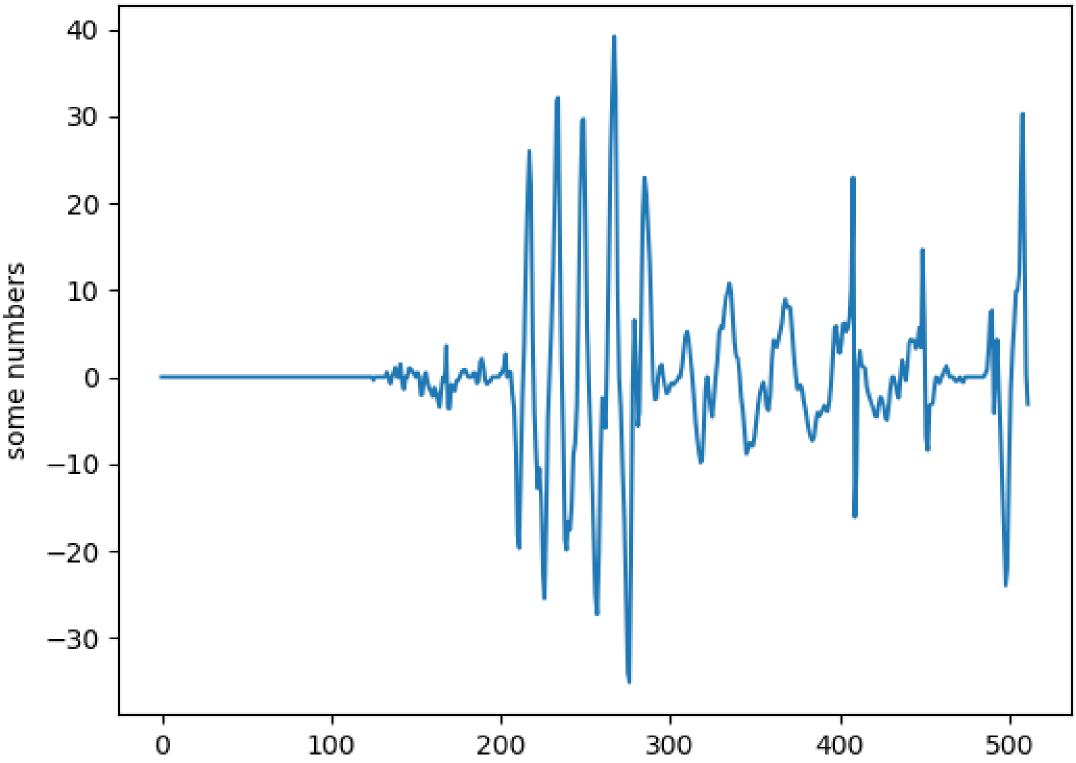

Figure 2: A section of the signal of accelerometer in real usage

5 DTAIDistance

Library for time series distances (e.g. Dynamic Time Warping) used in the DTAI Research Group.

DTAIDistance official documentation: dtaidistance.readthedocs.io

Source available on https://github.com/wannesm/dtaidistance.

5.1 Installation

Run the follow codes in terminal to install the module:

```
sudo apt install python3-pip python3-setuptools  python3-dev python3-tk
pip3 install wheel
pip3 install dtw
pip3 install dtaidistance
```

Listing 2: Used Linux Mint 19

5.2 Usage

```
from dtaidistance import dtw
from dtaidistance import dtw_visualisation as dtwvis
import numpy as np

s1 = np.array([0, 1, 2, 1, 0, 2, 1, 0, 0])
s2 = np.array([0, 1, 2, 1, 0, 0, 1, 2, 1])

d, matrix = dtw.warping_paths(s1, s2, window=25, psi=2)
print('DTW distance = ', d)

dtwvis.plot_warpingpaths(s1, s2, matrix, best_path, "image.png")
```

Listing 3: Script to calculate DTW distance and plot warping paths matrix

The result will be put in the newly created **image.png** file.

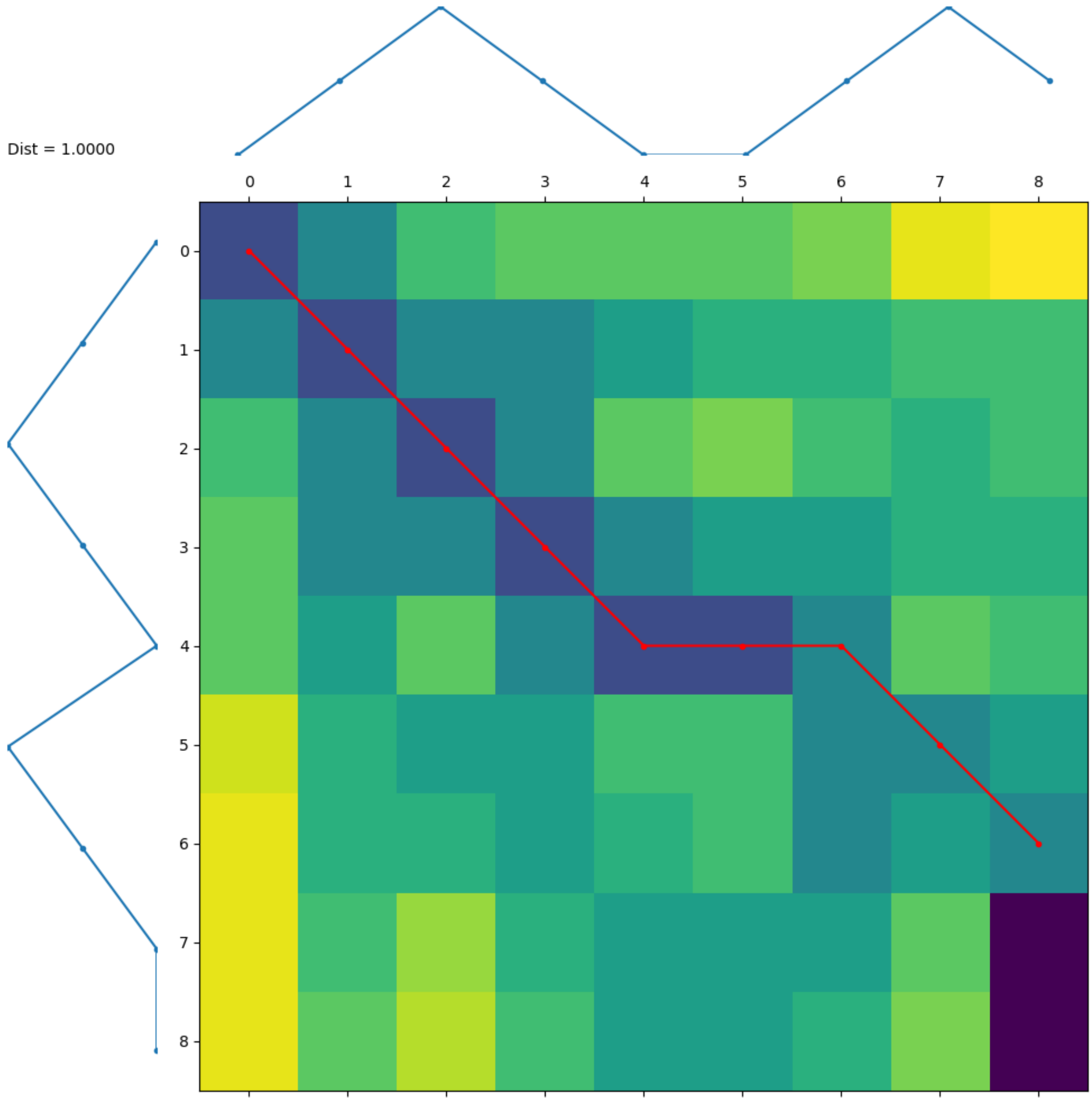

Figure 3: DTAIDistance's warping paths matrix

5.3 Documentation

```python
import numpy as np
s1 = np.array([0, 1, 2, 1, 0, 2, 1, 0, 0])
```

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

The *array()* function creates an array from a given list. See more details here.

```python
from dtaidistance import dtw
d, matrix = dtw.warping_paths(s1, s2, window=25, psi=2)
print('DTW distance = ', d)
print('DTW matrix = ', matrix
```

The *warping_paths()* function calculates the DTW distance and the DTW matrix for the two given series. See more details here.

```python
from dtaidistance import dtw_visualisation as dtwvis
dtwvis.plot_warpingpaths(s1, s2, matrix, best_path, "image.png")
```

The *plot_warpingpaths()* method plots the warping paths matrix into the *image.png* file. See more details here.

6 Postman

6.1 Installation

Installed according to this article: How to install Postman native app in Linux Mint 18.3

Used to test the main functionalities of the Node.js server.

6.2 Usage

6.2.1 GET

To get all buffers from database run this code in Postman/Linux terminal

```
curl -X GET http://localhost:3000/buffers
```

Listing 4: Get all buffers

The response is or an empty list, if no items in the database or a list like this:

```
[
    {
        "value": [
            5.733050346374512,
            1.704751968383789,
            -2.7134790420532227,
            -1.343064308166504,
            2.6042985916137695,
            3.92281436920166,
            2.15725040435791,
            -0.9106369018554688,
            -2.4146032333374023,
            -2.943338394165039,
            -1.5269522666931152,
            -0.8230304718017578
        ],
        "_id": "5b82607f5601ec575d3bf0e4",
        "__v": 0
    }
]
```

Listing 5: A sub section of the signal to process

6.2.2 POST

Post a new buffer a.k.a a sub section of the signal to store and process. Run this code in Postman or in a Linux terminal to post a new buffer to the Node.js server.

```
curl -X POST http://localhost:3000/buffers  -d '{
  "value":[-2.2, -1.1, 0, 1.1, 2.2]
}'
```

Listing 6: Send signal data via REST

7 PyMongo

7.1 Installation

```
pip3 install pymongo
```

Installing with pip: http://api.mongodb.com/python/current/installation.html

7.2 Usage

```
import pymongo
from pymongo import MongoClient
client = MongoClient('localhost', 27017)
client.database_names()
db = client['<db_name>']
db.collection_names()
coll = db['<collection_name>']

for col in coll.find({}):
    for keys in col.keys():
        print ('{', keys, ":" , col[keys] , '}' )
```