VILNIUS UNIVERSITY
FACULTY OF MATHEMATICS AND INFORMATICS
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF COMPUTATIONAL AND DATA MODELING

Bachelors Thesis

Implementation of application for visualization of regularities and randomness in data

Done by:
Audrius Baranauskas signature

Supervisor:
dr. Tadas Meškauskas

Vilnius
2021

Contents

Keywords

Pateikiamas terminų sąrašas (jei reikia)

Abstract

Santraukos tekstas rašto darbo kalba...

Santrauka

Darbo pavadinimas kita kalba

This is a summary in English...

Introduction

Signals can be observer all around us. For example, measuring the time taken between a weight-driven pendulum clock's ticks produces a signal. It does not require a great deal of effor to image how such a signal behaves. We would expect the clock's pendulum to swing back and forth, each time travelling a minutely shorter distance until the pendulum stops completely. Analysis of even a part of such a signal can help us determine the pendulum's position far into future.

Now consider a more complex siganl: the rates of a stock market. People have been analyzing this data for decades, grasping to predict its future state. For the scope of this paper, we defined the term signal processing as *the science of analyzing time-varying processes* [4].

In this thesis we analyzed the non-triviality of digital sygnals. Certain signals can be classified as simple (relatively trivial), like the aforementioned clock's pendulum. A more complex (non-trivial) signal would be the rates of a stock exchange.

1 Signal processing and Recurrence plot

1.1 Signal processing

A signal is a function that conveys information about the behaviour of a system or attributes of some phenomenon [6]. For example, measuring the time taken between a weight-driven pendulum clock's ticks produces signal. In turn, for the scope of this paper, we defined the term signal processing as *the science of analyzing time-varying processes* [4]. By processing a signal we analyzed the non-triviality of a given signal. Analyzing a signal reveals that some signals have properties that can be categorized.

1.2 Signal property categories

We have considered the following categories:

1. Stationary and non stationary signals
2.

Signals have varying properties. Some consist of simple repetitions while others have no apparent patterns. For example, measuring the time taken between a weight-driven pendulum clock's ticks produces a relatively simple (trivial) signal.

2 Web application development

This project is aimed at creating a web application allowing one to interact with the recurrence plot algorithm in a user friendly manner. The project offers a feature of classifying data based on the generated plot using convolutional neural networks. This is an effort to further spread the popularity of this algorithm and help users intuitively grasp how it behaves.

2.1 Analysis of analogous tools

As of the date of publishing, only one tool was located capable of generating a recurrence plot online [5]. There are multiple implementations of the recurrence plot in Python as well as other languages, but none offer the ability to classify data based on the generated image.

It is noteworthy, that the aforementioned implementations require at least a minimal undertanding of software programming, a computing machine and specific software to compile and run the code. This is laborious and is not likely to attract new users to experiment with algorithm. Based on these factors, a decision was made to create a web based application that requires as little user knowledge to get started with the algorithm as possible.

2.2 Architecture

This project uses the microservices. Microservices are small autonomous services deployed independently, with a single and clearly defined purpose [2]. This design approach was chosen due to the flexibility and scalability associated with the architecture. The nature of microservices allows one to easily test, modify or out right replace each one of the components giving more freedom to the developer. The project ecosystem consists of the following microservices:

1. Front end web application
2. Back end for the web application
3. Python web server for plotting operations

Communication between microservices is performed via HTTP requests. In general, a query with JSON body is sent to a service and a JSON response along side an image attachment is returned. Figure 1 illustrates the microservice architecture of the project and data flow within each service.

2.3 Project structure

The project is structured so that each microservice resides in an independent directory:

- app/
  - src/
    - components/
  - public/
- server/

  - db/
  - public/
  - utils/
- plotter/
  - jupyter/

As listed above the structure may be confusing, therefore we will now go more in depth into each one of the services.

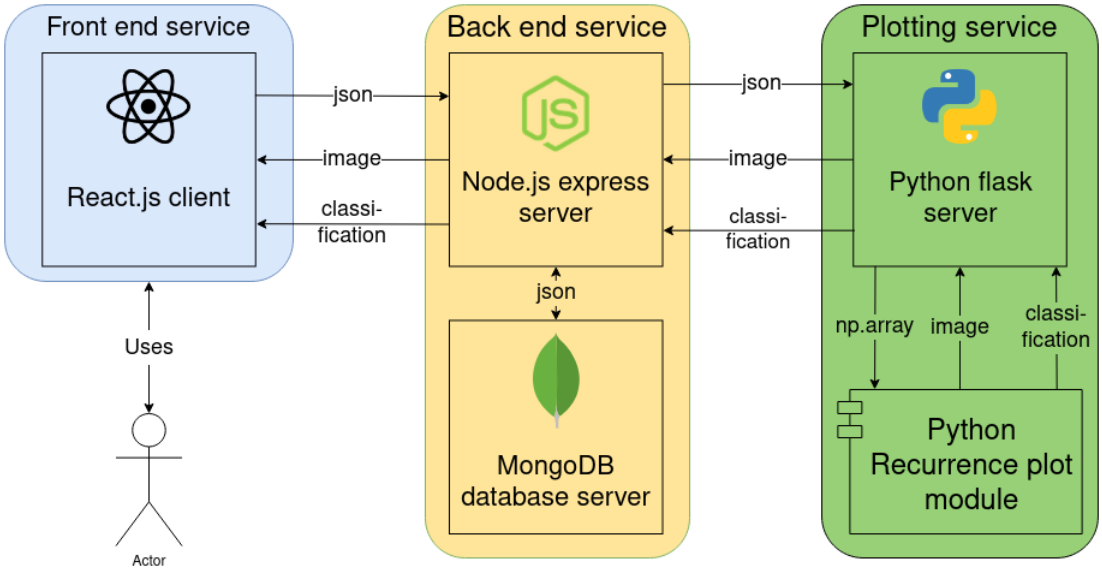

Figure 1. Microservice architecture structure

2.4 Microservices

The tools used for microservice development were largely open-sourced and relatively modern. Front end and Back end services were written in javascript based environments - React JS and Node JS respectively. These choises were made due to the widespread use of javascript in modern web application development providing a large pool of open-sourced libraries and tools.

On the other hand python was the tool used to develop the plotting service. It is known to perform better on data handling and machine learning than javascript alternatives. [3] Both Python and Node JS have certain strengths and thus have appropriate community driven libraries and modules to reinforce their leverages in appropriate operations.

2.4.1 Front end service

The front end service is developed using React - A JavaScript library for building user interfaces. [1] Following the best practices of React development, the app is into reuseable components. Each component returns a JSX object

2.4.2 Back end service

2.4.3 Plotter service

3 Pirmasis skyrius

asa [**?**]

3.1 Pirmojo skyriaus poskyris

Pateikiamas 3.1 poskyrio tekstas. Vienas iš šaltinių [**?**]. Visas [**?**] turinys priklauso 3 skyriui.

3.1.1 Pirmojo skyriaus pirmo poskyrio poskyris

Pateikiamas trečio lygio poskyrio tekstas.

$$x = \sum_{i=1}^{N} m_i \tag{3.1}$$

Table 1. Lentelė ...

test	test
test	test

Sprendimas pristatomas 1 algoritme, o įgyvendinimas -- 1 išeities kode.

Algorithm 1. Algoritmas uždavinio sprendimui

Require:
Ensure:
a **and** b

Listing 1. Pagrindinio metodo žingsniai

```
1 public static void main(String args[]){
2 }
```

Conclusions and Recommendations

Išvados bei rekomendacijos.

Ateities tyrimų planas

Pristatomi ateities darbai ir/ar jų planas, gairės tolimesniems darbams....

References

[1] Facebook Inc. React is a javascript library for building user interfaces. https://github.com/facebook/react, 2020. [Online; accessed 2-Jan-2021].

[2] L. Krause. *Microservices: Patterns and Applications: Designing Fine-Grained Services by Applying Patterns*. Lucas Krause, 2015.

[3] Miśtal Krzysztof. Performance comparison: Javascript vs. python for machine learning. https://dlabs.ai/blog/performance-comparison-javascript-vs-python-for-machine-learning/, 2020. [Online; accessed 2-Jan-2021].

[4] R.G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall professional technical reference. Prentice Hall/PTR, 2004.

[5] Norbert Marwan. Recurrence plots and cross recurrence plots. http://recurrence-plot.tk/online/index.php. [Online; accessed 2-Jan-2021].

[6] R. Priemer. *Introductory Signal Processing*. Advanced Series In Electrical And Computer Engineering. World Scientific Publishing Company, 1990.

Appendices

Dokumentą sudaro du priedai: A priede ....

A Pirmojo priedo pavadinimas

Pirmojo priedo tekstas ...

B Antrojo priedo pavadinimas

Antrojo priedo tekstas ...