

UNIVERSITY OF TARTU  
Institute of Computer Science  
Enterprise Engineering Curriculum

**Anastassia Ivanova**

# **Automatization for Finding Supplementary Materials for University Courses**

**Master's Thesis (30 ECTS)**

Supervisor: Siim Karus, PhD

Tartu 2019

## **Automatiseerimine täiendavate materjalide leidmiseks ülikooli kursustel**

### **Lühikokkuvõte:**

Kraadi omandamine valitud valdkonnas võib olla suur väljakutse tudengite jaoks. Koos lihtsate ainetega, õppekavas tulevad ka selliseid, kus osad tudengid saavad läbi kergesti, kuid teised panustavad väga palju aega, et napilt läbi saada. Üks võimalikest põhjustest, miks mõnedele õppijatele osad kursused valmivad nii palju raskusi, on kvaliteetse ning tudengite jaoks kättesaadava materjali puudus. Abi otsimisel pöörduvad tudengid sageli otsingumootori poole, mis leiab nendele artikleid, kus leidub vastuseid igale tudengi küsimusele. Tihti Google leitud materjal sisaldab ebakorrektselt informatsiooni ning kuna tudeng ei oska ise eraldada kvaliteetset õppematerjali ebakvaliteetsest, siis tulemusena saab tudeng teemast aru valesti ja satub segadusse.

Käesoleva töö eesmärk on luua portaal, mis otsib internetis kvaliteetset õppematerjali jooksva semestri kursuste jaoks. Portaali automaatselt uuendab kursuste nimekirja enne uue semestri algust. Pakutud rakendus kogub ning analüüsib erinevate ainete tekstilist materjali. Seejärel toimub andmete puhastus ning teisendamine analüüsimiseks sobivasse formaati. Edasi otsitakse Google vahendusel seotud materjale, filtreerides välja mitte sobivaid teemasid, ebakvaliteetseid ning ebausaldusväärseid materjale, ning tagastatakse tudengile tulemuste nimekiri õppimiseks ja lugemiseks.

Selle lõputöö tulemuseks on veebirakendus, mis leiab lisamaterjali iga aine jaoks jooksvas semestris. Algoritm selle teostamiseks on töötav, kuid vajab veel täiendamist, et leida materjale efektiivsemalt ja kiiremini. Praegused algoritmi leitud materjalid ei pruugi vastata ülikooli tasemele või hoopis otseselt mitte puutuda kursuse teemasse, mille jaoks teostatakse otsingut.

### **Võtmesõnad:**

tekstianalüüs, masinõpe, otsing, õppematerjalid

### **CERCS:**

P170 (Arvutiteadus, arvanalüüs, süsteemid, kontroll, soovitusüsteem, õppematerjalid)

# **Automatization for Finding Supplementary Materials for University Courses**

## **Abstract:**

Gaining a degree can be a great challenge for students. Along with easy ones, there are courses that may be easy for some students or can be a great challenge for others. One of the possible reasons for such difficulties could be lack of proper studying materials available for course takers. When looking for help, students often turn to search engines that find articles which have answers to every question that a student may have. However, most materials found via Google contain inaccurate information, and since students are not able to separate high-quality teaching materials from the low-quality one, the subject of the course is misunderstood and the student gets confused.

The purpose of this thesis is to create a portal that searches for high-quality teaching materials on the Internet for each course in the current semester. The portal will automatically update the course list before the start of the new semester. The proposed application collects and analyzes the textual information from the course. Then the data is cleaned and converted to a format suitable for analysis. The next step is to search for materials via Google by filtering out non-related topics, low quality materials and unreliable sources, and returning to the student to read the best results.

The final result of this thesis is a working web application that performs a search for high-quality teaching materials via Google engine for each course in the current semester. The algorithm that has been created for this purpose, however, requires improvement to work faster and find more suitable materials for students. Currently, the final output of the algorithm may contain articles that are too simple to the University student or does not refer to course subject at all.

## **Keywords:**

text analysis, machine learning, search, study materials

## **CERCS:**

P170 (Computer science, numerical analysis, systems, control, recommendation systems, study materials)

# Contents

1 Introduction	<b>6</b>
1.1 Other recommendation algorithms review	8
2 Architecture and algorithm	<b>11</b>
2.1 Flow of algorithm	11
2.2 Application architecture	12
3 Data scraping and preparation	<b>14</b>
3.1 Changes in Python code	14
3.2 Python integration	15
3.3 Saving tokens	17
3.4 Selecting tokens	18
4 Google engine	<b>19</b>
4.1 Search configuration	19
4.2 Results collecting	21
4.3 Processing Google articles	22
5 Dimensionality reduction	<b>24</b>
5.1 Converting articles to numeric format	24
5.2 SVD matrix	24
5.3 SVD vs PCA	26
6 Analysis and Assessment	<b>28</b>
6.1 Analysis and Result generation	28
6.2 Saving supplementary materials	32
6.3 Results validation	32
7 Front-end side of the application	<b>39</b>
7.1 Courses list	39
7.2 Course similarities	39
8 Future work	<b>43</b>
8.1 Fake news algorithm	43
8.2 Google scraping	43
8.3 Personalization	44
8.4 Integration with SIS	44
8.5 The optimization and securing	44
9 Conclusion	<b>47</b>
References	<b>48</b>
Appendix	<b>51</b>

I Glossary	51
II Deployment instructions	52
Server-side	52
Client-side	54
III List of configured parameters	55
IV SVD matrix example	57
V Licence	64

# 1 Introduction

Earning a degree is a great challenge for every entrant: students are asked to fulfil the curriculum, prove learned knowledge and skills, pass exams and submit the final thesis. The studying process for some students goes relatively easy and stressless. However, others cannot resist the pressure and give up. For example, in 2014 there were 1 927 students, who started a bachelor's degree at the University of Tartu [1]. Three years later only 1 258 students gained a degree [1] e.g. ca 65.3% of the accepted students in 2014. One of the reasons is that there are courses in each curriculum, which offer more challenges for some attendees. As there is a large number of students for each course, lecturers and practical supervisors cannot physically pay attention to every person to help to understand the materials. This leads to a lack of knowledge for students and results in the failure of the course.

The goal of this work is to build a portal that collects materials of the courses which are carried out in each ongoing semester, analyzes their content and finds high-quality supplementary study materials for them. The aim is to provide a portal, where, it is possible to find literature from the Google search engine for the specific course. The algorithm will extract the course related data from the `courses.cs.ut.ee` website, clean it up, transform into a suitable format for the analysis and compare to the data found on the web. The most related materials will be displayed in the table as a link for the student.

The entire application, including web scraping of the courses in the current semester, Google search and analysis, was implemented using Java 8 in server and Angular 6 in the frontend. The process is fully automated. Every first day of each month at 5:00 AM the CRON server runs the algorithm to scrape data of each course in the current semester, saves tokens for each course in a separate .txt file and then starts recommendation algorithm, which output is also saved in each separate file. This flow speeds up the process for end-user so that when the student wants to see material suggestions, the application simply reads already found articles from the .txt file and delivers the list in table format.

The tokens for the University courses are collected and preprocessed via earlier implemented Python API by Ragnar Vent in his Master's thesis in 2017 [2]. Nevertheless, there is much work left to write the algorithm that would be able to find high-quality supplementary materials that are suitable to be used by students of the University as a virtual helper in their studies. This thesis describes the contribution of the author into the development of this algorithm starting from correcting and improving Python code written by Ragnar Vent [2] to better suit this algorithm's needs, connecting to Google search engine, validating and scraping articles suggested by Google, transforming articles from raw text to cleaned and preprocessed tokens, converting tokens into suitable format for the analysis and finishing

with comparing the similarity between University course and possible supplementary materials.

The motivation to write this topic is an understanding of how much the good studying materials can give to the student and how vital it can be in passing the course. Being a student in both bachelor and master programs at the University of Tartu gave a background to write this kind of thesis and build a portal that could make studying easier. The idea is to build a web application, which each student can access and use to find more supplementary materials for each course in the ongoing semester. Currently, this application work only for the courses that are taught in the Institute of Computer Science.

The research question of this thesis is to find out whether an automated algorithm can find relevant supplementary materials of a good quality for the courses that are taught in the Institute of Computer Science from the web based solely on the keywords used in the course materials. High-quality materials provide additional information for the particular topic, contains legitimate data from the trusted source and is fully relevant to the content of the topic that this material has been looked for. Hypothetically, by properly collecting, tokenizing and cleaning content from the courses in the Institute of Computer Science and later using some of them as a keywords for Google search, and, after that, scraping and preprocessing articles found via Google engine the same way the course content has been preprocessed, it is possible to accurately measure the similarity between the contents of the particular course and all discovered candidates for the supplementary material. With an adequate assessment of the measurement results, it is possible to filter out suitable literature that could serve as additional reading for the students to help them better understand the content of the course. That is if the chosen tokens, with which Google performs its search, have been chosen efficiently and really are describe the content of the particular course.

The thesis starts with the introduction and a brief overview of the popular recommendation algorithms of different applications. The second section of this thesis introduces the overall structure of the application and describes the main steps used in the algorithm. The following chapters provide a more detailed overview of each step in the entire process. Firstly, it is explained how python code is executed in Java, because data collecting, extracting and cleaning are performed using a modified version of a Python 2.7 API written by Ragnar Vent [2]. Moreover, there are notes describing what has been changed in Python code to make it more suitable for the purpose of this application. Next step is selecting tokens that will be used for searching related materials. The selected tokens are used by the Google search engine to find candidates for supplementary readings. Once google results are extracted, the received data is cleaned and tokenized. After data preprocessing is finished, the output is transformed into a vector that is used in future analysis. The fifth chapter focuses on converting the high dimensional space to the 2-dimensional space using Singular Value Decomposition (SVD). Finally, to measure the similarity between the two articles, the Cosine similarity formula is applied. All result are sorted and the algorithm returns the output. At the

end of Chapter 6, the author assesses the results and provides the judgement of the algorithm by describing whether the algorithm fulfils its goal or not. Chapter 7 focuses on the frontend side of the application and briefly described the additional feature implemented for this thesis. The next chapter contains the suggestions of what can and should be done to improve and optimize the work of the algorithm and the application. The final Chapter 9 sums up all the information provided in this thesis.

## **1.1 Other recommendation algorithms review**

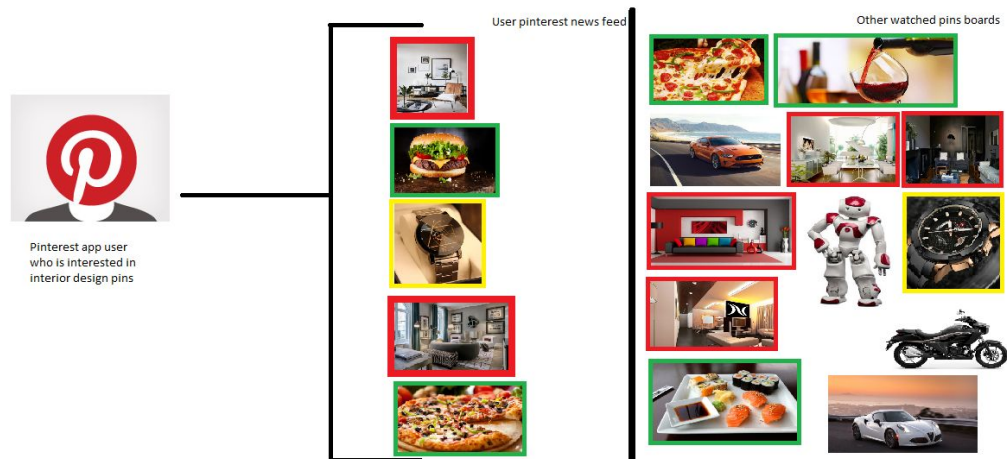
The application built for this thesis serves as a recommendation system which looks for the additional reading materials for each course in the ongoing semester in the Institute of Computer Science. Below is a small overview of other popular recommendation systems and application which use them.

Recommendation systems have become very popular in different areas (advertising, online shopping, movies and songs searching, online books databases etc.). The content-based similarity approaches are used, for example, by Netflix (suggesting movies or TV shows to users), AliExpress (discovering related products), IMDB movie database and Pinterest (searching subject or theme pins). This feature made the usage of these platforms more pleasant for the end users as they are able to discover more information on the active topic without consuming time for searching. There are many proven examples, that applying recommendation systems can significantly improve the product value provided to the customers. For instance, consumer research carried out by Netflix has reached a conclusion that regular user loses interest in the list of movies after 60 - 90 seconds after having reviewed around 10 - 20 titles [4]. For Netflix, it is either that the user finds eventually something of interest or will abandon the provided service. The recommender reduced that risk by providing content of customer's possible interests. For example, if the customer was interested in a particular movie, Netflix will later suggest the user movies with similar content. As a result, the customer would not have to parse the list of random movies but will pick up from the set of videos that are similar to the movies of the interest. Historically, Netflix's recommendation system was mostly a prediction model of how a particular user was going to rate certain movies or TV shows [4]. Back in days when Netflix started to send DVDs to their customers for rent, they realized that they would need a much stronger recommender because a star rating was the main feedback for their service. They even organized a competition called Netflix prize in 2009 to improve the accuracy of their rate prediction algorithm [4]. Nowadays, the recommendation system of Netflix consists of various algorithms that process a vast amount of data. This data describes what, when, how and where each user watches videos. Netflix uses Singular Value Decomposition (SVD) and Restricted Boltzmann Machines (RBM) [5] to decrease root mean squared error (**RMSE**) value from 0.9525 to 0.8572. RMSE is a widely used metric to evaluate the algorithm's accuracy. The whole net of recommendation features includes Personalized Video Ranker a.k.a. PVR (which sorts the entire catalogue of movies by genre for each user account) and

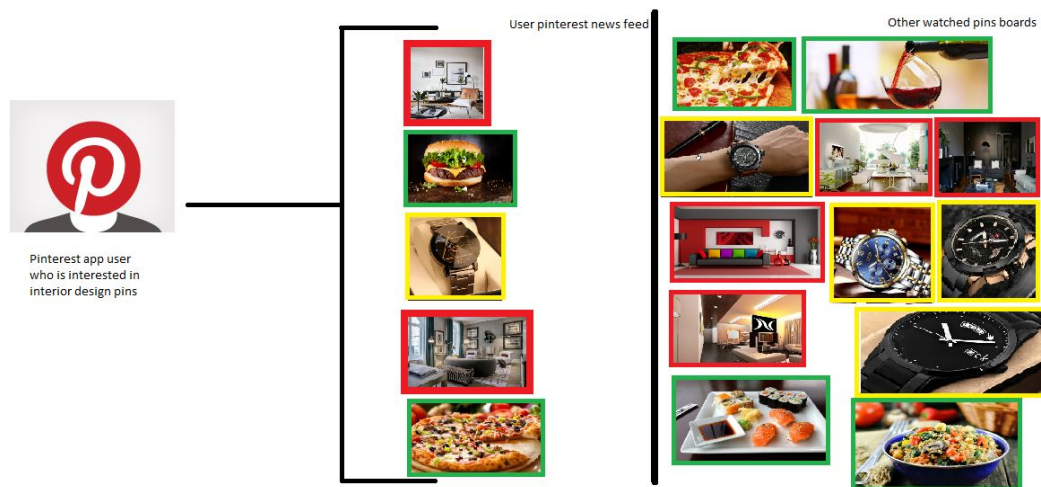
Top-N Video Ranker (that finds few the best suggestions for each member focusing only on the head of the ranking). Other features provided to the users are: Trending Now (a short-term temporal prediction of videos that user is likely to watch), Continue Watching (that analyzes the subset of recently viewed titles based on the prediction of whether the member intends to resume watching or not), Video - Video Similarity or Because You Watched a.k.a. BYW (that categorizes content based on what videos the user has watched) and Page Generation (that selects and orders rows from a pool of candidates to create an ordering optimized for relevance and diversity for each member). The page generation algorithm uses the output of all the algorithms mentioned above to construct every page of recommendations, taking into account the relevance of each row to the member as well as the diversity of the page [4]. Together these algorithms complete the recommendation system of Netflix and provide evidence of how recommenders can improve user experience nowadays.

Another famous application which actively uses recommendation algorithm is Pinterest, which operates a software system created to find information on the internet, mainly using images and, on a smaller scale, GIFs and videos as an input. Their web-scale recommendation system - Related Pins, an item-to-item recommendation system that uses collaborative filtering [7] - was firstly introduced in 2013 and now powers over 40% of user engagement on Pinterest [6]. Users save content as pins and have sets of these pins on boards. Related Pins analyzes this content to provide recommendations based on a given query [6]. In the beginning, they used board co-occurrence for generating candidates [7]. For instance, if pin X has been saved to N boards, then Related Pins picked candidates from other pins of these N boards. However, this approach has such disadvantages like boards drifting, granularity and segmentation. To resolve these issues, developers have built the Pin2Vec deep learning algorithm to put all pin candidates into 128-dimensional space. Training data is collected from the user's recent activity (saving, clicking, sliding window). Each pin has a unique ID. The learned groups of pins are put together according to a pinner's recent engagement. All related pins are closer to each other in the high dimensional space [7]. Finally, the N nearest pins for each of the category are chosen by their distances. This approach is proved to be more accurate than using board co-occurrence signals. In the Pinterest data model, each pin is represented as an instance of the image, which is uniquely identified by the image signature, and description [6]. The same image can be used across different boards: when a pin is saved to a new board, a copy of the pin is created. The approach used by Pinterest consists of two basic steps: Candidate Generation and a combination of Ranking and Memboost. Firstly, the algorithm narrows down a set of eligible pins from billions of candidates roughly to 1000 pins that are most likely to suit for Related Pins recommendation system. Next, a machine learning ranking model uses the query, the set of candidates filtered out during the previous step, the user profile, the session context and Memboost signals [6], which is memorization of previous requests on queries and result pairs. Considering the above-described example with the Pinterest application, the usefulness of a recommendation system (Related Pins in this case) can be visualized with Figures 1 and

2. Figure 1 shows how the user would see suggested pins without applied recommendation system and Figure 2 describes how the applied algorithm changes the set of pins for the user.



**Figure 1. The Pinterest application without Related Pints recommendation system.**



**Figure 2. The Pinterest application with Related Pins recommendation system.**

In the figures above, the first column of pictures is a so-called user's news feed on the Pinterest application. The second panel is images that the user has watched in Pinterest boards of images. In Figure 1 we can see that there are images on the user's news feed that are not related somehow to the images from Pinterest boards which were of interest of the certain user. However, in Figure 2, the images in the user's news feed are related to the images that the user has watched himself - food, watched, interior etc. Therefore, it can be clearly seen that in the case of Pinterest Inc. the applied Related Pints recommendation system significantly improves the user experience by suggesting images about the themes that user has been already interested in. Also, Related Pints saves customer's time spent on searching pins of interest. Due to these changes, the value of the Pinterest application also increased.

## 2 Architecture and algorithm

### 2.1 Flow of algorithm

As it is mentioned earlier in the Introduction section (see Page 5), the search of the supplementary materials is fully automated. The input of this algorithm is a set of tokens for each course in the ongoing semester. The set of tokens are received from Python code of Ragnar's thesis [2] (see Chapter 3 for further details). Once the tokens are collected and saved, CRON starts the algorithm to find possible materials for each course. The general flow of the entire algorithm is provided below step by step.

**STEP 1.0:** Getting tokens for the specified course.

**STEP 1.1:** The algorithm checks if the tokens for the course already exist.

**STEP 1.2:** If tokens exist, the set of tokens is read and the algorithm moves on to *STEP 2.0*. If tokens for the specified course have not been found, the algorithm calls the Python method to collect tokens only for the specified course.

**STEP 1.3:** Randomly choosing tokens to use in Google search. The reason for choosing in random order is explained in section 3.4 of this thesis.

**STEP 2.0:** Perform a Google search.

**STEP 3.0:** Each validated article found via Google search is converting to a numeric vector (see Chapter 5.1). The numeric vectors along with other essential information about the found articles (title and URL) is put into a map `Map<String, Map<String, double[][]>> vectors`. Note that the value of the second Map is a matrix, not the array. For better data accessibility, the numeric vector of the original input (a.k.a the specified course) is stored together with every numeric vector of the potential candidate (a.k.a a potential supplementary material). So the first column of the matrix is the numeric vector of the original input and the second column is a numeric vector of the potential candidate.

**STEP 3.1:** Preparing for *STEP 4*. Creating a map that contains all the necessary information (for example, article's title, URL and numeric vector) about the article found via Google search to simplify data access at the end of the algorithm. The final output of this sub-step is `List<Map<String, double[]>> input`, where each map contains only two entries - information about the specified course and the possible candidate as supplementary material. The key of the map is a title and its value is a numeric vector.

**STEP 3.2:** Randomly choosing a limited number of entries from the map prepared in *STEP 3.1* for further analysis. This step is required to ensure that the flow will not take too much time. The number of randomly chosen entries is a configurable parameter called *candidateArticlesLimit*. By default, the limit is

set to 50 articles, so that 50 or fewer articles will be randomly chosen for further steps of the algorithm.

**STEP 4.0:** Computing SVD matrix for each entry received from the previous step. The result of this step is also a list of a map `List<Map<String, double[][]>> result`, but, in this case, the map's value is SVD matrix instead of the numeric vector. Here it is important to pay attention that instead of the numeric vector, there is a matrix `double [][]`. This matrix is an SVD matrix of the two numeric vectors - one from the specified course and another is from potential supplementary material.

**STEP 5.0:** Computing Cosine similarity between the numeric vector of the courses tokens and tokens of the Google article. Each result is saved into the map `Map<Double, Map<String, String>> similarity`, where the key is a computed similarity and value is another map with the material's title as a key and its URL as a value.

**STEP 5.1:** Sorting the map made in *STEP 5.0* in descending order by the key, which is the computed similarity between the specified course and found material. So the first returned material suggestion is the best fit for the course lectures according to the algorithm.

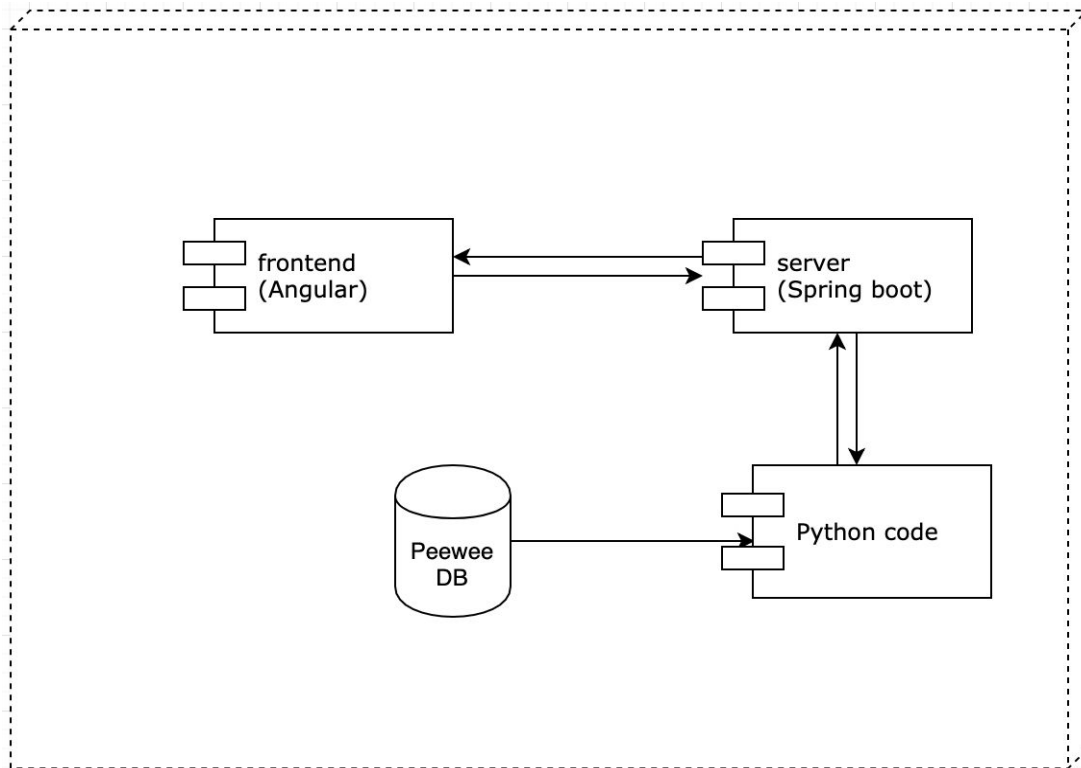
**STEP 6.0:** Returning the limited number of results to the end user. By default, the limit is set to 20. The limitation is applied because too much materials may have confused the student as it may be hard to decide which one to read.

The output of the algorithm is a list of found supplementary materials containing the title and the link of each of the found articles.

## 2.2 Application architecture

The architecture of this application is basic and has only three components: frontend side that is implemented in Angular 6, server-side that uses Spring Boot framework with Java 8 and Python code from the Ragnar Vent's Master's thesis [2]. As all data that is used by the application, as well as the data that is returned by the application is dynamic, there is no database used connected to Java server-side of the application. All information like course tokens and found materials are stored into .txt files. This way it is easier to manage (delete and edit) this information. However, Ragnar Vent's Python code uses peewee database to store information about scraped courses and their lectures. The Java code component consists of controllers, DTO-s and service classes. Main of them refers to sub-components like Google search, Cosine similarity computation, Web scraping and Tokens preprocessing services. The sketch of the project's architecture is provide below in Figure 3.

The architecture of this application has been composed in this particular way and each service performs specific purpose from web scraping, cleaning, preprocessing, converting the set of tokens to the numeric format, raw data extraction, scheduling to Cosine similarity computation and Google search.



**Figure 3. The architecture of the application.**

The application is designed so that each of the services was able to interact with other services easily and quickly without extra configuration. This is why Spring Boot framework has been chosen as a base for the server side of the application. Angular 6 has been chosen as a frontend side platform, because of its light-weight and scalability. It is a modern tool allowing to build an application for both web and mobile platforms.

### 3 Data scraping and preparation

The cleaning of the input data is the first step in data preprocessing. Noisy (with errors), incomplete (missing values or containing only aggregate data) and inconsistent (containing discrepancies) are commonplace in large real-world databases and data warehouses. There are numerous reasons why data can be garbaged. For instance, software malfunction or data considered not to be important at a certain time or not containing full information etc. This chapter shortly describes how data collecting and preprocessing has been implemented in this application. The primary goal of this work is to provide a framework with applied content-based similarity algorithm for finding suitable study materials to students attending courses in the Institute of Computer Science at the University of Tartu. This is achieved by using data mining techniques combined along with data processing methods such as data gathering, cleaning and analyzing. Textual analysis has already been implemented previously by Ragnar Vent in his Master's thesis "A Framework for Analysing Topics in University Courses" [2], where the whole server side of the application is implemented was Python 2.7. The purpose of Ragnar's work was to discover topic overlaps between university courses. He used LDA and Topic Modelling to identify which courses have similar materials. The first step of his algorithm is to collect data from specified resources, which can be `courses.cs.ut.ee` website, Moodle or SIS. Once data has been collected, it is cleaned and tokenized. Only after that, the actual analysis is done. In this thesis, the parts of Ragnar's code that have been used are web scraping, data cleaning and tokenization implemented in Python 2.7. Small adjustments have been made in order to make it more convenient to use in this application. The remaining part of this application has been implemented in Java. Integration between Python and Java as well as changes done in Ragnar Vent's code are described further in this work (see Chapter 3). The reason why the entire application has not been implemented in Java is that initially the whole algorithm has been done in Java. However, once Ragnar Vent's work has been analysed, it comes clear that the beginning of both algorithms is similar. Therefore, in order not to repeat the steps of Ragnar's algorithm in Java, it is easier to integrate Python with Java and re-use Ragnar's code.

#### 3.1 Changes in Python code

As it has been mentioned above, data gathering along with cleaning and tokenization is done using Ragnar Vent's code [2] ([https://github.com/ragnarvent/courses\\_analysis](https://github.com/ragnarvent/courses_analysis)). In order to make it more convenient for Python code to cooperate with Java code, minor changes in Python code have been performed. Here is the list of those updates:

1. In the tokenization process, if during the cleaning the list of the processed lecture tokens becomes empty, the program will output the lecture tokens before removing infrequent words. This is done to increase the chance for the algorithm to find any material at all.

2. A new command in Makefile similar to `scrape-courses` command has been implemented named `scrape-course`, which takes not only the semester as the argument but the course code as well. If the original command scrapes all courses in the specified semester, new command collects data only for the selected course.
3. Modified `scrape-courses` command: now it makes two courses dictionaries with different keys - course ID and course code. The first dictionary is required to make up acronyms used by Ragnar's analysis and the second one is required in the application as it helps to identify tokens per each course.
4. Modified Tokenizer module by replacing pool mapping with the general for-loop, because if the amount of the lecture's data is too large, the Python started quitting and the process was stuck. Python started to quit if total lectures count was 340 with total sentences count more than 10 000.
5. Modified `clean-stale` function by splitting its functionalities into two parts: cleaning the database and cleaning the raw data. The database cleaning logic has been moved into `cleandb.py` file, which is located in the root directory of the Python project. And the cleaning raw data has been left and modified so that it deleted all data in the `raw_data` directory. For this application, there is no need to store raw data as the application only uses data for the current semester.
6. Updated `requirements.txt` file as some requirement were incompatible. For example, `nlk` and `estnlk`. So, upgraded `nlk` from 3.0.1 to 3.1.

These minor changes are saved in a forked repository, which can be found here:

[https://github.com/jsutStacy/courses\\_analysis](https://github.com/jsutStacy/courses_analysis).

### 3.2 Python integration

As the main server-side logic in this application has been developed using Java 8, Python 2.7 needed to be integrated and called from Java to perform the first steps of the algorithm. As Python program is a standalone application, it can just be called from Java using the following piece of code:

```
Runtime rt = Runtime.getRuntime();
Process pr = null;
String[] cmds={**commands to execute**};
try {
    pr = rt.exec(cmds);
    pr.waitFor();
} catch (IOException e) {
    logger.error("" + e.getMessage());
} catch (InterruptedException e) {
    logger.error("" + e.getMessage());
}
```

The idea is basic: Ragnar's application can be executed via terminal and can be called either as a general command that performs all necessary steps to do the analysis or can call each step independently. For this application the chosen resources are courses.cs.ut.ee website, so the first step is to call the web scraping function that collects data from there. The three following Python commands are executed in Java using the code provided above. In this portal, we are only interested in the current semester, so the final command to be called is as follows:

```
"make scrape-course SEMESTERS="
```

where *semester* is a variable, which is populated by function `getCurrentSemester()`; and its output contains the current year and current semester (F if Fall and S if Spring). So the final output of the abovementioned function looks, for example, like "2018F". Once this command has been executed, the *raw\_data* directory is generated inside the root directory of the Python code, which contains all raw data for the selected course in the specified semester. On the frontend side of the application, the user can open complementary materials by clicking the button "View materials" (est. *Vaata materjalid*), which send a query to find the suggested materials located in the specified directory. The location of this directory is a configurable parameter called *materialsLocation* and its value can be changed in the *application.properties* file. If such materials for the chosen course have not been found, the system run algorithm again to search materials only for the specified course. In this case, the web scraping from courses.ut.ee website is called via the following command:

```
"make scrape-course SEMESTERS=" + semester + " COURSE_CODE=" +  
course_code;
```

which working principle is exactly the same as in the command above, but in this case, the materials are collected specifically for the specified course via a *course\_code* attribute. Next, data has to be extracted from the files and this is done via the below mentioned command:

```
"python -m extraction.TextExtractor"
```

which parses all raw files (.ppt, .txt, .pdf, .html) collected from the course website and extracts text from them. Finally, the textual context obtained in the previous step needs to be cleaned. For textual analysis data set must be transformed into a certain format. Data transformation process may include data normalization (data scaling within a specific range), aggregation (grouping data according to specific property) and generalization (replacing raw data with more abstract concepts). For example, a numeric attribute like age can be mapped to a higher level concept such as "young", "middle age" or "senior". For this purpose we can execute the command shown below:

```
"python -m cleaning.Tokenizer"
```

whose output is a list of pairs, where each pair contains a lecture object and a map, where the key is a token and the value is its frequency in the lecture. A sample output is:

```
[(<Lecture: 1>, {u'algoritmic': 1, u'wikitext': 1, u'exam': 2,
u'information': 1, u'pagetext': 1, u'content': 1, u'late': 1,
u'article': 1, u'appear': 1}), (<Lecture: 2>, {u'content': 1,
u'viited': 2, u'article': 1, u'pagetext': 1, u'wikitext': 1})]
```

This output is only valid for the application, if the algorithm has been given a course code as an input as in this case, the algorithm searches additional study materials only for this specific course. However, during the scheduling, the application scrapes all courses in the current semester, so it is essential to separate tokens for each course. Otherwise, it is impossible to find out what material suits which course. So during the scheduling, the algorithm runs the command to scrape all courses. The example command to scrap courses data from fall semester in the year 2018 is shown below:

```
make scrape-courses SEMESTERS=2018F
```

and a sample map of tokens per each course looks like follows:

```
{u'###LTAT.05.004': [{u'clsspacer': 1, u'veebileht': 1,
u'clsmaintable': 1, u'text': 6, u'global': 1, u'kujundatav': 1,
u'skript': 1, u'nbsp': 4, u'clsboddy': 1, u'keywords': 1,
u'kasutatavus': 1, u'kuluv': 1, u'tormiliselt': 1, u'xml': 4,
u'title': 2, u'sobilik': 1, u'varia': 1, u'rakendus': 3,
u'finally': 1, u'sqlexception': 1, u'black': 1, u'turvalisem': 1,
u'policy': 1, u'andmed': 2, u'arhitektuur': 2, u'contactus': 1,
u'meeskond': 1, u'teenindamine': 1, u'void': 1, u'keel': 1,
u'turvalisus': 2, u'digitaalkunstnik': 1, u'transitional': 1,
u'odbc': 1, u'server': 2, u'default': 1, u'admob': 1}]}
```

The scraping command during the scheduling is the only one that differs in the algorithm comparing to when the algorithm is run for a specific course. Other commands have exactly the same logic.

All next steps are performed entirely using Java 8 and Angular 6.

### 3.3 Saving tokens

After the lecture tokens for each course have been composed and ready to be used, they are saved into the directory specified by *tokensLocation* parameter into a .txt file, where the file name is the course code. For example, LTAT.03.005.txt. This step is done in order to speed up the search of the materials. As a first step, the algorithm checks if the tokens for the selected course have already been collected into the selected folder. If so, the algorithm uses the existing file, converts it into a list of strings and returns it as output for the next step.

These files are also used to rate similarities between courses percentwise (see Chapter 7.2). If the file with the selected course code has not been found in the specified folder, then the algorithm runs the Python code to scrap the data from `courses.cs.ut.ee` website to collect tokens.

### 3.4 Selecting tokens

As the average amount of cleaned tokens per course is about 10 000 words, it is impossible to use all of them for Google search and future analysis. So a limited amount of tokens must be selected. The number of tokens that will be chosen is a configuration parameter named *tokensLimit*, which value can be changed in the *application.properties* file along with other configuration parameters. Each chosen token is unique and is chosen in random order. Currently, this parameter's value is set to 5. The bigger this number is the more difficult it is for Google to find articles to analyze because the search engine looks for those articles that have all tokens in it (see section 8.2). The reason why tokens are chosen randomly and are not taken from the set of keywords from the course description is that usually the description of the course is brief and does not reflect all areas that are taught in this course. Some courses do not even have a description. Another option would be to split the subject into tokens and use this set for Google search. However, many courses have very general subjects like "LTAT.06.022 Andmeturve" or "MTAT.03.329 Bioinformaatika". These subjects do not open up what course is going to teach about exactly. Additionally, these subjects have only one word, therefore only one token, which is too few for Google search. Picking up tokens from the set of the all lectures tokens is a tricky move, but this gives an opportunity to widen the Google search area, therefore, to increase a possibility to find a high-quality supplementary materials. The random order is considered to be the best choice in this case due to the large size of the tokens sets scraped from the courses. Ragnar Vent's code selects the keywords that is believed to describe the course in the best way, but these results may be too general. For example, one of the keywords that has been chosen as a description for the course was "õppima" (eng. *study*). So, as the final set of tokens from the Ragnar's code may also contain not very exact keywords, it is better to expand the variety of choices to pick up from and use all preprocessed tokens. Another reason to settle with the random order to choose tokens is that the Google query must contain as few tokens as possible and as much as needed to be able to find relevant articles. As set of tokens per each course may have approx. 10 000 tokens, it is very hard task to narrow down only 5 tokens scientifically and semantically to describe all subject that this course may teach. This task could be one of the upgrades (see Chapter 8) to make the algorithm better.

## 4 Google engine

There are many approaches and libraries that can help to run the Google search in Java. For this application, it has been implemented using a web scraping approach via Jsoup library. Below there is a piece of code that has been used to connect to google engine and run the search query:

```
String searchURL = GOOGLE_SEARCH_URL +
"?q="+searchQuery+"&num="+limit;

Document doc = null;
try {
    doc = Jsoup.connect(searchURL).get();
} catch (IOException e) {
    logger.error("Viga otsimisel...");
}
```

where `searchQuery` is the Google URL containing tokens that have been selected from the set of all lecture tokens in the particular course and `num` is a google parameter that specifies the number of results that Google will display. In this application, this parameter called *googleLimit* and is configurable via the *application.properties* file. Currently, its value is set to 50.

### 4.1 Search configuration

Once the search tokens are selected, the next step is to run the Google search engine with the selected tokens. Since the Google search engine looks through a various amount of aliases, among which there can be also not very trustworthy websites, this application only accepts articles and files from specified resources that are known to contain reliable information. These sources are stored in the configurable parameter called *validUrls* and its value can be updated via the *application.properties* file. For the current configuration, these sites are:

```
validUrls=ut.ee,wikipedia.org,nutilabor.ee,msdn.microsoft.com,
docs.microsoft.com, docs.oracle.com, ibm.com/support,
developer.mozilla.org,computer.org,sirp.ee,ria.ee,cs.tlu.ee,cyber.
ee,support.office.com,tud.ttu.ee,edu,ph.emu.ee,nbi.dk,wed.academia
.eu,journal-imab-bg.org,w3schools.com,nngroup.com,sciencedirect.co
m,elsevier.com,acm.org,ieee.org
```

This list does not contain only certain URL-s, but some values are only a part of the URL. For example, there is a word “edu” that refers to all URLs of the education domain such as <https://www.cs.cmu.edu/> or <http://web.mit.edu>. The keyword “edu” in the list has a huge amount of aliases associated with it and some of them may not be qualified for University materials as some aliases may provide information, which is more related to the high school

level. It is important to pay attention that Wikipedia is a source of data that can be overly generalized and certainly not accurate enough for university subjects at some point. However, it is a popular information source which can also contain valid and useful information. This is why this address is in the list of the valid URL-s. Same goes for `w3schools.com`. Moreover, articles from `microsoft.com` and `oracle.com` portals have been partly approved. Only pieces of information for developers and IT specialists are whitelisted. Moreover, there is an URL “`ibm.com/support`”, which refers to IBM (International Business Machines) support domain, and there is a link “`developer.mozilla.org`”, which will allow all articles from this path. These two aforementioned links show sections for developers and provide technical texts for different areas. The parameters that are included in the list of the valid URLs are not filtered to provide only IT-related content, because University courses may discuss other areas like math, business, analysis and even law. So it has been decided not to restrict the subjects of the found URLs. The idea of these parameters is to limit the number of valid links as much as possible to reduce the possibility of a fishy link to be used in the analysis. At the same time, the goal is to allow as many domains as possible so that different areas of expertise can be considered as a candidate for the supplementary materials. For the current configuration, the goal was to put together parameters that would allow mostly well-known domains mostly related to science magazines, University domains and other science-related aliases.

This approach of making valid URL-s configurable makes the application more flexible because it can be considered to target different areas of interest to make search more specific. For example, the application is supposed to find supplementary materials only among the domain that belong to the University of Tartu, then certain URLs must be added to the list of valid parameters and other URL-s must be deleted. The list of valid URL-s is used in the Google search service to validate the source of an article.

Besides the list of the valid URL-s, there is also a list of websites that Google should ignore. These resources are known to be useless for the purpose of this application. This parameter is also configurable via the *application.properties* file and is called `invalidUrls`. A sample configuration is shown below:

```
invalidUrls=ajakiri.ut.ee,translate.google,webcache
```

This configuration makes it easier to adjust the application according to the user’s needs and purposes to receive better results. The logic of how the URL is validated has been defined in the following Java function:

```
public boolean validateUrl(String url, String code) {
    boolean valid = false;
    if (!url.equals("#")) {
        for (String s : validUrls) {
```

```

        if (url.contains(s.trim())
            || url.contains("courses.cs.ut.ee")
                && !url.contains(code)) {
            for (String s1 : invalidUrls) {
                if (!url.contains(s1.trim())) {
                    valid = true;
                    return valid;
                } else {
                    valid = false;
                }
            }
        } else {
            valid = false;
        }
    }
    return valid;
}

```

It should be paid to the attention that in the function above, there is specifically brought out “courses.cs.ut.ee” website. This is because the validation of this domain is a bit different than for all other domains. As the lectures of each course in the ongoing semester are scraped from this website, it is logical to prevent the algorithm to propose the same course’s lecture as a potential candidate. However, this website cannot be blacklisted entirely as the lecture materials from other courses may contain information related to the specified which can be useful as supplementary material.

## 4.2 Results collecting

A sample Google search URL is shown below:

```

https://www.google.com/search?q=equivalent tavajoonis fm-index
n-sx fcmboliline vicente add salesman suuruselt buffer robert
pdf&num=50

```

The application can process articles both in .pdf and HTML format. If the URL from Google is a .pdf file, then the text is extracted using the `String processRecord(String url)` method located in *PdfExtractorService.class* class. The method `processRecord` takes a link found via Google engine that contains a .pdf file with potential supplementary materials and as an output gives a plain text extracted from the .pdf file. If the URL suggested by Google engine is a plain HTML file, then the text from it is extracted using `String html2text(String html)` method located in *Html2TextService.class* class. This method takes HTML code and as an output gives a raw text without HTML markups. Validated articles found via Google engine are collected into the map `Map<String, Map<String, String>> googleResult`, where the key is the title of the found article and the value is another map with the article’s link as the key and its context as the value. This structure

makes it easier to access all essential information about the articles (such as link, title and context) quickly.

### 4.3 Processing Google articles

Before starting to measure similarities between the specified course's material and found articles, the context of each article has to be tokenized and tokens need to be preprocessed. During the cleaning process, each token is passed through the validation to define if it's a word containing important meaning for the analysis or not. So each token is tested against the set of stop words for both Estonian and English languages and if it belongs to either one of the sets, then this token is excluded from the future process. The sets of stop words are saved .json files and are located in the project's *src/main/resources/stopwords* directory. These are the same stop words sets that have been used by Ragnar Vent in his Master's thesis [2]. After tokens of the article have passed validation against stop words, they are passed to the function that counts the frequency of each token in the article and removes tokens that are least and most frequent. This is done to leave only tokens with the medium frequency for the analysis, therefore, are more valuable as they define the topic of the article and are, most possibly, not empty words like "I", "the", "a" etc. Basically, it is the same analysis that used Ragnar Vent [2] to preprocess tokens of the course's lectures:

- Token occurrence limit check (remove token with too high and too low occurrence)
- Stop word check (remove all words that occur in the sets of stop words)
- Alphabetic character check (remove all non-alphabetic characters)
- Word length check (remove too short words)
- Latin character check (remove latin characters)

Once all tokens have been passed through these controls, they are forwarded to be lemmatized. Lemmatization is a process of converting a word's form into its initial state. For example:

*are, am, was -> be*

Along with lemmatization process, another well-known algorithm to retrieve the root form of the words is a Stemming algorithm. The goal of both stemming and lemmatization is to reduce changed and derivationally related forms of a word to a common base form [8]:

*"love loving lovingly loved lover lovely" -> "love" "love" "love" "love" "lover" "love"*

The concept of the Stemming process is to separate the end of the words and convert the rest to its non-changing form. Lemmatization, on the contrary, is more sophisticated as there the words are converted to its dictionary form, also called a lemma [8]. Both stemming and lemmatization processes reduce the number of tokens, leaving only those that are relevant for the analysis. However, as it has been mentioned earlier and also in Ragnar Vent [2] work, the Stemming algorithm normalizes word by removing its inflectional suffixes (transforming it to

its root form, also known as a stem). While the Stemming algorithm serves its purpose for more simple and straightforward cases (e.g. *writing* will be stemmed to *write*), it does not guarantee that the stemmed word is a dictionary word (*calculated* can be stemmed to *calculat*). Lemmatization, on the other hand, is a much more sophisticated process of removing inflectional endings and resolving the lemma by relying on a dictionary that includes a morphological analysis for each word [2]. Even though lemmatization is a more pricey operation than stemming, its advantage is a potentially more reliable final output with the higher quality [2]. For above-described reasons, lemmatization has been chosen for token preprocessing as for this application it is critical to get proper forms of each word for the further analysis because these forms will be compared with forms of the specified course that the algorithm looks supplementary materials for. Each token has an effect on the result of the comparison. To illustrate the importance of lemmatization, the following example is provided: the words “write” and “written” would be considered different even if they have the same root meaning. This means that if one set of tokens has the word “written” and the other includes the word “write”, these two sets will be more different than they actually are. Before lemmatization begins, the application detects the language of input text (in this algorithm, the input to lemmatization is a set of before-cleaned tokens that has been composed of the articles found via Google search engine). For the purpose of this thesis, the application only starts lemmatization operation for either Estonian or English texts. The lemmatization process is the final step of tokens preprocessing.

After token preprocessing has been finished, the algorithm will choose X number of the tokens to be converted to the vector for future analysis. This is done because the amount of tokens even after cleaning is soaring, therefore that analysis may take an unreasonably long time. The number of tokens to be converted to the numeric format is a configurable parameter called *candidateInputChosenTokenForAnalysisLimit*. Its default value is set to 1000, same as *originalInputChosenTokenForAnalysisLimit* parameter's value for a number of token of the original input e.g. tokens of course materials. The selected tokens are later used in dimensionality reduction via SVD matrix. This process is described in Chapter 5.

## 5 Dimensionality reduction

### 5.1 Converting articles to numeric format

Once the articles identified with Google have been collected, cleaned, tokenized and the dimensionality has been reduced, we compare potential candidates for the supplementary material with the material of the lectures from the selected course. The first step is to convert both candidates and the course material into vectors in order to make it possible to evaluate their similarity. In this logic, each vector values represent word counts indistinct union of two sets of tokens - one from original input and one from the potential candidate. For example, suppose there are two lists of tokens:

```
t1 = ["vector", "cosine", "magnitude", "vector", "grade",  
"length"]  
t2 = ["matrix", "Euclidean", "vector", "similarity", "math"]
```

then the distinct union of these two lists would be:

```
{"vector", "cosine", "magnitude", "grade", "length", "matrix",  
"Euclidean", "similarity", "math"}
```

In this case the two vectors will look like:

```
v1 = [2, 1, 1, 1, 1, 0, 0, 0, 0]  
v2 = [1, 0, 0, 0, 0, 1, 1, 1, 1]
```

The example, of a map, where key is a token from the article and the value is this token's frequency is provided below:

```
removeHighAndLowFreq: {ccc=1, reuse=1, ccb=1, information  
technology=1, interacting=2, vastutama=0, hall=1, p\xfcstitama=0,  
descriptor=0, salary=1, spontaneous=1, morton=0, ccs=1,  
metabolomics=0, pick=1, myclassloader=0, loengumaterjal=0,  
spreading=1, require=0, kiire=0, seotav=1, prioritization=1,  
stabilized=1, size=3, generalizing=0, topeltklikk=0, resumption=3,  
toimuma=1, object=2, kuj=2, reordering=1, accessible=1,  
statistical=0, erindi=0, roll=0, teenusserver=1, example=0,  
result=0, component system=2}
```

Once all articles are converted into the numeric representation, it is time to apply dimensionality reduction techniques described in Chapter 5.2.

### 5.2 SVD matrix

Before starting to apply a certain algorithm on the prepared data, it is important to review the final set again as it may appear to be unnecessarily large. Performing analysis on such large

data sets is impractical and may take extensive time. In machine learning, there might be many features on which the final classification is done. The higher the number of features, the harder it gets to process the training set. Most of these features are correlated, and hence **redundant**. This is where dimensionality reduction algorithms come into play and can help to decrease the amount of data without compromising integrity. There are two types of a common practice: reducing the volume or reducing the number of attributes (dimension). The algorithm applied with reduced data is more efficient yet it will produce the same result as with the original data set.

In this thesis, once the data is ready for the analysis, dimensionality reduction techniques are applied to the final dataset to speed up the algorithm execution. Before starting to reduce dimensionality, it should be considered that the amount of possible suitable supplementary articles can be unnecessarily high. For this purpose, a configuration parameter called *candidateArticlesLimit* has been created to limit the amount of furtherly processed articles in the final steps of the algorithm. By default, this parameter is set to 50 articles. If the initial amount of articles found via Google engine is larger than the configured value, then the limited number of articles is chosen randomly. The example of the SVD matrix for the article found via Google engine can be found in Appendix IV. The title of the article and its content are provided below:

“Computational Biology: Genomes, Networks, Evolution MIT course“

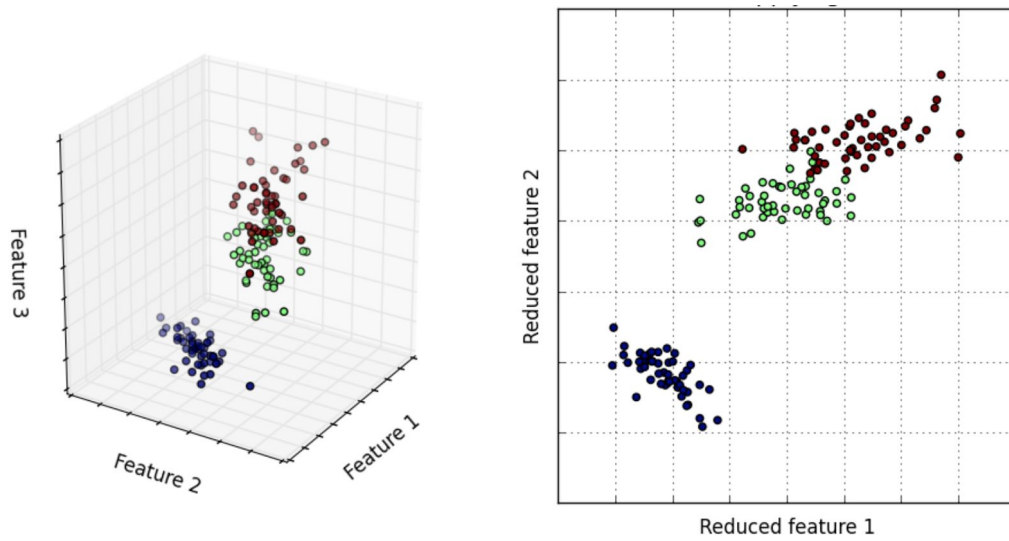
<https://stuff.mit.edu/afs/athena/course/6/6.047/.../CompiledScribeNotes2012 Master.pdf>.

In this thesis, the Single Value Decomposition a.k.a. SVD has been chosen to lower the number of dimensions. SVD is applied to a matrix, which consists of two vectors - the first is a representation of the course material (for which the search is run for) and the second column is a potential candidate for the supplementary material in the numeric format found on the web. Single Value Decomposition is the transformation of the matrix  $A$  into a product of three matrices  $A = UDV^T$  where  $U$  and  $V$  are orthonormal (meaning  $UU^T = U^TU = I$  and  $VV^T = V^TV = J$ ) and matrix  $D$  is diagonal with positive real entries [9]. Using the SVD matrix in different computation, rather than the original input matrix, is proven to be more robust [10].

Another approach that can be used besides SVD is Principal Components Analysis (PCA). These two approaches are closely related as both can be used to analyze and find data patterns and both can help to reduce the number of dimensions without compromising data integrity and losing valuable information.

The core of SVD is to find the most meaningful basis of the dataset and use it to reconstruct the original dataset while capturing as much of the original variance as possible. Figure 4

provides a sample visualization, where on the right side there is the original dataset and on the left side is the reduced one.



**Figure 4. Sample visualization the matrix for dimensionality reduction [17].**

Decreasing number of features provide numerous advantages: speeding up the analysis by reducing computation time, saving storage space that is required to keep the input data as well as reducing redundant features. For instance, there is no need to store the same features in different units like meter and inches. Also converting data to 2D may allow to plot and visualize it accurately, which makes possible to observe patterns more plainly. The primary damage of the reduction techniques is possible data loss.

### 5.3 SVD vs PCA

Singular value decomposition (SVD) and principal component analysis (PCA) are two **eigenvalue** methods that are used to convert a high-dimensional dataset into a dataset with fewer dimensions without compromising the integrity of the data. These two methods are closely associated with each other as PCA uses the SVD in its calculation. The basic idea of PCA is to convert the dataset from many correlated coordinates into fewer uncorrelated ones called principal components while still preserving most of the variability in the data. In other words, principal components are the directions where the data is most spread out. To do that, PCA needs to rate the importance of every feature a.k.a dimension. There are two ways of how PCA reduces dimensionality.

The first option is to use characteristic vectors (**eigenvectors**). This way is called Eigen Decomposition [13]. Suppose we have a matrix  $A$ . Almost all vectors change directions when they have been multiplied by the matrix  $A$ . Although there are exceptional vectors  $k$  which remain in the same direction as vectors  $Ak$ . These vectors are eigenvectors. Now if to multiply eigenvector by  $A$  and if the vector  $Ak$  is a number  $\eta$  times the original  $k$ , the basic equation would be  $Ak = \eta k$  or  $Ax = \eta x$  [13]. In this context,  $\eta$  is an eigenvalue of matrix  $A$ .

Eigenvalues can characterize many valuable properties of linear transformations [13] and provide such information as whether or not a system of linear equations has an exclusive solution.

The second option is to use SVD to calculate and measure the significance of each feature. For instance, all dimensions of the matrix  $U$  have to be assessed. Once finished, the dimensions that have been chosen by the PCA algorithm are the most important  $k$  ones. The actual data is mapped to  $k$  dimension according to these chosen dimensions.

As well as the general disadvantages of dimensionality reduction methods, PCA has also other several others. For example, PCA tends to find linear correlations between patterns [11]. Moreover, it fails if mean and **covariance** are not enough to define datasets [11]. Several applications such as Google (PageRank algorithm) and Kleinberg (Hits algorithm), use PCA or SVD approaches in use. In this application, SVD has been chosen for dimensionality reduction because SVD is numerically more efficient than PCA and more robust.

## 6 Analysis and Assessment

### 6.1 Analysis and Result generation

The number of tokens of both specified course (a.k.a original input) and potential candidate article can be colossal (more than 20 000 symbols). Given that, the analysis can take an unreasonably long time. So it is necessary to decrease the number of tokens that will go further for the analysis.

For this purpose, configurable parameters *originalInputChosenTokenForAnalysisLimit* and *candidateInputChosenTokenForAnalysisLimit* have been created and both default values are set to 1000. If the number of tokens from either original input or potential candidate is greater than the limit, the system will choose tokens for the analysis randomly. Before this control, all tokens have been previously cleaned and processed to be free from garbage words. Once all the vectors are built, each vector is used to measure the similarity between the vector of the possible supplementary material and the original input (a.k.a. course) vector. There is three basic distance/ similarity measurement in text mining:

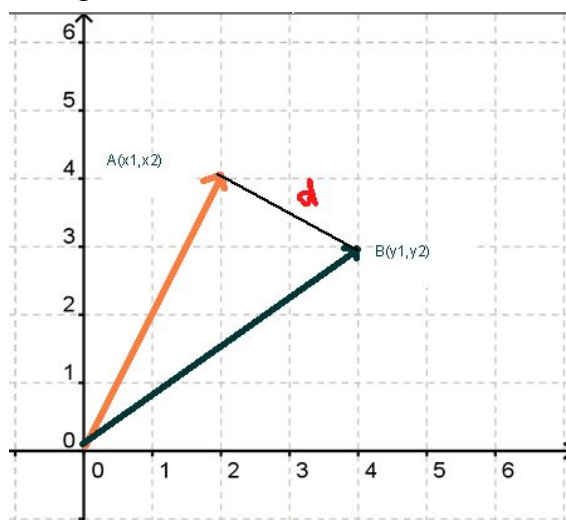
- **Euclidean distance**

The formula is provided below and it relies on the Pythagorean Theorem:

$$E(x, y) = \sqrt{\sum_{i=0}^n (x_i - y_i)^2}$$

**Figure 5. Euclidean distance formula**

This equation computes the shortest distance between the two objects, or, in this case, among two vectors. The basic idea is illustrated below:



**Figure 6. Euclidean distance logic explanatory image**

If the result of the equation is zero, then both objects (or vectors) are identical. For example, if to compare two identical sentences “I love dogs”, then their distance is zero, because both vectors will look like [1,1,1] and if to apply Euclidean distance formula to them then the output will be:

$$\text{Math.sqrt}((1-1)^2 + (1-1)^2 + (1-1)^2) = 0$$

However, if the second sentence would be “I love cats”, then the distance would be:

$$\text{Math.sqrt}((1-1)^2 + (1-1)^2 + (1-0)^2) = 1$$

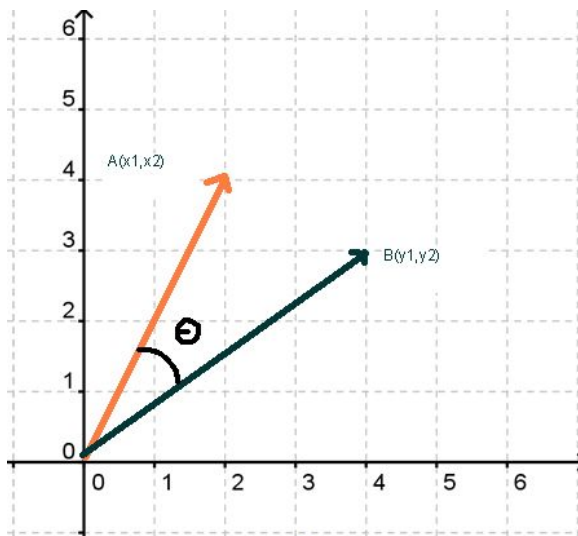
- **Cosine similarity**

Determines the angle between two objects and can be computed using the formula provided below in Figure 7.

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2} \sqrt{\sum_i B_i^2}}$$

**Figure 7. Cosine similarity formula**

The range of the result varies from zero to one. If it is one, then the vectors have the same in orientation, but not the same magnitude. The magnitude of vector AB is the length of its line segment. Figure 8 visualizes the idea of the Cosine similarity, where theta be the angle between two vectors.



**Figure 8. Cosine similarity principle explanatory image**

If  $0 < \theta < \pi$ , the vectors are positively oriented.

If  $\pi < \theta < 2\pi$ , the vectors are negatively oriented.

- **Jaccard similarity**

Refers to the number of common words over all words in the set. Suppose there are two sets of words A and B, then the Jaccard similarity can be determined using the formula provided below in Figure 9

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

**Figure 9. Jaccard similarity formula**

The result varies from zero to one. One means that sets are identical. If there is no common word between two sets, then the result is zero.

All three methods can be used to assess token based similarity. The advantages of these methods are computational efficiency and good performance when applied to large textual data. The main disadvantage of these methods is that they cannot handle synonyms words, for example, “crocodile” and “alligator”. Additionally, the magnitude of the vectors does have any meaning here. In this application, the cosine similarity metric for measuring the similarity between articles has been chosen because the direction of articles is more important than the exact distance between them. Following the example above about how vectors are composed (see section 5.1), the Cosine similarity between them can be computed as is shown further. So, if the two vectors are:

```
v1 = [2, 1, 1, 1, 1, 0, 0, 0, 0]
v2 = [1, 0, 0, 0, 0, 1, 1, 1, 1]
```

then the similarity would be:

```
Cosine Similarity
= ((2*1)+(1*0)+(1*0)+(1*0)+(1*0)+(0*1)+(0*1)+(0*1)+(0*1))
  / (Math.sqrt(22+12+12+12+12+02+02+02+02) *
    Math.sqrt(12+02+02+02+02+12+12+12+12))
= (2+0+0+0+0+0+0+0+0) / (Math.sqrt(4+1+1+1+1+0+0+0+0) *
    Math.sqrt(1+0+0+0+0+1+1+1+1))
= 2 / (Math.sqrt(8) * Math.sqrt(5)) = 2 / (2.82842 * 2.2360)
= 2 / 6.3243024 = 0,000000031624041
```

Each computed similarity is saved in a map Y, where the key is of type Double, which represents the computed similarity and the value is another map Z with only two entries in it: the original input and the candidate. The key of this map is of type String and it represents the title and the value is also of type String, which stores the URL. These titles and URLs are used later in the final validation and are shown to the user as the output of the algorithm.

Here the map Y has a declaration `Map<Double, Map<String, String>>` similarity and the sample content of the map Y is:

```
{0.5743767716395854: {"High Availability Optimization Assessment - Symantec":  
http://eval.symantec.com/mktginfo/enterprise/fact_sheets  
/b-ha_optimization_assesment_DS_20266020-1.en-us.pdf},  
0.8365288655512938: {"Ch. 1. The Planning Context and Process - Municipality of Anchorage":https://www.muni.org/Departments/OCPD  
/Planning/Publications/Gwood%20CATMP/girdctm_ch1.pdf}}
```

The first number is the similarity between two objects. If it is 0.0, it means that both objects are identical. Once all candidates have been assigned the value of the similarity with the course's content, the map Y is sorted by its keys (in this case, the similarity value) in ascending order. Finally, the results are put in the final list `List<Material>` `recommendedArticles`, where object `Material` has two attributes - title and link. There is a configuration parameter in the *application.properties* file called *recommendLimit*, which limits the number of articles to show in the final output of the algorithm in the front-end side of the portal. If the number of Google candidates is less than the recommended limit then all candidates are shown to the user. The sample output for the course MTAT.05.116 is shown below in Figure 10. The discussion over the results can be found in section 6.3

Kursuse materjalid <span>×</span>	
Pealkiri	Link
Gizem Karaali - Pomona College pages.pomona.edu/~gk014747/research/GK-CV.pdf	http://pages.pomona.edu/~gk014747/research/GK-CV.pdf
Anglo-Franco-German Network in Representation ... - University of Kent https://www.kent.ac.uk/smsas/personal/.../2016-EPSRC-RepNet-edited.pd...	https://www.kent.ac.uk/smsas/personal/sl261/RepNet/2016-EPSRC-RepNet-edited.pdf
Nicholas J. Owad nick.owad.org/Documents/CV_Owad.pdf	http://nick.owad.org/Documents/CV_Owad.pdf
Women's Representation in Mathematics Subfields: Evidence from the ... https://arxiv.org/pdf/1509.07824	https://arxiv.org/pdf/1509.07824
Twenty-five years of representation theory of quantum groups https://www.birs.ca/workshops/2011/1lw5096/report1lw5096.pdf	https://www.birs.ca/workshops/2011/1lw5096/report1lw5096.p
<div>Sulge</div>	

**Figure 10. The sample final output containing suggested supplementary materials for the course MTAT.05.116.**

## 6.2 Saving supplementary materials

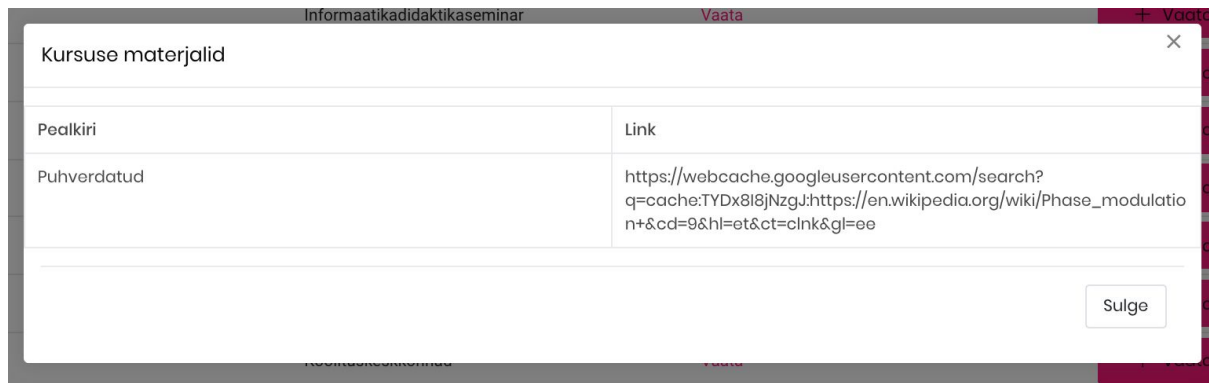
As it has been mentioned previously in section 3.2, the final output of the algorithm is saved into the specified directory (which can be assigned via the *materialsLocation* parameter in the *application.properties* file), where materials for each course is saved and the file's name is a course code. For example, "LTAT.02.001.txt". So when the user sends the request to the server to view supplementary materials, the application firstly parses the already existing materials and if a file with the specified course code found, the application just reads the content of the file and return it to the user. However, if the file with the specified course code has not been found, the application starts the algorithm to scrape course' data and runs the search for the supplementary materials. This helps to decrease the processing time of the request, which affects the user experience in a positive way as the final output is shown to the user in much less time. The sample content of the saved file for course MTAT.05.116 is provided below:

```
STARTTITLE:{{Twenty-five years of representation theory of quantum  
groups  
https://www.birs.ca/workshops/2011/11w5096/report11w5096.pdf  
}}ENDTITLE###STARTLINK:{{https://www.birs.ca/workshops/2011/11w509  
6/report11w5096.pdf}}ENDLINK-----
```

This structure helps to separate each candidate from another and to differ when ends article's title and starts its URL.

## 6.3 Results validation

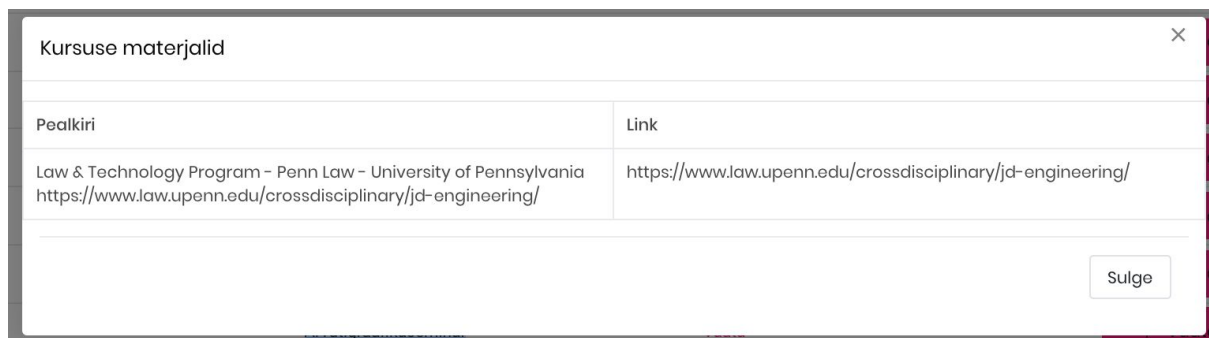
The manual testing of the algorithm has been performed for this application. Each found material has been reviewed by the author and relation to the course material has been assessed. The conclusion is that, at this stage, the algorithm is not ready yet to be passed through more accurate validation like "fake news" algorithm [18] or be assessed by the professors at the University of Tartu. The found materials do not really contain valuable information related to the material of the course and do not give knowledge that would be useful for the student. In general, one of the disadvantages of the algorithm is that some courses have fake results as one shown below in Figure 11. This result can be revealed, for example, for the course "MTAT.07028 Infoturve". Note that at the beginning of each month the system refreshes to tokens and runs a new search of the supplementary material for each course. As a result, this output may not be repeated for the same course. However, it may occur for some other course.



**Figure 11. The sample of the fake result of the algorithm.**

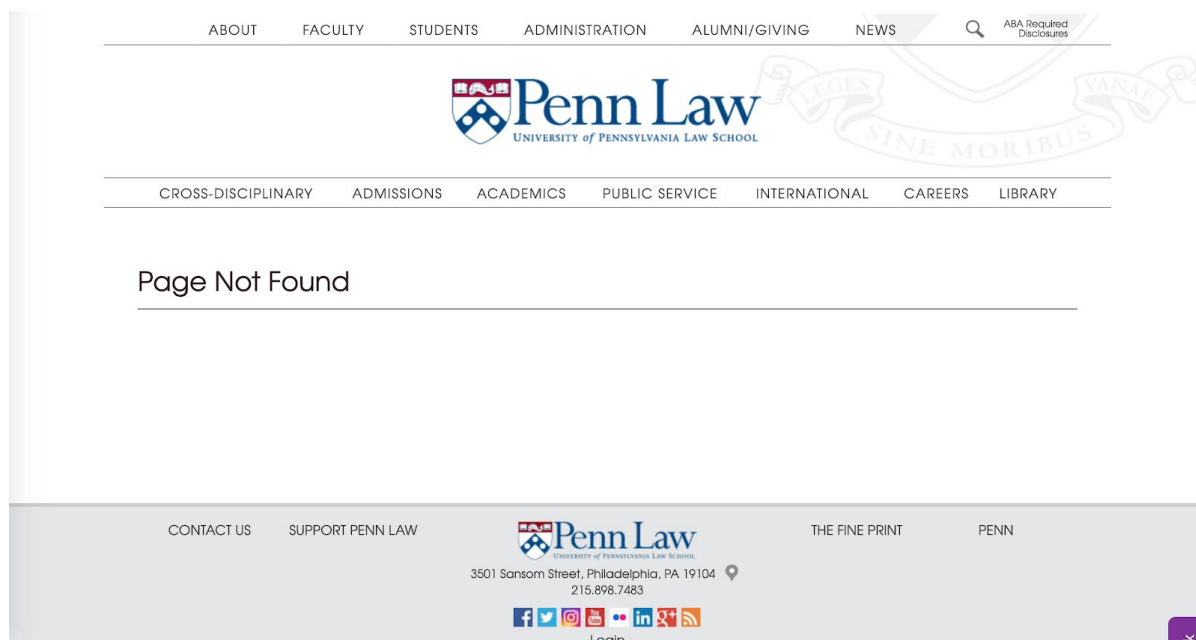
Another weak side of the current version is that mostly there are only one or two proposed supplementary materials in the output. Although, the limit for how many results may be shown to the user is set to 20 (configurable via a *recommendLimit* parameter in the *application.properties* file). Currently, there are too few choices for students for picking up an additional reading. Moreover, one article cannot definitely cover all subjects discussed in the course. So the majority of the valuable information from the University course is not taken into consideration by the algorithm, which results in lack of the supplementary materials returned as the output to the end user. What concerns this issue, the key problem may be connected to the Google search engine and how it processes queries in particular. The more detailed information about it can be found in section 8.2

Below are provided some particular examples to explain this assessment of the algorithm's work. For instance, the proposed supplementary materials for the course "MTAT.03.305 Arvutigraafikaseminar" is as shown in Figure 12 below:



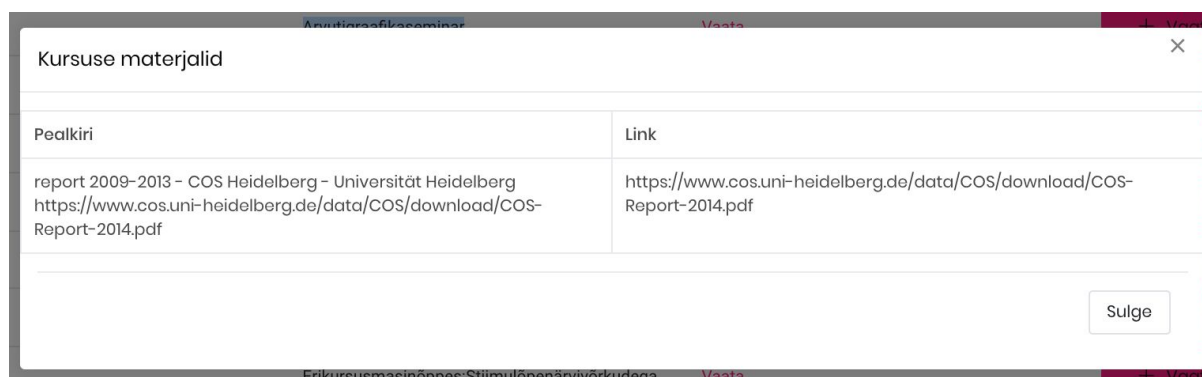
**Figure 12. The sample final output containing suggested supplementary materials for the course MTAT.03.305.**

The title of the proposed article does not say much about what the context is about. At first sight, the title does not seem to be related to the computer graphics area at all. Moreover, the content of the link returns a "Not found" error as shown in Figure 13 below.



**Figure 13. The article proposed as supplementary material for the course MTAT.03.305.**

Another example is with the course “MTAT.03.239 Bioinformaatika”, for which the suggested article is shown in Figure 14 below.



**Figure 14. The sample final output containing suggested supplementary materials for the course MTAT.03.239.**

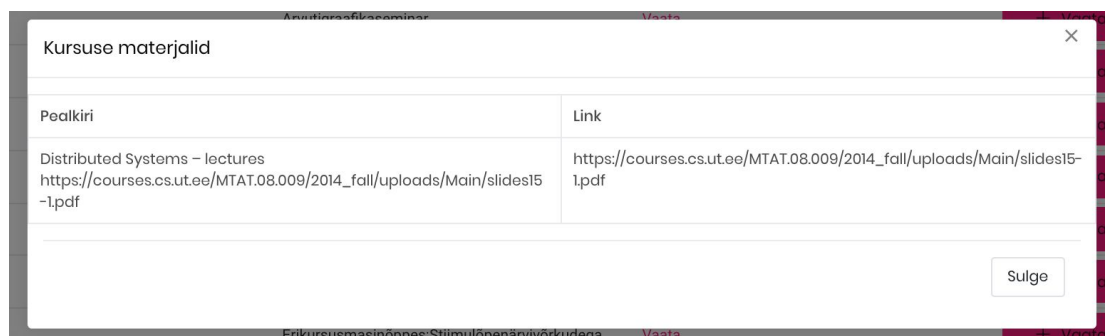
The title of this article also does not reveal what the text is going to be about. From this title, it is clear that this is some kind of a report. Moreover, there is an abbreviation “COS”, which does not say anything as well. If to open the URL assigned for this article, then the reader will discover that COS stands for Centre for Organismal Studies. So far, the reader may assume that this article somehow refers to biology. Going further through this document to the content of this report, there are totally three chapters (and appendix section as well), which are entitled as “Introduction”, “Research groups” and “Facilities”. In Figure 15 there is a list of all the research groups that are mentioned in the report.

<b>2 RESEARCH GROUPS</b>	<b>16</b>
2.1 Animal Evolution, Prof. Dr. Detlev Arendt	17
2.2 Independent Research Group: Dr. Jan Felix Evers	23
2.3 Circadian Clock Biology, Prof. Dr. Nicholas S. Foulkes	29
2.4 Animal Molecular Physiology, Prof. Dr. Stephan Frings	35
Project Leader: Dr. Frank Möhrle	38
2.5 Independent Research Group: Dr. Emmanuel Gaquerel	41
2.6 Independent Research Group: Dr. Guido Grossmann	47
2.7 Independent Research Group: Dr. Annika Guse	53
2.8 Plant Molecular Biology, Prof. Dr. Rüdiger Hell	59
Project Leader: Dr. Markus Wirtz	64
2.9 Molecular Evolution and Genomics, Prof. Dr. Thomas W. Holstein	67
Project Leader: Dr. Ulrike Engel	72
Project Leader: PD Dr. Suat Özbek	74
2.10 Independent Research Group: Dr. Amal J. Johnston	77
2.11 Biodiversity and Plant Systematics, Prof. Dr. Marcus A. Koch	83
Project Leader: apl. Prof. Dr. Claudia Erbar	88
Project Leader: Dr. Nicolai M. Nürk	90
2.12 Modelling of Biological Processes, Prof. Dr. Ursula Kummer	93
2.13 Independent Research Group: Dr. Steffen Lemke	99
2.14 Developmental Biology, Prof. Dr. Ingrid Lohmann	105
2.15 Stem Cell Biology, Prof. Dr. Jan Lohmann	111
2.16 Independent Research Group: Dr. Alexis Maizel	117
2.17 Developmental Neurobiology, Prof. Dr. Gabriele Elisabeth Pollerberg	123
2.18 Independent Research Group: Dr. Gislene Pereira	129
2.19 Plant Molecular Physiology, Prof. Dr. Thomas Rausch	133
Project Leader: Dr. Steffen Greiner	138
2.20 Plant Developmental Biology, Prof. Dr. Karin Schumacher	141
Project Leader: Dr. Stefan Hillmer	146
2.21 Cell Chemistry, Prof. Dr. Sabine Strahl	149
Project Leader: Dr. habil. Michael Büttner	154
Project Leader: Dr. Mark Lommel	156
2.22 Developmental Biology/Physiology, Prof. Dr. Joachim Wittbrodt	159
Project Leader: Dr. Lucia Poggi	164
Project Leader: Dr. Lazaro Centanin	166
Project Leader: apl. Prof. Dr. Thomas Braunbeck	168
2.23 Independent Research Group: Dr. Sebastian Wolf	171

**Figure 15. The list of all the research groups that are mentioned in the report.**

Taking a quick view on these titles, it seems like all the research groups are definitely connected to biology, but none of them has anything to do with informatics. Therefore, this report is not handy for the student, who takes MTAT.03.239 course.

Now for the course “MTAT.08.024 Hajussüsteemide seminar” the proposed material is found among the materials from other course “MTAT.08.009 Distributed Systems” that are taught at the University of Tartu. This particular lecture is dated of year 2014. This found lecture suits as supplementary material because there is a piece of real study information such as introduction into disturbed systems. Moreover, these slides have been composed by a lector at the University of Tartu, which makes them a legit information source and a student can be sure that there is no fake data there. The algorithm’s output for the course MTAT.08.024 is shown below in Figure 16.



**Figure 16. The sample final output containing suggested supplementary materials for the course MTAT.08.024.**

The course “MTAT.08.020 Paralleelarvutused” has seven supplementary material suggestions in the output which is shown below in Figure 17.


Kursuse materjalid	
Pealkiri	Link
restaureerimine - DSpace - Tartu Ülikool dspace.ut.ee/bitstream/handle/10062/58930/raamat_aeg_restaureerimine_8.pdf	http://dspace.ut.ee/bitstream/handle/10062/58930/raamat_aeg_restaureerimine_8.pdf
Teaduskirjanduskorpuse lemmad sageduse järjekorras - Eesti ... https://keeleressursid.ee/images/cl-ut-ee/sagedusloendid/lemma_tea_kahanevas.txt	https://keeleressursid.ee/images/cl-ut-ee/sagedusloendid/lemma_tea_kahanevas.txt
Informaatika - K tähega algavad mõisted   Sõnu seletav sõnastik https://annaabi.ee/k-lleksikon-2.html	https://annaabi.ee/k-lleksikon-2.html
Kogumiku on trükiks ette valmistanud / Publication prepared by Malle ... educationdocbox.com/_/70991937-Kogumiku-on-trukiks-et-ette-valmistanud-publicati...	http://educationdocbox.com/Distance_Learning/70991937-Kogumiku-on-trukiks-et-ette-valmistanud-publication-prepared-by-malle-ermel-marika-liivamagi-rein-saukas.html
EFI tagasimakse pilootprojekt osutus oodatust ... - Eesti Filmi Instituut filmi.ee/uudised#efi-tagasimakse-pilootprojekt-osutus-oodatust-tunduvalt-edukamaks	http://filmi.ee/uudised#efi-tagasimakse-pilootprojekt-osutus-oodatust-tunduvalt-edukamaks
Õpistsenaariumide kavandamise vahendi LePlanner disain ja arendus www.cs.tlu.ee/teemad/get_file.php?id=409	http://www.cs.tlu.ee/teemad/get_file.php?id=409
The Estonian Orthodox Eparchy under the influence of the Soviet ... www.academia.edu/_/The_Estonian_Orthodox_Eparchy_under_the_influence_of_th...	http://www.academia.edu/16595763/The_Estonian_Orthodox_Eparchy_under_the_influence_of_the_Soviet_religious_policies_in_1954-1964
Sulge	

**Figure 17. The sample final output containing suggested supplementary materials for the course MTAT.08.020.**

The first thing that strikes out is the very first suggested material, which title tells that this is something about restoration. Clearly, this article has nothing in common with the MTAT.08.020 course. The student would not even bother to open the URL, because the title is already disappointing. The fifth link also seems to be a random guest in this list as it refers to some film pilot project and, most probably, does not have anything to do with parallel computations. The last supplementary material also refers to some historical text and hardly can be useful in this course. All other suggestions do not seem very promising as well. For example, the third link may indeed be connected to computer science and different terms, but not directly to the parallel computations. Also the source URL “annaabi.ee” is not a very reliable source as it may contain a lot of wrong information. So, despite the whole variety of suggestions, none of them would really be helpful for the student.

Another semi-positive output of the algorithm has been received for the course “LTAT.03.003 Objektorienteeritud programmeerimine”. Here the algorithm found only one supplementary material for this course. Firstly, this is clear that this basic course should have more found propositions as the knowledge and practical skills acquired during this course are

very useful during further studies. The output for the LTAT.03.003 is shown below in Figure 18.



Pealkiri	Link
How to use FindBugs as IntelliJ plugin <a href="https://courses.cs.ut.ee/MTAT.03.159/2017_spring/uploads/Main/SWT2017-lab5-FindBugsIntelliJ.pdf">https://courses.cs.ut.ee/MTAT.03.159/2017_spring/uploads/Main/SWT2017-lab5-FindBugsIntelliJ.pdf</a>	<a href="https://courses.cs.ut.ee/MTAT.03.159/2017_spring/uploads/Main/SWT2017-lab5-FindBugsIntelliJ.pdf">https://courses.cs.ut.ee/MTAT.03.159/2017_spring/uploads/Main/SWT2017-lab5-FindBugsIntelliJ.pdf</a>

**Figure 18. The sample final output containing suggested supplementary materials for the course LTAT.03.003.**

In defence of this result, the link is taken from the University course “MTAT.03.159 Software Testing” dated of the year 2017. So the information source is reliable and, on the other hand, it is very practical because a lot of student use IntelliJ during the LTAT.03.003 course and such small tutorial about one of its plugins may be the good-to-know skill. Another appropriate supplementary material found by the algorithm is for the course “MTAT.03.277 Andmekaeve uurimisseminar”. The suggestion is shown below in Figure 19.



Pealkiri	Link
Refactoring: Improving the Design of Existing Code <a href="https://www.csie.ntu.edu.tw/~r95004/Refactoring_improving_the_design_of_existing_code.pdf">https://www.csie.ntu.edu.tw/~r95004/Refactoring_improving_the_design_of_existing_code.pdf</a>	<a href="https://www.csie.ntu.edu.tw/~r95004/Refactoring_improving_the_design_of_existing_code.pdf">https://www.csie.ntu.edu.tw/~r95004/Refactoring_improving_the_design_of_existing_code.pdf</a>

**Figure 19. The sample final output containing suggested supplementary materials for the course MTAT.03.277.**

Judging by the title, the proposed article speaks about how to make existing code better, which a good skill for every developer despite of the language used. The .PDF file contains an online version of a book “Refactoring: Improving the Design of Existing Code” written by Martin Fowler, Kent Beck (Contributor), John Brant (Contributor), William Opdyke and don Roberts. By quickly parsing the content of the book and reading through the lines, it seems that this material is fully connected to Informatics. It may be not the best suggestion that could be firstly proposed to the student who takes data mining research seminar course. However, as it is mentioned above, writing clean code is definite sign of a good programmer.

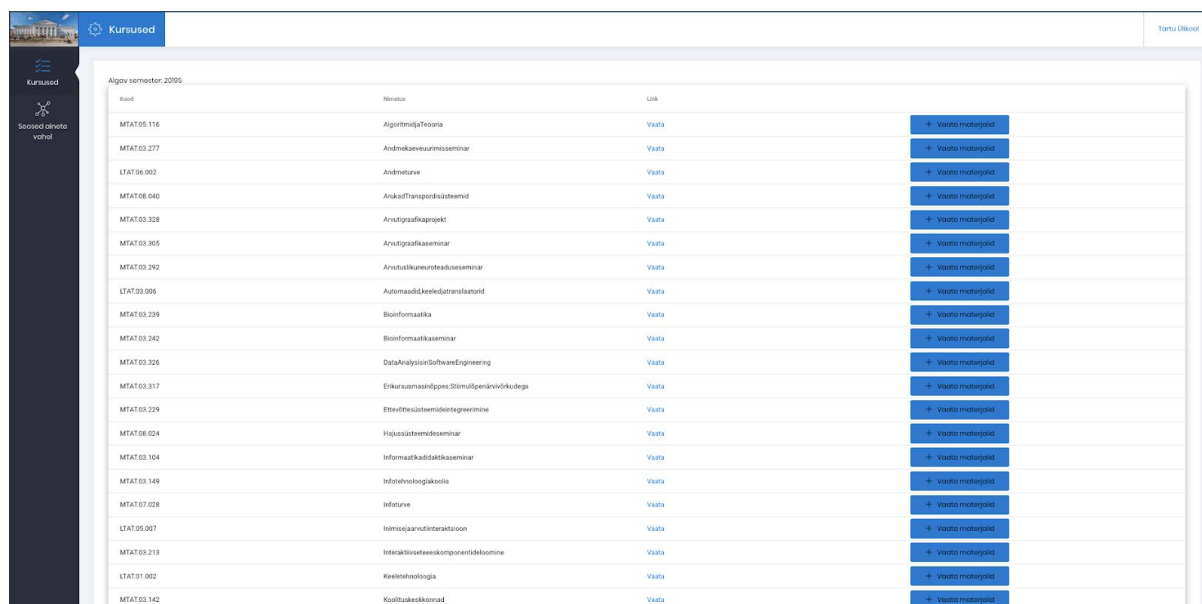
Even data mining researcher, who, perhaps, usually uses R language for the work, should be able to write a high-level code.

To sum up, the assessment has shown that the algorithm is technically working. It does perform scraping, analyzing and searching for suitable materials. However, it needs serious upgrade and improvement before the application can be accessible to students as the majority of the found suggestions are not valid. All propositions of how to improve the output for the algorithm are listed in Chapter 8.

## 7 Front-end side of the application

### 7.1 Courses list

The front-end side of the application has been fully implemented in Angular 6. The portal has a left-hand side menu, where two pages are listed: courses list and course's connections to other courses in the ongoing semester. The styling is done using Bootstrap 4. The list of courses shows all courses in the ongoing semester. There is a table that contains the code and name of each course. Also, there is a column named "Link", which contains the link to the course web site in *courses.cs.ut.ee* site. The last column has a button, which opens a supplementary materials for the chosen course. Figure 20 shows a screenshot of the courses list view.



Kood	Nimetus	Link	
MTAT.05.116	Algoritmid ja Teooria	Vaata	+ Vaata materjali
MTAT.02.277	Andmekavanduse seminar	Vaata	+ Vaata materjali
LTAT.06.002	Andmeturve	Vaata	+ Vaata materjali
MTAT.08.040	Andmetranspordisüsteemid	Vaata	+ Vaata materjali
MTAT.03.328	Andme graafika projekt	Vaata	+ Vaata materjali
MTAT.03.305	Andme graafika seminar	Vaata	+ Vaata materjali
MTAT.03.292	Andme süsteemide seminar	Vaata	+ Vaata materjali
LTAT.03.006	Automaadid, keeleõppimise alused	Vaata	+ Vaata materjali
MTAT.03.239	Bioinformaatika	Vaata	+ Vaata materjali
MTAT.03.242	Bioinformaatika seminar	Vaata	+ Vaata materjali
MTAT.03.326	Data Analysis in Software Engineering	Vaata	+ Vaata materjali
MTAT.03.317	Erikursusmaastik: Stimulatsioonide kude	Vaata	+ Vaata materjali
MTAT.03.229	Ettevõtte süsteemide areng	Vaata	+ Vaata materjali
MTAT.08.024	Häire süsteemide seminar	Vaata	+ Vaata materjali
MTAT.03.104	Informaatika didaktika seminar	Vaata	+ Vaata materjali
MTAT.03.149	Informaatika õppematerjalid	Vaata	+ Vaata materjali
MTAT.07.028	Infotur	Vaata	+ Vaata materjali
LTAT.05.007	Intervjuu ja suhtluse oskused	Vaata	+ Vaata materjali
MTAT.03.213	Interaktiivsete komponentide loomine	Vaata	+ Vaata materjali
LTAT.01.002	Keele tehnoloogia	Vaata	+ Vaata materjali
MTAT.03.142	Koolitus tekkimise	Vaata	+ Vaata materjali

Figure 20. Listed courses in the ongoing semester.

### 7.2 Course similarities

Another, the smaller, part of this thesis is to implement an algorithm, which result would be a list of courses sorted by their differences percentwise from the specified course. This functionality located on the separate page of the web application. The course similarities page layout is similar to the course list page and is shown below in Figure 21.

Kood	Nimetus	Link	
MTAT05.116	Algoritmid ja Teooria	Vaata	+ Seotud teiste ainetega
MTAT03.277	Andmekaasuvõimsused	Vaata	+ Seotud teiste ainetega
LTAT06.002	Andmeteadus	Vaata	+ Seotud teiste ainetega
MTAT06.040	Arvutiteaduse põhitõed	Vaata	+ Seotud teiste ainetega
MTAT03.328	Arvutisüsteemide projekt	Vaata	+ Seotud teiste ainetega
MTAT03.305	Arvutisüsteemide seminar	Vaata	+ Seotud teiste ainetega
MTAT03.292	Arvutisüsteemide seminar	Vaata	+ Seotud teiste ainetega
LTAT03.006	Automaatide teooria ja rakendused	Vaata	+ Seotud teiste ainetega
MTAT03.229	Bioinformaatika	Vaata	+ Seotud teiste ainetega
MTAT03.242	Bioinformaatika seminar	Vaata	+ Seotud teiste ainetega
MTAT03.326	Data Analysis Software Engineering	Vaata	+ Seotud teiste ainetega
MTAT03.317	Erikursusmaailm: Ohtu ja võimaluste maailm	Vaata	+ Seotud teiste ainetega
MTAT03.229	Eeltestiteooria ja rakendused	Vaata	+ Seotud teiste ainetega
MTAT06.024	Häireteooria seminar	Vaata	+ Seotud teiste ainetega
MTAT03.104	Informaatika alused	Vaata	+ Seotud teiste ainetega
MTAT03.149	Informaatika alused	Vaata	+ Seotud teiste ainetega
MTAT07.028	Infotur	Vaata	+ Seotud teiste ainetega
LTAT05.007	Intervjuu ja interaktsioon	Vaata	+ Seotud teiste ainetega
MTAT03.213	Interaktiivsed kompositsioonid	Vaata	+ Seotud teiste ainetega
LTAT01.002	Kahekeelne keeleõpe	Vaata	+ Seotud teiste ainetega
MTAT03.142	Koolitusekspert	Vaata	+ Seotud teiste ainetega

**Figure 21. Listed courses in the ongoing semester for courses similarities view.**

There is a list of courses in the ongoing semester represented in a table. The table consists of four columns: code, name, link and a button. Figure 22 shows a screenshot from the course similarities view. Here, the button performs a different function. Clicking this button opens a table, which shows how much each course in the ongoing semester is different from the specified course. The algorithm that has been used to find how all other courses in the ongoing semester differ from the specified course in the request has the following steps:

1. Using the web scraping service, identify courses in the ongoing semester
2. For each course in the current semester, if the lecture's data has already been collected and saved into the configured directory, then it is retrieved and saved lecture tokens into the list. If the lecture's material has not been located, then using the Ragnar Vent's modified Python code, course's data is collected from the *courses.cs.ut.ee* website, saved into a .txt file and also into the list. The Python code also performs cleaning, lemmatization, stemming and tokenization. The output of this step is a hashmap, where the key is the course code and the value is a list of string tokens for each course. The sample shortened version of the map is shown below:

```
Tokens map: lect_data [(<Lecture: 1>, {u'h\xe4rmel': 1,
u'ahti': 1, u'peter': 1, u'leiserson': 1, u'kiho': 2,
u'toimuma': 1, u'introduction': 1, u'cormen': 1, u'nestra':
1, u'mehlhorn': 1, u'structures': 1, u'moodle': 1, u'rivest':
1, u'kirjandus': 1, u'andmestruktuur': 2, u'algoritm': 2,
u'data': 1, u'keskkond': 1, u'j\xfcfcricri': 2, u'algorithms': 2,
u'suhtlus': 1, u'kurt': 1}), (<Lecture: 2>, {u'eelistatud':
3, u'binomiaalpuue': 2, u'binomiaalpuud': 2, u'redutseerima':
1, u'paarsus': 1, u'ullman': 1, u'k\xe4tte': 3,
u'binomiaalpuus': 1})]
```

3. The input course lecture is separately saved into the *originalInput* variable in the *CourseSimilaritiesService.java* class as follows.
4. Both tokens of the selected course and comparable course are converted into the vectors using the same logic as described in section 5.1 and the Cosine similarity formula is applied to find how similar other courses are with the specified course. For each result, the *Similarity.java* object is created which contains the related course code and the similarity value in Double format.
5. Each *Similarity.java* object is saved into a list and returned to the controller. Then the list is sorted in ascending order by similarity values and sent to the user. For example, request URL *http://172.17.64.156/#/courseSimilarities/similarities/MTAT.05.116* will give the output that is shown below in Figure 22.

Aine	Erinevus (%)
MTAT.03.231	4.745760332083471
LTAT.NR.003	7.700716629911479
MTAT.05.118	8.208581508681966
MTAT.03.239	9.878606633658512
MTAT.07.024	13.997435200225839
MTAT.03.158	14.03164804811139
LTAT.06.003	14.140568888488374
LTAT.05.007	16.073146029635808
MTAT.03.315	16.217035683025276
LTAT.03.003	19.037930596681267
MTAT.07.017	26.843721944923338
LTAT.TK.001	29.289459948049767
LTAT.05.006	29.59891347619933
MTAT.03.242	30.354387801438804
MTAT.03.305	31.156234563354545
MTAT.03.326	31.231696457851953
MTAT.03.328	31.415087431644366
MTAT.03.319	31.8591699253436
LTAT.03.006	31.9102316556081
MTAT.07.022	32.01197410502336
MTAT.03.132	32.035986775782646
LTAT.01.001	32.05234416499433
MTAT.TK.010	32.408757136011246
LTAT.05.018	32.44393559823938
MTAT.03.325	32.681154054748106
MTAT.03.206	32.74505931072203
MTAT.05.127	32.76106203743762
MTAT.08.027	33.24446291365604
LTAT.02.001	33.355380122319254
LTAT.06.004	33.47269300340683
MTAT.03.318	33.498519389949735

**Figure 22. Sample output of the similarity comparison between courses for the course MTAT.05.116.**

In Figure 22 shown above, there is a list of all courses in the current semester (Spring 2019) sorted by their differences from the course “MTAT.05.116 Algoritmid ja Teooria” in the ascending order. According to this list, the most similar course to the “MTAT.05.116” course is the “MTAT.03.231 Äriprotsesside juhtimine”. While assessing this result, one might reach a conclusion that these two courses indeed correlate to each other, because the MTAT.05.116 course discusses the importance of the complexity of an algorithm and how this complexity can be calculated while the course MTAT.03.231 teaches how to analyze the business processes and how they can be optimized to lower the complexity of the whole process.

As per the list from Figure 22, the largest difference from the course MTAT.05.116 has the course “MTAT.03.318 Praktika mitteinformatikutele”. These two courses indeed have nothing in common as the first course has a deep mathematical background (course lecture materials contains a lot of formulas and practical sessions have a lot of calculation exercises) and lots of theory discussed, while the second course gives students an opportunity to register and apply their practical experience in the IT world in general.

## **8 Future work**

As it has been concluded in section 6.3, the algorithm technically works. However, it requires some serious upgrade before real students can use it. Below are listed some ideas on how to improve the quality of supplementary materials.

### **8.1 Fake news algorithm**

Rapid spreading of fake content has become one of the top concerns of the online world. The growth of fake data is intensive and almost unstoppable today, especially due to a vast amount of microsites allowing unpleasing content. For this application as well there is a possibility that the algorithm will find an article that contains fake materials, reading which will confuse students and prevent from understanding the course materials in a proper way. This is why passing supplementary materials found via Google through some algorithm that can detect fake content with reasonable accuracy, will improve that quality of this application and, therefore, students will get a chance to read legitimate supplementary materials for any course. Fortunately, Google itself realises the damage that fake content can do the business, so they also gather forces to keep fake news off its platform [19].

### **8.2 Google scraping**

Currently, the tokens that the algorithm uses to build Google search query are chosen in random order. However, with such approach, even after the preprocessing, there is a possibility that the set of chosen tokens does not really reflect the content of the course materials and contains only general words that do not relate to any field of science. As a result, the Google engine will not find any articles that can serve as good supplementary materials for students. This is by far considered the most significant issue with the algorithm.

So, one of the ways how the algorithm in this application can be tuned is to add deeper semantics control to the algorithm that will remove all words that do not carry essential meaning for the materials. Currently, the algorithm only removes stop words such as “I” or “day”, but this cleaning is not enough. One way to do that is to create some clouds of words for each field of science and in Google search using only those tokens that are contained in some of the clouds.

Another problem related to how the Google engine processes query. Once a Google user types keywords into the search bar, the Google engine looks only for those documents where all these keywords are in. As each course covers different subjects, each subject has different keywords. Therefore, the challenge here is to correctly build up the Google query to optimize the search process. One way to do it is to use tokens from the course title or headlines of its subject. The reason why this approach has not been implemented in the current version is explained in section 3.4. Another way would be to run the algorithm for every subject of the course separately. However, this will significantly increase the time of the algorithm’s work

and may even cause a crash in the system. For example, Google does not allow to send too many requests to the server in a little amount of time.

### **8.3 Personalization**

One big improvement of this application would be implementing a personalized search of the supplementary materials for each student at the University of Tartu. Nowadays, personalized recommendations are used in each modern platforms and applications: Netflix, Pinterest, Amazon, IMDB etc. (see section 1.1). The advantage of this is that when an end-user starts to use the application, they already know that they probably will like suggestions as they have been made according to the specific tastes and needs of this particular user. It goes the same for the students, because every student is an individual, with his own needs and problems in each course. So, if the application could be personalized, it would increase the usability of this application.

Each student can be analyzed by his grades in each course that he has taken. The personalization system could find correlations between different courses and predict which course will go not so easy for the particular student according to his final grades of the correlated courses that the student has already taken in the previous semesters. If some course has been failed or has been given grades like E, D or even C, then this course can be considered difficult to this student. Once the predictions are made, the algorithm to find supplementary materials may start and the final output for the student would notify him that the system suggests paying more attention to this and that courses as his previous courses that correlate to these courses did not go so well.

### **8.4 Integration with SIS**

The result of this thesis is a standalone application. However, it can be integrated with SIS to increase the accessibility of the application by the students. After the integration, it may become even easier to personalize the application (see section 8.3) as the SIS team has already published the list of available APIs in the spring 2019, which can be found here: <https://ois2dev.ut.ee/doc/>. There is a GET request `/api/courses` in the “Courses backend” section that will provide a full list of all courses in the SIS. This list can be optionally filtered by course’s code or its title. This query can be used to get a list of courses for the currently authenticated user to get his courses, analyze them and process (see Chapter 8.3 for further details).

### **8.5 The optimization and securing**

The latest version of the application has been uploaded to the test server and is available for testing via <http://172.17.64.156/#/courses>. Note that this address is only available within the UT network.

The application consists of the two projects: client side and server side. In the current version, the client side of the application communicates with the server side directly, without using

any proxy. More information about the current configuration is available here: <https://angular.io/guide/build>. To sum up, there is a directory `src/app/config` in the client side of the application, which contains all configuration files. The most important of them are `config.ts` and `config-data.ts`. The first file contains a function that retrieves the server side address that is required to send requests to the server. The second file contains configurable parameters such as `backendUri` and `env`. The first parameter is an address of the server side application and it needs to be configured every time the application is deployed to a new environment. For local development that address would be `localhost` along with a port where the server side application runs on. The configuration for the current test environment is shown below:

```
export class ConfigData {
  public static CONFIG_DATA = {
    'backendUri': 'http://172.17.64.156:8091',
    'env': 'prod'
  };
}
```

The other two configuration files in this directory - `live-config-data.ts` and `test-config-data.ts`. Those are help files that can be used in case if the application needs to be deployed to several environments. So, while deploying, the data from one of these files needs to be copied to the `config-data.ts` file.

In the future, once the algorithm and the application are ready to be deployed into the live environment (whether it is an independent website or as a part of SIS), the communication between frontend and backend could be rewritten to avoid direct requests. One way to do that is to try proxying option described in <https://angular.io/guide/build>.

The current direct communication is an easy target for the hackers to perform DDoS attacks towards the server side of the application. Fortunately, the current version of the application mostly works on schedule and no time-consuming functions can be activated via the end user input from the backend. To open the found materials, the end user only activates a function that reads already collected materials from the `.txt` files and return it to the user. The only place in the application that could be a target of the DDoS attacks is a similarities calculating function. In this function's logic, if there are no tokens found for any course in the current semester courses list, then the system will run Python scraping and tokenization code to collect those tokens in order to calculate the difference between this course and the specified course. However, as the CRON server is scheduled to update the tokens for each course in the current semester every month, then there should not be a situation when there are no tokens for some course found. If tokens for each course in the current semester are present, then the following part of the difference calculation function is easy and no time-consuming, so DDoS attacks will become irrelevant.

The reason why the current deployment uses direct communication is that there is no potential reason for hackers to attack this application as it does not use any sensitive information. All used aliases are publicly available. As for DDoS attacks that are usually performed to block the flow of the algorithm, they are hard to do, because all time-consuming functions are scheduled and cannot be activated from the user input. Later, if the application will be personalized, the communication should be more protected, because personal students information (such as current user information, student courses and grades) is required to be used for the personalization.

## 9 Conclusion

Finding quality materials is a difficult task with a lot of details to take into consideration in each step of the algorithm starting with gathering data from different sources, processing the scraped data, looking up new materials and finishing with assessing the similarity and suitability of the found potential supplementary materials. The purpose of this thesis is to automate all these steps and build the application, which periodically collects data from the University courses, analyzes it and finds supplementary materials that could be helpful for the course takers. All the way along there are a lot of difficulties both expected and unexpected. How to assess the results? How to choose tokens? There are a lot of ways how this application can be improved to suit better needs of the users.

There are two main outcomes of this thesis. Firstly, a written paper that describes the logic of the implemented algorithm as well as the analysis of it and its results to answer the research question (see Introduction for further details). The answer to that is that the automated algorithm can indeed find supplementary materials for the students to help them in their studies. However, the algorithm still requires further development to achieve better performance (see section 6.3 for validation of the algorithm's results.). The most significant role in the algorithm's work play tokens that have been chosen for Google search as they are supposed to cover the content of the course and define the content of the articles that Google should find. The second outcome of this thesis project and a contribution is a working and fully automated portal built from scratch that runs the algorithm to search for possible supplementary materials for the courses in the ongoing semester in the Institute of Computer Science at the University of Tartu. Another contribution that has been done as a necessity in developing the portal is an updated Python code (initially written by Ragnar Vent [2]) that scrapes and preprocessed the content of each course in the ongoing semester in the Institute of Computer Science. All changes in Python code are listed in section 3.1 of this paper.

The goal of this thesis was to implement a web application with the main purpose to find supplementary materials for the courses that are taught in the Institute of Computer Science at the University of Tartu via Google search engine. The result of this thesis project is a working portal with the programmed algorithm that looks for the supplementary articles. The first step of the algorithm is to collect and process materials from the University courses and give a set of cleaned tokens as a result. This is done using Python code from Ragnar Vent's Master's thesis [2]. Next, after randomly choosing  $X$  tokens for the future analysis, build up a Google query, run it and scrape data from all valid links. The third step is to process articles found via Google engine. The preprocessing includes retrieving raw text (from, for example, .pdf file), tokenization, cleaning and lemmatization. The output of this step is a clean set of tokens without any noises or incomplete data. Now, before the next step of the algorithm,  $Y$  tokens must be chosen randomly from both sets of the specified course tokens and each of found Google article's tokens. Once it is done, each set of chosen tokens will be converted

into a numeric format to make it possible to measure the similarity between the set of tokens from the specified course and each set of tokens from Google articles. The fourth step is to compute the SVD matrix for the dimensionality reduction in order to speed up the workflow of the algorithm. Next and the final step is to compute Cosine similarities between the set of tokens from the course materials and each of the found candidate articles. When all similarities are computed, the final set is sorted by the similarity value in ascending order and the final output is shown to the end-user.

Example sets of tokens scraped using Python code during the testing have been uploaded to the server side project's repository into `src/main/resources/tokens` directory. The example supplementary materials for each course in the Institute of Computer Science that are taught in the Spring semester 2019 have been also uploaded to the server side repository into the `src/main/resources/materials` directory.

There is also another smaller but useful part of this application that runs another algorithm, which returns a list of courses in the current semester sorted by their differences percentwise from the specified course. This gives students a brief overview of how courses relate to each other and which courses should be taken or reviewed in case any particular course is in the interest of a student or if the student needs help with any particular course. In this case the materials from the similar courses may be useful to read and study. Despite the usefulness of this algorithm, the primary goal is the search of supplementary materials for each course in the ongoing semester. The latest assessment of the algorithm's output has shown that the algorithm still needs to be improved before it can be presented to students. The suggested improvements are described in Chapter 8. The built application consists of two project - server side and client side. The repository for the server side is:

```
https://bitbucket.org/anastassiaIv/ssd-server
```

and the repository for the client side is:

```
https://bitbucket.org/anastassiaIv/ssd-client
```

There is also a separate forked repository of the modified Python code (see section 3.1 to review changes) that also needs to be pulled before the installation process:

```
https://github.com/jsutStacy/courses\_analysis
```

The installation instructions for both client and server sides (including Python project) can be found in Appendix II.

## References

- [1] Tartu Ülikool. Tartu Ülikooli statistika. <https://statistika.ut.ee/u/apeek/ut/> (11.05.2019)
- [2] Ragnar Vent. A Framework for Analysing Topics in University Courses, 2017. Master's Thesis, University of Tartu, Estonia.
- [3] Wikipedia contributors. Netflix. In Wikipedia, The Free Encyclopedia, 2019. <https://en.wikipedia.org/wiki/Netflix> (11.05.2019)
- [4] Carlos A. Gomez-Uribe, and Neil Hunt. The Netflix Recommender System: Algorithms, Business Value, and Innovation, 2015. Netflix Inc.
- [5] Xavier Amatriain and Justin Basilico. Netflix Recommendations: Beyond the 5 stars (Part 1). In The Netflix tech blog, Medium, 2012. <https://medium.com/netflix-techblog/netflix-recommendations-beyond-the-5-stars-part-1-55838468f429> (11.05.2019)
- [6] David C. Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C. Ma, Zhigang Zhong, Jenny Liu, Yushi Jing. Related Pins at Pinterest: The Evolution of a Real-World Recommender System, 2017. Pinterest Inc.
- [7] Kevin Ma. Applying deep learning to Related Pins. In The Graph, Medium, 2017. <https://medium.com/the-graph/applying-deep-learning-to-related-pins-a6fee3c92f5e> (11.05.2019)
- [8] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. Introduction to Information Retrieval. In Cambridge University Press, Cambridge University, 2008. <https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html> (11.05.2019)
- [9] John Hopcroft, Ravi Kannan. Singular Value Decomposition (SVD). In Computer Science Theory for the Information Age (pp 110-142), 2012. Carnegie Mellon University. <https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/hopcroft-kannan-feb2012.pdf> (12.05.2019)
- [10] Sonia Leach. Single Value Decomposition - A Primer, 1995. Department of Computer Science, Brown University. <http://people.csail.mit.edu/hasinoff/320/SingularValueDecomposition.pdf> (12.05.2019)
- [11] DataFlair Team. What is Dimensionality Reduction – Techniques, Methods, Components. In Machine Learning Tutorials, DataFlair, 2018. <https://data-flair.training/blogs/dimensionality-reduction-tutorial/> (11.05.2019)
- [12] Rasmus Elsborg Madsen, Lars Kai Hansen, Ole Winther. Singular Value Decomposition and Principal Component Analysis, 2004. Technical University of Denmark. [http://www2.imm.dtu.dk/pubdb/views/edoc\\_download.php/4000/pdf/imm4000](http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/4000/pdf/imm4000) (13.05.2019)
- [13] Gilbert Strang. Eigenvalues and Eigenvectors. In Introduction to Linear Algebra, Fifth edition (pp 288-303), 2016. Wellesley Wellesley-Cambridge Press. [http://math.mit.edu/~gs/linearalgebra/linearalgebra5\\_6-1.pdf](http://math.mit.edu/~gs/linearalgebra/linearalgebra5_6-1.pdf) (13.05.2019)

- [14] Wikipedia contributors. Eigenvalues and eigenvectors. In Wikipedia, The Free Encyclopedia, 2019.  
[https://en.wikipedia.org/wiki/Eigenvalues\\_and\\_eigenvectors](https://en.wikipedia.org/wiki/Eigenvalues_and_eigenvectors) (11.05.2019)
- [15] Cosine Distance. Wolfram Language & System Documentation Center.  
<https://reference.wolfram.com/language/ref/CosineDistance.html> (11.05.2019)
- [16] Sakeena M. Sirajudeen, Nur Fatihah A. Azmi, Adamu I. Abubakar. Online fake news detection algorithm. In Journal of Theoretical and Applied Information Technology, Department of Computer Science, International Islamic University Malaysia, Malaysia, 2005.  
<http://www.jatit.org/volumes/Vol95No17/7Vol95No17.pdf> (12.05.2019)
- [17] Thom Hopmans. A recommendation system for blogs content based similarity. Part 2. In The Marketing Technologist, 2016.  
<https://www.themarketingtechnologist.co/a-recommendation-system-for-blogs-content-based-similarity-part-2/> (26.04.2019)
- [18] Srijan Kumar, Neil Shah. False Information on Web and Social Media: A Survey, 2018.  
<https://arxiv.org/pdf/1804.08559.pdf> (03.05.2019)
- [19] Michael Bertini. How Google Is Addressing Fake News. In EContent, 2018.  
<http://www.econtentmag.com/Articles/Editorial/Industry-Insights/How-Google-Is-Addressing-Fake-News-128238.html> (17.02.2019)
- [20] Steven Bird, Ewan Klein, and Edward Loper. Natural Language Processing with Python, 2009. O'Reilly Media Inc.  
<http://www.nltk.org/api/nltk.html#download-directory> (03.05.2019)

## Appendix

### I Glossary

**Covariance** - a measure of how to string the correlation between two or more sets of random variates is.

**Eigenvector** - a nonzero vector that only changes by a scalar factor, when that linear transformation is applied to it [14].

**Metadata** - data about data.

**RMSE** - root mean square error - a standard deviation of prediction errors (or residuals).

**Residuals** - a measure of how far from the regression line data points are.

**Redundancy** - a condition created within the database, where one piece of data is held in several separate places.

## II Deployment instructions

Below are provided instructions for each of the three separate projects to deploy the application.

### Server-side

#### Java project:

For the deployment, the following APIs must be installed into the deployment environment: Gradle 4.0 and Java 8.

1. Clone repository from <https://bitbucket.org/anastassiaIv/ssd-server/src/master/>
2. Navigate to the downloaded repository
3. Specify another port in the *application.properties* file, if needed
4. In *application.properties* file, replace the *pythonCodeLocation* parameter with the correct value, depending on where the Python code is located
5. In *application.properties* file, replace the *tokensLocation* parameter with the correct value, depending on where the tokens directory should be located
6. In *application.properties* file, replace the *materialsLocation* parameter with the correct value, depending on where the suggested materials should be saved
7. Review other parameters in the *application.properties* file and update them if needed. The list of parameters that are used by the application is provided in Appendix III.
8. Run `./gradlew build` command
9. For testing and development, follow step 9.1. For deployment follow steps 9.2 - 9.3.
  - 9.1 Run `./gradlew bootRun`
  - 9.2 Run `./gradlew war`
  - 9.3 Deploy war file to the server. In this thesis, the war has been deployed to Tomcat.

All other configuration can be found in the downloaded repository under the directory `src/main/resources/config`. The file name is *application.properties*. Note that when specifying the location directories for tokens and materials, make sure that these folders have permissions to let the application read files and write files into these directories. Additionally, when deploying the war file to Tomcat server, make sure that Tomcat has write and read permissions for its home directory. As Python project uses nltk module, there are some packages downloaded during the tokenization stage (see section 3.2) by the nltk module. According to the nltk documentation, “packages are installed in either a system-wide directory or in the current user’s home directory” [20]. So, when the war file is run by the Tomcat, the current user is Tomcat’s user home directory. Another way would be to change the default directory for the nltk by changing the location of the NLTK\_DATA folder. In the current test environment, this directory has been changed to `/usr/local/share/nltk_data`, where nltk data has been previously installed. Also, Tomcat user has been given write and read

permissions for this folder. Lastly, there might occur a problem when Tomcat is not able to find scrapy command required by Python project to scrape course data (see section 3.2). The reason may be that the directory, where the scrapy has been installed, is not in the Tomcat's path. So, to fix it, update the Tomcat's path list by adding this directory to it.

### Python project:

For the deployment, the Python 2.7, pip 8.1.1 and swig 3.0.8 are required.

1. Clone repository from [https://github.com/jsutStacy/courses\\_analysis](https://github.com/jsutStacy/courses_analysis)
2. Navigate to the downloaded repository
3. Install *pip* to the environment
4. Install *swig* to the environment
5. Run `python setup.py install --user`
6. Run `pip install setuptools`
7. Install required dependencies by running `pip install -r requirements.txt`

Before starting the application, make sure that `bs4`, `nlTK`, `pathos`, `scikit-learn`, `pptx`, `langdetect`, `estnltk`, `docx`, `pylatexenc` modules have been installed, because, for some reason, they may have not been installed with other requirements. If any of the modules is missing, then run the following command(s) accordingly:

```
pip install BeautifulSoup4
pip install nltk
pip install pathos
pip install -U scikit-learn
pip install python-pptx
pip install langdetect
pip install docx
pip install pylatexenc
```

If the environment where the application is being installed is new and while running `pip` command the following error occurs:

```
Traceback (most recent call last):
  File "/usr/bin/pip", line 11, in <module>
    sys.exit(main())
  File "/usr/lib/python2.7/dist-packages/pip/__init__.py", line
215, in main
    locale.setlocale(locale.LC_ALL, '')
  File "/usr/lib/python2.7/locale.py", line 581, in setlocale
    return _setlocale(category, locale) locale.Error:
unsupported locale setting
```

then run the following command:

```
export LC_ALL=C
```

## Client-side

For the deployment, the following dependencies must be installed into the deployment environment: Angular 6.1.10+, node 11.3.0, angular-cli 6.0.8+, nginx/1.10.3.

1. Clone repository from <https://bitbucket.org/anastassiaIv/ssd-client/src/master/>
2. Navigate to the downloaded repository
3. Run `npm install` command
4. In `config-data.ts` file specify the correct address of the server-side by replacing the `backendUri` parameter with the correct value or clone the data from either `live-config-data.ts` or `test-config-data.ts` files depending on the environment where the application is being deployed, if the correct parameter is stored there.
5. For testing and development, follow step 5.1. For deployment follow steps 5.2 - 5.3.
  - 5.1 Run `ng serve --host 0.0.0.0 --disableHostCheck true --port 4210`
  - 5.2 Run `ng build --prod --baseHref=/'`
  - 5.3 Copy the context of the dist directory into `/var/www/html`.

To test if the client side went up, navigate to the domain where it has been deployed. For this thesis, it is <http://172.17.64.156/#/courses>. If it is needed to specify the port the application to run on, add `--port` tag to the running command. For example, `ng server --port 4210`. In `src/app/config/config-data.ts` file make sure that server-side port is the same as the port the server side of the application is running on.

In case when the application needs to be deployed using Docker, then there is a sample Dockerfile configuration located in the project root directory for both server and client sides.

### III List of configured parameters

1. **coursesUrl** - defines the link where the University courses can be found;
2. **googleLimit** - defines the number of URLs Google is checking during the search of the supplementary materials;
3. **recommendLimit** - limits the number of recommended articles that are shown to the user;
4. **tokensLimit** - number of tokens that are chosen for the Google search;
5. **candidateArticlesLimit** - number of articles that should be forwarded for the dimensionality reduction;
6. **originalInputChosenTokenForAnalysisLimit** - number of tokens from the specified course that should be chosen for the analysis;
7. **candidateInputChosenTokenForAnalysisLimit** - number of tokens from the potential candidate that should be chosen for the analysis;
8. **pythonCodeLocation** - location of the Python code that performs the first step of the algorithm;
9. **validUrls** - list of acceptable URLs that algorithm checks while Google search;
10. **invalidUrls** - list of unacceptable URLs that algorithm checks while Google search;
11. **googleTryLimits** - number of attempts algorithm tries to send Google request for the materials;
12. **tokensLocation** - the location of the directory containing files with tokens for each course;
13. **materialsLocation** - the location of the directory containing files with suggested materials for each course;

Example of the *application.properties* file configuration:

```
tokensLimit=5
coursesUrl=https://courses.cs.ut.ee
googleLimit = 50
recommendLimit = 20
candidateArticlesLimit = 50
originalInputChosenTokenForAnalysisLimit = 1000
candidateInputChosenTokenForAnalysisLimit = 1000
pythonCodeLocation=~/.courses_analysis
googleTryLimits= 10
tokensLocation=/tmp/tokens
materialsLocation=/tmp/materials
```

```
validUrls=ut.ee,wikipedia.org,nutilabor.ee,msdn.microsoft.com,docs.microsoft.com,docs.oracle.com,ibm.com/support,developer.mozilla.org,computer.org,sirp.ee,ria.ee,cs.tlu.ee,cyber.ee,
```

support.office.com,tud.ttu.ee,edu.ph.emu.ee,nbi.dk,wed.academia.eu,journal-imab-bg.org,w3  
schools.com,nngroup.com,sciencedirect.com,elsevier.com,acm.org,ieee.org

invalidUrls=ajakiri.ut.ee,translate.google,webcache

## IV SVD matrix example

```
2019-Feb-12 08:35:08.914 INFO [http-nio-8091-exec-1]
e.s.a.s.m.AbstractService - Computational Biology: Genomes,
Networks, Evolution MIT course ...
https://stuff.mit.edu/afs/athena/course/6/6.047/.../CompiledScribe
Notes2012Master.pdf -> [3.0, 0.0, 0.0, 3.0, 4.0, 0.0, 2.0, 6.0,
0.0, 3.0, 7.0, 2.0, 2.0, 1.0, 1.0, 2.0, 6.0, 3.0, 0.0, 1.0, 0.0,
2.0, 1.0, 2.0, 7.0, 4.0, 3.0, 1.0, 4.0, 1.0, 4.0, 12.0, 1.0, 2.0,
5.0, 1.0, 7.0, 4.0, 2.0, 10.0, 0.0, 6.0, 0.0, 3.0, 8.0, 3.0, 0.0,
2.0, 2.0, 0.0, 2.0, 1.0, 2.0, 1.0, 0.0, 1.0, 3.0, 3.0, 0.0, 1.0,
2.0, 0.0, 2.0, 1.0, 2.0, 5.0, 6.0, 0.0, 3.0, 2.0, 2.0, 2.0, 3.0,
0.0, 3.0, 3.0, 5.0, 2.0, 0.0, 2.0, 0.0, 3.0, 4.0, 4.0, 5.0, 3.0,
6.0, 6.0, 2.0, 1.0, 1.0, 0.0, 2.0, 6.0, 2.0, 2.0, 1.0, 1.0, 5.0,
2.0, 2.0, 0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 0.0, 1.0, 6.0, 0.0, 4.0,
2.0, 4.0, 3.0, 5.0, 10.0, 0.0, 0.0, 6.0, 4.0, 4.0, 3.0, 1.0, 0.0,
2.0, 2.0, 2.0, 0.0, 2.0, 1.0, 3.0, 6.0, 3.0, 3.0, 1.0, 1.0, 2.0,
5.0, 10.0, 2.0, 6.0, 5.0, 4.0, 0.0, 0.0, 0.0, 3.0, 2.0, 5.0, 0.0,
0.0, 2.0, 4.0, 4.0, 3.0, 1.0, 2.0, 1.0, 2.0, 1.0, 10.0, 3.0, 1.0,
2.0, 5.0, 3.0, 7.0, 3.0, 0.0, 0.0, 6.0, 4.0, 6.0, 3.0, 5.0, 1.0,
0.0, 5.0, 3.0, 2.0, 3.0, 0.0, 2.0, 0.0, 4.0, 4.0, 1.0, 0.0, 3.0,
4.0, 2.0, 0.0, 0.0, 0.0, 1.0, 5.0, 2.0, 3.0, 1.0, 1.0, 6.0, 0.0,
6.0, 4.0, 0.0, 1.0, 2.0, 8.0, 0.0, 4.0, 1.0, 2.0, 5.0, 0.0, 3.0,
0.0, 3.0, 5.0, 0.0, 1.0, 1.0, 6.0, 5.0, 2.0, 2.0, 6.0, 1.0, 4.0,
2.0, 1.0, 1.0, 3.0, 6.0, 3.0, 2.0, 0.0, 2.0, 4.0, 0.0, 3.0, 4.0,
2.0, 1.0, 1.0, 2.0, 3.0, 0.0, 1.0, 4.0, 3.0, 0.0, 0.0, 3.0, 0.0,
4.0, 4.0, 1.0, 3.0, 3.0, 2.0, 2.0, 6.0, 6.0, 0.0, 2.0, 0.0, 3.0,
1.0, 7.0, 8.0, 2.0, 5.0, 0.0, 1.0, 5.0, 1.0, 6.0, 0.0, 1.0, 4.0,
1.0, 1.0, 3.0, 1.0, 5.0, 3.0, 4.0, 2.0, 1.0, 2.0, 6.0, 2.0, 2.0,
5.0, 1.0, 2.0, 2.0, 2.0, 4.0, 6.0, 0.0, 1.0, 3.0, 5.0, 7.0, 4.0,
2.0, 3.0, 1.0, 1.0, 7.0, 0.0, 2.0, 0.0, 2.0, 0.0, 2.0, 1.0, 2.0,
2.0, 6.0, 3.0, 5.0, 2.0, 2.0, 7.0, 0.0, 3.0, 2.0, 2.0, 3.0, 2.0,
5.0, 2.0, 8.0, 7.0, 3.0, 2.0, 2.0, 8.0, 2.0, 0.0, 1.0, 0.0, 2.0,
2.0, 4.0, 1.0, 7.0, 7.0, 2.0, 1.0, 2.0, 5.0, 2.0, 1.0, 4.0, 3.0,
3.0, 1.0, 2.0, 0.0, 2.0, 4.0, 7.0, 3.0, 4.0, 6.0, 1.0, 1.0, 7.0,
2.0, 0.0, 0.0, 1.0, 1.0, 5.0, 3.0, 1.0, 1.0, 0.0, 4.0, 2.0, 2.0]
2019-Feb-12 08:35:08.915 INFO [http-nio-8091-exec-1]
e.s.a.s.m.AbstractService - AINE_MATERJAL -> [2.0, 0.0, 1.0, 4.0,
3.0, 0.0, 3.0, 4.0, 3.0, 4.0, 2.0, 2.0, 0.0, 0.0, 0.0, 0.0, 2.0,
3.0, 2.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 2.0, 4.0, 4.0, 1.0, 2.0,
2.0, 4.0, 2.0, 4.0, 0.0, 6.0, 3.0, 0.0, 0.0, 0.0, 3.0, 0.0, 1.0,
0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 3.0, 4.0, 8.0, 0.0,
0.0, 2.0, 3.0, 0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 4.0, 0.0, 0.0,
5.0, 0.0, 0.0, 0.0, 5.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
0.0, 0.0, 8.0, 1.0, 4.0, 1.0, 0.0, 0.0, 4.0, 0.0, 0.0, 0.0, 0.0,
0.0, 6.0, 2.0, 1.0, 1.0, 0.0, 0.0, 2.0, 0.0, 1.0, 2.0, 0.0, 0.0,
2.0, 2.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 4.0, 0.0,
4.0, 2.0, 2.0, 0.0, 2.0, 0.0, 1.0, 0.0, 1.0, 0.0, 0.0, 2.0, 2.0,
3.0, 3.0, 0.0, 1.0, 4.0, 2.0, 2.0, 1.0, 2.0, 1.0, 0.0, 0.0, 3.0,
5.0, 4.0, 0.0, 0.0, 0.0, 2.0, 1.0, 0.0, 1.0, 1.0, 2.0, 0.0, 0.0,
0.0, 4.0, 5.0, 0.0, 0.0, 1.0, 0.0, 2.0, 0.0, 7.0, 2.0, 1.0, 1.0,
```

```

0.0, 0.0, 0.0, 2.0, 0.0, 1.0, 4.0, 6.0, 2.0, 2.0, 0.0, 1.0, 0.0,
1.0, 2.0, 0.0, 1.0, 2.0, 1.0, 0.0, 0.0, 0.0, 0.0, 3.0, 0.0, 2.0,
0.0, 1.0, 0.0, 0.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 1.0,
0.0, 8.0, 0.0, 1.0, 4.0, 0.0, 0.0, 5.0, 0.0, 0.0, 4.0, 0.0, 3.0,
5.0, 4.0, 3.0, 2.0, 5.0, 3.0, 0.0, 6.0, 0.0, 2.0, 2.0, 0.0, 3.0,
0.0, 1.0, 0.0, 2.0, 0.0, 1.0, 0.0, 1.0, 0.0, 1.0, 0.0, 3.0, 2.0,
5.0, 4.0, 0.0, 0.0, 4.0, 0.0, 1.0, 2.0, 4.0, 0.0, 2.0, 4.0, 0.0,
3.0, 2.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0,
2.0, 3.0, 5.0, 0.0, 1.0, 5.0, 3.0, 0.0, 4.0, 1.0, 0.0, 2.0, 2.0,
2.0, 2.0, 0.0, 4.0, 1.0, 3.0, 1.0, 1.0, 0.0, 1.0, 7.0, 0.0, 0.0,
3.0, 0.0, 2.0, 2.0, 5.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 3.0,
2.0, 2.0, 1.0, 0.0, 4.0, 3.0, 0.0, 5.0, 2.0, 0.0, 2.0, 0.0, 0.0,
1.0, 0.0, 2.0, 0.0, 2.0, 0.0, 2.0, 4.0, 3.0, 1.0, 0.0, 1.0, 0.0,
5.0, 0.0, 0.0, 0.0, 4.0, 0.0, 4.0, 1.0, 0.0, 0.0, 0.0, 2.0, 0.0,
0.0, 1.0, 0.0, 6.0, 3.0, 3.0, 0.0, 3.0, 1.0, 0.0, 0.0, 1.0, 3.0,
0.0, 0.0, 2.0, 0.0, 2.0, 2.0, 1.0, 0.0, 6.0, 1.0, 0.0, 0.0, 4.0,
0.0, 0.0, 4.0, 1.0]
2019-Feb-12 08:35:08.935 INFO [http-nio-8091-exec-1]
e.s.a.s.m.RecommendArticleService - svd:
2019-Feb-12 08:35:08.935 INFO [http-nio-8091-exec-1]
e.s.a.s.m.AbstractService - printMap
2019-Feb-12 08:35:08.938 INFO [http-nio-8091-exec-1]
e.s.a.s.m.AbstractService - Computational Biology: Genomes,
Networks, Evolution MIT course ...
https://stuff.mit.edu/afs/athena/course/6/6.047/.../CompiledScribe
Notes2012Master.pdf --- AINE_MATERJAL --- -> [[3.231088933988703,
1.5033200796926796], [-1.0057270370122517, -0.4679319202656237],
[0.34783091052304316, -0.8642239472129185], [-1.896056507958866,
-3.9139205843031584], [-3.3396808018322446, -3.202038235574064],
[0.0, 0.0], [-1.1480940351315607, -2.8973550386064115],
[-5.183436658009878, -4.370945379754631], [1.0434927315691296,
-2.5926718416387553], [-1.8960565079588665, -3.913920584303159],
[-6.974891862406306, -2.7948390838126205], [-1.4959249456546124,
-2.033131091393497], [-2.191586766700688, -0.30468319696765467],
[-1.095793383350344, -0.15234159848382733], [-1.095793383350344,
-0.15234159848382733], [-2.191586766700688, -0.30468319696765467],
[-5.87909847905599, -2.6424974853288066], [-2.243887418481907,
-3.049696637090239], [0.6956618210460863, -1.728447894425837],
[-1.095793383350344, -0.15234159848382733], [0.0, 0.0],
[-1.843755856177644, -1.1689071441805727], [-1.0957933833503444,
-0.15234159848382745], [-2.1915867667006887, -0.3046831969676549],
[-7.3227227729293904, -1.930615136599721], [-3.687511712355288,
-2.3378142883611455], [-1.8960565079588665, -3.913920584303159],
[0.29553025874182626, -3.6092373873355026], [-4.035342622878331,
-1.4735903411482267], [-0.40013156230426083, -1.880789492909666],
[-3.6875117123552874, -2.3378142883611455], [-11.75819695811198,
-5.284994970657613], [-0.40013156230426083, -1.880789492909666],
[-0.8002631246085217, -3.761578985819332], [-5.4789669167517445,
-0.7617079924191477], [0.9911920797879135, -5.337685281761338],
[-6.627060951883296, -3.6590630310255543], [-4.3831735334013775,
-0.6093663939353098], [-2.1915867667006887, -0.3046831969676549],

```

[-10.957933833503489, -1.5234159848382953], [1.0434927315691296,  
 -2.5926718416387553], [-6.574760300102062, -0.9140495909029629],  
 [0.34783091052304316, -0.8642239472129185], [-3.2873801500510313,  
 -0.45702479545148167], [-8.766347066802751, -1.2187327878706187],  
 [-3.2873801500510313, -0.45702479545148167], [0.0, 0.0],  
 [-2.191586766700688, -0.30468319696765467], [-2.191586766700688,  
 -0.30468319696765467], [0.0, 0.0], [-2.1915867667006887,  
 -0.3046831969676549], [-0.40013156230426083, -1.880789492909666],  
 [-1.1480940351315607, -2.8973550386064115], [0.29553025874182626,  
 -3.6092373873355026], [2.7826472841843453, -6.913791577703348],  
 [-1.0957933833503444, -0.15234159848382745], [-3.2873801500510313,  
 -0.45702479545148167], [-2.5917183290049404, -2.1854726898773165],  
 [1.0434927315691296, -2.5926718416387553], [-1.0957933833503444,  
 -0.15234159848382745], [-2.1915867667006887, -0.3046831969676549],  
 [1.7391545526152161, -4.321119736064593], [-2.1915867667006887,  
 -0.3046831969676549], [-1.0957933833503444, -0.15234159848382745],  
 [-2.1915867667006887, -0.3046831969676549], [-5.478966916751744,  
 -0.7617079924191472], [-5.183436658009882, -4.370945379754633],  
 [0.0, 0.0], [-3.287380150051032, -0.4570247954514819],  
 [-0.4524322140854784, -4.62580293303225], [-2.1915867667006887,  
 -0.3046831969676549], [-2.1915867667006887, -0.3046831969676551],  
 [-3.2873801500510322, -0.4570247954514821], [1.7391545526152161,  
 -4.321119736064593], [-3.287380150051032, -0.4570247954514819],  
 [-1.5482255974358219, -4.778144531516077], [-5.4789669167517445,  
 -0.7617079924191477], [-2.1915867667006887, -0.3046831969676551],  
 [0.0, 0.0], [-2.1915867667006887, -0.3046831969676551], [0.0,  
 0.0], [-3.2873801500510322, -0.4570247954514821],  
 [-4.3831735334013775, -0.6093663939353102], [-4.3831735334013775,  
 -0.6093663939353102], [-2.6963196325673615, -7.6754995701224775],  
 [-2.939549239527994, -1.3212487426644028], [-5.183436658009883,  
 -4.370945379754634], [-6.226929389579023, -1.7782735381158834],  
 [-2.1915867667006896, -0.30468319696765533], [-1.0957933833503448,  
 -0.15234159848382767], [0.2955302587418256, -3.6092373873355026],  
 [0.0, 0.0], [-2.1915867667006896, -0.30468319696765533],  
 [-6.574760300102068, -0.9140495909029656], [-2.1915867667006896,  
 -0.30468319696765533], [-2.1915867667006896,  
 -0.30468319696765533], [0.9911920797879139, -5.337685281761338],  
 [-0.40013156230425995, -1.8807894929096656], [-5.1311360062286875,  
 -1.6259319396320602], [-1.8437558561776455, -1.1689071441805734],  
 [-2.1915867667006896, -0.30468319696765556], [0.0, 0.0],  
 [-0.40013156230426017, -1.8807894929096656], [0.0, 0.0],  
 [-0.7479624728273044, -1.0165655456967477], [-0.40013156230425995,  
 -1.8807894929096656], [0.0, 0.0], [0.0, 0.0],  
 [-0.40013156230425995, -1.8807894929096656], [-5.879098479055986,  
 -2.642497485328805], [0.34783091052304316, -0.8642239472129185],  
 [-4.383173533401381, -0.6093663939353116], [-1.8437558561776457,  
 -1.1689071441805734], [-4.383173533401381, -0.6093663939353116],  
 [-3.2873801500510336, -0.45702479545148256], [-5.478966916751742,  
 -0.7617079924191463], [-10.957933833503484, -1.5234159848382927],  
 [0.34783091052304316, -0.8642239472129185], [0.0, 0.0],  
 [-5.183436658009885, -4.370945379754635], [-4.383173533401381,

-0.6093663939353116], [-2.9918498913092177, -4.066262182786991],  
 [-2.5917183290049426, -2.1854726898773174], [-0.40013156230425995,  
 -1.8807894929096656], [0.0, 0.0], [-1.4959249456546089,  
 -2.0331310913934955], [-2.1915867667006905, -0.3046831969676558],  
 [-1.843755856177646, -1.1689071441805736], [0.0, 0.0],  
 [-1.843755856177646, -1.1689071441805736], [-1.0957933833503453,  
 -0.1523415984838279], [-3.2873801500510322, -0.4570247954514821],  
 [-5.879098479055987, -2.6424974853288052], [-2.5917183290049435,  
 -2.185472689877318], [-2.243887418481907, -3.049696637090239],  
 [-0.05230065178121723, -2.7450134401225843], [-1.0957933833503453,  
 -0.1523415984838279], [-1.8437558561776457, -1.1689071441805734],  
 [-4.087643274659553, -4.218603781270813], [-10.262272012457371,  
 -3.2518638792641186], [-1.4959249456546089, -2.0331310913934955],  
 [-6.226929389579023, -1.7782735381158834], [-4.783305095705633,  
 -2.490155886844973], [-4.035342622878332, -1.4735903411482274],  
 [0.0, 0.0], [0.0, 0.0], [1.0434927315691296, -2.5926718416387553],  
 [-1.5482255974358206, -4.778144531516077], [-0.8002631246085199,  
 -3.761578985819331], [-5.47896691675174, -0.7617079924191454],  
 [0.0, 0.0], [0.0, 0.0], [-1.4959249456546093,  
 -2.0331310913934955], [-4.035342622878332, -1.4735903411482274],  
 [-4.383173533401379, -0.6093663939353111], [-2.9395492395279934,  
 -1.3212487426644026], [-0.7479624728273044, -1.0165655456967477],  
 [-1.4959249456546089, -2.0331310913934955], [-1.0957933833503448,  
 -0.15234159848382778], [-2.1915867667006896,  
 -0.30468319696765556], [-1.0957933833503448,  
 -0.15234159848382778], [-9.566610191411268, -4.980311773689947],  
 [-1.5482255974358201, -4.778144531516077], [-1.0957933833503448,  
 -0.15234159848382778], [-2.1915867667006896,  
 -0.30468319696765556], [-5.131136006228683, -1.625931939632058],  
 [-3.2873801500510336, -0.45702479545148256], [-6.974891862406317,  
 -2.794839083812626], [-3.2873801500510322, -0.4570247954514821],  
 [2.4348163736613024, -6.0495676304904284], [0.6956618210460863,  
 -1.728447894425837], [-6.226929389579024, -1.7782735381158838],  
 [-4.035342622878333, -1.4735903411482278], [-6.5747603001020645,  
 -0.9140495909029642], [-3.2873801500510336, -0.45702479545148256],  
 [-5.478966916751739, -0.761707992419145], [-0.40013156230425995,  
 -1.8807894929096656], [0.0, 0.0], [-5.131136006228682,  
 -1.6259319396320575], [-1.8960565079588656, -3.9139205843031584],  
 [-0.10460130356243269, -5.490026880245168], [-2.5917183290049435,  
 -2.185472689877318], [0.6956618210460863, -1.728447894425837],  
 [-2.1915867667006896, -0.30468319696765556], [0.34783091052304316,  
 -0.8642239472129185], [-4.383173533401381, -0.6093663939353116],  
 [-4.035342622878333, -1.4735903411482278], [-0.40013156230425995,  
 -1.8807894929096656], [0.0, 0.0], [-2.939549239527994,  
 -1.3212487426644028], [-3.687511712355293, -2.3378142883611477],  
 [-1.8437558561776464, -1.1689071441805738], [0.0, 0.0], [0.0,  
 0.0], [0.0, 0.0], [-1.0957933833503453, -0.1523415984838279],  
 [-4.435474185182584, -3.3543798340578883], [-2.1915867667006905,  
 -0.3046831969676558], [-2.5917183290049444, -2.1854726898773182],  
 [-1.0957933833503453, -0.1523415984838279], [-0.7479624728273038,  
 -1.0165655456967473], [-6.574760300102067, -0.9140495909029651],

[0.0, 0.0], [-6.226929389579026, -1.7782735381158847],  
 [-4.035342622878334, -1.473590341148228], [0.0, 0.0],  
 [-1.0957933833503453, -0.1523415984838279], [-2.1915867667006905,  
 -0.3046831969676558], [-8.766347066802762, -1.2187327878706231],  
 [0.34783091052304316, -0.8642239472129185], [-4.383173533401381,  
 -0.609366393935312], [-0.7479624728273038, -1.0165655456967473],  
 [-2.1915867667006905, -0.3046831969676558], [-2.6963196325673713,  
 -7.675499570122482], [0.0, 0.0], [-2.939549239527994,  
 -1.3212487426644028], [1.3913236420921726, -3.456895788851674],  
 [-3.2873801500510345, -0.4570247954514828], [-5.478966916751738,  
 -0.7617079924191446], [1.7391545526152161, -4.321119736064593],  
 [-1.0957933833503453, -0.1523415984838279], [-1.0957933833503453,  
 -0.152341598483828], [-5.1834366580098905, -4.370945379754637],  
 [-5.478966916751738, -0.7617079924191446], [-1.1480940351315607,  
 -2.8973550386064115], [-0.45243221408547707, -4.625802933032249],  
 [-5.1834366580098905, -4.370945379754637], [-0.052300651781216345,  
 -2.745013440122584], [-3.687511712355293, -2.3378142883611477],  
 [-0.45243221408547707, -4.625802933032249],  
 [-0.052300651781216345, -2.745013440122584], [-1.0957933833503453,  
 -0.1523415984838279], [-1.2003946869127862, -5.642368478728999],  
 [-6.574760300102069, -0.9140495909029656], [-2.5917183290049453,  
 -2.1854726898773187], [-1.495924945654607, -2.0331310913934946],  
 [0.0, 0.0], [-1.1480940351315612, -2.8973550386064115],  
 [-4.383173533401381, -0.6093663939353116], [0.34783091052304316,  
 -0.8642239472129185], [-3.2873801500510345, -0.4570247954514828],  
 [-3.687511712355293, -2.3378142883611477], [-2.1915867667006905,  
 -0.3046831969676558], [-0.7479624728273033, -1.016565545696747],  
 [-1.0957933833503453, -0.1523415984838279], [-1.8437558561776466,  
 -1.1689071441805738], [-3.287380150051035, -0.457024795451483],  
 [0.34783091052304316, -0.8642239472129185], [-1.0957933833503453,  
 -0.1523415984838279], [-3.33968080183225, -3.202038235574067],  
 [-2.5917183290049453, -2.1854726898773187], [1.7391545526152161,  
 -4.321119736064593], [1.3913236420921726, -3.456895788851674],  
 [-3.287380150051035, -0.457024795451483], [0.0, 0.0],  
 [-2.9918498913092133, -4.066262182786988], [-4.383173533401381,  
 -0.6093663939353116], [-0.7479624728273033, -1.016565545696747],  
 [-2.5917183290049457, -2.1854726898773187], [-1.8960565079588656,  
 -3.9139205843031584], [-2.1915867667006905, -0.3046831969676558],  
 [-1.4959249456546067, -2.033131091393494], [-5.183436658009891,  
 -4.370945379754637], [-6.57476030010207, -0.914049590902966],  
 [1.0434927315691296, -2.5926718416387553], [-1.4959249456546067,  
 -2.033131091393494], [0.0, 0.0], [-3.287380150051035,  
 -0.457024795451483], [-1.0957933833503453, -0.1523415984838279],  
 [-7.670553683452425, -1.0663911893867986], [-8.766347066802759,  
 -1.2187327878706222], [-2.1915867667006896, -0.30468319696765556],  
 [-5.478966916751738, -0.7617079924191446], [0.6956618210460863,  
 -1.728447894425837], [-1.0957933833503453, -0.1523415984838279],  
 [-5.478966916751738, -0.7617079924191446], [-1.0957933833503453,  
 -0.1523415984838279], [-5.87909847905599, -2.6424974853288066],  
 [1.0434927315691296, -2.5926718416387553], [0.6433611692648709,  
 -4.4734613345484195], [-4.383173533401381, -0.6093663939353116],

[-0.7479624728273031, -1.016565545696747], [0.6433611692648709,  
 -4.4734613345484195], [-2.2438874184819078, -3.0496966370902396],  
 [-1.0957933833503448, -0.15234159848382778], [-4.087643274659553,  
 -4.218603781270813], [-2.939549239527995, -1.3212487426644033],  
 [-4.383173533401381, -0.6093663939353116], [-1.4959249456546062,  
 -2.0331310913934937], [-0.4001315623042593, -1.8807894929096651],  
 [-1.4959249456546062, -2.0331310913934937], [-5.87909847905599,  
 -2.6424974853288066], [-2.1915867667006896, -0.30468319696765556],  
 [-0.8002631246085186, -3.7615789858193303], [-5.131136006228681,  
 -1.625931939632057], [-0.05230065178121612, -2.745013440122584],  
 [-1.8437558561776468, -1.168907144180574], [-1.8437558561776468,  
 -1.168907144180574], [-2.1915867667006905, -0.3046831969676558],  
 [-4.0353426228783364, -1.4735903411482294], [-4.139943926440772,  
 -6.963617221393397], [0.0, 0.0], [-1.0957933833503453,  
 -0.1523415984838279], [-2.243887418481907, -3.049696637090239],  
 [-5.478966916751736, -0.7617079924191437], [-6.974891862406326,  
 -2.79483908381263], [-3.6875117123552936, -2.337814288361148],  
 [-0.4524322140854762, -4.625802933032249], [-3.287380150051036,  
 -0.45702479545148345], [-1.0957933833503448,  
 -0.15234159848382778], [-0.4001315623042595, -1.8807894929096651],  
 [-7.670553683452422, -1.0663911893867972], [0.0, 0.0],  
 [-2.1915867667006896, -0.30468319696765556], [0.0, 0.0],  
 [-1.1480940351315616, -2.897355038606412], [0.6956618210460863,  
 -1.728447894425837], [-1.4959249456546058, -2.0331310913934937],  
 [-0.7479624728273029, -1.0165655456967468], [-2.1915867667006896,  
 -0.30468319696765556], [-0.8002631246085186, -3.7615789858193303],  
 [-5.531267568532946, -3.5067214325417244], [-3.2873801500510362,  
 -0.45702479545148367], [-3.739812364136512, -5.082827728483732],  
 [-1.4959249456546053, -2.0331310913934937], [-2.1915867667006896,  
 -0.30468319696765533], [-6.974891862406328, -2.7948390838126307],  
 [0.0, 0.0], [-3.287380150051036, -0.45702479545148345],  
 [-1.843755856177647, -1.168907144180574], [-2.1915867667006896,  
 -0.30468319696765556], [-2.5917183290049475, -2.1854726898773196],  
 [-2.1915867667006896, -0.30468319696765556], [-4.783305095705639,  
 -2.490155886844976], [-2.1915867667006896, -0.30468319696765556],  
 [-8.070685245756675, -2.9471806822964592], [-6.279230041360247,  
 -4.52328697823847], [-2.243887418481907, -3.049696637090239],  
 [-1.8437558561776473, -1.1689071441805743], [-2.1915867667006896,  
 -0.30468319696765556], [-8.41851615627972, -2.082956735083542],  
 [-2.19158676670069, -0.30468319696765567], [1.7391545526152161,  
 -4.321119736064593], [-1.095793383350345, -0.15234159848382783],  
 [0.0, 0.0], [-2.19158676670069, -0.30468319696765567],  
 [-0.8002631246085181, -3.7615789858193303], [-4.383173533401381,  
 -0.6093663939353116], [0.29553025874182626, -3.6092373873355026],  
 [-7.322722772929376, -1.9306151365997144], [-7.67055368345242,  
 -1.0663911893867963], [-2.1915867667006905, -0.3046831969676558],  
 [-1.0957933833503453, -0.1523415984838279], [-1.4959249456546049,  
 -2.0331310913934932], [-5.478966916751731, -0.7617079924191414],  
 [-2.1915867667006905, -0.3046831969676558], [-0.7479624728273024,  
 -1.0165655456967466], [-4.383173533401381, -0.6093663939353118],  
 [-1.2003946869127777, -5.642368478728995], [-2.243887418481906,

-3.049696637090239], [-0.05230065178121612, -2.745013440122584],  
[-2.1915867667006905, -0.3046831969676558], [1.0434927315691296,  
-2.5926718416387553], [-1.843755856177648, -1.1689071441805745],  
[-4.383173533401381, -0.6093663939353118], [-7.670553683452418,  
-1.066391189386796], [-2.9395492395279934, -1.3212487426644026],  
[-3.3396808018322517, -3.2020382355740677], [-6.574760300102073,  
-0.9140495909029678], [-1.0957933833503453, -0.15234159848382794],  
[-0.40013156230425884, -1.880789492909665], [-7.67055368345242,  
-1.0663911893867963], [-1.4959249456546044, -2.033131091393493],  
[0.6956618210460863, -1.728447894425837], [0.34783091052304316,  
-0.8642239472129185], [-1.0957933833503453, -0.15234159848382794],  
[0.9911920797879139, -5.337685281761338], [-5.131136006228685,  
-1.6259319396320588], [-3.287380150051037, -0.4570247954514841],  
[-1.0957933833503453, -0.15234159848382794], [0.29553025874182737,  
-3.6092373873355017], [0.0, 0.0], [-4.383173533401383,  
-0.6093663939353124], [-0.8002631246085177, -3.76157898581933],  
[-1.843755856177648, -1.1689071441805745]]  
2019-Feb-12 08:35:08.939 INFO [http-nio-8091-exec-1]

## **V Licence**

Non-exclusive licence to reproduce thesis and make thesis public

I, **Anastassia Ivanova**,

herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

**Automatization for Finding Supplementary Materials for University Courses**,

supervised by **Siim Karus**.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

**Anastassia Ivanova**

**15.05.2019**