University of Macau

Faculty of Science and Technology



Courses Recommendation SystemYour Project Title

by

LEI WENG LAMYour Name, Student No: D-B6-2539-6

Final Project Report submitted in partial fulfillment  
of the requirements of the Degree of   
Bachelor of Science in Computer Science

Project Supervisor

Prof. Yuan Wu

05 June 2020

# 

DECLARATION

I sincerely declare that:

1. I and my teammates are the sole authors of this report,
2. All the information contained in this report is certain and correct to the best of my knowledge,
3. I declare that the thesis here submitted is original except for the source materials explicitly acknowledged and that this thesis or parts of this thesis have not been previously submitted for the same degree or for a different degree, and
4. I also acknowledge that I am aware of the Rules on Handling Student Academic Dishonesty and the Regulations of the Student Discipline of the University of Macau.

Signature : \_\_\_\_\_\_\_\_\_\_\_\_ROY\_\_\_\_\_\_\_\_\_\_\_\_

Name : LEI WENG LAM

Student No. : D-B6-2539-6

Date : 05 June 2020

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my Academic Supervisor Prof. Yuan Wu, for providing their invaluable guidance, comments, and suggestion in this project.

In the idea decision stage, we have a common meeting with Prof. Yuan Wu to ask his professional opinions. Prof. Yuan Wu has sent a lot of material which can rich our idea. I feel thankful for Prof. Yuan Wu helping.

I would specially thank Prof. Yuan Wu for constantly motivating me to work smartly, and Prof. Yuan Wu give me various ideas.

**ABSTRACT**

Nowadays, online study and recommendation systems are becoming important roles in our life. In this work, we present how an online learning system works with courses recommendation through the implementation of a user interface, enrollment, and simple course management with a hybrid recommendation. Additionally, the platform requires a large number of the dataset of courses and students and ideally also the interaction (rating) of students. This project aims to help the learner to select the most appropriate courses by using a recommender system. As we can see, recommender Systems becomes a valuable extension for making personalized recommendations in which relevant to a given user. The basic idea to implement the recommender system is using either collaborative filtering or popularity based filtering. Through those methods, we can analyze the past historical data of the course ratings and enrollments. Typically, the current recommender systems are combined within one or more methods into a hybrid system. Then, our recommender system is based on the Matrix Factorization (MF) model combined with Collaborative Filtering (CF) and Popularity algorithm.

Recommendation Systems apply information retrieval and data mining techniques to select the online information relevant to a given user. Collaborative Filtering (CF) is used to build the Recommendation System. There are significant challenges, such as sparsity of rating matrix and growing nature of data faced by CF algorithms. These challenges are well-taken care by Matrix Factorization (MF).

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION 9

1.1 Background 9

1.2 Introduction 9

1.3 Objectives 10

CHAPTER 2. RELATED WORK SECTION 11

CHAPTER 3. Project Execution Schedule 12

3.1 Schedule 12

3.2 Self-contribution 12

CHAPTER 4. Functional Specification 14

4.1 Specific requirements 17

4.1.1 Users module 17

4.1.2 Courses module 17

4.1.3 Enrollment module & Rating module 17

4.1.4 User Interface (UI) 17

4.1.5 Courses recommend using hybrid approaches 18

4.1.6 Data Model and Description 18

CHAPTER 5. Software Design Specification 23

5.1.1 Data Structure Diagram 24

5.2 System Architecture 25

5.3 Recommender Architecture 25

5.4 User Interface (UI) 26

5.4.1 Rating interface 26

5.4.2 Course Information Interface 27

5.4.3 Course Management Interface 27

5.4.4 Login & Register Interface 28

5.4.5 Rating 28

CHAPTER 6. Implementation 29

6.1 Integrated Development Environment 29

6.2 Client Side 29

6.3 Server Side 30

6.4 Application Programming Interface (API): 30

6.5 Recommender system 30

6.6 Collaborative Filtering 30

6.6.1 Memory-based 31

6.6.2 Model-based 31

6.7 Matrix Factorization 32

6.7.1 Singular Value Decomposition (SVD) 33

6.7.2 Principal Component Analysis (PCA) 33

6.8 Matrix Factorization with TensorFlow 34

6.8.1 Create Matrix Factorization Model 34

6.8.2 Train Model 35

6.8.3 Evaluate Model 35

CHAPTER 7. Evaluation 37

7.1 Course Data Evaluation 37

7.2 Statistical Accuracy Metrics 37

7.2.1 Mean Absolute Error (MAE) 37

7.2.2 Root Mean Absolute Error (RMAE) 37

7.3 Decision Support Accuracy Metrics 37

7.3.1 Precision 37

7.3.2 Recall 38

7.3.3 Accuracy 38

7.3.4 F1 Score 38

7.4 Experimental Results 38

7.4.1 Quality Experiments 38

7.4.2 Performance Results 39

CHAPTER 8. Ethics and Professional Conduct 41

CHAPTER 9. Summary 42

CHAPTER 10. REFERENCES 43

LIST OF FIGURES

[Figure 1: Project Schedule 12](#_Toc42266244)

[Figure 2: Data Structure Diagram 24](#_Toc42266245)

[Figure 3: System Architecture 25](#_Toc42266246)

[Figure 4: Recommender system architecture 26](#_Toc42266247)

[Figure 5: Rating Interface 26](#_Toc42266248)

[Figure 6: Course Information Interface 27](#_Toc42266249)

[Figure 7: Course Management – Home Interface 27](#_Toc42266250)

[Figure 8: Login & Register Interface 28](#_Toc42266251)

[Figure 9: Rating Interface 28](#_Toc42266252)

[Figure 10: Matrix Factorization 32](#_Toc42266253)

[Figure 11: PCA flow 34](#_Toc42266254)

[Figure 12: Coo Matrix to CSR matrix 35](#_Toc42266255)

[Figure 13: Matrix U (user latent matrix) 35](#_Toc42266256)

[Figure 14: Matrix M (item latent matrix) 36](#_Toc42266257)

[Figure 15: Matrix Factorization Layer 37](#_Toc42266258)

[Figure 16: F1 Score Chart 40](#_Toc42266259)

[Figure 17: MSE Chart 41](#_Toc42266260)

[Figure 18: Before training 41](#_Toc42266261)

[Figure 19: After training 42](#_Toc42266262)

LIST OF TABLES

[Table 1: Use case - Register 13](#_Toc41843251)

[Table 2: Use case - Login 13](#_Toc41843252)

[Table 3: Use case - Enrol 14](#_Toc41843253)

[Table 4: Use case - Rating 14](#_Toc41843254)

[Table 5: Use case - View Course 15](#_Toc41843255)

[Table 6: Use case - Course Management 15](#_Toc41843256)

[Table 7: Use case - View Recommendation 15](#_Toc41843257)

[Table 8: Data object - User 17](#_Toc41843258)

[Table 9: Data object - Course 18](#_Toc41843259)

[Table 10: Data object- Rating 19](#_Toc41843260)

[Table 11: Data object - enrolment 19](#_Toc41843261)

[Table 12: Data object - lecture 20](#_Toc41843262)

[Table 13: Data object - assignment 20](#_Toc41843263)

# INTRODUCTION

## Background

With the rapid development of technology and online studying is gaining and becoming an essential part globally in recent years—many successful and established system applying the recommender technique in their system. However, fewer online studying platforms apply it. Most of the applications like Ummoodle only provide essential features like assignment management or group study but not extended to course recommendation.

As a user of the Ummoodle, we have to spend time inquiring about the courses with rare information. Even we cannot know the quality of the course. Additionally, recommendations had precise attractiveness in main applications such as Youtube, Netflix, and Amazon. This cause we have a novel idea of combining course recommendations and online studying.

Therefore, we want to build a web application with a recommender system. Students and professors can participate in the discussion forum and share their knowledge, experience, insights, and continuously provide a variety of information for learning. Moreover, the system can learn their historical performance and make a distinctive recommendation, including attractive and potential courses students may interest. The recommendation provided is aimed at supporting users in enrolling in processes.

Because of recommender algorithms that use Artificial Intelligence techniques, we believe Artificial Intelligence is the best approach to handle technique problems. For this reason, we use the recommender system, not a search system on our platform.

Before starting the development phase, we examine the feasibility of the project and be concerned about the technical part, do a schedule for the workloads, and consider the unique points. Afterward, we define the specifications and begin to experiment to evaluate the result we expect. Also, we have to design the database schema by well-organized data, which can be retrieved faster.

The most critical factor for the development of the recommended solution is using Matrix Factorization models. It can get the best result we expect. Further, we have implemented Matrix Factorization by Neural Networks. However, we do not get an excellent result with a limiting dataset.

Specifically, recommender systems can be classified as Content-based recommenders, Collaborative filtering recommenders, and Hybrid recommender systems. Below, we go deeper into each one of these.

## Introduction

In the era of the information explosion, users are over-run with junk on filtering out the attractive information from millions of data. Also, the learning methods are changing every day. Therefore, we aim to build an e-learning system with our new techniques and tools. Thus recommender systems are playing a critical role in getting the user's attention in recent years; Especially, recommender systems are successfully applied in many applications like Netflix and Amazon and primarily used in recommending a product or item in which users may be interested. Users can make better choices from the large catalogs and quickly find personalized information that fits their requirements.

Despite the online learning system is common in the world, such as Moodle. But our system is specifically designed to address the issue of recommending relevant courses depending upon specific interests and other latent features.

In general, recommendation systems have various types of filtering, such as collaborative filtering, content-based filtering, and hybrid filtering. For this work, we implement a hybrid recommender system with a hybrid technique to extract the information and suggest online courses. To build the hybrid recommender system, we use matrix factorization (MF) and Popularity approaches. By combining these two approaches, the system can make a personalized recommendation with high accuracy. The recommender systems derive the user preferences from analysing the implicit and explicit features regarding users' demographic information and the properties of items. In this project, we focus on discussing how our recommender systems and e-learning system work ideally together.

## Objectives

In this project, we are going to deliver a course recommender system where courses are to be recommended to students. The recommender systems play a primary role in the information era, it is widely adopted by many online services such as YouTube, Netflix, Facebook. We aim to solve an issue that they do not know fully and comprehensively about the course and the reviewing from students since they are enrolling in courses.

The general goal is to build the recommender system with APIs and collaborative filtering (CF) model. The recommender system provides a higher level of abstraction and divides the students into different groups to learn their performance (behaviour).

The primary objective of this system is to provide a comfortable and user-friendly interface where students can see the recommendation and choose an attractive course. We have devised our course recommender system with the hybrid recommender system which is involving objective functions of matrix factorization, k-Nearest Neighbours (KNN), singular value decomposition (SVD), popularity algorithm, correlation thresholding, and the nearest neighbours and also explored neural network for collaborative filtering. For the reason why we use the hybrid recommender system because hybrid can figure out the issues of visitor cold-start and item cold-start problems. The cold-start issues occur when a new item or a new user entered the platform

The final goal is to optimize the recommender system algorithm and integrate all the modules into a system by combing the main application and recommender system. In future work, we wish to successfully apply the recurrent neural network ( RNN ) in the recommender system and build a data distribution system, which can automatically trigger the event for training and evaluation under the various models.

# RELATED WORK SECTION

In this section, we briefly present the overview of web APIs, collaborative filtering, and Ruby on Rails.

In the past, we have to spend time to inquire about the courses in which we may enroll. It is not sufficient and efficient to solve the problem; Our work aims to build an extensive e-learning platform with a recommender system.

Firstly, we select Ruby on Rails as the web-application frameworks we are acquainted with to develop a back-end. Ruby on Rails is time-efficient, consistent, and cost-effective. AlsoIt is highly scalable and includes all packages needed and providing default structures for a database, a web service, and web pages**.** By the advantages,we did successfully execute our general goals, including member management, assignments management, and course management. To facilitate communication between website and back-end, we implement web APIs to specify a set of functions to access the Hypertext Transfer Protocol (HTTP) request messages, transfer data into the database, and process the data. Until the processing is completed, a result would be returned as response messages with JavaScript Object Notation (JSON) format. In general, web APIs have allowed web communities to facilitate sharing data and content between applications.

To predict the potential courses for users, we implement a hybrid recommender mechanism with collaborative filtering-based and popularity-based. This system works with data training, and predicting relied on the users' feedback and behavior. Furthermore, the core technology based on the collaborative filtering (CF) model is the Matrix factorization (MF) model. The model is high-speed, suitable, and can deliver the highest accuracies and applied with SVD and PCA.

To date, we have implemented a demo system that allows us to do the course management, member management, and recommendation. By limiting knowledge, we have to study collaborative filtering (CF) models such as user-based and item-based CF and matrix factorization models like SVD and PCA. Also, we have to make a schedule to conduct the experiment with different models to evaluate the result and design the structure of the application. But we enjoy the procedure to implement our works.

# Project Execution Schedule

## Schedule

We have separated the function to different tasks and do a schedule for the tasks. This is a 8-month project timeline (start on 25 September 2019, end on 31 May 2019) (see Figure 1).

A screenshot of a cell phone

Description automatically generated

Figure 1: Project Schedule

## Self-contribution

I primarily focus on machine learning applied in the recommender system and implement the e-learning platform with Ruby on Rails. In the early stage, we define our planning together and begin to study various knowledge. By limited knowledge, I have to spend the time to study the weakness section which is neural networking and machine learning with how to apply the learning into our application and how could we analyse the valuable information by the data. Thus, I study that knowledge on the Coursera and Udemy. After studying, I start to design the interface for our application. Afterward, I focus on implement fundamental functions and APIs. To implement the APIs, we have to design a RESTful API structure to ensure scalable and extensibility. APIs allow the frontend to connect the database and recommender system.

I use Python to apply the recommender algorithm into the recommender system, which is in modelling and evaluating users' performance on items based on their past interactions. To provide a personalized course recommendation to students, I have to build APIs to collect the user interaction records and store them in the database which my teammate designed. After the model trained, the personalized recommendation is able to recommend to users. Additionally, we have to implement a hybrid approach with combining matrix factorization (MF) models, Bayesian average algorithms, and machine learning used to enhance accuracy. These technologies are widely used in many applications, so we have an excellent chance of studying methodologies and adopting them in our project. The advantage of the hybrid approach is going to solve the cold-start issue. In deep, I employ a neural network in the matrix factorization model for improving performance. It is not easy work. I have to follow the neural network layer to implement the model. I have to design two input layers, four embedding layers, and one output in the factorization model. After work completed, I start to integrate the system. Finally, we have to do the testing and evaluation and figure out bugs for our application.

# Functional Specification

This section presents the primary functions of the course recommender system.

Table 1: Use case - Register

|  |  |
| --- | --- |
| User Case | Register Account |
| Purpose | This use case occurs when the actor indicates they want to create an account. |
| Actors | Visitors |
| Pre-Conditionals | Actor should enter a unique and valid email which does not be used.  Actor should enter a password with at least an eight-character.  Actor should enter a student name |
| Post-Conditionals | Actor can login the application by their email and password. |
| Exceptions | The required fields are empty: This occurs if the actor has not entered all the required field. The system will return error messages and ask the actor to re-enter the fields. In order that the actor should complete to fill all the required and the use case will continue. |

Table 2: Use case - Login

|  |  |
| --- | --- |
| User Case | Login Account |
| Purpose | This use case occurs when the actor indicates they want to access the information and interaction on the platform. |
| Actors | Visitors |
| Pre-Conditionals | Actor should register their accounts.  Actor should enter a registered email  Actor should enter a valid password which is associated with the email |
| Post-Conditionals | Actor have permission to access the platform. |
| Exceptions | The required fields are empty: This occurs if the actor has not entered all the required field. The system will return error messages and ask the actor to re-enter the fields. In order that the actor should complete to fill all the required and the use case will continue.  The account does not exist: This occurs if the actor enters an email which is not registered by actor, the system will return error message. In order that the actor should sign up an account with the email and the use case will continue.  The password is incorrect: This occurs If the password entered does not match the email, the system will return error message. In order that the actor should enter a correctly and the use case will continue. |

Table 3: Use case - Enrol

|  |  |
| --- | --- |
| User Case | Enrol in Course |
| Purpose | This use case occurs when the actor indicates they want to enroll in a course they are interested |
| Actors | Students |
| Pre-Conditionals | Actor should log in their account as student |
| Post-Conditionals | Actor can enter the course page.  Actor can view and access the information about the course. |

Table 4: Use case - Rating

|  |  |
| --- | --- |
| User Case | Rating |
| Purpose | This use case occurs when the actor indicates they want to rate a course |
| Actors | Students |
| Pre-Conditionals | Actor should log in their account as student  Actor should enrol in the course. |

Table 5: Use case - View Course

|  |  |
| --- | --- |
| User Case | View Course |
| Purpose | This use case occurs when the actor indicates they want to access lecture notes and assignment in the course |
| Actors | Students |
| Pre-Conditionals | Actor should log in their account as student  Actor should enrol in the course. |

Table 6: Use case - Course Management

|  |  |
| --- | --- |
| User Case | Course Management |
| Purpose | This use case occurs when the actor indicates they want to manage their course |
| Actors | Teacher |
| Pre-Conditionals | Actor should log in their account as student  Actor should create a course |

Table 7: Use case - View Recommendation

|  |  |
| --- | --- |
| User Case | View Recommendation |
| Purpose | This use case occurs when the actor indicates they want to view the recommendation |
| Actors | Student |
| Pre-Conditionals | Actor should log in their account as student |

## Specific requirements

This section describes the software requirements in detail as subsections. The application can be generally divided into five main modules: Users module, Courses module, Enrolment module & Rating module, User Interface and Courses recommend.

### Users module

    The user module allows users to manage profiles, register, and log in as a student.

    Register: Users able to create an account with their email. After the account created, the user is assigned a unique identification number (ID), which we can track the performance like rating by ID.

    Log In: Users can log in as students to enroll in the course they like after getting permission.

    Manage Profile: Users able to update their profile after they logged in as a student.

### Courses module

The course module is an essential part of the e-learning platform. This module allows students to access and receive courses' materials. Also, students can comment on the discussion forum for the asking and share knowledge. To access course's materials, users should log in their account as a student and enrolling in the courses. Course materials encompass both lecture notes and assignments, and each course may be organized differently. Besides, the course module allows the teacher to update the course detail and modify the course materials. Each updating of the course would send a notification to all students who are enrolling in the course.

### Enrollment module & Rating module

Machine learning techniques for recommender systems propose to interact with new users. Both the enrollment module & rating modules are critical elements for the system. The system can retrieve enrollment data and rating data to update the recommender model. Once the model updated, the accurate recommendation predictions are performed by the data. The enrollment module allows users to enroll in the course they like, and the rating module allows users to rating the course as the evaluation.

### User Interface (UI)

The User Interface (UI) provides a visualization depending on users' indicate to do. User interface design is an essential part of dealing with the interaction between users and data. We focus on providing a highly useful*and*efficient application. Thus, users able to be easy to use the application by a user-friendly interface. We implement our interface by using HTML and CSS. Also, Javascript used to provide the interaction between the client and application-side so that we can process the request from the application-side and show it on the interface.

### Courses recommend using hybrid approaches

We have many techniques used to implement the recommender system. As we know it, each technique includes both advantages and disadvantages. How to produce a perfect technique is trouble in recent years? We propose to use a hybrid recommender combining two or more recommender methodologies. In particular, the hybrid recommender

can be adopted to solve different use cases or different aspects of the dataset by exploiting various strategies. A cold-start issue occurs when the new users or items have either none or tiny interactions in the system, the collaborative approaches cannot predict with tiny interactions, but content-based approaches work in this situation. Our hybrid recommender system is based on the Matrix Factorization (MF) model with Tensorflow and bayesian average algorithms. Hence, recommending the items by popularity technique is the alternative way. In many applications, selecting the top few items well is crucial; Afterward, the system can collect their behavior by evaluating the previous recommendation. Take advantage of the hybrid recommender; solving the cold-start issues is easy.

The course's recommendation function is to provide students with suggestions on their interest courses after the process of predicted rating and ranking. The processing of recommendations can be separated into three sections. Firstly, data blustering, this step applied to the clustering

technique, is to partition the users into groups mapping similar users by the k-means clustering algorithm. The next step is data mining, including data analysis, data cleansing, and visualization of the results. In this step, we exploit data to find hidden patterns and correlations. Finally, the recommendation step is done after the previous is completed and computing the dataset by hybrid approaches models.

### Data Model and Description

This section describes the information domain for the Courses Recommender system. The model's data objects in the Courses Recommend that would be manipulated with the attributes of class diagrams in the following section.

#### Data objects

There are 6 types of data objects, User, Course, Rating, Enrolment, Lecture, Assignment.

##### User

This object holds the information of a specific user.

Table 8: Data object - User

|  |
| --- |
| User |
| id: integer  email: String  password: String  auth\_token: String  student\_name: String  is\_student: Boolean  created\_at: Datetime  updated\_at: Datetime |
| getUserID()  getStudentName()  addReview()  addRating()  addEnrollment()  updateUserProfile() |

##### Course

This object holds the information of a specific course item.

Table 9: Data object - Course

|  |
| --- |
| Course |
| course\_id: integer  course\_title: String  course\_image\_url: String  course\_language: String  course\_description: String  is\_student: boolean  rating\_score: integer  reviews\_count: integer  enrol\_count: integer  created\_at: Datetime  updated\_at: Datetime |
| getID()  getName()  getRating()  getInformation()  updateInformation() |

##### Rating

This object holds the information of a specific rating item.

Table 10: Data object- Rating

|  |
| --- |
| Rating |
| id: integer  user\_id: integer  course\_id: integer  rating: float |
| getRatingList()  addRating() |

##### Enrolment

This object holds the information of a specific rating item.

Table 11: Data object - enrolment

|  |
| --- |
| Enrolment |
| id: integer  user\_id: integer  course\_id: integer |
| getEnrolment()  enrol() |

##### Lecture

This object holds the information of a specific rating item.

Table 12: Data object - lecture

|  |
| --- |
| Lecture |
| Id: integer  course\_id: integer  title: String  description: String |
| getID()  getTitle()  getDescription()  addLecture()  removeLecture()  modifyLecture() |

##### Assignment

This object holds the information of a specific rating item.

Table 13: Data object - assignment

|  |
| --- |
| Assignment |
| Id: integer  course\_id: integer  title: String  description: String  attachment\_url: String |
| getID()  getTitle()  getDescription()  addAssignment()  removeAssignment()  modifyAssignment () |

# Software Design Specification

This section presents an overall system diagram for the e-learning platform with recommendation systems in the following figure. There are two primary architecture, including the application system and the recommender system side. In the beginning, we have to create the database by SQL and load the data produced by the random generation. Afterward, we work on the dataset and run the model training on the recommender engine, which processes in a distributed way using machine learning and matrix factorization (MF) models. The recommender engine is to come up with personalized recommendations. The goal of building the application is to collect users' interaction data that can be used to improve the member's experience. On the other hand, users benefit from being able to get better recommendations.

### Data Structure Diagram

The diagram (see Figure 2) presents the related of each table within a system.

A screenshot of a cell phone

Description automatically generated

Figure 2: Data Structure Diagram

To access the database, we need to create a database connection, cursor over the object, execute the SQL statement, and finally handle the exception and clean up the resources. In addition, our project allows users to be accessed by multiple users at the same time and process each users' request in multi-processor multi-thread mode. Assume that multi-threading is taken as an example. When each thread accesses the database, it must create a connection that belongs only to itself. It is not visible to other threads. Otherwise, it will confuse database operations. Therefore, we create a reliable and straightforward database access model that can operate the database both safely and efficiently in one thread.

## System Architecture

Software architecture design is a crucial process and it describes the requirements and basic software structure by separating functional areas into layers. It describes how a normal application system might interact with its users, external systems, data sources, and services. The diagram (see Figure 3) presents the pattern and services that we are workingA picture containing man, white, holding, standing

Description automatically generated

Figure 3: System Architecture

## Recommender Architecture

**Model training** is used to fit the data with features and generate a model that can be used for inference for classifying or predicting a new result, the architecture of recommender is shown below (see Figure 4). In future work, we would implement an Event and Data Distribution system that can handle all data and trigger events by the queue.

Despite this, We have to run our popularity-based model to solve the cold-start problem by providing popular items. Afterward, the result is exported with the recommended courses matched student interests.

A screen shot of a computer

Description automatically generated

Figure 4: Recommender system architecture

## User Interface (UI)

### Rating interface

The rating interface allows users to rate the course within the range of 1 – 5 stars. (see Figure 5)

A screenshot of a social media post

Description automatically generated

Figure 5: Rating Interface

### Course Information Interface

The course information interface allows users to view the information of course, comment in the discussion forum, access the lecture notes and download the assignments. (see Figure 6)

A screenshot of a cell phone

Description automatically generated

Figure 6: Course Information Interface

### Course Management Interface

The course management interface allows teacher to modify their course. For example, teacher can add the lecture notes and assignments in the course. (see Figure 7)

A screenshot of a cell phone

Description automatically generated

Figure 7: Course Management – Home Interface

### Login & Register Interface

The login & Register interface allows user to login and register an account. (see Figure 8)

A screenshot of a cell phone

Description automatically generated

Figure 8: Login & Register Interface

### Rating

The Rating interface allows user to rate the course (evaluation). (see Figure 9)

A screenshot of a cell phone

Description automatically generated

Figure 9: Rating Interface

# Implementation

This section presents how the courses recommend application is implemented with various approaches and recommender factor. Depending on its approaches, we have to develop the system with Ruby on Rails, which can provide a suitable library for implementation. In addition, we have to store our course and user information in a reliable database, so that we integrate the PostgreSQL into our Ruby of Rails application to have extreme compatibility and coherence. A login system is essential on a platform to record and save their information. By the advantage of Ruby on Rails, our application is reliable and easy to use. In order to implement the hybrid recommender system, we have to integrate two or more algorithms with Python. HTML and Javascript are essential programming languages to provide a user-friendly interface and interaction to implement the interface for the client-side. We separate our processing stages into 4 stages: Planning, Designing, Developing, and Evaluating.

In our planning stage, we define our requirements and design our schedule. Architecture is essential to ensure our development can operate successfully with other modules and libraries. We need to be clear about how our application works. Therefore, we focus on designing the architecture of our application and recommender in the designing stage and consider how to integrate both systems. In addition, we design our user interface with user-friendly and sustainable. In the developing stage, we can develop our application quickly and efficiently after all the requirements and specifications are defined and designed. The complexity of the implementation recommender poses one of the most significant challenges to their knowledge and mathematics. Evaluating and testing are essential for application development so that we have to evaluate the application and recommender system outcomes in the final stage. There are different methods for evaluating the accuracy of rating predictions for evaluating the recommender system, and we do quality testing for the application to ensure the performance.

In general, the course recommender application is organized from three-part: Client-side, Server-side, and recommender system. In the next section, we will briefly describe these three components.

## Integrated Development Environment

We use Visual Studio Code as an integrated development environment for developing our code.

## Client Side

The client-side system presents a user interface applied in the web application to provide visualization and interaction users indicate to view and action. The client-side application allows users to access the information from the server and interact with the server to receive recommendations. Users are able to register an account and log in as students into the platform. After they logged in, they are able to enrol in a course they are interested. To evaluate courses, users can rate the courses, and the client-side send the evaluating record back to the server. Therefore, the server can improve the performance of the recommender system by studying their behaviour. The client-side system can display the recommendation that obtains from the server.

## Server Side

The server-side system holds up the entire data in the PostgreSQL database and performs all the functionality operations after receiving the requests from the client-side. Specifically, when the requests come from the client-side application, the server-side system handles the requests to operate the functions the request requires. For example, when the users attempt to log in their account, the server-side receive the sign-in request and return the result by checking their credentials. The procedure of the previous processing we call the Application Programming Interface (API). Also, the server-side system can store and retrieve data to the database.

## Application Programming Interface (API):

The idea behind API is essentially a set of rules that dictate how two machines interact. Nearly all businesses employ APIs to retrieve data or interact with a database for customers to use. We present a code snippet below is an example of the API. The API returns a dataset as JSON format to users after retrieving the recommendation from the recommender engine.

class Api::V1::RecommendController < ApplicationController

before\_action :authorize\_request, only: %i[index]

# *Get /api/v1/recommend*

def index

courses = ::RecommenderEngine.new(User.first).recommend

return render\_ok(data: []) if courses.nil?

courses = ActiveModel::SerializableResource.new(

courses,

adapter: :json\_api,

each\_serializer: CourseSerializer

)

render\_ok(data: courses.as\_json[:data])

end

end

## Recommender system

The recommender system is the critical element of the whole system and be connected across all recommender systems built using this API. The system should not be passive by searching keywords on the database; instead, the system should dynamically learn the past data set with recommender algorithms and recommend the courses to users as well. We can predict the most relative course the user may want to enrol by studying past course information with the user’s preference. The server-side can trigger the event in evaluating and training the datasets with the recommender algorithm.

## Collaborative Filtering

Collaborative Filtering (CF) is a popular technique used for building a recommendation system to date. The predicted result depends on filtering the past ratings or behavior given by users in the system. The idea behind this method is intuitively obvious: to assume the connection and similarity between the users' behavior and present a reasonable prediction of their tailor preference. For example, a group of users likes to eat pizza and burgers, and one of them also likes to drink cola, and it would be recommended to the other user in the same group.

 In detail, Collaborative filtering (CF) based systems work by analyzing and collecting in the form of users' behavioral information. By observing their implicit or explicit behavior, such as feedback, ratings, preferences, and activities, the system can predict the user's behavior in the future and exploit similarities by the data.

 In general, there are different types of algorithms in the family of collaborative filtering (CF): Memory-based and Model-based.

### Memory-based

Memory-Based collaborative filtering can be divided into two main approaches: user-based filtering and item-based filtering. User-based collaborative filtering identifies users by finding the similarity between different user based on the similarity of ratings, and recommend items that those similar users liked. Amazon invented Item-based collaborative filtering in 1998. Unlike user-based collaborative filtering, item-based filtering looks at the similarity between different items. Memory-Based collaborative filtering identifies users or items by using Cosine similarity or Pearson correlation coefficients to find the closest user or items.

#### Similarity Measures

Cosine-based similarity:

Cosine similarity is a metric used to measures the similarity by calculating the cosine of the angle between two vectors. The smaller the angle, the higher the cosine similarity. It means how similar the vectors remain close to each other. The formula is given below.

Conditional-based probability:

Conditional probability is used to measure the probability by calculating the probability of an event occurring with one or more relevant events. The formula is given below.

#### score function

This function return a score that quantifies how strongly does a user likes/prefers item.

### Model-based

The model-based collaborative filtering approach involves building a model based on ML algorithms for data mining to discover hidden patterns and the relationship between users and items. Regularly, the Model-based Collaborative Filtering approach employs dimensionality reduction techniques like matrix factorization (Singular Value Decomposition — SVD, Principal Component Analysis- PCA, and Latent Factor models). The model's basic idea is to decompose the User-Item rating matrix into the user feature matrix and item feature matrix by f-dimensional vectors. We can extract the essential features from the dataset and generate a model used for inference to make recommendations without using the complete dataset. We can build a recommender system by working on a small dataset to benefit both scalability and efficiency. We present Matrix factorization (MF) Models in the next section.

## Matrix Factorization

Both of the collaborative filtering (CF) algorithms have been very successful in the past, but the problems we find about sparsity and Scalability. We cannot handle the recommender with millions of users and items as the growing dataset. The alternative approach to solve the problem is redundancy dimensions, as both users and items can be divided into groups with similar preference profiles. We can find a constant number k ideally so that k-dimensional vectors can represent both items and users.

A screenshot of a cell phone

Description automatically generated

Figure 10: Matrix Factorization

The basic idea behind Matrix Factorization (MF) (see Figure 10) is to find a matrix R by multiplied together between two matrices X and Y. The majority of Matrix Factorization (MF) models are well applying in many applications like Netflix and Amazon.

Matrix factorization (MF) is used for the calculation of massive matrix operation. It is intuitive and straightforward for finding the hidden structure behind the information with unsupervised learning methods. This advantage can overcome the problem from a high level of massive sparsity and make a high precision prediction of rating. Matrix Factorization (MF) is to discover latent features between two embeddings (user and item features) through massive user-item matrix decomposition and dimensionality reduction. However, there is a particular class of problems that caused in reducing and decomposing a large matrix. To solve it, Singular Value Decomposition (SVD) and Principal Component Analysis (PCA) are well-known models for applying in Matrix Factorization. These two methods are excellent flexibility and scalability and can identify latent factors in the field of Information Retrieval.

Firstly, we initialize two matrices m x k user matrix X and n x k item matrix. We suppose a rating *m x n* matrix R which contains all the ratings from users. Next, we assume latent features K and decompose matrix R into two origin matrix, user matrix X and item matrix Y. Finally, we have a product matrix of two matrices approximately equal to R is given by:

Secondly, we can compute the resulting dot product by these two matrices multiply together. They result in an m x n matrix is approximately equal to matrix R. In general, k is the default value of 10. The resulting dot product captures the interaction between item i and user u. This approximates user u’s rating of item i, which is denoted by , leading to the estimate

Using Matrix factorization (MF) models mapping both users and items on joint latent factor, The recommender system can easily estimate the rating a user given to the item by using Equation 1.

To learn the factor vectors ( and ), we use regularization term to create a minimize function to minimizes the regularized squared error on the initialize rating matrix.

Where κ is the set of the pairs for which is known.

For predicting unknown ratings with reasonable accuracy, the model is computing by fitting the previously observed rating matrix until the model can get a lower error value e by subtracting the original rating value by the dot product. Also, the system should control the extend of regularization to avoid overfitting by regularizing the learned parameters λ.

### Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a widely used technique to decompose a massive matrix to a lower-dimensional feature space, and exposing the original matrix's valuable properties. The basic idea of SVD is to factorize matrix R into three other matrices. For example: Let the matrix R as an input, and it gives you M, Σ and U where R is equal to the product . Both matrices M and U are orthogonal, and matrix Σ is diagonal, the following equations:

By using SVD, we can obtain the perfect matrix A by computing an approximation of original matrix R (A ≈ R). It is because matrix M is the eigenvectors of and the matrix U is the eigenvectors of .

### Principal Component Analysis (PCA)

The Principal Component Analysis (PCA) is a statistical technique used in the dimensionality reduction approach on Matrix Factorization (MF) models. PCA uses an orthogonal transformation to transform a set of observations of correlated variables into values through several linear combinations without losing original properties. In this transformation, a linear projection of high dimensional data into a lower-dimensional subspace with maximized variance retained and minimized the least-square reconstruction error. Its central aim is to overcome the dimensionality of the problem and output a lower-dimensional matrix suitable for calculation. (see Figure 11)

A close up of text on a white background

Description automatically generated

Figure 11: PCA flow

### Implementation

We have use Matrix Factorization (MF) models with PCA and SVD in this project. Firstly, we have to do data mining and clearing. Afterward, we can convert our rating matrix to Compressed Sparse Row matrix (CSR) (see Figure 12).

A screenshot of a cell phone

Description automatically generated

Figure 12: Coo Matrix to CSR matrix

Secondly, we use PCA to discover the features (genre) relation of the rating matrix R and the transformed rating matrix (see Figure 13 and Figure 14). Afterward, we will have matrix U and matrix M. Each column represents the weighted of the latent feature In both matrix U and matrix M. In the Roy row, we can see that Roy’s favourite genre is an adventure because the weighted of the column is the top.A screenshot of a cell phone

Description automatically generated

Figure 13: Matrix U (user latent matrix)

A close up of a map

Description automatically generated

Figure 14: Matrix M (item latent matrix)

Finally, we use SVD to compute matrix M, matrix U and matrix Σ together after we transpose both matrix R and to latent feature (factors) matrix U and (factors) matrix M. Also, we can predict a user rating with Dot Product.

## Matrix Factorization with TensorFlow

### Create Matrix Factorization Model

Firstly, we define parameters with a dictionary in the model. These parameters include information for counting, epochs, training hyperparameters, layer sizes, regularization, optimization hyperparameters, and batch size.

class DeepMatrixFactorization(object):

def \_\_init\_\_(self, userCount=0, itemCount=0, latentFeatures=10, regularizationScale=.0, epochs=25, batchSize=256, withBias=False, withLambda=True):

self.params = {

"userCount": userCount,

"itemCount": itemCount,

"latentFeatures": latentFeatures,

"regularizationScale": regularizationScale,

"epochs": epochs,

"batchSize": batchSize,

"withBias": withBias,

"withLambda": withLambda

}

self.\_build\_session\_()

def \_build\_session\_(self):

withLambda = self.params["withLambda"]

withBias = self.params["withBias"]

inputLayer = self.\_create\_input\_layer()

embeddingLayer = self.\_create\_embedding\_layer(

inputLayer["userLayer"], inputLayer["itemLayer"])

outputLayer = self.\_create\_output\_layer(withLambda, embeddingLayer)

if withBias:

biasInputLayer, biasOutputLayer = self.\_create\_bias\_output\_layer(inputLayer,

outputLayer)

model = self.\_create\_model(inputs=[

inputLayer["userLayer"], inputLayer["itemLayer"], biasInputLayer], outputs=[biasOutputLayer])

else:

model = self.\_create\_model(

inputs=[inputLayer["userLayer"], inputLayer["itemLayer"]], outputs=[outputLayer])

self.model = model

self.inputLayer = inputLayer

self.embeddingLayer = embeddingLayer

self.model

After the initialize, we have to create layers an input, embedding, and output, and the model is compiled with the Adam optimizer and evaluation metrics with mean squared error, root mean squared error, recall, and f1 score. The layer of matrix factorization (MF) model is shown in the below (see Figure 15):

A screenshot of a cell phone

Description automatically generated

Figure 15: Matrix Factorization Layer

### Train Model

After compiling the model, we try to train our dataset with two callbacks: model checkpoint and early stopping. The history of training records will be returned, and we can visualize the history.

def fit(self, ratingMatrix):

epochs = self.params["epochs"]

batchSize = self.params["batchSize"]

userCount = self.params["userCount"]

itemCount = self.params["itemCount"]

withBias = self.params["withBias"]

inputLayer = self.inputLayer

embeddingLayer = self.embeddingLayer

numpy.random.shuffle(ratingMatrix)

userMatrix = ratingMatrix[:, 0].reshape(-1, 1).astype(int)

itemMatrix = ratingMatrix[:, 1].reshape(-1, 1).astype(int)

biasMatrix = numpy.ones((ratingMatrix.shape[0], 1)) # *for bias*

labelMatrix = ratingMatrix[:, 2].reshape(-1, 1)

# *EarlyStopping: Interrupt training when validation performance has stopped improving.*

print("---Start to fit---")

if withBias:

history = self.model.fit([userMatrix, itemMatrix, biasMatrix], labelMatrix, epochs=epochs, batch\_size=batchSize,

validation\_split=.1, callbacks=[EarlyStopping(mode="min")])

else:

history = self.model.fit([userMatrix, itemMatrix], labelMatrix, epochs=epochs, batch\_size=batchSize,

validation\_split=.1, callbacks=[EarlyStopping(mode="min")])

return history

### Evaluate Model

After training the model, we visualize metrics on the validation set using our loaded model. We will describe all the metrics in the following section.

print(history.history["loss"], history.history["getRMSE"], history.history["val\_loss"], history.history["val\_getRMSE"])

plt.plot(history.history['loss'], label="train loss")

plt.plot(history.history['val\_loss'], label='test loss')

plt.legend()

plt.show()

# *plot rmse*

plt.plot(history.history['getRMSE'], label='train RMSE')

plt.plot(history.history['val\_getRMSE'], label='test RMSEE')

plt.legend()

plt.show()

# *plot mse*

plt.plot(history.history['f1\_score'], label='train F1 score')

plt.plot(history.history['val\_f1\_score'], label='test F1 score')

plt.legend()

plt.show()

# Evaluation

In this section, we are going to briefly discuss the evaluation process of the hybrid recommender engine with evaluation metrics. The hybrid approach is to combine matrix factorization models and Bayesian Average algorithm. After the training, we have to evaluate the result by using evaluation matrices. There are two categories of evaluation matrices: statistical accuracy metrics and decision support accuracy metrics.

## Course Data Evaluation

The basic online course data comes from Coursera through the data crawling approaches. The data its attributes like course name, course info is matched our needed. We can analyse the course and determine if it is essential for the students. In addition, we want to add the other matrix information like student ratings and related courses. All the attributes are essential for students to make their choices, and they are essential to our recommendation system.

## Statistical Accuracy Metrics

### Mean Absolute Error (MAE)

Mean Absolute Error is a measure of the average deviation (error) between the predicted rating virus correct ratings. The mean absolute error is a standard measure in the statistics. Let be the predicted ratings and be the true ratings, n is the counting. Then, the mean absolute error, denoted is defined as follows:

### Root Mean Absolute Error (RMAE)

Root Mean Absolute Error is a frequently used measure of the root average deviation (error) differences between values (sample and population values). The formula is defined as follows:

The lower the MAE and RMSE, the more accurately the recommendation engine predicts user ratings. These metrics are beneficial to use when the recommendations are based on predicting rating or number of transactions.

## Decision Support Accuracy Metrics

### Precision

Precision is a measure of result relevancy, is defined as the number of true positives over the number of true positives, plus the number of false positives. The formula is defined as follows:

### Recall

Recall is a measure of how many truly relevant results are returned, is defined as the number of true positives over the number of true positives, plus the number of false negatives. The formula is defined as follows:

### Accuracy

Accuracy is a classification metric that is easily suited for binary as well as a multiclass classification problem. The formula is defined as follows:

### F1 Score

To consider both Precision and Recall the measure F1 Score is defined as the harmonic mean by formula:

These evaluation metrics are available for recommendation systems.

## Experimental Results

### Quality Experiments

A screenshot of a cell phone

Description automatically generated

Figure 16: F1 Score Chart

A screenshot of a cell phone

Description automatically generated

Figure 17: MSE Chart

The performance of the recommender system is evaluated using the course data set. We set the latent feature k to be 10. The evaluation result is shown (see Figure 16 and Figure 17). MAE decreases significantly as the number of epochs increases.

### Performance Results

Firstly, we create an account without interaction. The system should return a list of ranks by the popularity-model, and we can see the result in Figure 18. After we generate the ratings, the system will return the prediction result with matrix factorization models by our performance records. After prediction is returned, we can see the result in Figure 19.

A screenshot of a social media post

Description automatically generated

Figure 18: Before training

A screenshot of a social media post

Description automatically generated

Figure 19: After training

# Ethics and Professional Conduct

In this project, we discover two primary problems when we attempt to implement the recommender system. The first obvious problem is to solve the cold-start problem which occurred when the new users or items have either none or tiny interactions in the system, the collaborative approaches cannot predict with tiny interactions, but content-based approaches work in this situation. In this case, we have to build a hybrid way to combine two or serval recommender algorithms to solve all possible cases. Therefore, we have to study all the recommender algorithms we might use, such as knowledge-based approaches, popularity-based approaches, collaborative-based approaches, and deep learning-based approaches. In the end, we have selected the matrix factorization (MF) model and Bayesian average algorithms and implement the matrix factorization (MF) with TensorFlow.

The second problem is the integration with various programming languages. We have to integrate both online study application and recommender systems. There is a conflict due to combining two different languages, Ruby on Rails and Python. To solve it, we have to write a shell script widely supported in different systems to communicate with each one. After the hybrid recommender system is trained, and the prediction is output, we can run the script to transfer the result into our online study application and return it to users—the simple that effective way to solve problems.

# Summary

To summarise, the final year project is the most challenging part of university life. In the beginning, we did a series of planning about design, user requirements, and schedule, after the planning, we did much research about the database, frontend development, and backend development. By limited knowledge, we faced complex issues such as data mining and machine learning. Therefore, self-studying is the most significant section in the whole processing. We have to review our machine learning knowledge, such as evaluation and metrics, so that I have studied in-depth learning course on Udemy. After reviewing and studying, I experimented with a different recommender system to get the highest performance and accuracy. We find that matrix factorization (MF) models can dramatically improve the scalability of recommender systems. In order to implement the matrix factorization (MF) models, we have been testing with the MovieLens dataset, which is run by GroupLens, a research lab at the University of Minnesota, and it is a suitable and largest dataset. Through the experiment, an independent recommender system cannot fulfill all the situation causes, which occur the cold-start issues. Thus, we build a hybrid recommender system. We choose to use Ruby on Rails to develop the web application. In addition to the planning, we have carried out on useful functions such as member module, course module, and assignment module. Because of Ruby's advantages, we have implemented the first application with Ruby on Rails by time-effectiveness and efficiency. After implementing fundamental, we study an online lesson about TensorFlow, studying neural networking and machine learning with how to interpret sensory data through a kind of machine perception, labeling, or raw clustering input.

In this project, we focus on discussing how our recommender systems and e-learning system work ideally together. We successfully implement the recommender system employed matrix factorization (MF) and bayesian average algorithms, and we get the result as our expectation.

In future work, we wish to employ the recurrent neural network ( RNN ) successfully in our recommender system. A recurrent neural network ( RNN ) is one of the approaches of the neural network. Also, we want to build an automatic workflow with the data distribution system which can handle the schedule jobs and automatically trigger the event for training and evaluation under the diversity models.

# REFERENCES

[1] Jeremy Daer, Kasper Timm Hansen, "Ruby on Rails Guides" 2019. [Online]. Available: <https://guides.rubyonrails.org/>

[2] Chhavi Saluja ,“Recommendation Systems made simple!” 2018. Available: <https://medium.com/@chhavi.saluja1401/recommendation-systems-made-simple-b5a79cac8862>

[3] Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl, “Item-based Collaborative Filtering Recommendation Algorithms” 2001. Available: <http://glaros.dtc.umn.edu/gkhome/fetch/papers/www10_sarwar.pdf>

[4] Dheeraj Bokde, Sheetal Girase, Debajyoti Mukhopadhyay, “Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey” 2015. Available: <https://www.sciencedirect.com/science/article/pii/S1877050915007462>

[5] Michael D. Ekstrand, John T. Riedl, Joseph A. Konstan, “Collaborative Filtering Recommender Systems” 2010. Available: <http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf>

[6] Xiwang Yang, Yang Guo, and Yong Liu, “Bayesian-inference Based Recommendation in Online Social Networks”. Available: <http://eeweb.poly.edu/faculty/yongliu/docs/SN_recomm_tdke.pdf>

[7] PETER BOSTRÖM, MELKER FILIPSSON, “Comparison of User Based and Item Based Collaborative Filtering Recommendation Services”. Available: <https://www.diva-portal.org/smash/get/diva2:1111865/FULLTEXT01.pdf>

[8] Google, “Matrix Factorization”. Available: <https://developers.google.com/machine-learning/recommendation/collaborative/matrix>

[9] Google, “Building a Recommendation System in TensorFlow: Overview”. Available: <https://cloud.google.com/solutions/machine-learning/recommendation-system-tensorflow-overview>

[10] Yehuda Koren, Robert Bell, Chris Volinsky, “Building a Recommendation System in TensorFlow: Overview”. Available: [https://datajobs.com/data-science-repo/Recommender-Systems-[Netflix].pdf](https://datajobs.com/data-science-repo/Recommender-Systems-%5bNetflix%5d.pdf)

[11] Oleksii Kharkovyna, “Beginner’s Guide to Machine Learning with Python”. Available: <https://towardsdatascience.com/beginners-guide-to-machine-learning-with-python-b9ff35bc9c51>

[12] Geeks for Geeks, “Introduction To Machine Learning using Python”. Available: <https://www.geeksforgeeks.org/introduction-machine-learning-using-python/>

[13] Todd Fredrich, ” Learn REST: A RESTful Tutorial”. Available: <https://www.restapitutorial.com/>