PROJECT No. ISNE P701-2/2567

Recommendation System

Newin Yamaguchi 630615028
Patcharaporn Satantaipop 630615035

A Project Submitted in Partial Fulfillment of Requirements
for the Degree of Bachelor of Engineering
Department of Computer Engineering
Faculty of Engineering
Chiang Mai University
2024

โครงงานเลขที่ วศ.สค. P701-2/2567

เรื่อง

ระบบช่วยแนะนำ

โดย

นายเนวิน ยามากุชิ รหัส 630615028
นางสาวพัชราภรณ์ สท้านไตรภพ รหัส 630615035

โครงงานนี้
เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
ปีการศึกษา 2567

Project Title	: Recommendation System	
Name	: Newin Yamaguchi	630615028
	Patcharaporn Satantaipop	630615035
Department	: Computer Engineering	
Project Advisor	: Asst. Prof. Kampol Woradit, Ph.D.	
Degree	: Bachelor of Engineering	
Program	: Information Systems and Network Engineering	
Academic Year	: 2024	

The Department of Computer Engineering, Faculty of Engineering, Chiang Mai University has approved this project to be part of the degree of Bachelor of Engineering (Information Systems and Network Engineering)

..... Department chair
(Assoc. Prof. Santi Phithakkitnukoon, Ph.D.)

Project examination committee:

..... Main advisor / Chair
(Asst. Prof. Kampol Woradit, Ph.D.)

..... Committee member
(Asst. Prof. Yuthapong Somchit, Ph.D.)

..... Committee member
(Assoc. Prof. Kenneth Cosh, Ph.D.)

หัวข้อโครงงาน	: ระบบช่วยแนะนำ	
	: Recommendation System	
โดย	: นายเนวิน ยามากุชิ	รหัส 630615028
	นางสาวพัชราภรณ์ สท้านไตรภพ	รหัส 630615035
ภาควิชา	: วิศวกรรมคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	: ผศ.ดร. กำพล วรดิษฐ์	
ปริญญา	: วิศวกรรมศาสตรบัณฑิต	
สาขา	: วิศวกรรมระบบสารสนเทศและเครือข่าย	
ปีการศึกษา	: 2567	

บทคัดย่อ

ในภูมิทัศน์ดิจิทัลร่วมสมัย การเรียนรู้ออนไลน์กลายเป็นรูปแบบการศึกษาที่ได้รับการยอมรับอย่างกว้างขวาง เนื่องมาจากความสามารถในการเข้าถึงและความคุ้มค่า การส่งมอบหลักสูตรที่ตรงเวลาตามความต้องการของผู้เรียนถือเป็นหัวใจสำคัญในการรักษาความมุ่งมั่นและความมุ่งมั่นในการเรียนรู้ โครงการนี้เน้นย้ำถึงบทบาทที่สำคัญของระบบการแนะนำในการดึงดูดผู้เรียนให้มุ่งเน้นไปที่เส้นทางการเรียนรู้และบรรลุเป้าหมาย โดยเน้นย้ำถึงการเข้าถึงทางเศรษฐกิจและโอกาสสำหรับบุคคลทั่วไปในการมีส่วนร่วมกับทักษะอาชีพที่จำเป็น ข้อกังวลเหล่านี้มีความสำคัญเป็นพิเศษเมื่อพิจารณาจากค่าใช้จ่ายที่สูงของการศึกษาแบบดั้งเดิมและการรับรู้ถึงความมีอภิสิทธิ์ของสถาบันบางแห่ง ซึ่งสร้างอุปสรรคสำหรับผู้เรียนที่ด้อยโอกาสทางเศรษฐกิจ

โครงการนี้ใช้วิธีการหลายแง่มุมเพื่อเพิ่มความแม่นยำของคำแนะนำ ในตอนแรกจะรวบรวมข้อมูลผู้ใช้เพื่อวัดการให้คะแนนคุณลักษณะ ซึ่งสะท้อนถึงปฏิกิริยาของผู้เรียนต่อหลักสูตร นอกจากนี้ คำอธิบายหลักสูตรยังได้รับการวิเคราะห์เพื่อยืนยันความสำคัญของคำและความเกี่ยวข้องระหว่างหลักสูตร แม้ว่าแนวทางตามเนื้อหาแบบสแตนด์อโลนจะไม่เพียงพอ แต่โครงการก็นำแนวทางการกรองแบบร่วมมือกันมาใช้ในลำดับถัดไป ในที่นี้ เราจะเรียนรู้การให้คะแนนคุณลักษณะผ่านโมเดล K-Nearest Neighbors (KNN) โดยใช้ประโยชน์จากการวัดความคล้ายคลึงกันเพื่อระบุผู้เรียนที่คล้ายคลึงกัน สิ่งสำคัญอย่างยิ่งคือ โครงการได้รวมวิธีการเหล่านี้เข้ากับแบบจำลองการกรองแบบไฮบริด โดยผสมผสานจุดแข็งของทั้งสองแนวทางเพื่อประสิทธิภาพสูงสุด การประเมินประสิทธิภาพแสดงให้เห็นถึงประสิทธิภาพของระบบ ซึ่งวัดผ่านความแม่นยำของ Hit Rate และ F1 Score ซึ่งแสดงให้เห็นถึงประสิทธิภาพในการเพิ่มประสิทธิภาพการเรียนรู้

Project Title	: Recommendation System	
Name	: Newin Yamaguchi	630615028
	Patcharaporn Satantaipop	630615035
Department	: Computer Engineering	
Project Advisor	: Asst. Prof. Kampol Woradit, Ph.D.	
Degree	: Bachelor of Engineering	
Program	: Information Systems and Network Engineering	
Academic Year	: 2024	

ABSTRACT

In the contemporary digital landscape, online learning has emerged as a widely embraced mode of education due to its accessibility and cost-effectiveness. Timely delivery of courses tailored to learners' needs is pivotal in maintaining their focus and commitment to learning. This project highlights the significant role of recommendation systems in attracting learners to focus on their learning paths and achieve their goals. It underscores the economic accessibility and the opportunities for individuals to engage with essential career skills. These concerns are particularly salient given the high cost of traditional education and the perceived elitism of certain institutions, which create barriers for economically disadvantaged learners.

The project employs a multi-faceted approach to enhance recommendation accuracy. Initially, it collects user information to gauge Feature Ratings, reflecting learners' reactions to courses. Additionally, course descriptions undergo analysis to ascertain word importance and inter-course relevance. While a standalone content-based approach proves insufficient, the project adopts a collaborative filtering approach next. Here, Feature Ratings are learned through a K-Nearest Neighbors (KNN) model, leveraging a similarity metric to identify similar learners. Crucially, the project integrates these methodologies into a Hybrid Filtering model, combining the strengths of both approaches for optimal performance. Performance evaluation showcases the system's efficacy, measured through Hit Rate and F1 Score accuracies, demonstrating its effectiveness in enhancing learning outcomes.

Acknowledgments

I (Mr. Newin Yamaguchi) would like to express my deepest gratitude to all those who have contributed to the completion of this project. Without their support, this endeavor would not have been possible.

First and foremost, I would like to thank my advisor (Mr. Kampol Woradit) for their guidance and invaluable insights throughout the duration of this project. Their expertise and encouragement have been instrumental in shaping the direction of my research.

Furthermore, I would like to acknowledge the support of my colleagues (Ms. Patcharaporn Satantaipop) for their constructive feedback and encouragement during the writing process. I am deeply thankful to my friends and family for their unwavering support and understanding throughout this journey. Their encouragement has been a constant source of motivation.

Finally, I would like to express my gratitude to the Department of Computer Engineering, Chiang Mai University for their financial support, which made this project possible.

Newin Yamaguchi
Patcharaporn Satantaipop
24 February 2024

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Project rationale

Traditional recommendation system problems

The obstacles of online learning become the main issue on e-Learning platform. The primary aim of this project is to address these challenges by employing various feedback models to recommend courses tailored to individual user interests [1].

Many e-Learning platforms lack sophisticated algorithms for suggesting courses based on learners' preferences [2]. Consequently, users frequently resort to selecting courses prominently displayed on the homepage, leading to a detrimental impact on study intention.

1.2 Objectives

1. To provide personalized course recommendations for learners.
2. To collect and analyze user data to improve course recommendations.

1.3 Project scope

The project scope involves identifying and utilizing appropriate algorithms based on relevant methodologies to develop a recommendation system. Clear and verifiable success criteria will be defined, aligned with the chosen methodologies. Necessary libraries for implementation will be specified, and a suitable dataset will be selected for experimentation.

1.3.1 Approaches

Our recommendation system aims to enhance user experience by predicting ratings for specific items based on individual preferences. We will utilize content-based filtering, analyzing user features and preferences to recommend items, and collaborative filtering, leveraging past user interactions to personalized recommendations [3].

1.3.2 Libraries

- **Scikit-learn:** Utilized for predictive data analysis, providing a vast collection of machine learning algorithms.
- **SciPy:** Useful for solving mathematical equations and algorithms.
- **NumPy**: Fundamental for scientific computing in Python.
- **Pandas:** Providing fast and flexible data structures for data analysis.

1.3.3 Dataset

Kaggle

Kaggle will be used as a source of quality datasets for building AI models, allowing users to explore, analyze, and share data.

1.4 Expected outcomes

1. `Improved user experience and satisfaction.`
2. `Enhanced content discovery.`

1.4.1 Hardware technology

1. **Cloud Instance:** A cloud instance with at least 8GB RAM and 4 CPU cores, such as AWS EC2 or Google Cloud Compute Engine, is required for running recommendation algorithms and serving recommended items.
2. **Graphics Processing Units (GPUs):** NVIDIA GPUs with CUDA support are recommended for speeding up the training process for deep learning-based recommendation systems.
3. **Storage:** A minimum of 100GB storage space is necessary for storing user behavior data, item features, and trained models.
4. **Memory:** At least 16GB of RAM is crucial for storing intermediate computations and caching frequently accessed data.

1.4.2 Software technology

1. **Integrated Development Environment (IDE):** Visual Studio Code, Google Colab, or any other suitable IDE can be used for building, editing, and debugging the system application.
2. **Data Preprocessing:** Microsoft Excel or Python's Pandas library is utilized for dataset processing.
3. **Programming Language:** Python 3.x is required, along with the Python Package Index (PyPI) to install the necessary recommendation packages.

1.5 Project plan

Task	Jun 2023	Jul 2023	Aug 2023	Sep 2023	Oct 2023	Nov 2023	Dec 2023	Jan 2024	Feb 2024	Mar 2024
Research/Discuss										
Testing/Experiment										
Implement										
Draft Report										
Final Report										

Table 1.1: Gantt chart

1.6 Roles and responsibilities

This project is made possible by 2 students and 1 adviser

- `Newin Yamaguchi`: Responsible for integration, scope, time, data structure, and collaboration.
- `Patcharaporn Satantaipop`: Responsible for forecasting, tools, datasets, and conclusion.
- `Kampol Woradit`: Adviser providing suggestions and support.

1.7 Impacts of this project on society, health, safety, legal, and cultural issues

The project aims to improve decision-making, educational model, and technical way [4] to benefit society by enhancing health, safety, legal compliance, and cultural inclusivity

Chapter 2
Background Knowledge and Theory

This chapter provides an overview of *e-Learning course recommendation systems*, focusing on the application of machine learning techniques [5]. It draws upon relevant research articles to establish a comprehensive understanding of the topic, aiming to provide insight into the intricacies of these systems and their relevance in the field of machine learning.

2.1 Problems

The primary challenge in e-Learning course recommendation systems is the inefficient presentation of products to users, requiring them to make decisions to discern their needs or preferences. To address this challenge, it is essential to consider the pros and cons of different approaches and their practical behavior [6].

2.2 Review of Related Studies

Focusing on an e-Learning recommendation system that assists learners in accessing diverse topics without requiring personal information. The core functionality of the recommendation algorithm facilitates learners' engagement with various online courses [7]. The implementation incorporates different techniques to address challenges, including the utilization of collaborative filtering [8]. The economic challenges faced by many individuals, especially during the COVID pandemic, highlight the importance of affordable education [9]. Developers should prioritize key components such as datasets, explicit and implicit ratings, learner behavior data, and results when implementing recommendation mechanisms [10].

2.3 Content-Based Filtering

The first step is the *Feature Rating*, where features describe users and items in a recommendation system [11]. This method works best when all items share the same set of features. However, it's not suitable for items with different features. Compatibility across items' features must be considered.

In this case, content-based filtering assesses the relevance of courses based on their descriptions. *Term Frequency and Inverse Document Frequency (TF-IDF)* [12] are implemented to calculate the weights of courses' descriptions.

1. **Term Frequency (TF):**

$$\text{tf}(\text{term}, \text{document}) = \frac{f(\text{term}, \text{document})}{\sum_{\text{term}' \in \text{document}} f(\text{term}', \text{document})} \tag{2.1}$$

2. **Inverse Document Frequency (IDF):**

$$\text{idf(term, allDocuments)} = \log\left(\frac{N}{1 + \text{df}(t)}\right) + 1 \tag{2.2}$$

3. **TF-IDF (Term Frequency-Inverse Document Frequency):**

$$\text{tfidf(term, document)} = \text{tf(term, document)} \times \text{idf(term, allDocuments)} \tag{2.3}$$

2.3.1 Advantages

1. Quick processing as it focuses on individual users without considering others.
2. Capable of capturing specific interests of niche user groups.

2.3.2 Disadvantages

1. Results depend on feature definition and user knowledge.
2. Limited to the user's current interests, lacking the ability to expand interests.

2.4 Collaborative Filtering

Collaborative filtering recommends items based on the ratings of similar users. *K-Nearest Neighbors (KNN)* [13, 14] is employed for this purpose, utilizing a similarity metric to identify the most similar users. In addition to traditional collaborative filtering, the system also incorporates an item-based algorithm that analyzes users' behavior records to calculate item similarities. This approach filters out item sets with high scores among those generated by the target user and utilizes the item similarity matrix to identify other items most similar to each item in the collection. By sorting and filtering items selected by the target user, the system then recommends the remaining items to enhance the recommendation accuracy and user satisfaction [15].

K-Nearest Neighbors (KNN):

- `Constructs a user-item matrix.`
- `Calculates cosine distances between courses.`
- `Selects top similar courses based on cosine distances.`

2.4.1 Advantages

1. Does not require feature definition.
2. Can recommend different items without specifying features.

2.4.2 Disadvantages

1. New items may not be recommended until users rate them.
2. Accuracy decreases in sparse matrices.
3. Tends to recommend popular courses over less attended ones.

2.5 Hybrid Filtering

One common thread in recommender systems is the need to combine recommendation techniques to achieve peak performance [16]. All of the known recommendation techniques in different ways.

Technique	**Background**	**Input**	**Process**
Collaborative	Ratings from U of items in I.	Ratings from u of items in I.	Identify users in U similar to u, and extrapolate from their ratings of i.
Content-based	Features of items in I	u‘s ratings of items in I	Generate a classifier that fits u‘s rating behavior and use it on i.
Demographic	Demographic information about U and their ratings of items in I.	Demographic information about u	Identify users that are demographically similar to u, and extrapolate from their ratings of i.
Utility-based	Features of items in I.	A utility function over items in I that describes u‘s preferences.	Apply the function to the items and determine i‘s rank.
Knowledge-based	Features of items in I. Knowledge of how these items meet a user‘s needs.	A description of u‘s needs or interests.	Infer a match between i and u‘s need.

Table 2.1: Recommendation Techniques.

Hybrid recommendation systems [17] combine multiple techniques to enhance performance by customizing recommendations based on specific conditions and dataset requirements.

2.6 Evaluation

Evaluation helps you assess how well your recommendation system is performing. It allows you to measure the accuracy of the recommendations made by the model and understand its effectiveness in suggesting relevant courses to users [18].

2.6.1 Training and Testing

Training and testing allow you to evaluate the performance of your recommendation system. By training the model on a subset of data and testing it on another subset that it hasn't seen during training, you can assess how well the model generalizes to new data [19]. This helps in understanding if the recommendations made by the system are accurate and relevant.

2.6.2 Hit Rate

The performance of models is evaluated by predicting a certain number of courses for each user in the training dataset. Then, for each user in the test dataset, it will check if any of the courses they actually took are among the N predicted courses. if at least one of the predicted courses matches a course taken by the user in the test dataset, it is considered as a Hit [20].

2.6.3 F1 Score

This is a metric used to evaluate the performance of a classification model. The precision, recall, and F1 value are used as evaluation indexes to measure the performance of each algorithm [21]. The precision refers to the probability that the user is interested in the recommended course list, and recall refers to the probability that the course that the user is interested in appears in the recommendation list [22]. To find the F1 Score from Recall and Precision,

2.7 ISNE knowledge used, applied, or integrated in this project

Various aspects of Computer Engineering knowledge are integrated into the project, including basic programming, object-oriented programming, data structures and algorithms, and fundamentals of database systems. Some of the key areas of knowledge applied include:

2.7.1 Basic Computer Programming for Information Systems and Network Engineering

C++ enables efficient algorithm implementation and performance optimization for recommendation systems. Basic programming skills aid in data preprocessing and custom component development. Additionally, C++ facilitates integration with existing systems and supports parallelization for improved computational efficiency.

2.7.2 Object-Oriented Programming

Object-Oriented Programming (OOP) allows for modular design, facilitating the organization of recommendation system components into reusable and understandable classes. Encapsulation ensures data integrity and abstraction simplifies complex algorithms, enhancing maintainability and scalability.

2.7.3 Data Structures and Algorithms

Data Structures optimize storage and retrieval of user-item interactions, enhancing recommendation efficiency. Algorithms like collaborative filtering or matrix factorization enable personalized recommendations based on user preferences. Efficient implementation of sorting and searching algorithms enhances recommendation performance, ensuring timely and relevant suggestions.

2.7.4 Fundamentals of Database Systems

Understanding database fundamentals aids in designing efficient storage schemas for user preferences, item attributes, and interaction data. Proficiency in database management ensures robust data retrieval and manipulation, supporting accurate recommendation generation. Knowledge of transaction management and concurrency control enhances data integrity and system reliability in handling user interactions.

2.8 Extracurricular knowledge used, applied, or integrated in this project

Understanding Python programming is crucial for our project since we'll be primarily using it to develop the recommendation system. Additionally, familiarizing ourselves with various libraries is essential, as they are key tools for implementing different functionalities within the system. Selecting and utilizing the right libraries will be pivotal in ensuring the effectiveness and efficiency of our system.

To do so, it is vital to have in-depth knowledge of data analysis, machine learning, and generative AI. These areas will help us understand the configuration, behavior, and characteristics of user items from large datasets [23]. Thus, enabling us to provide personalized recommendations by learning from the data.

Chapter 3
Project Structure and Methodology

Overall, the methodologies involved data preprocessing, implementing two different recommendation algorithms *(TF-IDF and KNN)*, combining their results into a hybrid approach, and integrating the system to become a part of backend system behind the e-Learning platform for user interaction and visualization of results. Here's a step-by-step summary of the experimentation and results:

3.1 Data Cleaning

1. Load user and item datasets that are compatible with the specified format.
2. In the user dataset, reformat dates, standardize payment statuses and education levels, clean addresses, and extract email domains. Moreover, rename all columns
3. In the item dataset, rename course and description columns
4. After finishing all processes, save it to new files.

name	**degree**	**Old**	**your mail**	**residence**	**course**	**enrollment**	**Is it paid**
John	4-year	26	john@gmail.com	Bangkok	OOP	yesterday	success
Peter	PHD	30	peter@cmu.ac.th		Web	recent days	failure
thomas	mid	42	nelson@msn.com	Chiang Mai	OS	a minute	disapprove

Table 3.1: Before cleaning the user dataset

username	**education**	**age**	**email**	**address**	**course**	**time**	**payment**
John	Bachelor degree	26	gmail.com	Bangkok	OOP	1/1/2566	success
Peter	Master degree	30	cmu.ac.th		Web App	1/5/2566	failure
thomas	High school level	42	msn.com	Chiang Mai	OS	2/2/2566	disapprove

Table 3.2: After cleaning the user dataset

Our Course Name	The Explanation
OOP	Object-Oriented Programming (OOP) allows for modular design, facilitating the organization of recommendation system components into reusable and understandable classes. Encapsulation ensures data integrity and abstraction simplifies complex algorithms, enhancing maintainability and scalability.
Web App	A web application is a software program accessed via web browsers over the internet. It offers various services, from basic websites to complex systems. It uses a client-server architecture for interaction, allowing modular design for scalability and flexibility. Technologies like HTML, CSS, JavaScript, and server-side scripting languages enable development.
OS	An operating system (OS) is software managing computer resources and providing a user interface. It handles tasks such as process, memory, file, and device management, enabling multitasking and efficient resource allocation. Encapsulation ensures data integrity, simplifies hardware interactions, and enhances maintainability and scalability. Examples include Windows, macOS, Linux, and Unix.

Table 3.3: Before cleaning the item dataset

Course	Description
OOP	Object-Oriented Programming (OOP) allows for modular design, facilitating the organization of recommendation system components into reusable and understandable classes. Encapsulation ensures data integrity and abstraction simplifies complex algorithms, enhancing maintainability and scalability.
Web App	A web application is a software program accessed via web browsers over the internet. It offers various services, from basic websites to complex systems. It uses a client-server architecture for interaction, allowing modular design for scalability and flexibility. Technologies like HTML, CSS, JavaScript, and server-side scripting languages enable development.
OS	An operating system (OS) is software managing computer resources and providing a user interface. It handles tasks such as process, memory, file, and device management, enabling multitasking and efficient resource allocation. Encapsulation ensures data integrity, simplifies hardware interactions, and enhances maintainability and scalability. Examples include Windows, macOS, Linux, and Unix.

Table 3.4: After cleaning the item dataset

3.2 Course Recommendation using TF-IDF and Linear Kernel

1. Load the cleaned dataset.
2. Match courses between user dataset and item dataset to return descriptions.
3. Use TF-IDF vectorization calculate the weights of words in the course descriptions.
4. Apply linear kernel to calculate the cosine similarities between all courses.
5. Return the top N recommended courses for the given courses taken by a specific user.

3.2.1 Term Frequency

$$logNormalization(\text{term, document}) = 1 + log(f(term, document)) \tag{3.1}$$

Let's take an example of a course with the following description:

- **Doc**: `The programming language is difficult, The local language is easy.`

logTF("The, Doc)	->	1 + log(2) ≈ 1.3
logTF("programming, Doc)	->	1 + log(1) = 1
logTF("language, Doc)	->	1 + log(2) ≈ 1.3
logTF("is, Doc)	->	1 + log(2) ≈ 1.3
logTF("difficult, Doc)	->	1 + log(1) = 1
logTF("local, Doc)	->	1 + log(1) = 1
logTF("easy, Doc)	->	1 + log(1) = 1

3.2.2 Inverse Document Frequency

$$idf(term, allDocuments) = log\left(\frac{N}{df(term)}\right) \tag{3.2}$$

Let's take an example of 2 courses with the following description:

- **Doc1**: `The web programming language is taught by a foreign professor.`
- **Doc2**: `The network security is taught by a thai professor.`

idf("The", Doc1)	->	log(2/2) = 0
idf("web", Doc1)	->	log(2/1) ≈ 0.3
idf("network", Doc2)	->	log(2/1) ≈ 0.3
idf("professor", Doc2)	->	log(2/2) = 0

3.2.3 Term Frequency and Inverse Document Frequency

$$tfidf(term, document, allDocuments) = tf(term, document) \times idf(term, allDocuments) \tag{3.3}$$

Let's take an example of a word 'language' that appears in 2 documents out of 3 documents.

- **Doc1**: `The programming language is difficult, The local language is easy.`
- **Doc2**: `The web programming language is taught by a foreign professor.`
- **Doc3**: `The network security is taught by a thai professor.`

1. **tf**("language", Doc1) = 1 + log(2) ≈ 1.3
2. **idf**("language", [Doc1, Doc2, Doc3]) = log(3/2) ≈ 0.18
3. **tfidf**("language", Doc1, [Doc1, Doc2, Doc3]) = 1.3 × 0.18 ≈ 0.23

3.2.4 Cosine Similarity

$$cosine\_similarity(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|} \tag{3.4}$$

Let's take an example of 3 courses with the following matrix where each row represents a course and each column represents a word:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	0.0	0.3	0.2	0.0	0.4	0.5	0.0	0.0	0.0	0.3	0.0	0.0	0.0	0.4	0.0
1	0.3	0.0	0.0	0.4	0.2	0.3	0.0	0.0	0.3	0.3	0.0	0.3	0.0	0.2	0.4
2	0.3	0.0	0.0	0.0	0.2	0.3	0.4	0.4	0.3	0.0	0.4	0.3	0.4	0.2	0.0

Table 3.5: User-Item Matrix

Compute the *dot product* between each pair of course vectors, so we get a course-course similarity matrix.

$$\begin{bmatrix} 1 & 0.4493628 & 0.20315676 \\ 0.4493628 & 1 & 0.44147846 \\ 0.20315676 & 0.44147846 & 1 \end{bmatrix}$$

Table 3.6: Item-Item Matrix

We can finally use the this matrix to recommend courses based on a given course.

3.3 Course Recommendation using Feature Ratings and KNN

1. Load the cleaned dataset.
2. Determine and calculate the feature ratings for each course given by users.
3. Train the model on the user-course to learn the relationships between courses based on user preferences.
4. Predict and rank the distances of courses based on the courses that the user has taken.
5. Return the top N recommended courses for the given user.

3.3.1 Feature Ratings Calculation

The *feature ratings* are also considered to recommend courses to individual users. The principle underlying this approach is to calculate the rating scores for each course based on the user's historical background. The measurement of impressive score is calculated by observing the behaviors of users in each column as follows.

- **Email** Column
  - Give 0 points if no email is provided.
  - Give 1 point if a common email domain is provided.
  - Give 2 points if an educational email domain is provided.
- **Age-Education** Column
  - Give 0 points if a person is in the educational system.
  - Give 1 point if a person just graduated.
  - Give 2 points if a person is in the working age.
- **Time** Column
  - Give 0 points if the time registration is unsuitable for a learning period.
  - Give 1 point if the time registration is suitable for a learning period.
- **Payment** Column
  - Give 0 points if the payment is overdue.
  - Give 1 point if the payment is unapproved.
  - Give 2 points if the payment is on time.

- **`Address` Column**
  - `Give 0 points if the address is blank.`
  - `Give 1 point if the address is filled.`

Find the *impressive level* individually by considering the historical background of the user and the course.

User	Course	Email	Age-Education	Time	Payment	Address	Score
User 1	Course 1	0	0	1	1	1	2.75
User 2	Course 1	1	1	1	2	0	3.75
User 3	Course 3	2	0	0	0	0	1.75
User 4	Course 3	0	1	0	2	1	3.5
User 5	Course 2	1	2	0	0	0	1.875

Table 3.7: Feature Ratings

Remove the feature rating columns to remain only user, course, and score.

User	Course	Score
User 1	Course 1	2.75
User 2	Course 1	3.75
User 3	Course 3	1.75
User 4	Course 3	3.5
User 5	Course 2	1.875

Table 3.8: Cleaned Feature Ratings

Pivot the table to get the user-course matrix.

	User 1	User 2	User 3	User 4	User 5
Course 1	2.75	3.75	NaN	NaN	NaN
Course 2	NaN	NaN	NaN	NaN	1.875
Course 3	NaN	NaN	1.75	3.5	NaN

Table 3.9: User-Course Matrix

3.3.2 Nearest Neighbors Model

Fit a k-nearest neighbors (KNN) model. Where yellow, red, and blue colours represent 3 different courses.

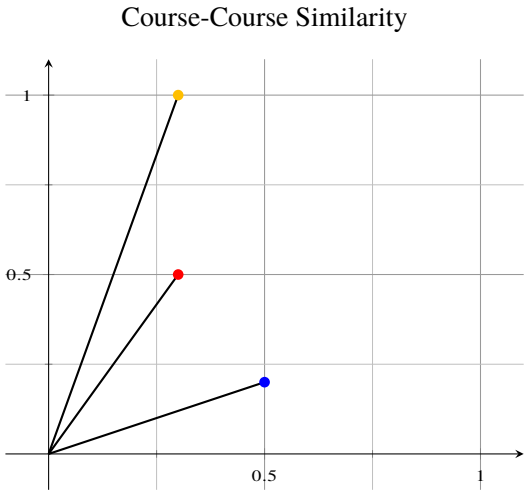

Figure 3.1: KNN Model

Use the KNN model to calculate the cosine distance between the red course and all other courses.

Course	Cosine Distance
yellow	0.25
blue	0.4

Table 3.10: Predicted Distances

After calculating the cosine distances for all courses, select the top most similar courses.

3.4 Hybrid Course Recommendation

1. Loaded the cleaned dataset.
2. Combined the results from TF-IDF and KNN approaches using a hybrid recommendation system.
3. Returned the top N recommended courses for the given user, integrating recommendations from both TF-IDF and KNN approaches.

3.4.1 The combination of TF-IDF and KNN

Set the weights and normalize the matrices of TF-IDF and KNN.

	User 1	**User 2**	**User 3**	**User 4**	**User 5**
Course 1	1.00	3.75	2.00	2.50	3.25
Course 2	2.25	2.75	1.75	1.50	1.87
Course 3	1.00	1.40	1.95	4.55	4.45

Table 3.11: Before the normalization

	User 1	**User 2**	**User 3**	**User 4**	**User 5**
Course 1	0.167	0.626	0.334	0.417	0.543
Course 2	0.486	0.594	0.378	0.324	0.404
Course 3	0.145	0.204	0.284	0.662	0.647

Table 3.12: After the normalization

Stack arrays in sequence horizontally

	Column 1	Column 2
Row 1	0.1	0.2
Row 2	0.3	0.4
Row 3	0.5	0.6

Table 3.13: Matrix 1

	Column 3
Row 1	0.7
Row 2	0.8
Row 3	0.9

Table 3.14: Matrix 2

	Column 1	Column 2	Column 3
Row 1	0.1	0.2	0.7
Row 2	0.3	0.4	0.8
Row 3	0.5	0.6	0.9

Table 3.15: Stacked Matrix

3.4.2 The process of Recommendation

We apply Nearest Neighbors to the stacked matrix to get the final recommendation. Nevertheless, the process of recommendation is the same as the KNN approach.

3.5 Evaluation

By following this evaluation process, developers and researchers can measure and improve the effectiveness of recommendation systems in providing personalized and relevant recommendations to users.

1. Consider users who have taken more than one course.
2. The training and testing datasets are splitted by identifying the ratio.
3. Allow the model to familiarize itself with the training dataset to ensure there is no bias.
4. Predict the courses for all users using the trained model.
5. Calculate the accuracy by measuring the similarities between the predicted courses and the actual test courses.
6. Investigate the results and adjust the parameters to enhance the quality of the system.

3.5.1 Training and Testing

In designing the training and testing split, several considerations can be made to ensure an effective evaluation of the recommendation system. These considerations include:

1. **Consider only users who have taken more than 1 course:** This step ensures that you have enough data from each user for meaningful analysis.
2. **Divide the courses into 2 parts individually according to the specified proportions:** This ensures that each user's data is divided according to the specified proportion.
3. **Split the dataset corresponding to the curriculum from step 2:** This ensures that the dataset is approximately divided into two parts: one for training and one for testing.

3.5.2 Accuracy Measurement

There are two performance measurements in total including *hit rate* and *f1 score*.

Hit Rate

A Hit measures the share of users that get at least one relevant recommendation. Hit Rate is calculated as follows:

$$\text{HitRate} = \frac{\texttt{Number of Hits}}{\texttt{Number of Users}} \tag{3.5}$$

A larger value of recommended courses can potentially improve the hit rate because it provides more opportunities to include relevant items. Therefore, the performance of the hit rate indeed depends on the number of recommended courses.

F1 Score

Recall, Precision, and F1 Score can use the following formulas:

$$Recall = \frac{\texttt{Number of correctly predicted items}}{\texttt{Total number of relevant items}} \tag{3.6}$$

$$Precision = \frac{\texttt{Number of correctly predicted items}}{\texttt{Total number of recommended items}} \tag{3.7}$$

$$F1\_score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{3.8}$$

- `Number of correctly predicted items` - `The course that system predict correctly.`
- `Total number of relevant items` - `The courses that have been taken by the user from the course recommendation system`
- `Total number of recommended items` - `All courses that the recommendation system suggests for user`

To find the average F1 score from all users by

$$F1\_score_{average} = \frac{\sum_{i=1}^{n} F1\_score_i}{\texttt{Number of Users}} \tag{3.9}$$

The harmonic mean nature makes sure if either Precision or Recall has a really high value, then it does not dominate the score. F1 Score has a high value when both precision and recall values are close to 1.

3.6 Function Structure

3.6.1 The TfidfLinearKernel class

The fit(item_df) method

Parameters	**item_df:** *pd.DataFrame* The DataFrame that contains course and description columns.
Returns	**matrix:** *sparse matrix of (n_sample, n_features)* The TF-IDF-Weighted document-term matrix.

The predict(name, item_df, user_df, matrix, n_recommendations=10) method

Parameters	**name:** *str* A name of user **item_df:** *pd.DataFrame* The DataFrame that contains course and description columns. **user_df:** *pd.DataFrame* The DataFrame that contains user, course, and features columns. **matrix:** *sparse matrix of (n_sample, n_features)* The TF-IDF-Weighted document-term matrix. **n_recommendations:** *int, default=10* The number of recommended courses.
Returns	**prediction_df:** *pd.DataFrame* The DataFrame that contains recommended course with score columns

The train_test_split(user_df, test_size=0.2) method

Parameters	**user_df:** *pd.DataFrame* The DataFrame that contains user, course, and features columns. **test_size:** *float, default=0.2* The proportion of the DataFrame to include in the test split.
Returns	**X_train:** *pd.DataFrame* The training DataFrame. **X_test:** *pd.DataFrame* The testing DataFrame.

The hit_rate(train, test, item_data, matrix, k=10) method

Parameters	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame. **item_df:** *pd.DataFrame* The DataFrame that contains course and description columns. **matrix:** *sparse matrix of (n_sample, n_features)* The TF-IDF-Weighted document-term matrix. **k:** *int, default=10* The number of recommendations to consider.
Returns	**hit_rate:** *float* The hit rate of the recommendation system.

The f1_score(train, test, item_data, model, k=10) method

Parameters	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame. **item_df:** *pd.DataFrame* The DataFrame that contains course and description columns. **matrix:** *sparse matrix of (n_sample, n_features)* The TF-IDF-Weighted document-term matrix. **k:** *int, default=10* The number of recommendations to consider.
Returns	**f1_score:** *float* The F1 score of the recommendation system.

3.6.2 The FeatureRatingsKNN Class

The fit(user_data) method

Parameters	**user_data:** *pd.DataFrame* The DataFrame that contains user-item data.
Returns	**model:** *NearestNeighbors* The fitted nearest neighbors estimator.

The predict(name, user_data, model, n_recommendations=10) method

Parameters	**name:** *str* The name of the user. **user_data:** *pd.DataFrame* The DataFrame that contains user-item data. **model:** *NearestNeighbors* The fitted nearest neighbors estimator. **n_recommendation:** *int, default=10* The number of top items to recommend.
Returns	**prediction_df:** *pd.DataFrame* The DataFrame that contains recommended course with score columns

The train_test_split(user_data, test_size=0.2) method

Parameters	**user_data:** *pd.DataFrame* The DataFrame that contains user-item data. **test_size:** *float, default=0.2* The proportion of the DataFrame to include in the test split.
Returns	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame.

The hit_rate(train, test, model, k=10) method

Parameters	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame. **model:** *NearestNeighbors* The fitted nearest neighbors estimator. **k:** *int, default=10* The number of recommendations to consider.
Returns	**hit_rate:** *float* The hit rate of the recommendation system.

The f1_score(train, test, model, k=10) method

Parameters	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame. **model:** *NearestNeighbors* The fitted nearest neighbors estimator. **k:** *int, default=10* The number of recommendations to consider.
Returns	**f1_score:** *float* The F1 score of the recommendation system.

3.6.3 The Hybrid Class

The fit(item_data, user_data) method

Parameters	**item_df:** *pd.DataFrame* The DataFrame that contains course and description columns. **user_data:** *pd.DataFrame* The DataFrame that contains user-item data.
Returns	**model:** *NearestNeighbors* The fitted nearest neighbors estimator.

The predict(name, item_data, user_data, model, n_recommendations=10) method

Parameters	**name:** *str* A name of user **item_df:** *pd.DataFrame* The DataFrame that contains course and description columns. **user_data:** *pd.DataFrame* The DataFrame that contains user-item data. **model:** *NearestNeighbors* The fitted nearest neighbors estimator. **n_recommendations:** *int, default=10* The number of recommended courses.
Returns	**prediction_df:** *pd.DataFrame* The DataFrame that contains recommended course with score columns

The train_test_split(item_data, test_size=0.2) method

Parameters	**item_df:** *pd.DataFrame* The DataFrame that contains course and description columns. **test_size:** *float, default=0.2* The proportion of the DataFrame to include in the test split.
Returns	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame.

The hit_rate(train, test, model, k=10) method

Parameters	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame. **model:** *NearestNeighbors* The fitted nearest neighbors estimator. **k:** *int, default=10* The number of recommendations to consider.
Returns	**hit_rate:** *float* The hit rate of the recommendation system.

The f1_score(train, test, model, k=10) method

Parameters	**train:** *pd.DataFrame* The training DataFrame. **test:** *pd.DataFrame* The testing DataFrame. **model:** *NearestNeighbors* The fitted nearest neighbors estimator. **k:** *int, default=10* The number of recommendations to consider.
Returns	**f1_score:** *float* The F1 score of the recommendation system.

Chapter 4
Experimentation and Results

In this section, we delve into the experimentation phase of our project, where the proposed methodologies were put to the test, and their efficiency was evaluated. The primary objective is to assess the performance of the recommendation system in providing personalized course recommendations to users.

4.1 Install our Python package

To install the isne-recommendation Python package, you can use the pip command in your terminal or command prompt.

```
pip install isne-recommendation
```

If you're using a Jupyter notebook, you can run this command in a Jupyter Notebook by prefixing it with an exclamation mark:

```
!pip install isne-recommendation
```

4.2 Data Preparation

In this section, we are going to write Python in the Jupyter Notebook and import classes from the isne_recommendation module that we will use to assess the model's performance and import the datasets necessary for our evaluation.

4.2.1 Import the classes

Import one or more classes from the isne_recommendation module that originally were hosted on Python Package Index (PyPI) and now are available on GitHub.

```
from isne_recommendation import TfidfLinearKernel, FeatureRatingsKNN, Hybrid
```

4.2.2 Import datasets

The dataset provided by the developer is needed to be a csv format and consists of two main components:

```
import pandas as pd
item_df = pd.read_csv('https://startg2545.github.io/item_tutorial.csv')
user_df = pd.read_csv('https://startg2545.github.io/user_item_tutorial.csv')
```

4.3 Extract a user dataset

It is crucial to split the dataset into training and testing sets to erase the test data from model's memories.

```
train, test = TfidfLinearKernel.train_test_split(user_df, test_size=0.25)
```

4.4 Fit the model

The next step is to train the model using the training dataset.

```
tfidf_matrix = TfidfLinearKernel.fit(item_df)
knn_model = FeatureRatingsKNN.fit(train)
hybrid_knn_model = Hybrid.fit(item_df, train)
```

4.5 Predict the model

So far, you can predict recommendations for every user, analyze and incorporate them into the performance assessment. Let's take an example of a user named *Tracy Marchant* below:

```
# This compares the model's predictions against the actual preferences in the test set.
true = test[test['username'] == 'Tracy Marchant']['course'] # actual preferences
first_prediction = TfidfLinearKernel.predict('Tracy Marchant', item_df, train, tfidf_matrix, 10)
second_prediction = FeatureRatingsKNN.predict('Tracy Marchant', train, knn_model, 10)
hybrid_prediction = Hybrid.predict('Tracy Marchant', item_df, train, hybrid_knn_model, 10)
```

4.6 Observe the results

This user has taken totally six courses, and the *test_size* is 25%, so the test set contains two courses by rounding.

```
true
```

	Course
647	Introduction to English Common Law
645	Retrieve Data using Single-Table SQL Queries

The first approach recommends the following courses:

```
first_prediction
```

	Course	Score
79	Write A Feature Length Screenplay For Film Or Television	0.2627
8	COVID-19 - A clinical update	0.2559
43	Introduction to International Criminal Law	0.2535
40	Health Care IT: Challenges and Opportunities	0.2400
3	Biomedical Visualisation	0.2353
67	Sharpened Visions: A Poetry Workshop	0.2334
22	EU policy and implementation: making Europe work!	0.2303
62	Python and Machine Learning for Asset Management	0.2301
17	Design and Analyze Secure Networked Systems	0.2262
68	Social Media Management	0.2217

The second approach recommends the following courses:

```
second_prediction
```

	Course	Cosine Distance	Score
0	The Product Lifecycle: A Guide from start to finish	0.7808	0.2191
2	Retrieve Data using Single-Table SQL Queries	0.6214	0.3785

The hybrid approach recommends the following courses:

```
hybrid_prediction
```

	Course	Score
43	Retrieve Data using Single-Table SQL Queries	0.5920
46	The Product Lifecycle: A Guide from start to finish	0.2088
15	Digitalisation in Space Research	0.1745
9	Collaborate on Files in Slack: Local & Google Drive Integrations	0.1670
41	Python and Machine Learning for Asset Management	0.1134
18	EU policy and implementation: making Europe work!	0.0855
1	Addiction Treatment: Clinical Skills for Healthcare Providers	0.0794
48	Water Supply and Sanitation Policy in Developing Countries Part 1: Understanding Complex Problems	0.0769
8	Coaching Practices	0.0764
25	Fundamental Neuroscience for Neuroimaging	0.0711

4.7 Calculate the accuracy

4.7.1 Tracy Marchant

There is no certain function to calculate the hit rate and F1 score for a single user. However, you can calculate them manually. Here are the results for *Tracy Marchant*:

Hit Rate

- **First Approach**: *Miss*, a predicted course does not appear in the actual preferences
- **Second Approach**: *Hit*, a predicted course appears in the actual preferences
- **Hybrid Approach**: *Hit*, a predicted course appears in the actual preferences

F1 Score

- **First Approach**: *Precision* = 0.0, *Recall* = 0.0, *F1 Score* = 0.0
- **Second Approach**: *Precision* = 0.5, *Recall* = 0.5, *F1 Score* = 0.5
- **Hybrid Approach**: *Precision* = 0.1, *Recall* = 0.5, *F1 Score* = 0.1667

4.7.2 All users

For the model evaluation, you are capable to calculate the hit rate and F1 score for all approaches by following the code below:

```
# first model evaluation
first_hit_rate = TfidfLinearKernel.hit_rate(train, test, item_df, tfidf_matrix)
first_f1_score = TfidfLinearKernel.f1_score(train, test, item_df, tfidf_matrix)

# second model evaluation
second_hit_rate = FeatureRatingsKNN.hit_rate(train, test, knn_model)
second_f1_score = FeatureRatingsKNN.f1_score(train, test, knn_model)

# hybrid model evaluation
hybrid_hit_rate = Hybrid.hit_rate(train, test, item_df, hybrid_knn_model)
hybrid_f1_score = Hybrid.f1_score(train, test, item_df, hybrid_knn_model)
```

Hit Rate

- **First Approach**: *Hit Rate Accuracy* = 14.69%
- **Second Approach**: *Hit Rate Accuracy* = 48.24%
- **Hybrid Approach**: *Hit Rate Accuracy* = 69.18%

F1 Score

- **First Approach**: *F1 Score Accuracy* = 2.58%
- **Second Approach**: *F1 Score Accuracy* = 7.92%
- **Hybrid Approach**: *F1 Score Accuracy* = 11.62%

Chapter 5
Conclusions and Discussions

5.1 Conclusions

In this project, I integrate insights from diverse studies to develop a recommendation system grounded in comprehensive research. The system is designed to recommend the most suitable courses for learners who have previously taken at least one course. It offers personalized recommendations, conducts thorough user data analysis to enhance accuracy, and contributes to improving the overall learning experience. However, it's important to note that the system's applicability is currently limited to online learning platforms that adhere to specific data formats required for implementation.

1. **First Approach Analysis:** The first approach has relatively low accuracy in both Hit Rate and F1 Score, indicating that the model's performance in recommending relevant courses to users is not very effective. A Hit Rate of *14.69%* means that only about *14.69%* of the recommended courses match with the actual courses users take. Similarly, an F1 Score of *2.58%* suggests poor precision and recall, further highlighting the limitations of this approach in accurately predicting user preferences.
2. **Second Approach Analysis:** The second approach shows improved accuracy compared to the first approach, with a Hit Rate of *48.24%* and an F1 Score of *7.92%*. This indicates that the model in the second approach performs better in recommending courses that users are likely to take. However, the accuracy is still relatively moderate, suggesting room for further improvement.
3. **Hybrid Approach Analysis:** The hybrid approach demonstrates the highest accuracy among the three approaches, with a Hit Rate of *69.18%* and an F1 Score of *11.62%*. This indicates that combining multiple recommendation techniques, such as content-based filtering and collaborative filtering (as implied by the hybrid approach), leads to more accurate and personalized recommendations for users. The higher Hit Rate and F1 Score suggest that the hybrid approach has a better understanding of user preferences and is more successful in recommending relevant courses.

5.2 Challenges

The challenges encountered in the project are diverse, with the most significant being the process of gathering and comprehending studies related to the recommender system. We conducted a deep investigation into the evidence provided by authors to ensure reliability. Additionally, our main obstacle lies in Machine Learning Operations (MLOps) concerning deployment, as we are unfamiliar with the process.

5.3 Suggestions and further improvements

A suggestion for improving the project is to expand its compatibility with a broader range of online learning platforms. This would reduce the concerns developers have about data consistency between their datasets and the recommender system, making it more accessible and user-friendly across various platforms.

References

[1] Y. Liu, H. Yang, G. Sun, and S. Bin. Collaborative filtering recommendation algorithm based on multi-relationship social network. *Ingenierie des Systemes d'Information*, 25(3):359–364, 2020.

[2] Dominic Wong. A critical literature review on e-learning limitations. *Journal for the Advancement of Science and Arts*, 2(1):55–62, 2007.

[3] Q. Zhang, Y. Liu, L. Liu, S. Lu, Y. Feng, and X. Yu. Location identification and personalized recommendation of tourist attractions based on image processing. *Traitement du Signal*, 38(1):197–205, 2021.

[4] China Internet Network Information Center (CNNIC). The 46th statistical report on the current situation of internet development in china. Technical report, China Internet Network Information Center (CNNIC), Beijing, China, 2020.

[5] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. A scalable, accurate hybrid recommender system. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 94–98, 2010.

[6] S. Bhaskaran and B. Santhi. An efficient personalized trust based hybrid recommendation (tbhr) strategy for e-learning system in cloud computing. *Cluster Computing*, 22(1):1137–1149, 2019.

[7] H. Tan, J. Guo, and Y. Li. E-learning recommendation system. In *Proceedings of the International Conference on Computer Science and Software Engineering*, pages 430–433, Wuhan, China, December 2008.

[8] L. Sharma. A survey of recommendation system. *Int. J. Eng. Trends Technol.*, 4:1989–1992, 2013.

[9] S. Shishenhchi, S.Y. Banihashem, and N.A.M. Zin. A proposed semantic recommendation system for e-learning: A rule and ontology based e-learning recommendation system. In *Proceedings of the International Symposium on Information Technology*, pages 1–5, Kuala Lumpur, Malaysia, June 2010.

[10] S.S. Khanal, P.W.C. Prasad, A. Alsadoon, and A. Maag. A systematic review: Machine learning based recommendation systems for e-learning. *Educ. Inf. Technol.*, 25:2635–2664, 2020.

[11] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*,

SIGIR '16, page 305–314, New York, NY, USA, 2016. Association for Computing Machinery.

[12] Hualong Ma, Xiande Wang, Jianfeng Hou, and Yunjun Lu. Course recommendation based on semantic similarity analysis. In *2017 3rd IEEE International Conference on Control Science and Systems Engineering (ICCSSE)*, pages 638–641, 2017.

[13] Rajae Zriaa and Said Amali. A comparative study between k-nearest neighbors and k-means clustering techniques of collaborative filtering in e-learning environment. In Mohamed Ben Ahmed, İsmail Rakıp Karaş, Domingos Santos, Olga Sergeyeva, and Anouar Abdelhakim Boudhir, editors, *Innovations in Smart Cities Applications Volume 4*, pages 268–282, Cham, 2021. Springer International Publishing.

[14] J. Shen, T. Zhou, and L. Chen. Collaborative filtering-based recommendation system for big data. *International Journal of Computational Science and Engineering*, 21(2):219–225, 2020.

[15] P. Thakkar, K. Varma, V. Ukani, S. Mankad, and S. Tanwar. Combining user-based and item-based collaborative filtering using machine learning. In *Information and Communication Technology for Intelligent Systems*, pages 173-180. Springer, Singapore, 2019.

[16] J. Xiao, M. Wang, B. Jiang, and J. Li. A personalized recommendation system with combinational algorithm

[17] Freddy Lécué. Combining collaborative filtering and semantic content-based approaches to recommend web services. In *2010 IEEE Fourth International Conference on Semantic Computing*, pages 200–205, 2010.

[18] J. Lin, H. Pu, Y. Li, and J. Lian. Intelligent recommendation system for course selection in smart education. *Procedia Computer Science*, 129:449-453, 2018.

[19] AvinashR.Sonule, Mukesh Kalla, Amit Jain, and Deepak Singh Chouhan. Unsw-nb15 dataset and machine learning based intrusion detection systems. *International Journal of Engineering and Advanced Technology*, 2020.

[20] Nitish K. Panigrahy, Jian Li, and Don Towsley. Hit rate vs. hit probability based cache utility maximization. *ACM SIGMETRICS Performance Evaluation Review*, 45(2):21–23, 2017.

[21] Reda Yacouby and Dustin Axman. Probabilistic Extension of Precision, Recall, and F1 Score for More Thorough Evaluation of Classification Models. In Steffen Eger, Yang Gao, Maxime Peyrard, Wei Zhao, and Eduard Hovy, editors, *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 79–91, Online,

November 2020. Association for Computational Linguistics.

[22] G. Xu, A. Zhai, J. Wang, Z. Zhang, and X. Li. Cross-media semantic matching based on sparse representation. *Technical Gazette*, 26(6):170-1713, 2019.

[23] J. Aguilar, P. Valdiviezo-Diaz, and G. Riofrio. A general framework for intelligent recommender systems. *Applied Computing and Informatics*, 13(2):147–160, 2017.

[24] Y. Tian, B. Zheng, Y. Wang, Y. Zhang, and Q. Wu. College library personalized recommendation system based on hybrid recommendation algorithm. *Procedia CIRP*, 83:490–494, 2019.

[25] J. Gope and S. Jain. A survey on solving cold start problem in recommender systems. In *Proceedings of the 2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 133-138, Greater Noida, India, 2017.

Appendix A
Technical Implementation Details

This appendix provides additional technical details and information related to the implementation of the recommendation system aimed at facilitating course recommendations tailored to individual user interests on e-Learning platforms.

A.1 Data Collection and Preprocessing

We collected user interaction data from the e-Learning platform, including user ratings, course enrollments, and browsing history. The data were preprocessed to remove duplicates, handle missing values, and ensure consistency.

A.2 Content-Based Filtering

In the content-based filtering approach, we utilized the TF-IDF (Term Frequency-Inverse Document Frequency) technique to extract relevant features from course descriptions and user preferences. We then applied feature weighting and cosine similarity to recommend courses based on similarity to the user's preferences.

A.3 Collaborative Filtering

For collaborative filtering, we employed the K-Nearest Neighbors (KNN) algorithm to identify similar users or courses based on their interaction patterns. We calculated similarities between users or courses and generated recommendations by considering the preferences of similar users.

A.4 Hybrid Approach

The hybrid approach combines content-based and collaborative filtering techniques to leverage the strengths of both methods. We integrated the recommendations generated from both approaches using a weighted or ensemble method to provide more personalized and accurate course suggestions.

A.5 Model Evaluation

We evaluated the performance of the recommendation system using standard metrics such as accuracy, precision, recall, and F1-score. Additionally, we conducted A/B testing or user studies to assess user satisfaction and the effectiveness of the recommendations in improving the user experience and content discovery on the e-Learning platform.

Appendix B
Model Training and Evaluation

This appendix outlines the key aspects of model training, evaluation metrics, and challenges addressed during the development of the recommendation system.

B.1 Model Training

The recommendation system model was trained using various machine learning algorithms, including content-based filtering, collaborative filtering, and hybrid approaches. Training involved feature extraction, algorithm selection, hyperparameter tuning, and data preparation.

B.2 Evaluation Metrics

Performance evaluation utilized metrics such as accuracy, precision, recall, F1-score, and hit rate to assess the system's effectiveness in recommending relevant courses based on user preferences and interactions.

B.3 Challenges and Solutions

Challenges encountered included data sparsity [24], cold start problem [25], scalability issues, and algorithm complexity. Solutions included data augmentation, cold start handling techniques, scalability improvements, and algorithmic enhancements.

Biographical Sketch

Name: Newin Yamaguchi
Affiliation: Chiang Mai University

Name: Patcharaporn Satantaipop
Affiliation: Chiang Mai University

Newin and Patcharaporn are students at the Department of Computer Engineering, Chiang Mai University, specializing in machine learning and natural language processing. We are currently in the fourth-year of Bachelor's Degree. With a passion for data science, we have dedicated our career to exploring innovative solutions to motivate students to become more addicted to self-learning.

Our current project focuses on the development of a Recommendation System for e-Learning platforms. Recognizing the challenges associated with online learning. We aim to investigate diverse models to facilitate course recommendations tailored to individual user interests. By leveraging a hybrid approach that combines content-based and collaborative filtering techniques, we aim to enhance the content discovery process and improve user satisfaction in online education.

Overall, Newin and Patcharaporn are dedicated and driven researchers committed to making meaningful contributions to the field of Computer Engineering. Our recommendation system project for e-Learning platforms represents an ongoing commitment to improving online education and enhancing the learning experience for students worldwide.