Content Based Recommender System For Online Education

By
Farhan Bashir
Registration No. 00000320390

Fall-2019-MS-Systems Engineering

Supervisor
Dr Mehak Rafiq

Department of Engineering

A thesis submitted in partial fulfilment of the Master of Systems Engineering degree requirements.

In

School of Interdisciplinary Engineering and Sciences

National University of Sciences and Technology

Islamabad, Pakistan.

March, 2023

Thesis Acceptance Certificate

Certified that the final copy of the M.S. thesis entitled "Content Based Recommender System For Online Education", written by Farhan Bashir (Registration No 00000320390), has been reviewed by the undersigned, found to be complete in all respects and accordance with NUST Statutes/Regulations, is free of plagiarism, errors, and mistakes, and is accepted as partial fulfilment for the award of an M.S. degree. It is also certified that necessary changes suggested by the scholar's GEC members have been incorporated into the thesis.

Signature: _____

Name of Supervisor: Dr Mehak Rafiq

Date: _____

Signature (HoD): _____

Date: _____

Signature (Dean/Principal): _____

Date: _____

Approval

It is certified that the contents and form of the thesis entitled "Content Based Recommender System For Online Education" submitted by Farhan Bashir have been determined to be sufficient for the degree requirement.

Advisor: Dr Mehak Rafiq

Signature: _____

Date: _____

Committee Member 1: Dr Muhammad Tariq Saeed

Signature: _____

Date: _____

Committee Member 2: Dr Shahzad Rasool

Signature: _____

Date: _____

Dedication

I dedicate this research project report to ALLAH (SWT), my parents, my supervisor Dr Mehak Rafiq, teachers, siblings, and friends (Syeda Yumna Nasir & Haseeb Sultan) in particular for their endless support, love, encouragement and invaluable trust in me. All these people are the precious gift of ALLAH (SWT).

Certificate of Originality

I hereby declare that the results presented in this research work titled as "Content Based Recommender System for Online Education" are generated by myself. Moreover, none of its contents are plagiarised nor set forth for any kind of evaluation or higher education purposes. I have acknowledged/referenced all the literary content used for support in this research work.

_____

Farhan Bashir

00000320390

Acknowledgement

I am pleased to express my sincere gratitude to my supervisor, Dr Mehak Rafiq, who gave me the freedom to explore on my own and guided me whenever I was following the wrong steps. She was always an immense source of knowledge and I am very thankful for her continuous support during my research work. Concisely and precisely, I feel deeply indebted to the company of my beloved Father, Mother and siblings who were encouraging, supportive and heart-warming all the time.

Besides my advisor, I would like to pay sincere thanks to, my GEC members Dr Muhammad Tariq Saeed and Dr Shahzad Rasool and my fellows Ameer Ghaznavi, Mehar Masood, Noor-us-Subha, Mohsin Ahmad Khan, Haseeb Khan, Rumsha Fatima, Huzaifa Arshad, Madahiah Bint e Masood, Tayaba Alvi and Farhana for being helping and supportive throughout my project.

Farhan Bashir

Table of Contents

Table of Contents 6
List of Figures 8
List of Equations 10
List of Tables 11
List of Abbreviation 12
Abstract 14
Chapter 1: Introduction 2
1.1. Background 2
1.2. Recommender Systems 4
1.2.1. Stages of Recommender System 4
1.2.2. Types of Recommendation Techniques 7
1.2.2.1 Content-based recommendation 8
1.2.2.2 Collaborative recommendation 9
1.2.2.3 Demographic recommendation 13
1.2.2.4 The utility-based recommendation 13
1.2.2.5 Knowledge-Based Recommendation, 14
1.2.2.6 Hybrid Recommendation 15
1.4. Research Question 17
1.5. Objectives 18
1.6. Thesis Structure 18
Chapter 2: Literature Review 19
2.1. Knowledge Tracing Problem 19
2.2. Modern Education Recommender Systems (ERS) 21
2.3. Recommendation Techniques in Modern ERS 23
2.3.1. Collaborative Filtering 23
2.3.2. Content-Based Filtering 24
2.3.3. Hybrid Filtering 25
2.4. Input Features Incorporated in Modern ERS 25
2.5. Display of Recommendations by ERS 30
2.6. Literature Gap 31
Chapter 3: Methodology 33
3.1. Research Methodology Workflow 33
3.2. Data Collection 34
3.2.1. Properties of EdNet Dataset 34
3.3. Exploratory Data Analysis 35
3.3.1. Python Implementations 35
3.3.2. EdNet Sub-Datasets 36
3.3.2.1. KT1 Sub-Dataset 36
3.3.2.2. KT2 Sub-Dataset 37

3.3.2.3. KT3 Sub-Dataset 38
3.3.2.4. KT4 Sub-Dataset 39
3.3.2.5. Content Sub-Dataset 40
3.4. Feature Selection 41
3.5. Feature Engineering 44
3.6. Groupization Scheme for Recommender System 44
3.7. Recommender System Design 45
3.7.1. Database 45
3.7.2. Software Framework 46
3.7.3. User-Interface Backend 49
Chapter 4: Results & Discussion 50
4.1. Exploratory Data Analysis Results 50
4.2. Feature Engineering Results 50
4.2.1. All Bundles Performance Distribution 51
4.2.2. Bundle Performance of 60% & above 52
4.2.3. Accuracy Distribution on Question Count Level 52
4.2.3.1. Accuracy Distribution over All Bundles - All Questions 53
4.2.3.2. Accuracy Distribution over All Bundles - Question Count Three & Four 53
4.3. Visual Interpretation of Content-Based Recommender System 54
Chapter 5: Conclusion & Future Perspective 58
5.1. Limitations 59
5.2. Future Perspectives 59
References 61
Appendix 65
Appendix A 65

List of Figures

Figure 1: Schematic representation of ontology based e-learning platforms……...3

Figure 2: Schematic representation of different stages of recommender systems……..6

Figure 3: Techniques defining different types of recommender systems……10

Figure 4: An illustration on how knowledge tracing might work in an intelligent tutoring system (ITS).……...17

Figure 5: An overview of traditional knowledge tracing strategies.……..…..18

Figure 6: Collaborative filtering using cosine similarity……..……..19

Figure 7: Research methodology workflow…….……..…….22

Figure 8: KTI Subset of EdNet.……23

Figure 9: KT2 Subset of EdNet.……..…….….24

Figure 10: KT3 Subset of EdNet.……...25

Figure 11: KT4 Subset of EdNet.……..30

Figure 12: Content Subset of EdNet.……..31

Figure 13: Common users in all subsets of EdNet.……..31

Figure 14: Python implementation of sorting bundle IDs…….,,,,,,,,,,.….32

Figure 15: Python implementation of calculating pattern count between bundles…….33

Figure 16: Maximum pattern count.…….,,,,,,,,,,..35

Figure 17: Recommender system database..……...45

Figure 18: Structure of an ASP.NET MVC application.……49

Figure 19: Accuracy distribution over all bundle counts……....50

Figure 20: Accuracy distribution over bundle counts with 60% & above correctness.……..52

Figure 21: Accuracy distribution (x-axis) over all bundles with the varying count of questions……..…53

Figure 22: Bimodal histogram at three & four questions within several bundles (y-axis) & accuracies % (x-axis).....54

Figure 21 to 26: Content based recommender system GUI…….....55

List of Equations

Equation 1: Bayesian Knowledge Tracing (BKT) Probability Function…….13

Equation 2: Cosine Similarity Vector Form……. 16

Equation 3: Logistic form of the IRT predictions…….17

List of Tables

Table 1: Types of recommendation techniques used in technology-enhanced learning (TEL).....12

Table 2: Comparison of Modern Education Recommender Systems (ERS)……23

Table 3: Input features for content-based ERS.……35

Table 4: Data exploration results.…….44

Table 5: New features related to Bundles
……..……45

List of Abbreviation

A.I.	Artificial Intelligence
LMS	Learning Management System
N.N.	Neural Network
SVM	Support Vector Machine
IDF	Inverse Document Frequency
ORM	Object-Relational Mapper
MVC	Model View Controller
E.F.	Entity Framework
SQL	Structure Query Language
HTML	HyperText Mark-up Language
CSS	Cascading Style Sheet
J.S.	JavaScript
AJAX	Asynchronous JavaScript and XML
KT	Knowledge Tracing
MOOCs	Massive Open Online Courses

ITS	Intelligent Tutoring System
BKT	Bayesian Knowledge Tracing
FAM	Factor Analysing Model
IRT	Item Response Theory
K.C.	Knowledge Component
TEL	Technology Enhanced Learning

Abstract

Recommendation systems are now commonly used in e-learning to suggest resources and learning materials to learners and improve teaching and learning quality. To this end, predicting learners' needs and recommending e-learning resources in e-learning systems has become a research focus. One way to address the need for predicting user needs and improving the usability of e-learning systems is to recommend pages or resources to learners that are related to their interests at a given time. The purpose of this study is to evaluate the effectiveness of bundle correctness percentages and frequencies in a web-based e-learning system for analysing learner interests and recommending e-learning resources. The primary data source for this research is EdNet. The primary distinguishing factor of this study is the organization of some of the dataset's questions into bundles (groups of questions that must be answered collectively). This dataset has about 95,293,926 interactions, 13,169 questions, 784,309 students, and 188 knowledge components. The approach involves clustering learner sessions based on the accuracy of learning relevant bundles in a pattern specified by our content-based recommender tool. The outcome of this study is a graphical user interface. Users will provide the bundle they want to study, and our system will recommend the prerequisite bundles required to understand the intended bundle. Out of a total of 9,534 bundles, 8,935 are considered suitable learning objects, as 60% of students can respond to the questions correctly. An efficient pattern for studying these bundles is recommended as a study plan by a content-based education recommender tool, which can further enhance user performance.

Chapter 1: Introduction

1.1. Background

The ultimate goal of education is always in transition to serve the needs of society. This never-ending debate on the purpose of education has resulted in a broad description of the outcomes of education, from making an individual creative, responsible and irreproachable to capable, skilled and efficient (Sloan, 2012). However, learning is always the main goal of education, whether it is for the workers of a particular profession or the development of social skills for the benefit of society. In order to aid this core goal, researchers have done several studies to develop useful pedagogical (teaching) techniques.

The definition of successful pedagogy varies, however active learning has been identified by academics as a crucial component of these techniques (Bonwell & Eison, 1991). While there are several definitions of active learning, Bonwell & Eison (1991) described it as pupils participating in an activity intended to help them comprehend a new topic. Additionally, using technology as a tool can encourage active learning. In order to encourage active learning, creative educational strategies have been developed (Kadiyala & Crynes, 2000). Active learning takes time, which makes it harder to finish the course material in the allotted time (Cooperstein & Weidinger, 2004). As a result, technology may be used to assure active learning without the risk of not covering all the material in the allotted time. This complexity in learning resource recommendations can be reduced by employing knowledge structures such as ontologies to personalise the learner profile should reflect the requirements and traits of the student. Figure 1 shows schematic representation of ontology-based e-learning platforms incorporated with the elements of traditional educational systems. Ontology is one of the methods used by e-learning technologies to offer personalised recommendations. This approach is used to model students and learning resources to obtain details and produce additional content for learners. For example, in the recommendation process, Verbert et al. (2012) underline the necessity of including extra information on the student, teacher, and their context.

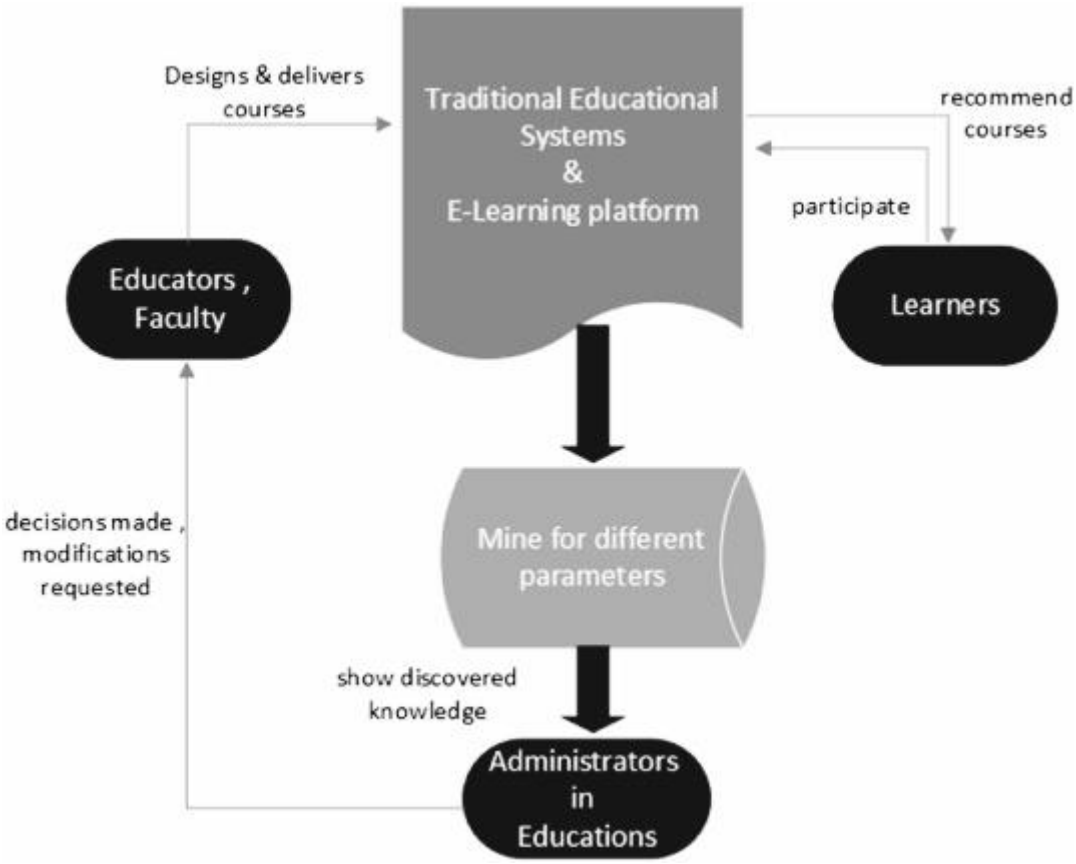

Figure 1: Schematic representation of ontology-based e-learning platforms incorporated with the elements of traditional educational systems. Ontology is one of the methods used by e-learning technologies to offer personalised recommendations. This approach is used to model students and learning resources to obtain details and produce additional content for learners.

Technology-enhanced learning aims to increase learning for everyone by designing, developing, and testing socio-technical innovations that will support and improve learning practices for individuals and organisations. This application category includes the technologies that underpin all forms of instruction and learning. For example, information retrieval, which is the process of seeking relevant educational resources to help teachers or students, has increased the use of recommender systems.

This should more or less have been anticipated since one of the usual issues with TEL has been the most easier discovery of learning resources. In various TEL environments, for instance content for digital learning is routinely produced, arranged, and released, such as:

1. Learning Management Systems and Course Management Systems like Blackboard and Moodle.
2. OpenCourseWare sites like MIT OCW7 or OpenLearn.
3. Learning Object Repositories like Learning Resource Exchange.

There are many ways for users to use this variety of digital learning resources. All TEL system users might potentially benefit from the appealing services that assist them in selecting appropriate learning resources from the overwhelming range of available options. The idea of recommender systems consequently became quite attractive for TEL research.

1.2. Recommender Systems

Depending on the sort of system being used, recommender systems are a type of customised information filtering technology that are used to either identify the top N things that will be of interest to a given user or forecast if a certain user would enjoy a particular item (Deshpande & Karypis, 2004).

In order to understand the tastes of various users and anticipate or propose products that correspond to their needs, recommender systems identify patterns in many datasets. The word "item" in this context refers to any course, educational component, book, service, application, or product. To accomplish their objectives, recommender systems primarily employ machine learning and data mining approaches (Ricci, Rokach, Shapira & Kantor, 2010). Massive open online courses (MOOCs) are using these systems more and more frequently for educational objectives. These systems are utilised extensively in e-commerce and by businesses to increase their sales and audience. The foundation of data filtering is the recommender system, according to Sloan (2012). Building a recommender system's goal is to give users all the information they need for customised learning and interests based on their social interactions.

1.2.1. Stages of Recommender System

A recommender system's main objective is to make suggestions to users. There are many other types of suggestions, including those for movies, news, and music tracks. The recommendation system offers suggestions while maintaining user interest and taking contextual information into account. In addition to making important suggestions, recommender systems effectively handle the information overload problem. Given the wealth of information accessible, it is imperative to eliminate unnecessary information (Kadiyala & Crynes, 2000). The majority of recommender systems now in use solely focus on recommending the data that is most closely connected to the user's search and contextual information. For instance, those systems do not take time or location into account. However, contextual data and personalization

features are also a part of the often used modern recommendation systems in research (Dakhel & Mahdavi, 2013).

The role of a recommender system, whether it is collaborative filtering or a deep learning model, is to rate or score the potential user interest in a set of items. However, in most situations in the actual world, these scores are insufficient to provide recommendations. Moreover, scoring every item for every user is impossible or computationally feasible. Hence, the idea of 2-stage & 3-stage recommender systems (Oldridge, 2022). Figure 2 gives a schematic representation of different stages of recommender systems from simple to complex frameworks. 1-stage recommender systems score every item-user data. 2-stage recommender system scores based on top candidate's generation data. 3-stage recommender systems retrieve, filter and then score the user-item interaction data to make recommendations. These four stages of Retrieval, Filtering, Scoring, and Ordering Policy make up a design pattern for high-efficiency 4-stage recommender systems.

1. **2-stage recommender system:** This stage is commonly referred to as the candidate *Retrieval* stage and is used to choose a reasonably relevant set of items that the user will eventually engage with. Retrieval models take many forms, including matrix factorisation, two-tower, linear models, approximate nearest neighbour, and graph traversal.
2. **3-stage recommender system:** The *Retrieval* stage is usually followed by *filtering*. In some circumstances, these are straightforward exclusion queries. However, they can also be more complicated, like with Bloom filters, which are used to eliminate things that users have already interacted with.
3. **4-stage recommender system:** To match the model's output with business goals or limitations, an explicit *Ordering* (policy or business logic) stage is included in 4-stage recommender systems. A design pattern comprising the four stages of retrieval, filtering, scoring, and ordering covers almost all recommender systems deployed at the corporate scale.

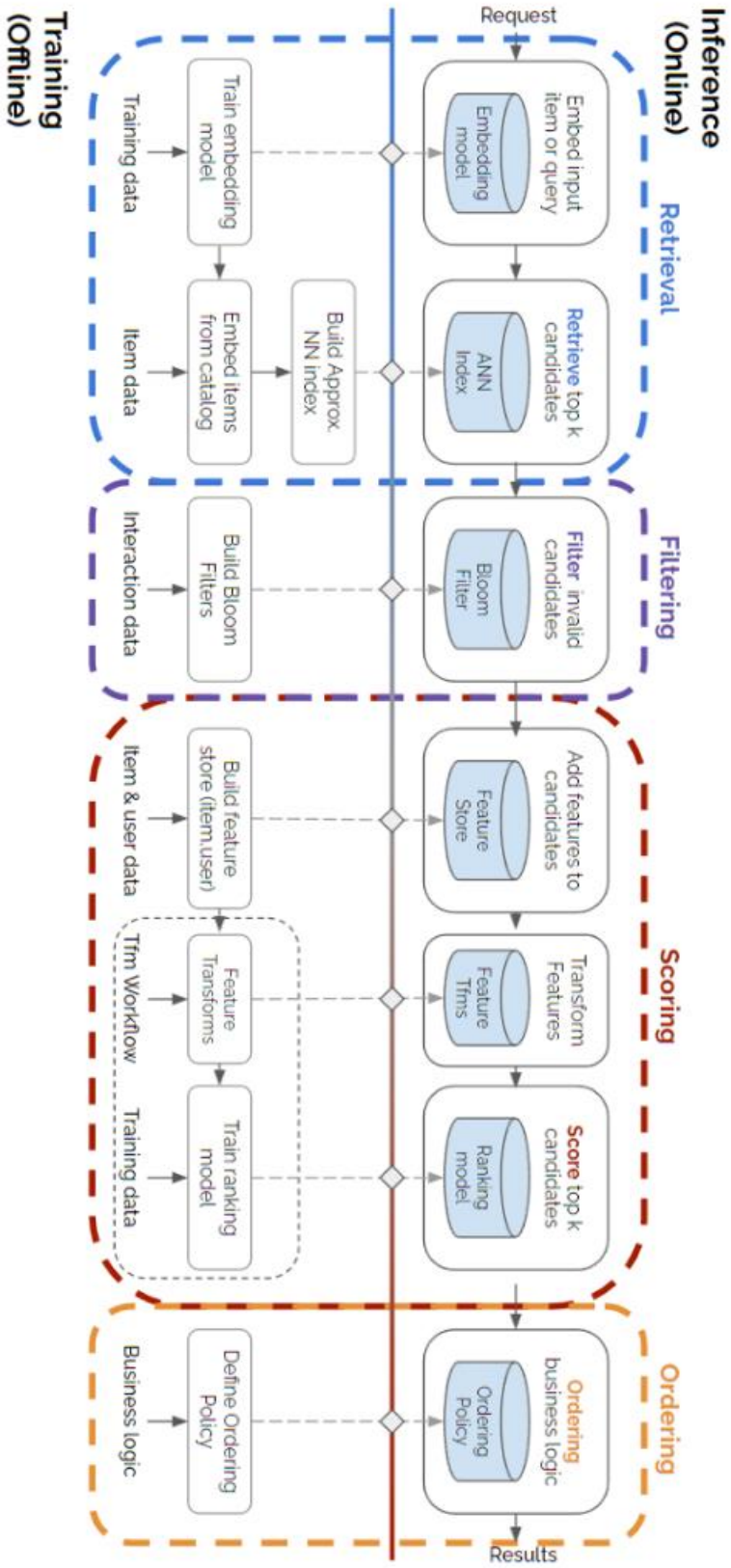

Figure 2: Schematic representation of different stages of recommender systems from simple to complex frameworks. 1-stage recommender systems score every item-user data. 2-stage recommender system scores based on top candidate's generation data. 3-stage recommender systems retrieve, filter and then score the user-item

interaction data to make recommendations. These four stages of Retrieval, Filtering, Scoring, and Ordering Policy make up a design pattern for high-efficiency 4-stage recommender systems.

1.2.2. Types of Recommendation Techniques

Only user and item information is insufficient for many applications, such as the holiday suggestion system and the movie recommendation system, and contextual information is crucial. For instance, a travel recommendation system suggests appropriate vacation packages based on the forecast. Additionally, Recommender frameworks provide clients tailored advice that may be of interest to them. These tools support people in managing informational overload and reducing complexity while looking for crucial information. To achieve personalization, three essential elements are needed.

- Database to save representations of the objects that are available.
- Using profiles to predict clients' preferences.
- Customised plans & offerings according to client personas

The most popular methods at the moment are content-based, collaborative filtering, and hybrid methods. Utilising user preferences, evaluations, and recommendation systems, practical information retrieval is achievable. Semantic-based recommendation systems are also widely adopted. The extraction phase of the recommendation system is the most important phase. The example's suggestion order makes it possible for teachers to employ examples that provide clear solutions to certain problems. Additionally, it improves the advice provided by instructors. It produces respectable results, improves learning abilities, and provides a springboard for encouraging frameworks for improvement and recommendations. We took into account various proposal features, evaluations of the kinds associated with the suggested programmes, and seeing propensities while making our assessment for the factual examination. All of these elements work together to produce suggestions that meet the needs of the customer.. Figure 3 details major techniques defining different types of recommender systems. Recommender systems rely on a variety of inputs (filtering techniques), including the most realistic, high-quality explicit feedback (contextual filtering), which includes direct user input about their interest in the item, or implicit feedback (collaborative filtering), which is derived by indirectly inferring user preferences by observing user behaviour. Combining explicit and implicit feedback can also result in hybrid feedback (hybrid filtering techniques). To provide effectively

relevant and accurate suggestions, information availability and personalization must be increased.

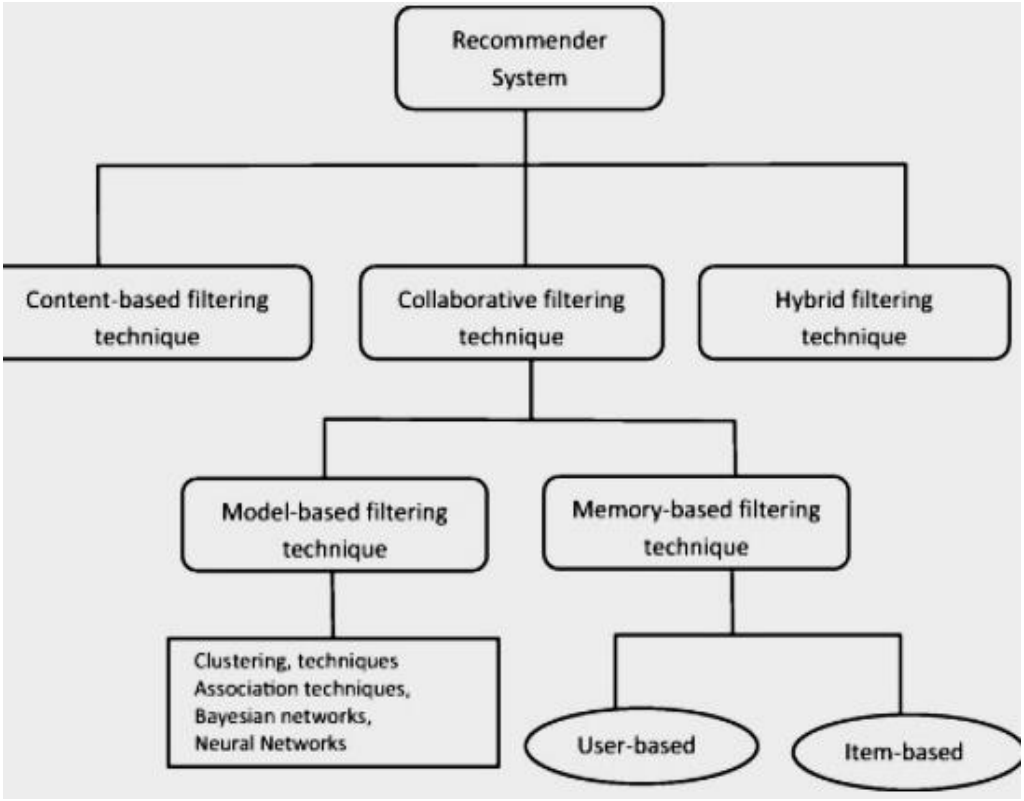

Figure 3: Techniques defining different types of recommender systems. Recommender systems rely on a variety of inputs (filtering techniques), including the most realistic, high-quality explicit feedback (contextual filtering), which includes direct user input about their interest in the item, or implicit feedback (collaborative filtering), which is derived by indirectly inferring user preferences by observing user behaviour. Combining explicit and implicit feedback can also result in hybrid feedback (hybrid filtering techniques).

1.2.2.1 Content-based recommendation

A sort of recommendation system called content-based recommendation promotes products to customers based on the features of the products and the user's previous preferences. The system analyses the items' attributes, such as genre, year of release, actors, directors, etc., and the user's past interactions with similar items to generate recommendations.

An example to better illustrate the concept is when a user has watched and liked several romantic comedies, such as "When Harry Met Sally," "Notting Hill," and "500 Days of Summer." A content-based recommendation system would analyse these movies and identify common features, such as the genre, lead actors, and director, and then suggest other romantic comedies that share these features.

The advantage of content-based recommendation is that it's based on explicit information about the items and the user, which means it can make highly personalised recommendations. However, the system is limited by the quality of the data available and the user's ability to provide relevant information. If the user has limited interactions with the system or provides insufficient information, the recommendations may not be accurate (Jannach, 2010).

Overall, content-based recommendation is a useful method for generating personalised recommendations and can be combined with other recommendation methods for improved results.

1.2.2.2 Collaborative recommendation

A sort of recommendation system called collaborative recommendation proposes products to users based on their preferences and interactions with other users who share their interests. Based on their prior interactions with things, the algorithm recognizes people who are similar to them and then proposes items that are well-liked by these users. For example, consider a book recommendation system. If a user has read and liked several mystery novels, the system would identify other users who have read and liked similar mystery novels. The system would then suggest books that are popular among these similar users.

One popular example of a collaborative recommendation system is the recommendation engine used by Amazon. The system suggests products to users based on their past purchases and the purchases of similar users. The system also takes into account items that users have viewed, added to their cart, or placed on their wishlist to generate recommendations.

Another example is the recommendation system used by Netflix. The system suggests movies and TV shows to users based on their past viewing history and the viewing history of similar users. The system also uses information about the user's ratings and reviews to further personalise the recommendations.

The advantage of collaborative recommendation is that it can make highly personalised recommendations based on the preferences and interactions of a large number of users. The system can also identify trends and popular items that a user might not be aware of. However, the system may not be effective for new users with limited interactions or users with unique preferences that are not well represented in the system.

Overall, collaborative recommendation is a powerful method for generating recommendations and can be an effective complement to content-based recommendation and other recommendation methods.

Table 1 lists down different types of recommendation techniques used in technology-enhanced learning (TEL). Collaborative techniques include reasoning that may involve users, items or demographics-based recommendations. Content-based techniques are applied to either case-based or attribute-based recommendation strategies.An extended list of collaborative and contextual filtering techniques comprehensively compares applied use cases in TEL and the advantages and disadvantages of these recommendation techniques (Drachsler et al., 2008).

Name	**Short description**	**Advantages**	**Disadvantages**	**Usefulness for TEL**
Collaborative filtering (CF) techniques				
1.User-based CF	Users that rated the same item similarly probably have the same taste. Based on this assumption, this technique recommends unseen items already rated by similar users.	-No content analysis -Domain independent -Quality improves over time -Bottom-up approach - Serendipity	-New user problem -New item problem - Popular taste -Scalability -Sparsity - Cold-start problem	- Benefits from experience - Allocates learners to groups (based on similar ratings)

2.Item-based CF	Focus on items, if items rated similarly are probably similar. It recommends items with highest correlation (based on ratings to the items).	-No content analysis -Domain independent -Quality improves over time -Bottom-up approach -Serendipity	-New item problem - Popular taste -Sparsity - Cold-start problem	- Benefits from experience
3.Stereotypes or demographics CF	Users with similar attributes are matched, then recommends items that are preferred by similar users (based on user data instead of ratings).	-No cold-start problem -Domain independent -Serendipity	-Obtaining information -Insufficient information -Only popular taste -Obtaining metadata information - Maintenance ontology	- Allocates learners to groups - Benefits from experience - Recommendation from the beginning of the RS
Content-based (CB) techniques				

4. Case-based reasoning	Assumes that if a user likes a certain item, s/he will probably also like similar items. Recommends new but similar items.	-No content analysis -Domain Independent - Quality improves over time	-New user problem -Overspecialisation -Sparsity - Cold-start problem	-Keeps learner informed about learning goal -Useful for hybrid RS
5. Attribute-based techniques	Recommends items based on the matching of their attributes to the user profile. Attributes could be weighted for their importance to the user.	-No cold-start problem -No new user / new item problem - Sensitive to changes of preferences -Can include non-item related features -Can map from user needs to items	- Does not learn - Only works with categories - Ontology modelling and maintenance is required -Overspecialisation	- Useful for hybrid RS - Recommendation from the beginning

Table 1: Types of recommendation techniques used in technology-enhanced learning (TEL). Collaborative techniques include reasoning that may involve users, items or demographics-based recommendations. Content-based techniques are applied to either case-based or attribute-based recommendation strategies.

In addition, various alternative recommender system types have been put forth (Burke, 2007):

1.2.2.3 Demographic recommendation

A sort of recommendation system known as a demographic recommendation system makes product recommendations to consumers based on their age, gender,

region, and income. The system uses this information to make assumptions about a user's interests and preferences and then suggests items that are popular among similar demographic groups.

For example, consider a clothing recommendation system. If a user is a female in her 30s living in New York City, the system might suggest clothing items that are popular among women in their 30s living in New York City. The system might also take into account the user's income and suggest items that are affordable for the user.

Another illustration is a system that recommends places to users based on their demographic data. If a user is a young couple in their 20s, the system might suggest popular destinations for young couples, such as tropical beaches and adventurous destinations.

The advantage of demographic recommendation systems is that they can provide broad recommendations based on general demographic trends and popular items. The system can also make recommendations that are relevant to the user's stage in life and life circumstances. However, the system may not be effective for users with unique preferences that are not well represented by their demographic group.

Overall, demographic recommendation systems are a useful method for generating recommendations and can be an effective complement to content-based and collaborative recommendation systems which groups users based on the characteristics of their profiles and generates suggestions for them. Demographic recommendations refer to the practice of recommending products, services, or content to a user based on their demographic information such as age, gender, income, education level, location, etc. This approach aims to target specific groups of users more effectively by considering their specific characteristics, preferences, and interests. By utilising demographic data, organisations can personalise their offerings and create a more engaging and relevant user experience.

1.2.2.4 The utility-based recommendation

A sort of recommendation system called utility-based recommendation promotes products to customers based on their anticipated usefulness or worth. Based on the user's prior encounters with comparable goods, the system estimates the utility of each item and then recommends the items with the greatest estimated value.

For example, consider a music recommendation system. If a user has listened to and liked several jazz songs, the system would estimate the utility of each jazz song

for the user based on their past interactions. The system would then suggest the jazz songs with the highest estimated utility for the user.

Another example is a movie recommendation system that suggests movies based on the estimated utility of each movie for the user. The system calculates the utility of each movie based on the user's past movie preferences and ratings, as well as the ratings and preferences of similar users.

The advantage of utility-based recommendation is that it can make highly personalised recommendations based on the user's past interactions and preferences. The system can also take into account the user's changing preferences over time and adjust the recommendations accordingly. However, the system may not be effective for new users with limited interactions or for users with unique preferences that are not well represented in the system.

Overall, utility-based recommendation is a powerful method for generating recommendations and can be an effective complement to content-based, collaborative, and demographic recommendation systems.

1.2.2.5 Knowledge-Based Recommendation,

Knowledge-based recommendation systems are a type of recommendation system that suggests items to users based on the knowledge that the system has about the user and the items. The system uses this knowledge to make recommendations based on the user's stated preferences, interests, and background information.

For example, consider a book recommendation system. If a user is a history enthusiast, the system might suggest books on history based on the user's stated interests and background information. The system might also take into account the user's reading level and suggest books that are appropriate for the user's level.

Another example is a restaurant recommendation system that suggests restaurants based on the knowledge that the system has about the user and the restaurants. The system might suggest restaurants based on the user's dietary restrictions, preferred cuisine, and location.

The advantage of knowledge-based recommendation systems is that they can provide highly personalised recommendations based on the user's specific preferences and needs. The system can also make recommendations that are relevant to the user's unique circumstances and provide a more personalised experience. However, the system may not be effective for new users with limited background information or for

users with unique preferences that are not well represented in the system's knowledge base.

Overall, knowledge-based recommendation systems are a useful method for generating recommendations and can be an effective complement to content-based, collaborative, demographic, and utility-based recommendation systems.

1.2.2.6 Hybrid Recommendation

Hybrid recommendation systems are a type of recommendation system that combines multiple recommendation techniques to provide more accurate and personalised recommendations to users. To provide suggestions, these systems can combine approaches from content-based, collaborative, demographic, utility-based, and knowledge-based recommendation strategies.

Think of a system that recommends films, for instance. The system may employ content-based methods to make recommendations for films that are comparable to previous favourites of the user. In order to recommend films that are well-liked by other users, it could also employ collaborative methodologies. Additionally, the system might use demographic techniques to suggest movies that are popular among users in the user's age group and geographic location.

Another example is a product recommendation system that uses a hybrid approach to suggest products to users. The system might use utility-based techniques to suggest products that the user is likely to find valuable based on their past interactions with similar products. It might also use knowledge-based techniques to suggest products that are relevant to the user's specific needs and preferences.

The advantage of hybrid recommendation systems is that they can provide more accurate and personalised recommendations by combining the strengths of multiple recommendation techniques. The system can also make recommendations based on a wider range of information, including the user's past interactions, preferences, and background information. However, hybrid systems can be more complex to implement and maintain than single-technique recommendation systems.

Overall, hybrid recommendation systems are a powerful method for generating recommendations and can provide users with highly personalised and relevant recommendations.

In addition, **hybrid recommendations** have been found in TEL systems. This recommender system combines two or more of the aforementioned categories to

improve performance and handle shortcomings found in pure recommendation techniques (Burke, 2007).

1.3. Motivation

Due to the COVID-19 pandemic and the lockdowns that ensued worldwide, educational institutes are looking for online learning platforms. Online learning is essentially a form of electronic learning which delivers education through the Internet instead of a conventional classroom and has arisen as a vital asset for students and schools everywhere in the world. In recent months, the demand for online learning has risen significantly, and it will continue to do so. The new normal in education is the increased use of online learning tools.

Recommender frameworks assist the user with finding an item and suggesting relevant items. As online learning resources proliferate on the World Wide Web, online learners struggle to choose the most pertinent and acceptable learning materials that match their needs due to information overload. Since the advent of the Internet, picking information from a plethora of possibilities has proven to be a substantial difficulty due to information overload (Bull, 2010).

Learners can overcome this challenge with the help of recommender systems, which automatically recommend the most relevant learning resources to them based on their individual preferences and profile. Researchers in this discipline are increasingly interested in recommender systems for e-learning. A recommender system is a set of software tools and strategies that make suggestions for products users find helpful (Choi, 2020). To address information retrieval issues brought on by information overload, the idea of customised and intelligent agents, search engines, and recommender systems has received significant support (Chrysafiadi, 2013).

The recommendations made by recommender systems are tailored to the requirements and preferences of the user, as opposed to search engines or retrieval systems, which provide relevant results that match the user's query. They are crucial in e-commerce and e-learning (Corbett, 1994). Real-world recommender systems include Amazon's book recommendations and Netflix's movie recommendations. Furthermore, the study of recommender systems in e-learning has become necessary as a research area (Gervet, 2020). These technologies assist students in an e-learning environment in swiftly locating relevant learning resources appropriate for their needs.

Combining user expectations (profiles) with items that need to be appropriately advised is one of the difficulties of this kind of system (HAO, 2019). The e-learning domain has different recommender demands than other domains. Researchers have created recommender systems that use a variety of techniques to distinguish between useful and unimportant input (Hochreiter, 2001). It is also challenging to suggest learning materials to a particular student since different learners have varied characteristics in terms of prior knowledge, history, competency level, learning style, and learning activities in e-learning (Khajah, 2016).

During online evaluations, students' activities can be recorded, i.e., when a student joins a class, when he/she quiets, play/stop the ongoing meeting and how much time elapsed on each question. Even the changes in the options before being submitted can be recorded. Those patterns can be studied, and a knowledge-based recommender system can be developed. As a result, even if two learners have similar results, if their learner characteristics vary, they will need different suggestions. Klanja-Milievi et al. (2011) contend that recommender systems for e-learning must take the learner's specific preferences and needs into account. In e-learning environments, evaluating these qualities for a specific learner is becoming increasingly necessary throughout the personalisation and recommendation process.

1.4. Research Question

This study tries to address the following research queries:

1. How can the use of a recommender system improve the process of active e-learning?
2. Is a 1-stage scoring recommender system sufficient to accurately score and recommend item-user interactions? Or is there a need to incorporate additional stages into the recommender frameworks for online education platforms?
3. Do we need simple recommender systems or complex deep-learning recommender models for online education platforms?
4. What data pre-processing and filtering strategy can be used to study item-user interactions in the online education-based dataset?

1.5. Objectives

The following are the objectives of this research:

- To devise a simple (model-free) multi-stage recommendation scheme based on data retrieval, filtering and scoring to recommend relevant items to users.

- The main goal of this study is to create a prototype recommendation system that can suggest appropriate e-learning materials in an e-learning management system (LMS).

1.6. Thesis Structure

The thesis titled "Content-Based Recommender System for Online Education" is structured as follows:

Chapter 1 Introduction discusses active learning practices in traditional education systems and the challenges of incorporating technology-enhanced learning using recommender systems. The importance of using different types and stages of recommendation techniques in various systems is also discussed. **Chapter 2 Literature Review** compares various model-free recommender system techniques, their pros & cons and their performance in real-life applications. **Chapter 3 Methodology** provides details on dataset acquisitions, pre-processing and filtering steps. **Chapter 4, Results & Discussion,** provides research outcomes of data analysis followed by scientific reasoning. **Finally, chapter 5 Conclusion** states the prospects and limitations of the study. In the end, references to the literature and citations are provided.

Chapter 2: Literature Review

2.1. Knowledge Tracing Problem

Numerous online education platforms, including Massive Open Online Courses (MOOCs), intelligent tutoring systems and educational games, have been developed during the past 30 years to support and occasionally fully replace traditional educational institutions. For instance, the COVID-19 pandemic halted traditional classroom-based teaching and accelerated the digital transformation of educational systems. As a result, teachers and students worldwide were forced to quickly adapt to an online learning environment to lessen the impact of COVID-19. However, although the use of computer technologies in teaching is urgently needed, it has also presented a new obstacle: efficiently monitoring a student's learning progress through online interactions with teaching platforms (Wang., 2019).

In the past, Anderson et al. first proposed the idea of knowledge tracing (K.T.) in a technical study for cognitive modelling and intelligent tutoring in 1986. This report was then published in the Artificial Intelligence journal in 1990 (John R Anderson, 1990). Since then, numerous initiatives have been made to develop machine learning models for resolving the K.T. problem. An instructor can keep a check on the student's knowledge in order to tailor the lesson plan to their needs. Similar technology is required to monitor student understanding and personalise their learning experiences with the advent of online learning platforms. This is referred to as the K.T. problem in the literature. Figure 4 illustrates how knowledge tracing might work in an intelligent tutoring system (ITS). It depicts a situation in which a student interacts with an Intelligent Tutoring System (ITS) and is given a series of questions from a question set (q1, q2, q3, q4) and instructed to respond to them. The ITS assesses the student's knowledge states over the abilities k1, k2, k3, and k4 (for example, maths skills like addition, subtraction, and multiplication) needed to respond to these questions during the interaction. By solving the K.T. problem, the potential of computer-aided educational initiatives like intelligent tutoring systems, curriculum learning, and material learning suggestions would be unleashed (Anderson, 1986).

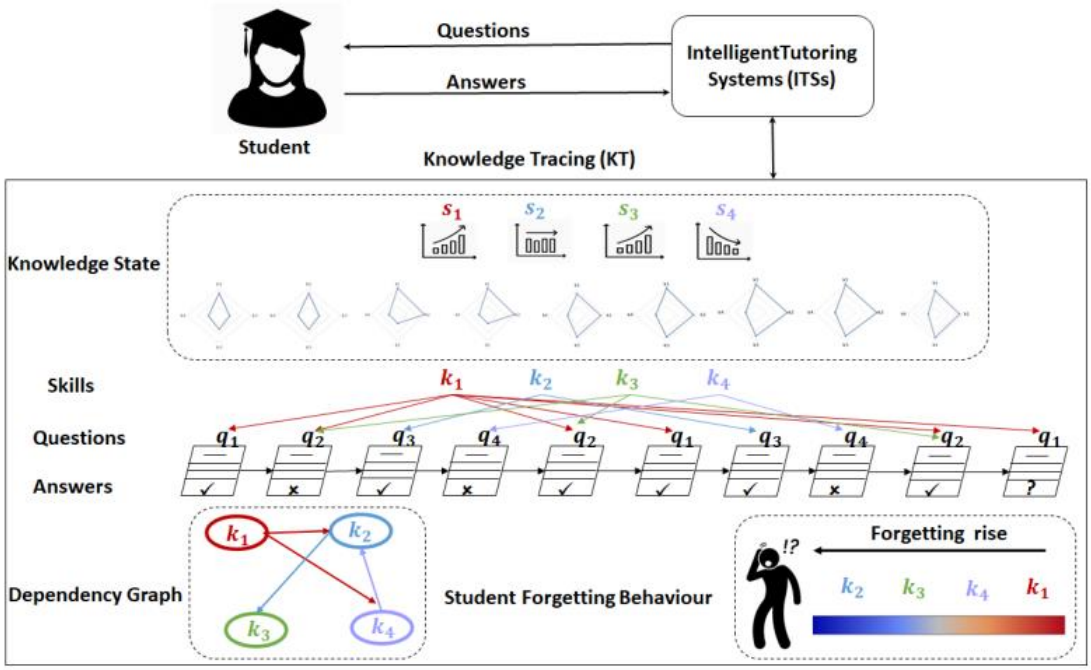

Figure 4 illustrates how knowledge tracing might work in an intelligent tutoring system (ITS). It depicts a situation in which a student interacts with an Intelligent Tutoring System (ITS) and is given a series of questions from a question set (q1, q2, q3, q4) and instructed to respond to them. The ITS assesses the student's knowledge states over the abilities k1, k2, k3, and k4 (for example, maths skills like addition, subtraction, and multiplication) needed to respond to these questions during the interaction.

Early efforts adopted **Bayesian inference methods**, which frequently depended on oversimplifying the model assumptions (for example, assuming only one skill) to make the posterior computation manageable. Bayesian Knowledge Tracing (BKT) models frequently employ probabilistic graphical models like the Markov Analysis to track students' shifting knowledge states. The Bayes' theorem, which states that the following holds for two events, A and B, is fundamental to these models:

$$P(A|B) = P(B|A) * p(A) / p(B)$$

Equation 1: Bayesian Knowledge Tracing (BKT) Probability Function. This probabilistic approach is an ancient method used to solve knowledge-tracing problems.

Later, with the emergence of conventional machine learning techniques like logistic regression models, K.T. turned to parametric factor analysis methodologies, which track a student's knowledge levels and conduct response prediction based on modelling numerous factors. Factor analysis is a statistical approach for reducing the number of observable components to gain a deeper understanding of a dataset (Hao Cen, 2006). Additional elements, such as the amount of time between a student's various interactions and the frequency of those interactions, have also been indicated by studies about students' learning behaviour and forgetting behaviour (J. Bobadilla, 2013). The main challenge with this technique is striking a balance between

performance and scalability. It is challenging to sustain accurate recommendations for new users with less knowledge and seasoned users with numerous interactions. Figure 5 gives an overview of traditional knowledge tracing strategies. Bayesian Knowledge Tracing (BKT) is a probabilistic approach to studying various events that occur during student-item interactions in Technology Enhanced Learning (TEL) systems. Factor analysis models consider high-importance parameters like student learning rates and time to propose solutions to knowledge-tracing problems.

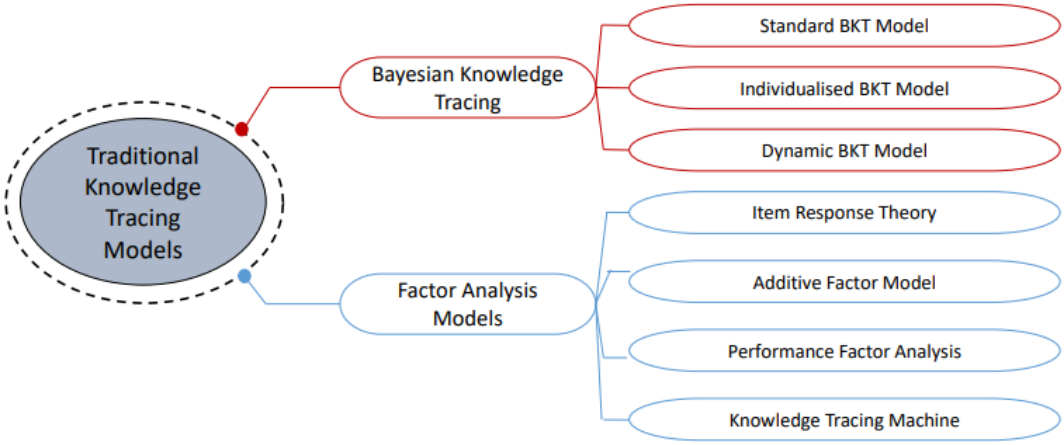

Figure 5: An overview of traditional knowledge tracing strategies. Bayesian Knowledge Tracing (BKT) is a probabilistic approach to studying various events that occur during student-item interactions in Technology Enhanced Learning (TEL) systems. Factor analysis models consider high-importance parameters like student learning rates and time to propose solutions to knowledge-tracing problems.

2.2. Modern Education Recommender Systems (ERS)

Education Recommender Systems (ERS), developed from 2016 onwards, make use of new recommendation techniques to propose higher prediction accuracy than the simple statistical techniques for knowledge tracing like BKT or factor analysis models.

By selecting learning items from educational repositories, Sergis and Sampson (2016) provide a recommendation system that aids teachers in their teaching methods. This ERS makes recommendations based on the instructors' level of Information and Communication Technology (ICT) competency. Zapata et al. (2015) developed a recommendation system for learning objects for teachers. The study outlines a system based on voting aggregation strategies and collaborative methodology for group recommendations. The Delphos recommender system employs this strategy. A recommendation system based on various machine learning algorithms was created by Yanes et al. (2020) to help teachers enhance the effectiveness of their teaching methods.

The recommendations of Tarus et al. (2017) are generated for students. This is an e-learning resource recommendation system based on ontology-mapped information of users and items. A technique for recommending elective courses was presented by Huang et al. in 2019. The recommendation indicators are based on the student's curriculum time limits and how well the student performs academically compared to senior students. The recommendations of disciplines are covered by Fernandez-Garcia et al. (2020) using a sparse dataset with few cases. To aid students in picking their courses, the authors created a model based on several data mining and machine learning techniques.

The length of time people spend studying is another crucial factor, according to Nabizadeh et al. (2020). In this study, a suggested learning route including lessons and learning materials is presented. The learner's good performance score is calculated by such a system, and it then produces a learning route that meets with their time constraints. Additional resources are suggested by the suggestion strategy for people who don't perform as expected. A recommender that evaluates students' comprehension of a subject and creates a list of tasks with varied degrees of difficulty was created by Wu et al. in 2020.

Google searches that are most suited to students' academic levels are suggested using an ERS offered by Rahman and Abdullah (2018). The recommended technique separates students into groups and locates websites with information pertinent to their shared interests and group members' shared characteristics. Ismail et al. (2019) also developed a recommender to support informal learning. It provides Wikipedia content that takes user activity and platform data from unstructured text into account.

Nafea et al. (2019) suggest a novel ERS for making recommendations. The ERS considers item ratings and students' learning preferences when recommending learning objects. The recommender system presented by Klanja-Milievi et al. (2018) is based on tags chosen by the students and has been commercially deployed in the Protus e-learning system.

Wan and Niu (2016) provide a recommender system based on combined idea mapping and immunology algorithms (set of computational systems). For pupils, it compiles materials for learning. The self-organisation idea is applied to ERS in another study by the same authors. Self-organizing learning objects are a topic covered by Wan and Niu (2018). Resources in this study act like people who can move towards learners. Based on the learning characteristics and behaviours of the students, this interaction

produces recommendations. According to Wan and Niu (2020), self-organisation is the method used by students motivated by a need to learn. The authors propose an ERS that suggests learning items based on self-organized cliques of learners.

Wu et al. (2015) propose a recommendation system for e-learning settings. To replicate the complexity and uncertainty associated with user profile data and learning activities in this study, fuzzy logic and tree architectures are combined. Recommendations are produced as these structures are matched.

2.3. Recommendation Techniques in Modern ERS

Three prominent recommendation techniques form the basis of the recommendation framework in modern ERS. Applications of hybrid, collaborative and content-based recommendation approaches are discussed next.

2.3.1. Collaborative Filtering

The collaborative filtering (C.F.) recommendation paradigm is crucial for research and is most frequently used. Cross-user-domain collaborative filtering recommendation approach groups neighbour users on cosine similarity technique (Huang et al., 2019). Figure 6 shows collaborative filtering using cosine similarity is represented in a vector space. An item named Bundle 2 is more relevant to the user preferences since the angle between their respective vectors θ is more minor than between the user and the item named Bundle 1 at angle φ.

$$\text{cosine similarity}(\vec{u}, \vec{v}) = |cos(\theta)| = \left| \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \, \|\vec{v}\|} \right|$$

Equation 2: Cosine Similarity Vector Form.

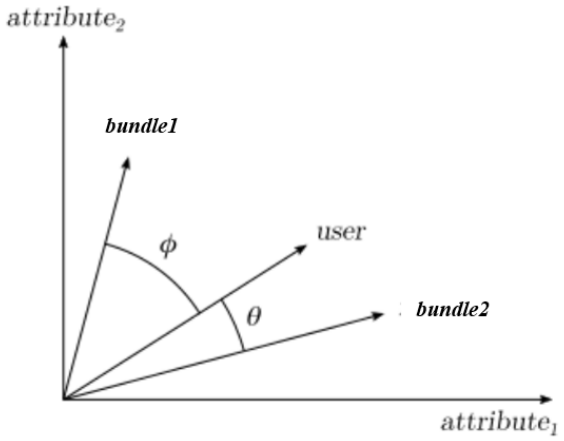

Figure 6: Collaborative filtering using cosine similarity is represented in a vector space. An item named Bundle 2 is more relevant to the user preferences since the angle between their respective vectors θ is more minor than between the user and the item named Bundle 1 at angle φ.

In most cases, the C.F. a user-based variant is usually applied. Unlike the object-based variant, which bases predictions on item similarities, this version calculates predictions based on user similarity (Isinkaye et al., 2015). All identified CF-based recommendation systems, whether standalone or combined with other techniques, employ this variant. The benefit of a student-centered teaching and learning process in education is demonstrated by this study conclusion (Krahenbuhl, 2016; Mccombs, 2013). This idea is supported by recommendation techniques based on user profiles, including interests, requirements, and capabilities.

2.3.2. Content-Based Filtering

In contrast to CF-based ERS, Content-Based Filtering (CBF) recommenders have not had the same level of popularity. However, this method is an established approach for recommendations and generates outcomes based on the similarity between items the user is familiar with and other recommendable items (Bobadilla et al., 2013). A CBF-based recommendation system is suggested by Nafea et al. (2019) for students.

The Personalised E-Learning system (PEL-IRT) uses item response theory (Chen et al., 2005). It suggests appropriate course content for students, considering the course material's difficulty and the student's aptitude. Students can select course categories and units in PEL-IRT and search for engaging course material using relevant keywords. The system asks students to complete two questions after suggesting course material and letting them peruse the content. PEL-IRT uses this specific feedback to reassess the students' skills and modify the recommended level, of course, material difficulty.

$$p\left(\frac{C}{Q}\right) = \frac{1}{1+e^{(a-\beta_{qs})}}$$

*Equation 3: Logistic form of the item response theory (IRT) predictions. The **C** value indicates whether a response to question **Q** is correct. The **a** value indicates a learner's aptitude, while the **β** indicates how challenging a question is.*

2.3.3. Hybrid Filtering

Many ERS use mixed recommendation techniques. In order to overcome or circumvent the constraints of pure recommendation systems, these recommenders are characterised by computing predictions using a combination of two or more algorithms (Isinkaye et al., 2015). Sergis and Sampson (2016) proposed a recommender based on two main techniques: fuzzy sets to handle uncertainty regarding teacher competency

levels and collaborative filtering (C.F.) to identify learning materials based on neighbours who may share competencies.

Tarus et al. (2017) use ontologies to represent student and learning resource profiles. The system then uses a sequential pattern mining algorithm and collaborative filtering to produce predictions and offer learning resources. Furthermore, although being a well-established method (Bobadilla, Ortega, Hernando, and Gutiérrez, 2013), the hybrid strategy that combines collaborative filtering with Content-Based Filtering (CBF) does not appear to be widely used in research on recommender systems for teaching and learning.

Wu et al. (2015) employ fuzzy trees to organise user data and learning activities. These structures use fuzzy sets to represent the values given to the tree nodes. A collaborative filtering technique for similarity computation and a tree-structured data matching approach are both fed by the fuzzy tree data model and user ratings.

2.4. Input Features Incorporated in Modern ERS

Based on input parameters used in modern ERS frameworks, several variations of recommenders are available in the literature.

Important determining variables for the inputs utilised in the suggestion process are the users' educational profile traits. Various academic data, such as learning objectives, learning styles, and learning levels are examples (Yanes et al., 2020; Fernández-Garca et al., 2020); ICT competences (Sergis & Sampson, 2016); Wu et al., 2015; Nafea et al., 2019); and learning levels (Tarus et al., 2017). Some systems, such the CBF-based recommender developed by Nafea et al. in 2019, include item-related data into the recommendation process.

In study, academic knowledge and learning preferences are given greater weight than other considerations. Student grades (Huang et al., 2019), academic background (Yanes et al., 2020), learning categories (Wu et al., 2015), and topics taken (FernándezGarca et al., 2020) are a few of the academic data analysed.

It was noted that explicit feedback is preferred above other data-gathering methods in terms of how inputs are recorded. In this method, users are required to directly give the data that will be utilised to create recommendations (Isinkaye et al., 2015). The use of graphical interface components (Klanja-Milievi et al., 2018), surveys

(Wan & Niu, 2016), and manual entry of datasets (Wu et al., 2020; Yanes et al., 2020) are the main methods for explicit data collecting.

The recommender system gathers information implicitly when it infers inputs. For instance, data extraction from another system (Ismail et al., 2019), users' data session monitoring (Rahman & Abdullah, 2018), users' data usage tracking, such as access, browsing, and rating history (Rahman & Abdullah, 2018; Sergis & Sampson, 2016; Wan & Niu, 2018), and data estimation (Nabizadeh et al., 2020) are examples of implicit data collection methods.

In the area of recommender systems for teaching and learning support, the implicit collection of data has traditionally been examined in addition to the explicit one as a complimentary strategy.

Table 2 summarises the comparative study based on the recommendation techniques, input parameters and data collection by different ERS. It gives a comparison of Modern Education Recommender Systems (ERS). The recommender systems developed from 2016 onwards in the Technology Enhanced Learning (TEL) field have different capabilities to solve knowledge tracing (KT) problems. The comparison between ERS is based on the recommendation approach, recommendation techniques, data collection strategy and methods, and the input parameters used by the recommender system.

ESR (citation)	Recommendation approach	Techniques	Input parameters	Data collection strategy	Data collected
Sergis and Sampson (2016)	Hybrid (collaborative filtering and fuzzy logic)	Neighbours users based on Euclidean distance and fuzzy sets	(i) ICT Competency; (ii) Rating (users' preferences)	Hybrid	(i) Collection of users' usage data; (ii) User defined

Zapata et al. (2015)	Hybrid (techniques for the group-based recommendation)	Collaborative methodology, voting aggregation strategies and meta-learning techniques	Rating (users' preferences)	Explicit	User-defined
Yanes et al. (2020)	Hybrid (machine learning algorithms)	One-vs-All, Binary Relevance, Classifier Chain, K Nearest Neighbors	Academic information	Explicit	Input file/Dataset
Taurus et al. (2017)	Hybrid (Collaborative Filtering, sequential pattern mining and knowledge representation)	Neighbours users based on cosine similarities, Generalised Sequential Pattern algorithm and student/ learning resource domain ontologies	(i) Learning style; (ii) Learning level; (iii) Item attributes (iv) Rating (users' preferences)	Explicit	(i) Questionnaire; (ii) Online test; (iii) N/A; (iv) User defined
Huang et al. (2019)	Cross-user domain collaborative filtering	Neighbours users based on cosine similarity	Academic information	Explicit	Input file /Dataset

Fernandez-Garcia et al. (2020)	Hybrid (data mining and machine learning algorithms)	Encoding, Feature Engineering, Scaling, Resampling, Random Forest, Logistic Regression,	Academic information	Explicit	Input file /Dataset
Nabizadeh et al. (2020)	Hybrid (graph-based, clustering technique and matrix factorisation)	Depth-first search, k-means and matrix factorisation	(i) Background knowledge; (ii) users' available time; (iii) Learning score	Implicit	(i) Collection of users' usage data; (ii, iii) estimated data
Wu et al. (2020)	Hybrid (neural network techniques)	Recurrent Neural Networks and Deep Knowledge Tracing	Answers records	Explicit	Input file/Dataset
Rahman and Abdullah (2018)	Group based	Grouping algorithm	(i) Academic information; (ii) learners' behaviours (iii) Contextual information	Implicit	(i) Learning management system records; (ii) Collection of users' usage data; (iii) Tracking changes in user academic records and behaviour

Ismail et al. (2019)	Hybrid (Graph-based and fuzzy logic)	Structural topical graph analysis algorithms and fuzzy set	(i) Learning interests; (ii) Thesaurus	Implicit	(i) Collection of users' usage data; (ii) Data extraction from another system
Nafea et al. (2019)	Collaborative filtering, content-based filtering and Hybrid (combining the last two approaches)	Neighbours users based on Pearson correlation, Neighbors items based on Pearson correlation and cosine similarity	(i) Learning style; (ii) Item attributes; (iii) Rating (users' preferences)	Explicit	(i) Questionnaire; (ii) Specialist defined
Klašnja-Miliéevié et al. (2018)	Hybrid (Social tagging and sequential pattern mining)	Most popular tags algorithms and weighted hybrid strategy	(i) Tags; (ii) Learners' behaviours	Hybrid	(i) User-defined; (ii) Collection of users' usage data
Wan and Niu (2016)	Hybrid (knowledge representation and heuristic methods)	Mixed concept mapping and immune algorithm	(i) Learning styles; (ii) item attributes	Explicit	(i) Questionnaire; (ii) Specialist defined / Students' feedback

Wan and Niu (2018)	Self-organisation based	Self-organisation theory	(i) Learning style; (ii) Item attributes; (iii) learning objectives (iv) learners' behaviours	Hybrid	(i) Questionnaire; (ii) Specialist defined / students' feedback; (iii) N/A; (iv) Collection of users' usage data
Wan and Niu (2020)	Hybrid (fuzzy logic, self-organisation and sequential pattern mining)	Intuitionistic fuzzy logic, self-organisation theory	(i) Learning style (ii) Learning objectives (iii) Tags (iv) Academic information (v) Information from academic, social relations	Hybrid	(i, ii) Questionnaire (iii) Extracted from m learners' learning profiles (iv, v) Extracted from e-learning platform records

Table 2: Comparison of Modern Education Recommender Systems (ERS). The recommender systems developed from 2016 onwards in the Technology Enhanced Learning (TEL) field have different capabilities to solve knowledge tracing (K.T.) problems. The comparison between ERS is based on the recommendation approach, recommendation techniques, data collection strategy and methods, and the input parameters used by the recommender system.

2.5. Display of Recommendations by ERS

In recommender systems for teaching and learning support, there are two approved ways to offer suggestions. First off, the majority of ERS base their lists of rated things on predictions made about each user. This method is applied in any circumstance where it is beneficial to locate useful objects to assist users in teaching and learning activities (Ricci et al., 2015; Drachsler et al., 2015).

The second method is centred on the generation of ***learning pathways***. In this case, recommendations are presented as linked items connected by a few preconditions. Such learning pathways are created using learning object association attributes (Wan and Niu, 2016) and a mix of the user's past knowledge, the amount of time available, and the learning score (Nabizadeh et al., 2020). The purpose of these ERS, linked to

the item sequence recommendation task, is to direct users toward a particular level of expertise (Drachsler et al., 2015).

Many ERS deviate from the traditional search engine model by omitting filtering options. In Huang et al.'s (2019) study, for instance, the development of anticipated student scores and a list of the top optional courses are very loosely described, and it is not made clear how the list is shown. This could be explained by the fact that the majority of these recommenders do not mention how they integrate their services with other systems (like learning management systems) or why they make them available as independent tools (like online or mobile recommendation systems). The lack of these specifications lessens the necessity of creating a polished presenting interface. For instance, Nafea et al. (2019), Wan and Niu (2018), and Tarus et al. (2017) are the sole authors to suggest recommenders.

A panel named "recommendations for you" is used by Rahman and Abdullah (2018). Ismail et al. (2019) present the user with ideas in a pop-up box. According to Beel et al. (2013), some ERS display "organic recommendations," that is, naturally ordered objects for user engagement.

2.6. Literature Gap

According to Pöntinen et al. (2017), informal learning is a type of learning that often takes place outside of a formal educational setting. As a result, students are neither guided by a domain expert nor have to follow a predefined curriculum (Pöntinen et al., 2017; Santos & Ali, 2012). Such factors affect how ERS can assist users. For instance, content may come from various sources in informal situations so that it may be presented without regard to the proper pedagogical sequence. In return, ERS aiming at this scenario should prioritise arranging and sequencing recommendations guiding users' learning processes (Drachsler et al., 2009).

Despite literature highlighting the existence of significant differences in the design of educational recommenders that involve formal or informal learning circumstances, the current study appears to be focused on the formal learning context (Drachsler et al., 2009; Okoye et al., 2012; Manouselis et al., 2013; Harrathi & Braham, 2021). This is true because recommendations are based on information kept up to date by institutional learning systems. A lack of pedagogical sequencing to support self-directed and self-paced learning is another issue with most proposals (e.g., those that

create a learning route that leads to a certain body of information). The coronavirus epidemic, on the other hand, has caused the scientific community to pay more attention to informal learning (Watkins & Marsick, 2020).

The research titled "Content-Based Recommender System for Online Education" is aimed at developing an ERS for informal learning settings where students will be presented with a correct sequence of learning items from open-source education datasets (explicit data collection). Furthermore, using a content-based filtering approach on input features, learning paths will be generated for users who want to score above average in quizzes without the help of instructors.

Chapter 3: Methodology

The research methodology adopted for the ERS study revolves around procedures of relevant data collection, data exploration, feature selection, feature engineering and the ERS interface design.

3.1. Research Methodology Workflow

Developing a content-based recommender system for online education is a multi-step procedure. The pipeline followed in this research is illustrated in Figure 7 as a research methodology workflow for developing a content-based education recommender system. The retrieval stage involves data collection & exploration. The filtering stage filters input features based on relevance in ESR. The scoring and ordering stage gives scores or weightage to selected features and provides a recommendation policy to display outcomes on the user-friendly interface.

Stage 1 - Retrieval

- Step 1: Data is collected from online education resources containing information about users and learning objects.
- Step 2: Data is explored for missing values, redundancy check and variables analysis.

Stage 2 - Filtering

- Step 3: Relevant input parameters are selected to study specific attributes of learning objects. In this particular research, learning objects are bundles included in EdNet data.
- Step 4: New features are added, providing additional analysis criteria for the input parameters.

Stage 3 - Scoring & Ordering

- Step 5: Learning objects or bundles are grouped based on certain similarities or patterns. Such patterns in the learning objects can be deducted from personalisation based on specific users or by grouping several users of the same attributes. Most prevalent patterns are scored higher than less prevalent patterns.
- Step 6: A business logic based on rule-based ordering is deployed to design a recommender system. An interface design helps present the recommender system's output or recommendations to the end users.

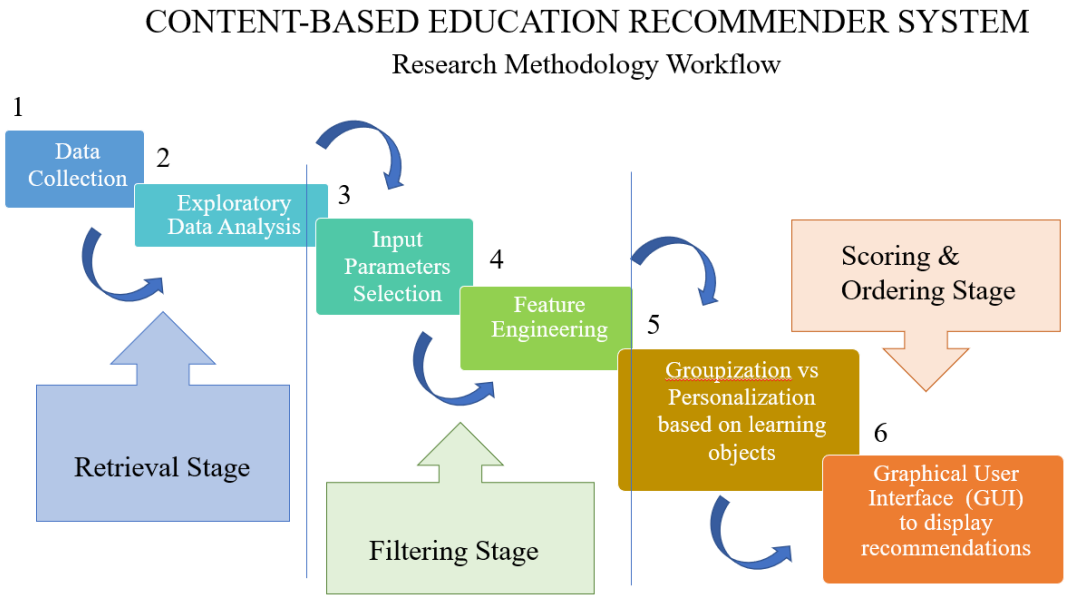

Figure 7: Research methodology workflow for developing a content-based education recommender system. The retrieval stage involves data collection & exploration. The filtering stage filters input features based on relevance in ESR. The scoring and ordering stage gives scores or weightage to selected features and provides a recommendation policy to display outcomes on the user-friendly interface

3.2. Data Collection

The information is taken from EdNet, publicly accessible through the project's official GitHub repository. Santa gathered a vast dataset of all student-system interactions over two years, which he called EdNet. Santa is an artificial intelligence tutoring service available on various platforms, including iOS, Android, and the web, and it has over 780,000 users in Korea. After downloading and decompressing the data, the 19.4 GB of data is broken down into over 1.8 million individual users (CSV) files.

3.2.1. Properties of EdNet Dataset

The EdNet dataset has several properties which make it a suitable choice to develop an ERS and validate predictions made by the recommender.

1. **EdNet is a massive dataset**

Since 2017, almost 1.4 million interactions have been collected from roughly 1.8 million Santa pupils in the dataset. Through various channels, each student has produced around 442 interactions with Santa. In addition, this A.I. the teaching programme contains 13,169 issues expressed as questions from 1,021 lectures, each of which has been seen nearly 95 million times. According to our understanding, this is the largest publicly available education dataset regarding students, interactions, and types of interactions.

2. **Diversity in the EdNet dataset**

EdNet has the broadest range of data. Learning activities on EdNet include viewing lectures, reading explanations, attempting questions, repeating questions, and logging in and out of timeframes. No other dataset, to our knowledge, is as rich as the EdNet dataset.

3. **EdNet is data collected from multiple platforms**

In this age of modernity, where everyone has access to many gadgets, from personal computers to cell phones and Artificial Intelligence speakers, students may reach Santa utilising any platform. Santa is a cross-platform application that works on iOS, Android, and the web. This dataset includes data from both mobile and desktop devices.

4. **Hierarchy in EdNet dataset**

EdNet's data points are organised hierarchically. It gives the dataset at five distinct levels, each labelled KT1, KT2, KT3, KT4, and Content, to provide many data points in a consistent and organised manner. As the dataset's intensity grows, so does the number of actions and types of activities engaged.

3.3. Exploratory Data Analysis

Different tiers of datasets referred to as KT1, KT2, KT3, KT4, and Content subsets were explored during data exploration. The entire dataset is separated by students identified by user ID.

3.3.1. Python Implementations

Table 3 lists all python libraries used for reading, writing and analysing the EdNet dataset.

Library	Usage
Pandas	Data analysis begins with importing the data files, followed by data cleaning, integrating several datasets into one and statistical analysis.
Numpy	Numpy makes it possible to work with multi-dimensional arrays effectively.

Matplotlib	It is a tool that is used for data visualisation and plotting.
Seaborn	Visualisation package that is built on top of Matplotlib to generate aesthetically appealing graphs
DateTime	This module supplies classes that are used for accessing and manipulating data and time
Feather	Feather format is an alternative file format to store the dataset. It offers more extraordinary read & write speeds than the traditional CSV or excel file format.

Table 3: Python libraries used for analysing EdNet datasets.

- **Reading & Writing Data Frames**

Performance problems occurred when the data frame was exported since it was approximately 10.0 Gb when produced as a single CSV or excel file. It was a time-consuming operation and was not the best course of action. As a result, the data frame was converted into the feather format, which has far faster read/write rates than CSV and dramatically reduces the data frame size.

3.3.2. EdNet Sub-Datasets

Here is a quick look at each dataset level included for analysis.

3.3.2.1. KT1 Sub-Dataset

Figure 8 displays the KTI Subset of EdNet. Input features included in KT1 are a 13-digit Unix timestamp, question integer I.D., bundle I.D. containing several questions, user responses as string variables and time elapsed in recording responses.

- **Timestamp**: Time recorded in milliseconds since the user was provided with the question. This input feature is displayed as a Unix timestamp.
- **question_id**: The input feature represents the question presented to a student.
- **bundle_id**: The input feature represents from which specific bundle that question belongs. In standard terms, these can also be known as chapters.
- **user_answer**: student's answer to that question, presented and recorded as a character between a and d inclusively.

timestamp	question_id	bundle_id	user_answer	elapsed_time
1548996377530	48	q2844	d	47000
1548996378149	48	q2845	d	47000
1548996378665	48	q2846	d	47000
1548996671661	49	q4353	c	67000
1548996787866	50	q3944	a	54000

Figure 8: KTI Subset of EdNet. Input features included in KT1 are a 13-digit Unix timestamp, question integer I.D., bundle I.D. containing several questions, user responses as string variables and time elapsed in recording responses.

3.3.2.2. KT2 Sub-Dataset

Figure 9 shows the KT2 Subset of EdNet. Input features included in KT2 are a 13-digit Unix timestamp, three different action types performed by users, item I.D. representing bundles and questions, source representing user activity status, user responses as string variables and platform from which data was collected for this record.

- **action_type**: This column contains three values:
  - **Enter**: this is documented when a student receives and views a question bundle through SantaUI
  - **Respond:** This is documented when the student answers from a to d respectively; a student can change their answer to the same question multiple times; the last response a student gives would be considered the last response.
  - **Submit**: This is documented when a student submits the final answer to the given question from the bundle
- **item_id**: This column contains two values (bundle integer or question integer); for KT2-only, the I.D.s of questions and bundles are documented; whenever the action type is "enter" or "submit", it is documented as bundle id, but whenever the action type is "respond" it records as question id.
- **Source:** This column represents where the student is watching a lecture on solving a question in SANTAUI.
  - **Sprint:** This entry represents students choosing a part that they want to study; after selecting the specific path, they can only answer questions to specific questions of that bundle unless they change their path

- **Diagnosis:** This represents when a user enters the SantaUI application for the first time, they need to solve several questions for diagnosis (as per the author of the dataset)

- **user_answer:** This column only records values from a to d, respectively, whenever the action type is "respond".
- **Platform:** This column represents which specific device students use to access the SANTAUI, which contains mobile & web.

timestamp	action_type	item_id	source	user_answer	platform
1358114668713	enter	b4957	diagnosis		mobile
1358114691713	respond	q6425	diagnosis	c	mobile
1358114701104	respond	q6425	diagnosis	d	mobile
1358114712364	submit	b4957	diagnosis		mobile
1358114729868	enter	b5180	sprint		mobile
1358114745592	respond	q6815	sprint	c	mobile
1358114748023	respond	q6816	sprint	a	mobile
1358114748781	respond	q6814	sprint	a	mobile
1358114751032	submit	b5180	sprint		mobile

Figure 9: KT2 Subset of EdNet. Input features included in KT2 are a 13-digit Unix timestamp, three different action types performed by users, item I.D. representing bundles and questions, source representing user activity status, user responses as string variables and platform from which data was collected for this record.

3.3.2.3. KT3 Sub-Dataset

Figure 10 shows the KT3 Subset of EdNet. Input features included in KT3 are a 13-digit Unix timestamp, three different action types performed by users, item I.D. representing bundles and questions, source representing user activity status, user responses as string variables and platform from which data was collected for this record.

- **Reading explanations:**
  - When the student attempts the given question, corresponding explanations are provided, and the source column will record as "sprint" or "review". Students can also re-read the explanations of the questions; in that case, the source column will have "my_note".
  - As the student enters or exits the explanation view in SANTAUI, the item_id will record as "enter" or "quit".
- **Watching lectures:**
  - Whenever a student watches a lecture, source column entries have two outcomes: "archive" and "adaptive_offer". "Archive" is watching a recorded lecture presented in the SANTAUI, adaptive_offer is

SANTAUI recommending the following lectures based on the student's path.

- Action_type would register as "enter" and "quit" with lecture I.D. when the student starts watching or stops the lecture.

timestamp	action_type	item_id	source	user_answer	platform
1573364188664	enter	b790	sprint		mobile
1573364206572	respond	q790	sprint	b	mobile
1573364209673	respond	q790	sprint	d	mobile
1573364209710	submit	b790	sprint		mobile
1573364209745	enter	e790	sprint		mobile
1573364218306	quit	e790	sprint		mobile
1573364391205	enter	l540	adaptive_offer		mobile
1573364686796	quit	l540	adaptive_offer		mobile
1573364693793	enter	b6191	adaptive_offer		mobile
1573364702213	respond	q8840	adaptive_offer	c	mobile
1573364705838	submit	b6191	adaptive_offer		mobile

Figure 10: KT3 Subset of EdNet. Input features included in KT3 are a 13-digit Unix timestamp, three different action types performed by users, item I.D. representing bundles and questions, source representing user activity status, user responses as string variables and platform from which data was collected for this record.

3.3.2.4. KT4 Sub-Dataset

Figure 11 shows the KT4 Subset of EdNet. Input features included in KT4 are 13-digit Unix timestamp, new action types performed by users, item I.D. representing bundles and questions, source representing user activity status, user responses as string variables and platform from which data was collected for this record.

- **Erase_choice, undo_erase_choice**: A student can erase his choice while attempting a question and undo choice.
- **Play_audio, pause_audio, play_video, pause_video**: As the lectures and the explanations provided by expert teachers are recordings saved in SANTAUI, a student can select any of these available actions, and while choosing these options, a cursor_time is recorded in the cursor_time column when the students play or pause the media.
- **Pay, refund**: SANTAUI offers a free trial to every student who wishes to get an idea of the U.I. In a free trial, the student is provided ten questions; if they hope to continue, they need to pay to access all the contents. Refunds may also apply to several specific situations.
- **Enrol coupon**: this entry is recorded when a student pays for full access to SANTAUI contents and applies valid coupons during the payment period.

timestamp	action_type	item_id	cursor_time	source	user_answer	platform
1358114668713	pay	p25				mobile
1358114691713	enter	b878		sprint		mobile
1358114701104	text_enter	q878		sprint		mobile
1358114712364	play_audio	q878	0	sprint		mobile
1358114729868	pause_audio	q878	10000	sprint		mobile
1358114745592	eliminate_choice	q878		sprint	a	mobile
1358114748023	respond	q878		sprint	c	mobile
1358114748781	submit	b878		sprint		mobile
1358114751032	enter	e878		sprint		mobile
1358114779211	play_audio	e878	0	sprint		mobile
1358114792300	pause_audio	e878	8000	sprint		mobile
1358114842195	quit	e878		sprint		mobile

Figure 11: KT4 Subset of EdNet. Input features included in KT4 are 13-digit Unix timestamp, new action types performed by users, item I.D. representing bundles and questions, source representing user activity status, user responses as string variables and platform from which data was collected for this record.

3.3.2.5. Content Sub-Dataset

Figure 12 is a preview of the Content Subset of EdNet. Input features included in the subset are question, bundle and explanation I.D., correct answers, parts of questions in numeric values, tag annotations by experts, and timestamp when Santa deploys the question session.

- **question_id**: This column enlists all the question numbers in q{integer}.
- **bundle_id**: This column enlists all the bundle numbers in the form of b{integer}. A single bundle can contain many questions, from a minimum of 1 to a maximum being 5.
- **explanation_id**: This column enlists an expert teacher's corresponding explanations for each bundle. Bundle_id and explanation_id are the same for every question provided
- **correct_answer**: This column enlists the correct answers to the questions; it is recorded as a character between a and d.
- **part**: This column enlists the different parts of a question; it contains numeric values from 1 to 7 inclusively.
- **Tags**: This column contains all the annotated tags by expert teachers for their understanding.
- **deployed_at**: This input specifies the moment each question in Santa begins. It is provided as a Unix timestamp in milliseconds

question_id	bundle_id	explanation_id	correct_answer	part	tags	deployed_at
q2319	b1707	e1707	a	3	179;53;183;184	1571279008033
q2320	b1707	e1707	d	3	52;183;184	1571279009205
q2321	b1707	e1707	d	3	52;183;184	1571279010285
q2322	b1708	e1708	b	3	52;183;184	1571279012823
q2323	b1708	e1708	c	3	179;52;182;184	1571279013890
q2324	b1708	e1708	d	3	52;183;184	1571279014989

Figure 12: Content Subset of EdNet. Input features included in the subset are question, bundle and explanation I.D., correct answers, parts of questions in numeric values, tag annotations by experts, and timestamp when Santa deploys the question session.

3.4. Feature Selection

Figure 13 gives an overview of common users in all subsets of EdNet. Familiar users provide maximum interactions with learning objects (bundles) included in KT1, KT2, KT3, KT4 and the content subset. Data from 1.6 million students' CSV files consisted of several columns, and each column represented a different feature. Familiar users in all subsets of EdNet are filtered out to have data of maximum user interactions with learning objects.

	user_iD	Total Questions Attempted	Corrected	Incorrected	Percentage	elapsed_time
0	1	1082	753	329	69.593346	36.7500
1	10	16	9	7	56.250000	27.7500
2	100	33	18	15	54.545455	34.5000
3	1000	1488	930	558	62.500000	34.0000
4	10000	405	258	147	63.703704	25.0000
...	...	...	...	...	...	...
724134	99994	40	25	15	62.500000	21.5000
724135	99995	933	719	214	77.063237	19.0000
724136	99996	30	14	16	46.666667	23.6665
724137	99997	30	9	21	30.000000	22.1665
724138	99998	40	14	26	35.000000	13.0000

724139 rows × 6 columns

Figure 13: Common users in all subsets of EdNet. Familiar users provide maximum interactions with learning objects (bundles) included in KT1, KT2, KT3, KT4 and the content subset.

Broadly there are two categories of features in the data:

- features explaining user behaviour
- features explaining attributes of learning objects and resources

Our interest was to explore attributes of learning objects and study items. Table 3 gives a list of selected input features for content-based ERS. EdNet five subsets are included, and standard features related to the attributes of bundles (learning objects) are studied.Hence, the following features are included for further investigation:

Input Features	Relevance to ERS
KT1	
timestamp	To study the sequence in which users have interacted with bundles.
question_id	The unique question in each bundle
bundle_id	Unique bundles (main feature)
user_answer	User responses against each question I.D. within a bundle.
KT2	
timestamp	Time gives a sequence in which users have interacted with bundles.
action_type (respond)	User action on bundles
item_id (bundle_id, question_id)	Unique questions in unique bundles
user_answer	User responses against each question I.D. within a bundle.
KT3	
timestamp	Time gives a sequence in which users have interacted with bundles.
action_type (respond)	User action on bundles

item_id (bundle_id, question_id)	Unique questions in unique bundles
user_answer	User responses against each question I.D. within a bundle.
KT4	
timestamp	Time gives a sequence in which users have interacted with bundles.
action_type (respond)	User action on bundles
item_id (bundle_id, question_id)	Unique questions in unique bundles
user_answer	User responses against each question I.D. within a bundle.
Content	
question_id	The unique question in each bundle
bundle_id	Unique bundles (main feature)
correct_answer	Correct responses for each question I.D. within a bundle.
deployed_at	Time gives a sequence in which bundles are presented to the users.

Table 3: Input features for content-based ERS. EdNet five subsets are included, and standard features related to the attributes of bundles (learning objects) are studied.

3.5. Feature Engineering

New features were engineered for further study. These included new parameters like question counts in different bundles, correctness percentages for user responses against each bundle I.D. etc. Supplementary code given in Appendix A presents the

python implementation of the feature engineering step. Values for new features are appended and stored in dictionary format.

3.6. Groupization Scheme for Recommender System

Selected bundle I.D.s were analysed to find any grouping or learning pattern between learning resources. Figure 14 is the python implementation of sorting bundle I.D.s based on time and user I.D. Timestamp (the feature is converted from Unix to datetime format) gives a sequence in which a user has interacted with bundles.

- **Sorting bundles based on time and users:** Bundle I.D.s are sorted in ascending order based on timestamp and user I.D.s.

```
P = P.sort_values(by=['user_iD', 'timestamp'])
```

```
P
```

	index	timestamp	solving_id	question_id	user_answer	elapsed_time	user_iD	correct_answer	bundle_id	explanation_id	deployed_at	correct
1	1	2019-08-06 12:57:01.062	2	4706	c	24.0	1	c	3238	3238	2019-10-30 05:34:30.723	1.0
2	2	2019-08-06 12:58:13.432	3	4366	b	68.0	1	b	2898	2898	2019-10-30 05:37:51.451	1.0
6	6	2019-08-06 13:01:41.746	7	6488	a	35.0	1	a	5020	5020	2019-10-17 03:19:19.221	1.0
7	7	2019-08-06 13:11:41.361	8	356	b	23.0	1	b	356	356	2018-05-18 09:08:49.083	1.0
8	8	2019-08-06 13:12:51.393	9	1382	c	22.0	1	c	1382	1382	2019-10-30 05:35:26.363	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
95293916	90094717	2019-12-02 16:57:40.437	6	8556	c	24.0	840471	c	5907	5907	2019-06-24 08:48:22.531	1.0
95293918	90094719	2019-12-02 16:58:23.437	8	10207	c	19.0	840471	c	7558	7558	2019-10-17 03:15:30.683	1.0
95293919	90094720	2019-12-02 16:59:05.437	1	3661	c	35.0	840472	c	2193	2193	2019-09-17 02:49:18.900	1.0
95293924	90094725	2019-12-02 17:00:37.437	1	3830	c	25.0	840473	c	2362	2362	2019-08-09 08:05:34.925	1.0
95293925	90094726	2019-12-02 17:01:33.437	2	593	c	18.0	840473	c	593	593	2019-10-17 02:19:23.083	1.0

62248682 rows × 12 columns

Figure 14: Python implementation of sorting bundle I.D.s based on time and user I.D. Timestamp (the feature is converted from Unix to datetime format) gives a sequence in which a user has interacted with bundles.

- **Calculating pattern count between bundles:** Previous_1 and previous_2 are two new variables that previously give bundles studied (two-time units) from a given bundle I.D. Figure 15 is the python implementation of calculating pattern count between bundles. Previous_1 & previous_2 are bundle I.D.s studied two-time units before bundle I.D. 1 shown here. The count column gives the number of repetitions of these patterns.This gives a pattern of bundle interaction by users based on time.

```
Q = P.groupby(['bundle_id','previous_1','previous_2']).size().reset_index().rename(columns={0:'count'})
```

```
Q
```

	bundle_id	previous_1	previous_2	count
0	1	1.0	31.0	1
1	1	1.0	116.0	1

Figure 15: Python implementation of calculating pattern count between bundles. Previous_1 & previous_2 are bundle I.D.s studied two-time units before bundle ID 1 shown here. The count column gives the number of repetitions of these patterns.

- **Retaining information on maximum pattern count between bundles:** The patterns for a single bundle which are most redundant or followed by many users repeatedly are retained. Figure 16 gives maximum pattern count. The count column gives the maximum number of sequences in which familiar users study bundle I.D. & previous two bundles. This information serves as a rule-based logic to make predictions.

```
ZZ = Z.drop_duplicates(subset=['bundle_id'],keep='first')
ZZ
```

	Unnamed: 0	bundle_id	previous_1	previous_2	count
0	2731	1	5322.0	5322.0	8
1	8428	2	5322.0	5322.0	9
2	22841	3	5322.0	5322.0	41
3	38524	4	5322.0	5322.0	26
4	50902	5	5596.0	5595.0	4981
...	...	...	...	...	...
939110	38884500	12202	10.0	5573.0	1
939672	38885062	12203	118.0	8246.0	1
940238	38885628	12204	63.0	5573.0	1
940787	38886177	12205	70.0	5573.0	1
941383	38886773	12206	7.0	8099.0	1

8935 rows × 5 columns

Figure 16: Maximum pattern count. The count column gives the maximum number of sequences in which familiar users study bundle I.D. & previous two bundles.

3.7. Recommender System Design

Lastly, the recommended learning paths and lists were added to a database connected to a user interface with the help of software (ASP.NET MVC application).

3.7.1. Database

The database for the recommender is an excel sheet containing a list of bundle I.D.s with 60% corrections (60% of users responded to the questions correctly for the

given bundle). Figure 17 is the outlook of the recommender system database. Rule-based recommendation logic is derived from the information available in the bundle interaction sequence database.Alongside, the file lists two previously studied bundles by common users for a given bundle I.D. Lastly, it contains the maximum number of pattern counts for bundle interactions.

	A	B	C	D
1	bundle_id	Previous 1	Previous 2	count
2	1	3	4	8
3	2	2	12	9
4	3	10	10	41
5	4	3	1	26
6	5	4	10	4981
7	6	3	5	6
8	7	1	5	2170
9	8	3	9	27
10	9	8	12	15
11	10	10	7	18
12	11	1	11	22
13	12	3	1	9
14	13	10	12	19
15	14	1	11	10
16	15	15	3	22
17	16	36	43	19
18	17	33	39	9
19	18	31	26	10
20	19	16	38	35

Figure 17: Recommender system database. Rule-based recommendation logic is derived from the information available in the bundle interaction sequence database.

3.7.2. Software Framework

An ASP.NET MVC is a web application built using the Model-View-Controller (MVC) architectural pattern. This pattern separates an application into three main components: the model, the view, and the controller. It allows developers to build dynamic, data-driven web applications by separating the application's data and business logic from its presentation layer. Figure 18 gives the structure of an ASP.NET MVC application with a Microsoft SQL Server database involving a separation of concerns between the model, view, and controller. The database stores the data, models represent it and provide a way to interact with it, controllers handle requests and render views, and the views render the user interface.

To get started, the following tools are needed:

- **Visual Studio:** This is the primary development environment for building ASP.NET MVC applications. It can be downloaded for free from the Microsoft website.
- **Microsoft SQL Server:** This database management system (DBMS) will store and manage data. There are several versions available, including Express, Standard, and Enterprise.

Once these tools are installed, the ASP.NET MVC application can be built. Here are the basic steps:

(1) The model represents the data and logic of the application. In an ASP.NET MVC application, the model is typically implemented using Entity Framework (E.F.), an object-relational mapper (ORM) that simplifies interacting with a database. For example, E.F. allows the definition of the database schema using classes, which are then used to generate tables and relationships.

(2) The view renders the application's user interface (U.I.). In an ASP.NET MVC application, the view is implemented using Razor templates, which are HTML files that contain C# code for rendering dynamic content.

(3) The controller is responsible for handling incoming requests and rendering views. In an ASP.NET MVC application, the controller is implemented using C# classes that contain action methods, which are responsible for performing specific tasks in response to requests.

(4) Define the database schema. This will involve creating tables to store data and defining their relationships. This is done using the SQL Server Management Studio or writing SQL scripts.

(5) Generate Entity Framework (E.F.) models for your database tables. EF is a powerful object-relational mapper (ORM) that simplifies interacting with your database. "ADO.NET Entity Data Model" can be used as a template in Visual Studio to generate E.F. models based on your database schema.

(6) Test and debug the application. Once built, the application can be tested by running it in a browser and interacting with it. In addition, Visual Studio debuggers can identify and fix any issues that may arise.

Here is a more detailed breakdown of the structure of an ASP.NET MVC application with a Microsoft SQL Server database:

- **Database:** The database is the storage location for data. In this case, Microsoft SQL Server as the database management system (DBMS) is used. The database schema is defined using SQL scripts or the SQL Server Management Studio.
- **Entity Framework Models:** The E.F. models are C# classes representing the database's tables and relationships. These models are generated based on the database schema and are used to interact with the database.
- **Controllers:** The controllers are C# classes that contain action methods that handle incoming requests and render views. Each controller is responsible for a specific application area and is mapped to a specific URL pattern.
- **Views:** The views are HTML files that contain C# code for rendering dynamic content. The controllers render these HTML files in response to requests.
- **Layout:** The layout is a master template that defines the overall layout of the application. It contains common elements such as the header, footer, and navigation menu used across all views.
- **CSS and JavaScript:** CSS (Cascading Style Sheets) and JavaScript are used to add styling and interactivity to the application. These files are typically included in the layout or individual views as needed.
- **Ajax:** Ajax (short for Asynchronous JavaScript and XML) is a set of web development techniques that allows a web page to send and receive data from a server asynchronously without the need to refresh the entire page. This allows for a more seamless and interactive user experience. For example, the page can update specific parts of its content based on user input or other events rather than reloading the entire page. Ajax is typically implemented using a combination of JavaScript, HTML, and XML (though other data formats can also be used). The process usually involves sending a request to a server using JavaScript and processing the received response. This can be done using various methods, such as the XMLHttpRequest object in JavaScript or the fetch API. One of the main benefits of Ajax is that it allows web pages to communicate with a server in the background without interrupting the user's experience. This can make web pages more responsive and interactive, as the page can update its content based on user input or other events without needing a full page refresh.

In conclusion, the structure of an ASP.NET MVC application with a Microsoft SQL Server database involves a separation of concerns between the model, view, and controller. The database stores the data, the E.F. models represent it and provide a way to interact with it, the controllers handle requests and render views, and the views render the user interface. In addition, CSS and JavaScript can add styling and interactivity to the application.

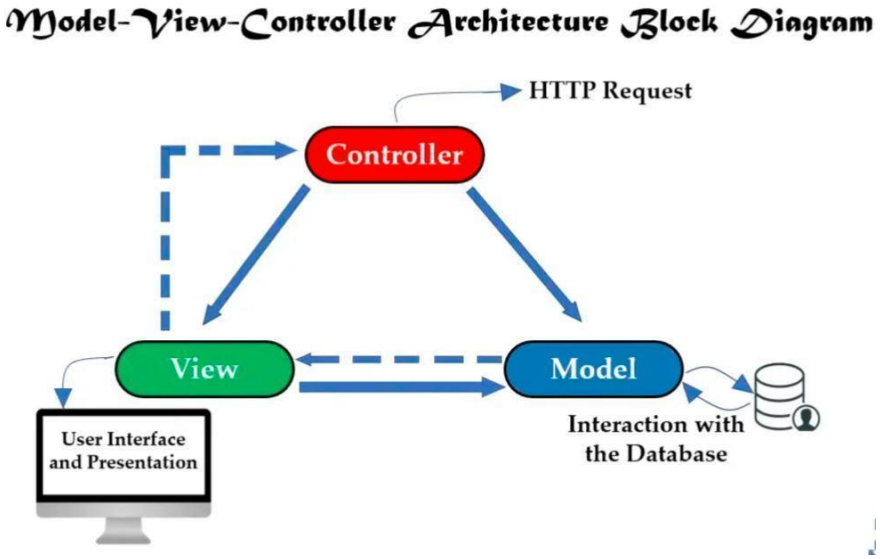

Figure 18: The structure of an ASP.NET MVC application with a Microsoft SQL Server database involves a separation of concerns between the model, view, and controller. The database stores the data, models represent it and provide a way to interact with it, controllers handle requests and render views, and the views render the user interface.

3.7.3. User-Interface Backend

Input by the user is taken in the form of bundle I.D. integer number. Output is displayed as a learning path containing two previously read bundles from the input bundle I.D. HTML-CSS implementation included in Appendix A provides an input text box with an input type specified as bundle I.D. integers. In addition, the output is programmed to display a table containing bundle I.D., previous_1, previous_2 and count separated by columns. Appendix A has an html code for exception handling and error queries.

Chapter 4: Results & Discussion

4.1. Exploratory Data Analysis Results

While exploring all subsets of EdNet, we came across standard user I.D.s that frequently interacted with question I.D.s from various bundle I.D.s. From 780000 total user data, 724000 are familiar users in five subsets of EdNet. Unique bundle I.D.s in all subsets of EdNet are 9534. As bundle I.D.s are composed of various questions, there are 13169 unique question I.D.s. Since the main focus of the research is to know the bundle I.D.s with the highest interactions in all data subsets, we gathered the following information in Table 4.

Feature	Frequency
Total Users in EdNet	780000
Familiar users in all subsets of EdNet	724000
Unique bundle I.D.s in all subsets of EdNet	9534
Unique question I.D.s in all subsets of EdNet	13169

Table 4: Data exploration results. Different features and their prevalence in the data are given.

4.2. Feature Engineering Results

New features engineered for studying bundle I.D.s and their distribution across users are given in Table 5. From 9534 total bundles, 8935 bundle I.D.s were responded to correctly by 60% of users. Hence the correctness percentage was recorded at 60% for 8935 bundles, while the rest of the bundle I.D.s with low correctness percentages were dropped. The excluded data constitute 599 bundle I.D.s only. Further investigation revealed that there are a maximum of five questions in bundles. Eight thousand one hundred bundle I.D.s have only one multiple choice question. Eight hundred eighty-five bundles contain three questions. Four hundred twenty-three bundles have four questions. One hundred twenty-six bundles have a maximum number of questions, i.e. five. Only 96 bundle I.D.s have two questions.

New Feature	Frequency
Correctness Percentage of Bundles	
Unique bundle I.D.s in all subsets of EdNet	9534
Bundles I.D.s with 60% & above correctness	8935
Question Count in Bundles	
Bundles with five questions	126
Bundles with four questions	423
Bundles with three questions	885
Bundles with two questions	96
Bundles with 1 question only	8100

Table 5: New features related to Bundle I.D.s. Correctness % gives accurate vs incorrect responses by users for questions within bundles. Question count gives the number of multiple-choice questions in a bundle ID.

4.2.1. All Bundles Performance Distribution

The bundle accuracy distribution was tested, and the histogram was plotted. Accuracy percentages are distributed from as low as 30% of the correctness of responses against question I.D.s to as high as 90% of correct responses. As depicted in Figure 19, the distribution is uniform, with low kurtosis (thin tails) and a mean of around 60. Most of the bundles lay in the accuracy range of 60-80%. Bundles above 60% were taken to continue the analysis further. Taking in bundles whose accuracy is lower than 60% meant that the people who studied them did not perform well enough to score well and make a cumulative accuracy of that bundle up to 60%. Thus, by

excluding those low-accuracy bundles, we could find the ones that helped achieve the best results.

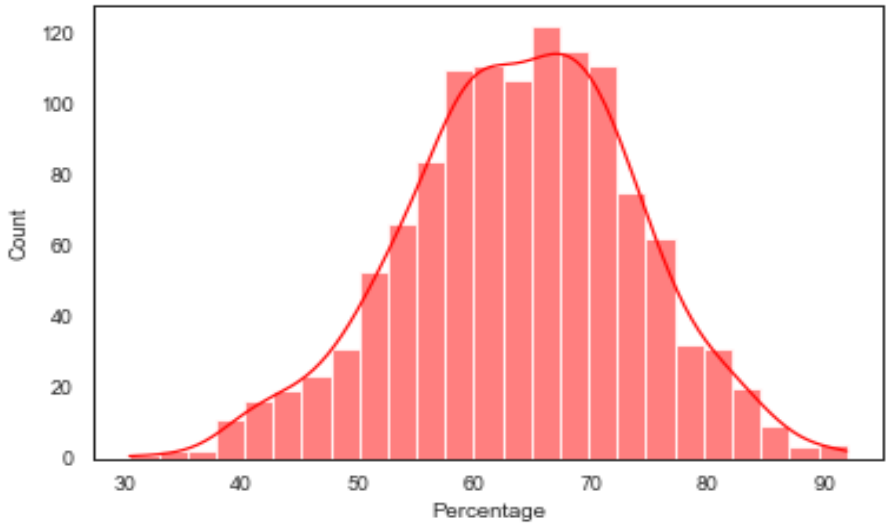

Figure 19: Accuracy distribution over all bundle counts on the X-axis versus the count of bundles against each accuracy percentage on the Y-axis

4.2.2. Bundle Performance of 60% & above

Figure 20 gives accuracy distribution over bundle counts with 60% & above correctness. After thresholding at 60% and above the accuracy percentage, a count of relevant bundles has been filtered out. This rightly skewed distribution of accuracy percentages indicates that a significant count of bundles lies in the 60% to 75% cut-off region. This includes bundles with all question sizes i.e. even bundles with 1 question. About 600 bundles were dropped who had lower performance scores.

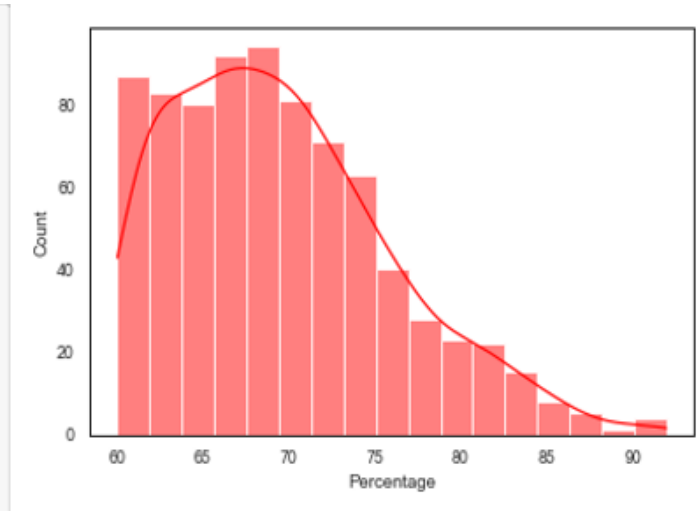

Figure 20: Accuracy distribution over bundle counts with 60% & above correctness

4.2.3. Accuracy Distribution on Question Count Level

In addition to bundle correctness, further investigation of correct and incorrect responses at the level of the questions within bundles provides more understanding of the content within learning objects.

4.2.3.1. Accuracy Distribution over All Bundles - All Questions

Figure 21 gives accuracy distribution (x-axis) over all bundles with the varying count of questions (y-axis). Bundles with five question counts have a correctness percentage of around 75%. Four question counts have a correct response percentage of 65-70%. Two question counts have been corrected by 90% of users. Bundles with only one question have around 55% accuracy. Each bundle's number of questions varies, reaching a maximum of five. Accuracy percentages against question counts are depicted in the distribution below. The histogram indicates that the bundles that contain five questions have around 75% accuracy. As can be seen, the distribution is platykurtic.

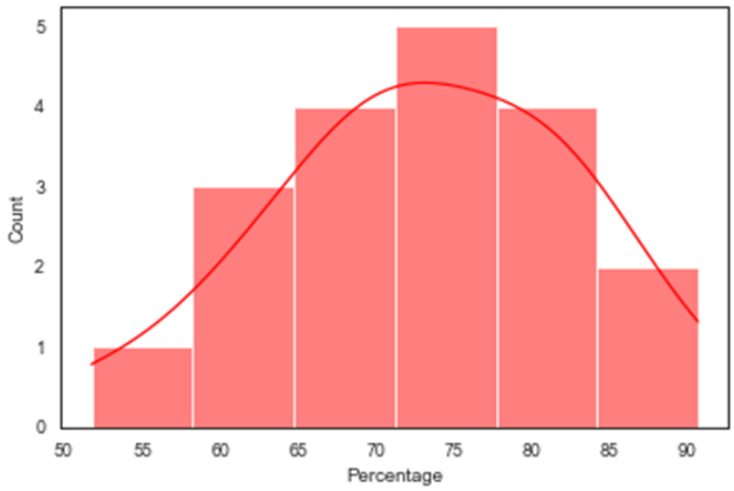

Figure 21: Accuracy distribution (x-axis) over all bundles with the varying count of questions (y-axis). Bundles with five question counts have a correctness percentage of around 75%. Four question counts have a correct response percentage of 65-70%. Two question counts have been corrected by 90% of users. One question count has an accuracy of 55%.

4.2.3.2. Accuracy Distribution over All Bundles - Question Count Three & Four

A moderate number of questions to evaluate bundle content efficacy should be three and more. Correctness percentages for questions having three and four-question counts are shown in Figure 22. The histogram depicts bimodal percentage accuracies

for bundle counts with four & three questions. The bundle count with either three or four questions of around 60% correctness is represented as two mods of the histogram.

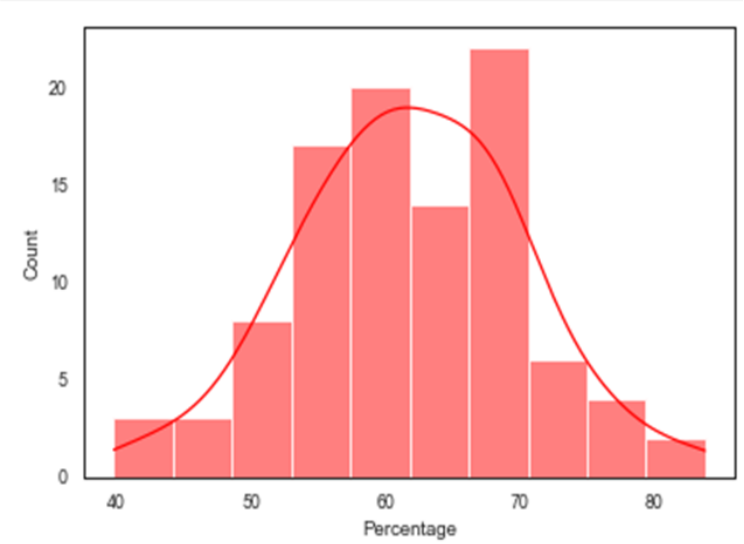

Figure 22: Bimodal histogram at three & four questions within several bundles (y-axis) & accuracies % (x-axis).

4.3. Visual Interpretation of Content-Based Recommender System

Graphical user interface for content-based recommender system in an interactive web-based tool for users who want to have a study plan in an informal education setting.

This interactive GUI displays recommendations in the form of lists & learning paths. Figure 23 shows the display containing a list of all bundle I.D.s with 60% correctness which is included in the database. Successive columns show previous_1 and previous_2 as additional bundle I.D.s to be studied with a specific bundle I.D.

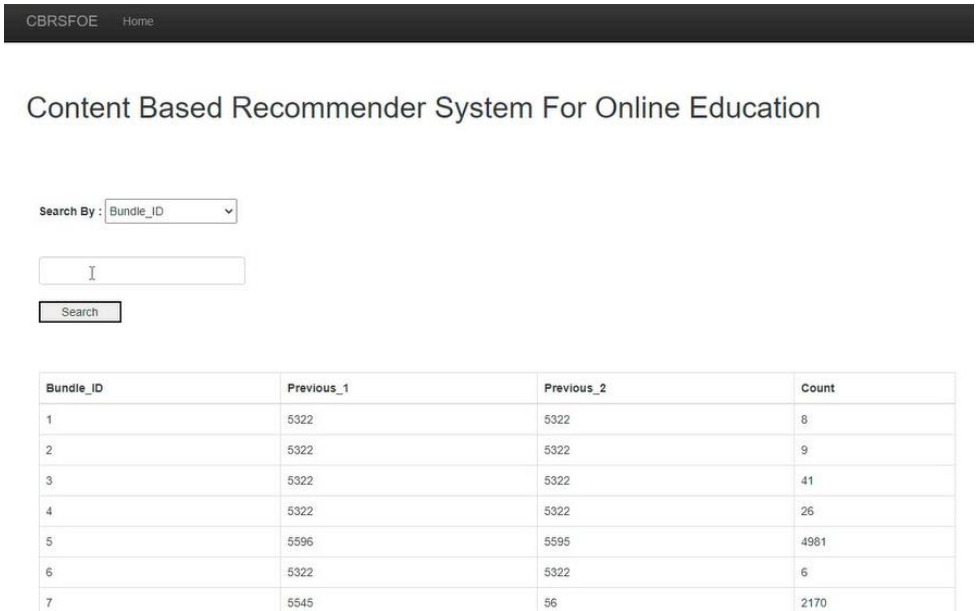

Figure 23: Content-based recommender system GUI. A list of all bundle I.D.s with 60% accuracy is included in the database and is displayed on the GUI for scrolling search. In addition, a query input feature is also added in the GUI where users can search for a specific bundle I.D. and, in the output, receives a learning path for studying that bundle with additional learning resources.

As shown in Figure 24(a), user input is a bundle I.D. in the form of an integer (234 in this case). The output displays bundle I.D., previous_1 & previous_2 bundle I.D.s as a learning path and counts this study pattern for the study path repeated in the database. This pattern shown in Figure 25 appears only once in the database. Figure 24(b) shows only one bundle ID (5322) to be studied with bundle 12. This study pattern is repeated nine times in the database.

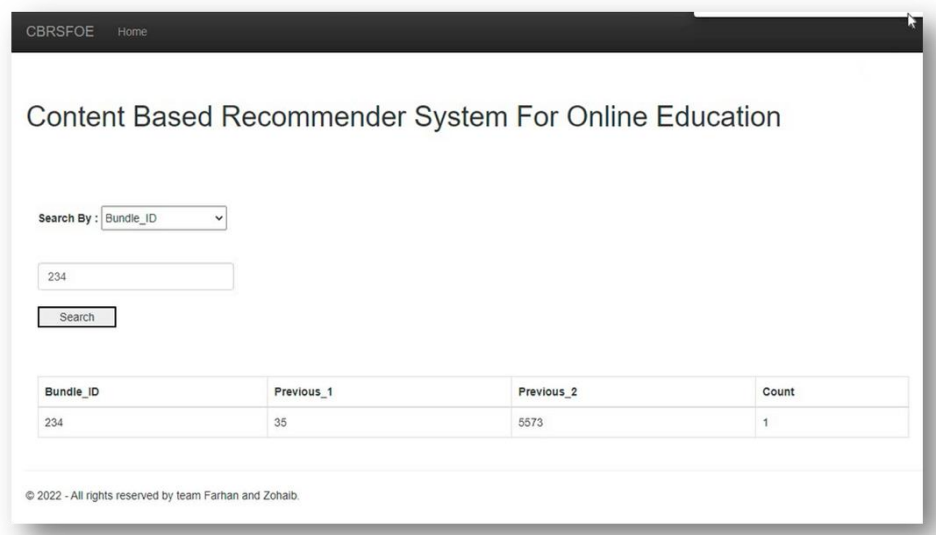

Figure 24(a): Content-based recommender system GUI. User input for bundle ID 234 gives a study pattern by recommending two additional bundles (learning objects) 35 and 5573 to study before bundle ID 234 for scoring 60% and above in online EdNet assessments.

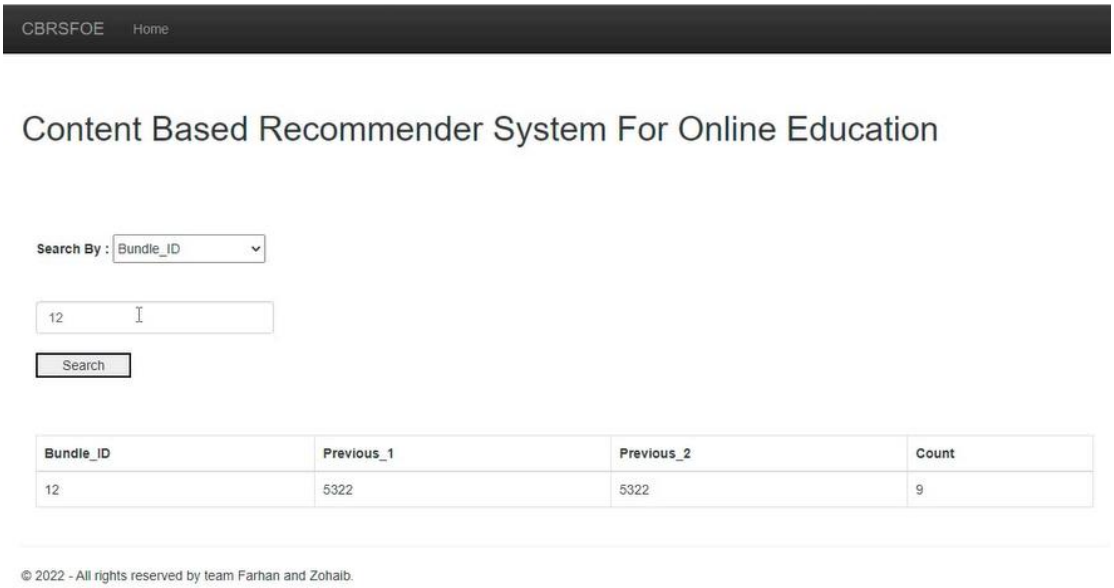

Figure 24(b): Content-based recommender system GUI. User input for bundle I.D. 12 gives a study pattern by recommending only one additional bundle (learning objects) 5322 to study before bundle ID 12 for scoring 60% and above in online EdNet assessments.

Figures 25 & 26 depict error types included to define user input. Users can only place bundle I.D. queries in the form of integers. Strings like 'Farhan' are taken as exceptions and are handled with a defined output ', No Match Data'. Also, any bundle I.D. beyond the I.D.s added to the database is considered out of the scope of the recommender tool. Hence no study plans are suggested as an output.

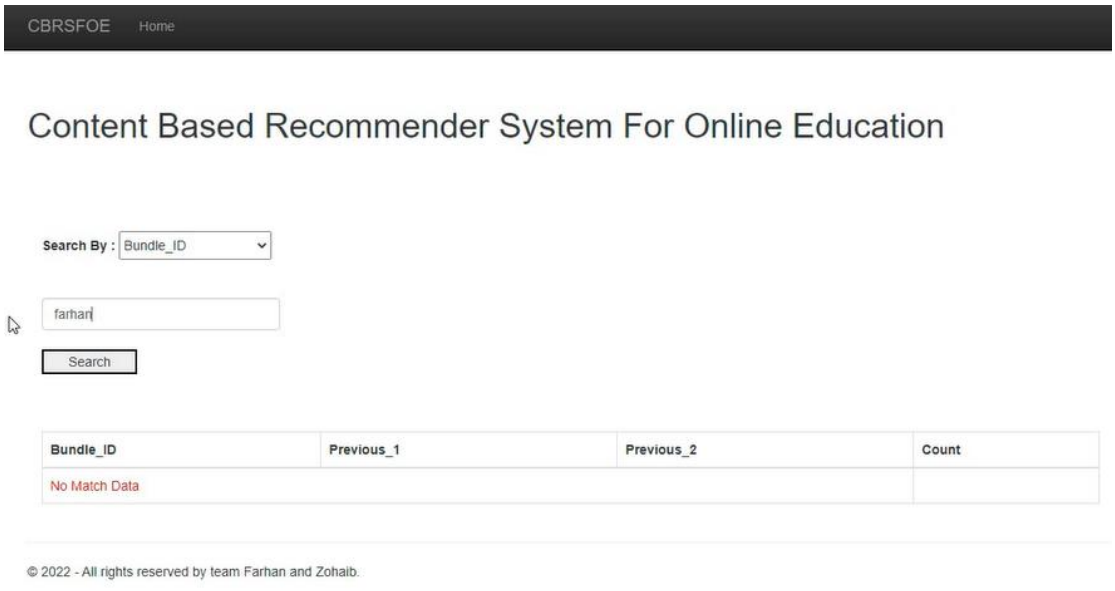

Figure 25: Content-based recommender system GUI. Input given as the string "Farhan" is not the correct search query. The output for wrong queries is shown as 'No Match Data'.

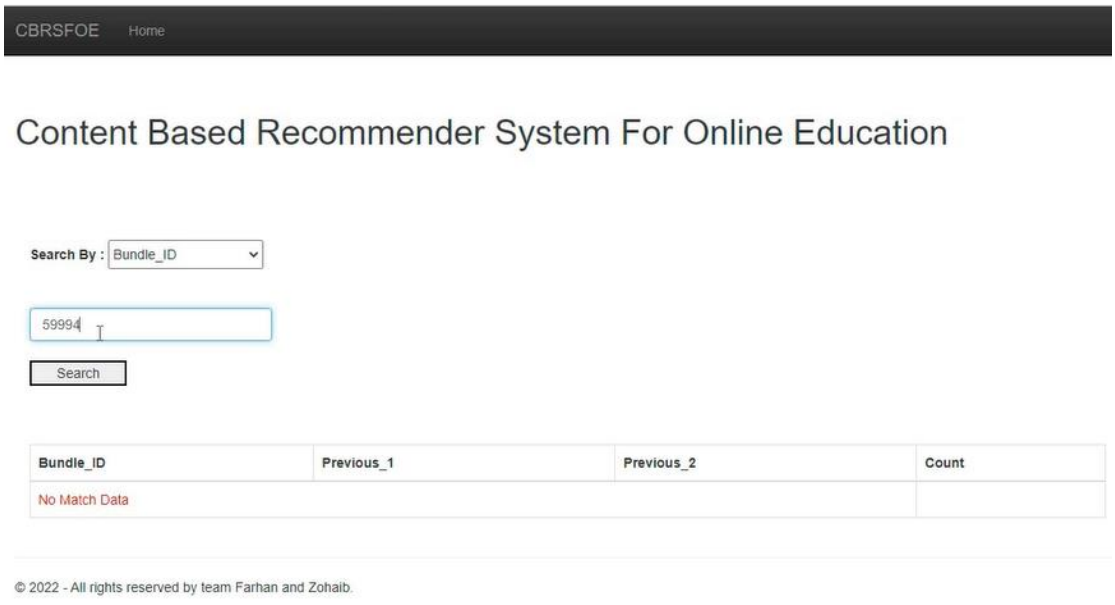

Figure 26: Content-based recommender system GUI. Search queries other than the given Bundle I.D.s in the database are not recognised. For example, 'No Match Data' is shown as an output for the '59994' input bundle I.D. query.

Hence, the content based recommender system proposed in this study is developed using implicit data on online education. By using EdNet learning management system records, users' usage data and tracking changes in user academic records and behaviour, a groupization algorithm has been designed to evaluate and score learning objects.

Groupization algorithm aims to broaden the recommendation results using the interests of the user’s communities. The algorithm designed here uses data insights on (i) Academic information; (ii) learners' behaviours and (iii) Contextual information.

In return the recommender system scores and recommends bundles that contain high quality knowledge components and enable students to get 60% and above in evaluation exams.

Content based recommender system helps both students and educators by recommending a feasible study plan for getting good grades. An easy to use GUI that has a simple procedure of user query input and information retrieval makes the content based recommender system a user-friendly application for online education platforms.

Chapter 5: Conclusion

E-learning environments accessible on the web are becoming more widespread in educational institutions. The fast growth of e-learning has altered traditional learning practices and created new challenges for teachers and students. For example, teachers find it increasingly difficult to direct students in selecting appropriate learning materials because of the abundance of online options. Meanwhile, students struggle to determine which learning materials suit their needs and preferences. In order to address these issues, this research focuses on interpreting learning objects (bundle characteristics) that were crucial in predicting the highest-performing solution rather than implementing multiple learner performance models. Bundles were investigated based on a correlation and grouping method. Such a grouping of learning resources provides students with a study plan in informal educational settings. These recommendations can make new users score equivalent to their top-performing peers. 'Content-based recommender system' gives a strong base for researchers to create ERS that can predict future student performance exceptionally accurately with the aid of this understanding of learning resources at the local level.

When concluding a series of prior question-answering histories or considering the most recent observation, BKT models often assume a first-order Markov chain to maintain the tractable Bayesian inference. However, their capacity to predict complicated dynamics in student learning behaviours is constrained by this assumption. Moreover, recently developed dynamic BKT models, like the DBN, frequently compromise prediction accuracy for computing efficiency since they are computationally intractable (Ribeiro, 2016). Our practical solution is computationally less expensive and provides higher accuracies based on local data insights of the essential variations in bundle features.

BKT models address a sequence prediction problem based on prior student learning. Factor analysis models do not consider the sequence of questions in which a student's replies are observed. For instance, the order in which students respond to two questions and their related answers has no bearing on the factor analysis models. However, factor analysis models can be improved to assess temporal elements of student learning by including additional temporal variables of learning activities. Our analytical solution compares student performances over various bundles and designs a

learning curve for students to score on average 60% by studying relevant bundles suggested by our automation tool.

Furthermore, the following conclusions are drawn from the findings concerning the research questions:

- The potential usefulness of a learner interest model that considers time in an e-learning material recommendation system for recommending e-learning resources is promising.
- The proposed e-learning material recommendation system is beneficial for situations where educators are inexperienced or not present.

5.1. Limitations

This work is a good foundation for designing ERS based on learning objects. However, certain limitations need to be addressed in future work.

1. Bundle I.D.s with less than 60% correctness should be considered to improve the content of the learning resources that are not contributing enough to users' performance.
2. Other than bundles, additional information on learning objects like a watchlist, reading history and purchased learning resources should also be considered to get deeper insights into user interactions with high-in-demand learning objects.

5.2. Future Perspectives

Our interest area is the supply of a learning curriculum (for example, an ordered list of subjects to study) for a particular subject depending on a student's knowledge status. One approach in this application field uses simulations to select a suitable curriculum of learning materials for a course to maximise student knowledge gain. The instructor uses a K.T. tool trained on the historical exercise data of the students.

Another approach evaluates the influence of each module (i.e., a collection of learning materials) on students' knowledge growth to determine how well a course structure performs in obtaining its targeted objectives.

Although existing state-of-the-art K.T. solutions have produced some encouraging results, several unresolved issues still present chances for future research.

Data Representation: Any K.T. tool performance is directly impacted by choice of data representation The proposed models or the accessible datasets, however, ignore some information in the description of a question, such as pictures and mathematical formulas

that might result in more useful embedding representations. K.T. tools frequently learn question and knowledge concept embedding representations from abstract formats like one-hot encoding. By asking the following questions, research directions (R.D.) become available.

- How should such data be represented for K.T. tasks?
- How can a dataset be produced for K.T. tasks that permit more accurate embedding representation learning?

Interactive Knowledge Tracing: Interactive techniques, driven by question-answering response behaviour, to understand better the dynamics of students' knowledge states are still unexplored. Most K.T. models adopt a passive approach of observing question-answering response history to estimate students' knowledge states. Interactive techniques are beneficial in cold start settings where an interactive approach can ask questions about various K.C.s to disclose students' knowledge states.

References

Abeysekera, L., & Dawson, P. (2015). Motivation and cognitive load in the flipped classroom: definition, rationale and a call for research. *Higher Education Research & Development, 34*(1), 1-14.

Adomavicius, Gediminas and Tuzhilin, Alexander. (2005). "Towards the next generation of recommender systems: a survey of the state-of-the-art and possible extensions." *IEEE Transactions on Knowledge and Data Engineering.*

Anderson, J. R. (1986). Cognitive Modelling and Intelligent Tutoring.

Bonwell, C. C., & Eison, J. A. (1991). *Active Learning: Creating Excitement in the Classroom.* ASHE-ERIC Higher Education Report No. 1, Washington, D.C.: The George Washington University, School Of Education and Human Development.

Bull, S. a. (2010). In Advances in. *Open learner models*, 301-322.

Choi, Y. L. (2020). Ednet: A large-scale hierarchical dataset in education. *International Conference on Artificial Intelligence in Education*, 69-73.

Chrysafiadi, K. a. (2013). Student modelling approaches A literature review for the last decade. *Expert Systems*, 4715–4729.

Cooperstein, S. E., & Weidinger, E. K. (2004). Beyond active learning: a constructivist approach. *Reference Services Review, 32*(2), 141-148.

Corbett, A. T. (1994). Knowledge tracing: Modelling the acquisition of procedural knowledge. *User modelling .253*–278.

Corbett., A. (2000). Cognitive mastery learning in the act programming tutor. *Adaptive User Interfaces.*

Gervet, T. K. (2020). When is Deep Learning the Best Approach to Knowledge Tracing? *JEDM| Journal of Educational Data Mining*, 31-54.

Hao Cen, K. R. (2006). Learning Factors Analysis - A General Method for Cognitive Model Evaluation and Improvement. *Intelligent Tutoring Systems, 8th International Conference*, 164-175.

HAO, K. (2019). China has started a grand experiment in A.I. education. It could reshape how the world learns. *MIT Technology Review*.

Hochreiter, S. B. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *In A field guide to dynamical recurrent neural*, 237-243.

J. Bobadilla, F. O. (2013). Recommender systems survey. *Knowledge-Based Systems*, 109–132.

Joeckel., A. H. (2017). Increasing the effectiveness of digital educational games: The effects of a learning instruction on students' learning, motivation and cognitive load. *Computers in Human Behavior*, 79-86.

John R Anderson, C. F. (1990). Cognitive modelling and intelligent tutoring. *Artificial Intelligence*, 7-49.

Kadiyala, M., & Crynes, B. L. (2000). A Review of Literature on Effectiveness of Use of Information Technology in Education. *Journal of Engineering Education*, *89*(2), 177-189.

Khajah, M. L. (2016). How Deep is Knowledge Tracing? *International Educational Data Mining Society.*

Koedinger, K. R. (2013). . New potentials for data-driven intelligent tutoring system development and optimisation. *A.I. Magazine* , 21-47.

Mierswa, I. W. (2006). Rapid prototyping for complex data mining tasks. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 935-940.

Mohammad Khajah, R. W. (2014). . Integrating latent-factor and knowledge tracing models to predict individual differences in learning. *Proceedings of the 7th International Conference on Educational Data Mining*, 99-106.

Pandey, S. a. (2019). A self-attentive model for knowledge tracing . *Proceedings of the 12th International Conference on Educational Data Mining*, 384-389.

Paszke, A. G. (2017). .Automatic differentiation in PyTorch. . *Autodiff Workshop at the 31st Conference on Neural Information Processing Systems*.

Pavlik Jr, P. C. (2009). Performance Factors Analysis--A New Alternative to Knowledge Tracing. *Online Submission*.

Pelanek, R. (2017). Bayesian knowledge tracing, logistic models, and beyond: an overview of learner modelling techniques. *User Modeling and User-Adapted Interaction*, 313-350.

Piech, C. B. (2015). Deep knowledge tracing. *In Advances in Neural Information Processing Systems* , 505–513.

Jannach, Dietmar and Fuchs, Markus, (2010) "Recommender Systems", *Springer*.

Reise., S. E. (2013). Item response theory. *Psychology Press*.

Resnick, Paul and Varian, Hal. (1997) "Recommender systems." *Communications of the ACM*.

Ribeiro, M. S. (2016). "Why should I trust you?" Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference*, 1135-1144.

Ritter, S. Y. (2016). How mastery learning works at scale. *ACM Conference on Learning*, 71-79.

Rose, C. P. (2019). Explanatory learner models: Why machine learning (alone) is not. *British Journal of Educational Technology* , 2943-2958.

Ryan Shaun Joazeiro de Baker, A. T. (2008). More Accurate Student Modeling through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. *Intelligent Tutoring Systems , 9th International Conference*, 406–415.

Sloan, W. M. (2012, July). What Is the Purpose of Education? *Education Update: Quality Feedback: What It Is and How to Give It, 54*(7).

Van Der Linden, W. J. (2013). Handbook of modern item response theory. *Springer Science & Business Media*.

Vanlehn, K. (2011). The relative effectiveness of human tutoring,. *Educational Psychologist*, 197-221.

Wang., G. A. (2019). Knowledge Tracing with Sequential Key-Value Memory Networks. *42nd International Conference on Research and Development in Information Retrieval*, 175–184.

Wenbin Gan, Y. S. (2020). Modelling learner's dynamic knowledge construction procedure and cognitive item difficulty for knowledge tracing. *Applied Intelligence*, 3894–3912.

Xiong, X. Z. (2016). Going deeper with deep knowledge tracing. *Proceedings of the 9th International Conference on Educational Data Mining*, 545–550.

Youngduck Choi, Y. L. (2020). Towards an Appropriate Query, Key, and Value Computation for Knowledge Tracing. . *Seventh ACM Conference on Learning*, 341–344.

Yun Huang, J. P.-B. (2014). General features in knowledge tracing to model multiple subskills, temporal item response theory, and expert knowledge. *n Proceedings of the 7th International Conference on Educational Data Mining*, 84-91.

Yuying Chen, Q. L.-z. (2017). Tracking Knowledge Proficiency of Students with Educational Priors. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management.*, 989–998.

Appendix

Appendix A

Supplementary codes:

```
output_dict1 = {"bundle_id": [], "question_id": [], "correct": [], "incorrect": []}
output_dict2 = {"bundle_id": [], "question_count": [], "correct": [], "incorrect": []}
for b_id in qDF['bundle_id'].unique():
    output_dict2['bundle_id'].append(b_id)
    output_dict2['question_count'].append(0)
    output_dict2['correct'].append(0)
    output_dict2['incorrect'].append(0)
    qDF_filtered_by_bundle = qDF.loc[qDF["bundle_id"]==b_id]
    item_counts = rDF['item_id'].value_counts()
    for _, qDF_row in qDF_filtered_by_bundle.iterrows():
        question = qDF_row['question_id']
        rDF_filtered_by_question = rDF.loc[rDF["item_id"]==question]
        original_answer = qDF_row['correct_answer']
        output_dict2['question_count'][-1] += 1
        for _, rDF_row in rDF_filtered_by_question.iterrows():
            user_answer = rDF_row['user_answer']
            if question not in output_dict1['question_id']:
                output_dict1['bundle_id'].append(b_id)
                output_dict1['question_id'].append(question)
                output_dict1['correct'].append(0)
                output_dict1['incorrect'].append(0)
            if user_answer == original_answer:
                output_dict1['correct'][output_dict1['question_id'].index(question)] += 1
                output_dict2['correct'][-1] += 1
            else:
                output_dict1['incorrect'][output_dict1['question_id'].index(question)] += 1
                output_dict2['incorrect'][-1] += 1

output1 = pd.DataFrame(data=output_dict1)
output2 = pd.DataFrame(data=output_dict2)
```

Figure 15: Python implementation of feature engineering algorithm. Output_dict 1 & output_dict 2 are two dictionaries that store values of new features like correct and incorrect responses; question count in each bundle I.D. After appending values from all CSV files, these dictionaries are stored as dataframes.

```
@Html.TextBox("Search", null, new (@class="form-control"))

<input style="width: 100px;" type="submit" id="SearchBtn" value="Search" />

@<div class="form-horizontal">

<div class="form-group">

<label class="col-md-2 control-label">Search</label>

<div class="col-md-6">

<input type="text" name="SearchBy" class="form-control" placeholder="Search By ID" style="max-
width:10ek"/>

</div>

<div class="col-md-2">

<button class="btn btn-primary" type="submit">Search</button>

</div>

</div>

</div>”@

<table class="table table-bordered">

<thread>

<tr>

<th>Bundle ID</th>

<th>Previous 1</th>

<th>Previous 2</th>

<th>Count</th>
```

Figure 21(a): HTML - CSS code for the design layout of the content-based recommender tool GUI. Search text box, input type specified by bundle I.D. integer format, study plan list in the form of a table design containing column headers labelled as bundle iD, previous_1, previous_2 and count.

```
<script>
$(document).ready(function () {
$("#SearchBtn").click(function () {
var SearchBy $("#SearchBy").val();
var SearchValue = $("#Search").val();
var SetData $("#DataSearching");
SetData.html("");
$.ajax({
type: "post",
url: "/Home/GetSearchingData?SearchBy-" + SearchBy + "&SearchValue=" + SearchValue,
contentType: "html",
success: function (result) {
if (result.length = 0) {
SetData.append('<tr style="color:red"><td colspan="3">No Match Data</td></tr>')
}
else {
$.each(result, function (index, value) {
var Data "<tr>" +
"<td>" + value.stuId + "</td>" +
"<td>" + value. StuName + "</td>" +
"<td>" + value. Semester + "</td>" +
*<td>" + value.count + "</td>" +
"</tr>";
SetData.append(Data);
});
```

Figure 21(b): HTML - CSS code for the design layout of the content-based recommender tool GUI. Search button design code, input query and output display colour and content code are shown.