

UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

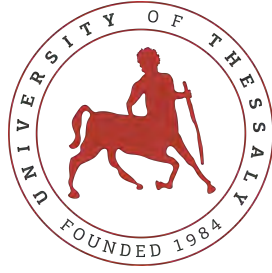
**Implementation of a smart MOOC using learning analytics and
recommendation algorithms**

Diploma Thesis

Kanellopoulou Aikaterini

Supervisor: Daskalopoulou Aspasia

Volos 2020



UNIVERSITY OF THESSALY
SCHOOL OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

**Implementation of a smart MOOC using learning analytics and
recommendation algorithms**

Diploma Thesis

Kanellopoulou Aikaterini

Supervisor: Daskalopoulou Aspasia

Volos 2020



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

**Υλοποίηση έξυπνης διαδικτυακής πλατφόρμας μάθησης (smart
MOOC) με τη χρήση learning analytics**

Διπλωματική Εργασία

Κανελλοπούλου Αικατερίνη

Επιβλέπουσα: Δασκαλοπούλου Ασπασία

Βόλος 2020

Approved by the Examination Committee:

Supervisor **Daskalopoulou Aspasia**

Assistant Professor, Department of Electrical and Computer Engineering, University of Thessaly

Member **Tsalapata Hariklia**

Laboratory Teaching Staff, Department of Electrical and Computer Engineering, University of Thessaly

Member **Vassilakopoulos Michael**

Associate Professor, Department of Electrical and Computer Engineering, University of Thessaly

Date of approval: 20-9-2020

Acknowledgements

First and foremost, I would like to thank my professor Hariklia Tsalapata for her guidance and counseling. She supported me and helped me with my thoughts and work process throughout this dissertation. I would also like to thank my supervisor Michael Vasilakopoulos and my professor Aspasia Daskalopoulou for the confidence they showed me to take on this subject.

Finally, I would like to thank my family and friends who were always there with their encouragement throughout the years of studies in the University of Thessaly.

DISCLAIMER ON ACADEMIC ETHICS AND INTELLECTUAL PROPERTY RIGHTS

«Being fully aware of the implications of copyright laws, I expressly state that this diploma thesis, as well as the electronic files and source codes developed or modified in the course of this thesis, are solely the product of my personal work and do not infringe any rights of intellectual property, personality and personal data of third parties, do not contain work / contributions of third parties for which the permission of the authors / beneficiaries is required and are not a product of partial or complete plagiarism, while the sources used are limited to the bibliographic references only and meet the rules of scientific citing. The points where I have used ideas, text, files and / or sources of other authors are clearly mentioned in the text with the appropriate citation and the relevant complete reference is included in the bibliographic references section. I fully, individually and personally undertake all legal and administrative consequences that may arise in the event that it is proven, in the course of time, that this thesis or part of it does not belong to me because it is a product of plagiarism».

The declarant

Kanellopoulou Aikaterini

15-9-2020

Abstract

Nowadays, e-learning environments have become a great addition to the traditional educational field. Their relatively low-cost and highly-adaptive nature has enabled more and more people around the world to start or continue their studies. MOOC (massive online open courses) platforms are an essential element in this transition to online learning and an evolving software domain. This Thesis submits a proposal for combining this area with the developing area of learning analytics, a powerful instrument to understand and enhance students' performance via the collection and analysis of vast data. The tool to achieve that merge is the use of recommender system technology.

To elaborate, this Thesis initially investigates the possible data selection for forming a predictive model for the user's behavior that will be beneficial for him. Then, this Thesis proceeds to the assessment of the available recommendation techniques to choose the most suitable for the given data. This assessment leads to the software development of the recommendation system. Alongside comes the web platform development for an interface that integrates the recommender. After configurations and optimizations, the DCG (discounted cumulative gain) algorithm is used to judge the efficiency of our results. Finally, the system is considered to result in adequate recommendations according to the preference profile of the user.

Keywords

MOOC, Learning Analytics, Collaborative Filtering, Recommender Systems, Personalization

Περίληψη

Τη σημερινή εποχή τα περιβάλλοντα διαδικτυακής μάθησης έχουν ενισχύσει σημαντικά το τομέα της καθιερωμένης εκπαίδευσης. Το σχετικά χαμηλό κόστος τους και η προσαρμοστική φύση τους έχει δώσει την ευκαιρία σε πλήθος κόσμου από όλο τον κόσμο να ξεκινήσει ή να συνεχίσει τις σπουδές του και τη μάθησή του. Οι πλατφόρμες MOOC (massive online open courses), δηλαδή οι πλατφόρμες που παρέχουν μαζικά διαδικτυακά ανοιχτά μαθήματα στο κοινό, είναι ένα δομικό στοιχείο της μεταβολής στη διαδικτυακή μάθηση, ενώ παράλληλα είναι ένας τομέας συνεχούς εξέλιξης. Η παρούσα Διπλωματική εργασία θέτει μια πρόταση συνδυασμού του παραπάνω αντικειμένου με το πεδίο των Learning Analytics, εκπαιδευτικών αναλύσεων, ένα τομέα-εργαλείο για την κατανόηση και την ενίσχυση των μαθητικών αποδόσεων, μέσω της συλλογής και της ανάλυσης μεγάλου όγκου δεδομένων. Το μέσο για να πραγματοποιηθεί αυτή η σύνθεση είναι η χρήση ενός συστήματος συστάσεων (Recommender System).

Πιο συγκεκριμένα, η παρούσα Διπλωματική αρχικά ερευνά την κατάλληλη διαλογή δεδομένων, τα οποία μπορούν να χρησιμοποιηθούν για να δημιουργηθεί ένα μοντέλο πρόβλεψης της συμπεριφοράς του χρήστη που θα είναι ταυτόχρονα ωφέλιμο για εκείνον. Στην συνέχεια, προχωρά στην αξιολόγηση των υπάρχοντων τεχνικών για συστάσεις ώστε να επιλεγεί αυτή που ενδεικνυται για τα δεδομένα. Η παραπάνω αξιολόγηση οδηγεί στη ανάπτυξη του λογισμικού του συστήματος συστάσεων, ενώ παράλληλα, πραγματοποιείται και η ανάπτυξη λογισμικού της διαδικτυακής πλατφόρμας που θα ενσωματώσει το προαναφερόμενο σύστημα. Έπειτα από ρυθμίσεις και βελτιστοποιήσεις, χρησιμοποιείται ο αλγόριθμος μειωμένου αθροιστικού όφελους (DCG algorithm) για να κριθεί η αποδοτικότητα των αποτελεσμάτων. Τελικά, το σύστημα θεωρείται να αποδίδει επαρκή αποτελέσματα σε σχέση με την συνολική εικόνα και τις προτιμήσεις του χρήστη.

Table of contents

Acknowledgements	v
Abstract	vii
Περίληψη	viii
Table of contents	i
List of figures	i
Abbreviations	iii
1 Introduction	1
1.1 Learning analytics and recommendation systems	1
1.2 Thesis structure	2
2 Model approach	3
2.1 Introduction	3
2.2 Related work	3
2.3 Personalization parameters and student satisfaction	4
2.4 Features and data generator	5
3 Recommender Systems	6
3.1 Introduction	6
3.2 Categories	6
3.3 Used method analysis	7

4	Platforms and software tools	10
4.1	Introduction	10
4.2	The overall system	10
4.3	Database details	11
4.4	Recommendation system software	12
4.5	Web application software	14
4.5.1	User interface	14
5	Results and evaluation	17
5.1	Introduction	17
5.2	The DCG algorithm	18
5.3	Evaluation methodology	19
5.4	Results overview	22
6	Summary	23
6.1	Conclusion	23
6.2	Future work	24
	Bibliography	25
	Appendix	
	Programming code	28
1	Recommender system programming code	28
1.1	Data generator code	28
1.2	TensorRec code	30
1.3	Recommender program output	32
2	User interface code	32
2.1	Client side	32
2.2	Server side	34

List of figures

2.1	Extract from courses data set.	5
2.2	Extract from users' attendance data set.	5
3.1	Logic of user-based CF.	8
3.2	Explanatory figure of the CF process.	8
3.3	Logic of Content Based model.	9
3.4	Explanatory figure of the CB process.	9
3.5	Logic of the Hybrid model.	9
4.1	System context figure.	11
4.2	Courses' set example.	12
4.3	User's information set example.	12
4.4	User's Attendance set example.	12
4.5	Logic of the TensorRec framework.	13
4.6	Use cases diagram.	15
4.7	Login page.	15
4.8	Table of recommended courses.	16
4.9	Rating functionality.	16
4.10	Progress score submission.	16
5.1	DCG results for the user 456.	20
5.2	DCG results for the user 645.	20
5.3	DCG results for the user 3586.	20
5.4	DCG results for the user 7000.	20
5.5	DCG results for the user 8790.	21
5.6	DCG results for the user 13564.	21

5.7	DCG results for the user 18569.	21
.1	Console output of the recommendation procedure.	32

Abbreviations

DB	Database
DCG	Discounted cumulative gain
JS	JavaScript
MERN	Mongo, Exprss, React, NodeJS
MOOC	Massive open online course
RS	Recommender system

Chapter 1

Introduction

A smart MOOC platform falls into the category of e-learning platforms. It is a technological tool with educational aspects, taking advantage of various multimedia. Even though the majority of current online learning systems are closed learning environments where courses and learning materials have a static nature [1], there is a vast turn in analytics tools to change this status quo. Nowadays, several fields, including teaching academics and data scientists, are combined to provide adaptive personalized learning environments to contribute to the performance and comfort of the learner.

1.1 Learning analytics and recommendation systems

Learning Analytics is the cycle of measurement, collection, analysis, and reporting of data about learners and their acting environment, for purposes of understanding and optimizing learning and the environments in which it occurs. [2] It benefits the learner by offering more timely, precise, actionable feedback, and it facilitates adaptation to the individual's needs and preferences. [3] Learning Analytics achieves all of the above by utilizing a variety of tools involving primarily data mining as well as recommender system techniques.

Burke defines a recommender system for learning as any system that produces personalized recommendations as output or has the effect of guiding the learner in a personalized way to interesting or useful learning resources in a large space of possible options [4]. This Thesis will try to answer the fundamental steps of implementing a system like that. The following are the problems occurring in this course of actions that are to be answered are [5]:

- Dataset Selection

- Offline Data Study
- Performance Evaluation
- Deployment of the Recommender System

1.2 Thesis structure

The remainder of this Thesis is organized as follows:

Chapter 2: gives an outline of our model approach and the theoretical background for drawing conclusions about the personalization parameters used. This section defines the organization of the wanted platform.

Chapter 3: gives an analytic overview of the recommender systems' theory and techniques and then specifies the formation of this thesis' hybrid system.

Chapter 4: focuses on the design and the development of each component of the final platform. This section provides technical details about the user interface, the database, and the recommender system.

Chapter 5: contains the evaluation process for the resulting recommendations and the measurements of effectiveness.

Chapter 6: presents the conclusions for the overall system.

Chapter 2

Model approach

2.1 Introduction

In the bigger picture, the wanted outcome of this Thesis is an adaptive e-learning environment that aims to improve students' performance and satisfaction. It is stated that for these particular improvements, studies suggest the adjustment of learning methods based on individual profiles and information [6]. This acquired intelligence can be exploited by two leading techniques: Learning Analytics and Educational Data Mining. The first one aims at providing insights to teachers and learners, and the second one focuses on the automatic adaptation of the learning process with not necessarily any human interference.. [7]

Practically for this Thesis, what is mentioned above translates to the implementation of a unified system, representing a downsized, limited MOOC platform where the user is provided with a list of recommended courses to choose from. More specifically, the system gathers students' ratings and progress-statuses from currently taken courses and merges this information with courses' metadata to offer the mentioned recommendations. This proposed model can be considered to be a gray area between Learning Analytics and educational data mining because it offers adaptive options and guidance to a learner, and parallelly it functions as an external automated process.

2.2 Related work

There have been other approaches to connecting MOOC and RS technology in the past. To begin with, there is a group of studies that utilizes an interface to accumulate learners' in-

formation and search for external sources of material. For example, Symeonidis and Malakoudis [8] proposed a system that recommends courses from different MOOC platforms. A similar plan was implied by Bousbahi and Chorfi [4], and Ya Tang and McCalla suggested a system that could recommend helpful material from the open web [1]. Furthermore, studies that did focus on the course recommendation field, differentiate from this study due to their recommending methods. Gulzar, Leema, Deepak [9] utilized the ontology method, and Jie Lu utilized fuzzy matching rules to produce recommendations. [10].

Even though our approach utilizes the common technique of collaborative filtering, it can be distinguished from the previous discussion because of its integrated implementation with the interface and the explicit submissions of the users. This interconnection with the user's activity targets the likeability and satisfaction enhancement of the student since it is stated that these elements are major factors of the usability and success of a course platform. [11] [12]

2.3 Personalization parameters and student satisfaction

As outlined above, the recommendation process takes advantage of the following criteria: the individual's ratings, the individual's score, and the content types of courses' material, e.g. video, audio means, etc. This selection of attributes for the system is an effort to approach a personalization strategy of e-learning scenarios suggested by previous studies [13]. It is stated that among other parameters, learner's level of knowledge, media preference (content types), and progress on tasks (course's score) are key values to implement a learning strategy that is adequate and satisfactory for the student. Moreover, satisfactory user experience is also considered to be one of the key factors of usability and efficiency. That is why there is an attempt to amplify this factor's value by providing feedback from the user in the form of ratings. Furthermore, these values are in coordination with the shared schemas of observation, submission, collaboration, and feedback, that are suggested for MOOC related information [14].

2.4 Features and data generator

After research, there was no adequate data set found to satisfy these requirements. The following solution was to generate dummy data that corresponds to the model. Python provided randomization functions for the dummy data resulting to the creation of two data sets.

1. Courses data set: with the respective fields of courseId, title, course field, professor involvement, types of content, and difficulty. The model utilized the courseId field as the primary identification key for courses and the types of content field for recommendations. There were a total of one thousand entries in this set.

courseid	title	coursefield	professor_involvement	types_of_content	difficulty
0	title	history	high	scripts	high
1	title	economics	low	exercises videos scripts audios	high
2	title	economics	low	videos	medium

Figure 2.1: Extract from courses data set.

2. Users Attendance data set: with the respective fields of userId, courseId, score, and ratings. These entries represent the enrollment of a student to a course. The model utilizes the courseId and the userId as a unique key for this relationship and the other two fields are used for the recommendation procedure. There were a total of twenty thousand different userId fields, representing different users, as an effort to approximate the number of active students in a greek university.

userid	courseid	score	ratings
0	298	53	2

Figure 2.2: Extract from users' attendance data set.

Chapter 3

Recommender Systems

3.1 Introduction

Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user [15]. They are a subcategory of information filtering systems meaning that they are capable of processing a stream of information and result in a targeted result of actions and/or feedback information without the need of human intervention. [16] Their integration in contemporary platforms has changed the needs and expectations of entertainment and consuming habits online. The use of RSs today is continually growing, mainly in the business field but also academically and recreationally. There are several variations in respect to the algorithmic approach of the issue, starting with the input data information all the way to efficiency measurements. For example, consumers and users might be giving explicit and declared ratings to a system or they may not even know their implicit data is saved to be used. Typical explicit data that are asked for are ratings or rankings. On the other hand, typical implicit data are closer to user's navigation behavior such as viewing time or mouse clicks counting. [17]

3.2 Categories

The fundamental goal of a RS is to leverage knowledge with an appropriate algorithm to generate various utility predictions and hence recommendations [18]. The classification of RSs is based upon this knowledge, essentially meaning their input information. Specifically, the prominent categorization is the following [19] [20]:

1. Content Based: In this case, the RS focuses on item attributes such as descriptive keywords. The goal of the algorithm is to find similarity between the existing items. Here the outcome is recommending items similar to what the user liked in the past exploiting only the interested user's history.
2. Knowledge Based: Here, an RS provides recommendations after computing a specific model for the user, based on his own needs and preferences. This model is paired with the available items (model solutions) to detect the most successful match.
3. Utility Based: In this less utilized proposition, the information submitted is related to an item's benefits, for instance, the availability of a product.
4. Demographic Base: In this condition, the recommendations are dependent on the similarity with the user's demographic group.
5. The leading scenario of the recommendation world is CF. In simple words, this technique makes recommendations of items to a user derived from similar users' history. If one user has resembling history with another, one can predict that their future choices - ratings will align. More to follow.
6. Hybrid model: It is not rare to combine some of the aforementioned techniques to achieve better results and overcome individual problems.

For this particular project, the Hybrid model is made of use, utilizing the frequent combination of Collaborative Filtering and the Content-Based model. On top of its ability to eliminate weaknesses between those two, for instance, resolving the cold start problem of the CF method to a certain extent [21], Hybrid models have promise better recommendations for plenty of projects in the past.

3.3 Used method analysis

More meticulously, this Thesis approaches a weighted hybrid model, in which the main focus of the technique remains the CF prediction of the users' ratings, but there is also a weighted contribution from some items' metadata. In particular, the memory-based CF technique utilizes explicit users' ratings, gained from previous history and knowledge, in the form of positive ordinal values. These accumulated values (the input data) are taken advantage of

to form classes of similar users, meaning if several users liked the same content before they are put in the same group because theoretically they are expected to have converging tastes either in the future or in areas of unknown ratings. Consequently, whether there is a missing rating or there is a need for a rating prediction, the system exploits those groups' ratings by calculating a value for an item's missing value. The fact that this system focuses on the information of the user and not the information of the item, renders it a user-based RS. The following step is the ranking procedure where the known and the predicted values are compared and numerically ranked in order to separate the highest values which depict the actual recommendations.

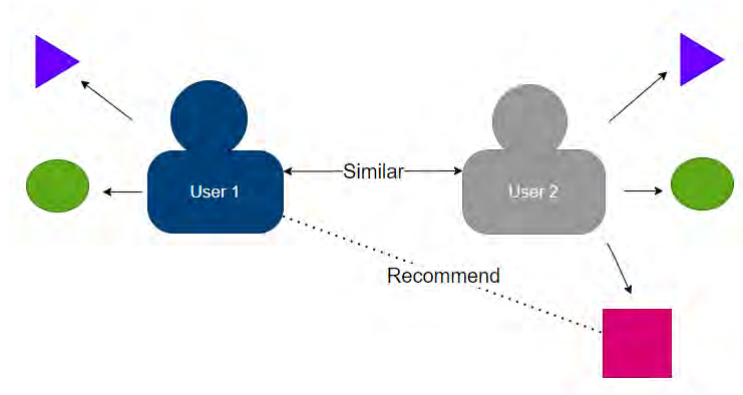


Figure 3.1: Logic of user-based CF.

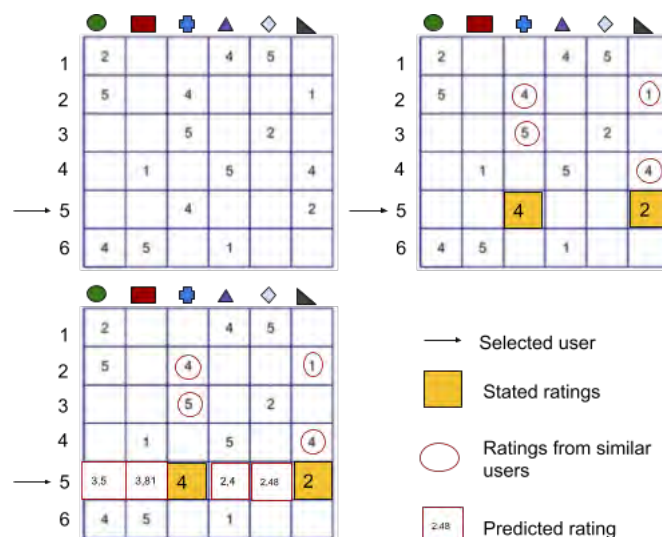


Figure 3.2: Explanatory figure of the CF process.

On the other hand, regarding the Content-Based part of the system, the established theory declares that the recommendations are not related to others' interactions but solemnly

to the past preference history of the interested user. In opposition to CF, according to this technique, the similarity correlation is between items' features. Firstly, the system samples items with high values from the user's interaction pool to compare them with the rest of the available items. For this comparison, the items are remodeled as a set of keywords. Consequently, the final similarity measurement hands out the closest sets, which represent the recommendations.

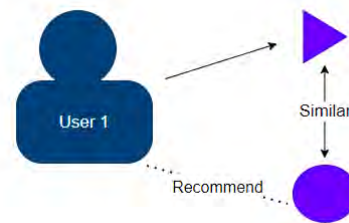


Figure 3.3: Logic of Content Based model.

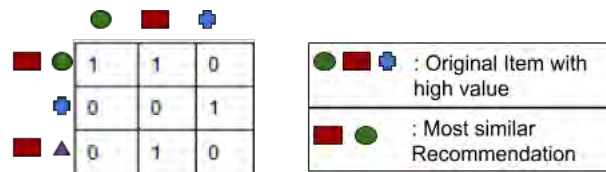


Figure 3.4: Explanatory figure of the CB process.

As mentioned before, ultimately, the two recommendation groups are fused to a single hybrid algorithm. This fusion is a procedure of weighted averaging of the models, which remains a black box matter for the programmer. Further information in chapter 4.

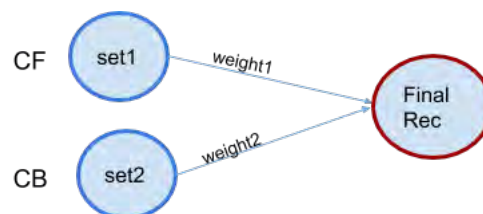


Figure 3.5: Logic of the Hybrid model.

Chapter 4

Platforms and software tools

4.1 Introduction

A primary objective for this Thesis was the integral implementation of a functional platform from the user's perspective. Therefore, there was thorough research of available tools for the three components of the system, including the web platform, the database, and the recommendation system. The goal of this chapter is to elaborate on the applied software tools.

4.2 The overall system

As noted, the overall system consists of three components:

1. The Web Platform
2. The Database
3. The Recommender System.

The linking of these components intends to complete the use cases triggered by the main actor of the system: the interested student. A use case is a process or an action that the actor expects to fulfill via the indicated system. Thereby, to comprehend the functionality of the implemented structure, it is necessary to study the responses of the system to these use cases.

Scenarios, like logging in, obtaining a recommendation list, viewing the material of a course, require, first of all, the communication between the client-side and the server-side of the web application. The user handles the client-side from the browser interface and prompts requests, which expect responses with valuable information from the server. To achieve this

back and forth communication the two sides utilize a REST API protocol ensuring quick and safe data transmission. In reality, the only side getting requests is the side of the server, which also connects to the system's database. To respond to the client's requests, in most instances the server needs to retrieve data information from the DB consequently, the connection between them remains always open.

In parallel, there is a separate program running the recommender system. After obtaining the required information about content types of the courses and users' ratings from the DB, it proceeds to its computations and finally updates the database in terms of the new recommended courses. This connection does not have to retain a stable connection since the program will run only a few times a day because of its demand for major changes in the current data. An RS of this size will readjust its results after hundreds of modifications on the available data, so logically one cannot expect this to happen very often.

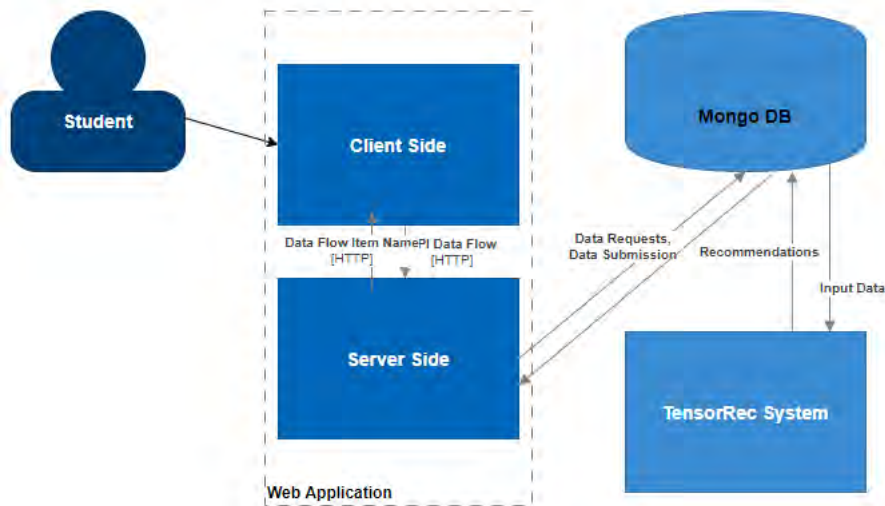


Figure 4.1: System context figure.

4.3 Database details

The database is a core component of the system. Not only, both the recommender and the application necessitate data retrieval to function properly, but also, the DB is the common link between the other two components. The wanted retrievals, depending on their content, are intertwined with three sets of information that are stored. In MongoDB theory, these sets are called collections.

- Firstly, all the information about the available courses is stored in the courses set, for example, the ids of the courses. It needs to be initialized by an external actor such as the professor of the course.
- Secondly, the users' info set where the personal details of each user are stored, for example, his username and his recommended courses. This set initializes automatically when a user signs up and is modified when a user has updates regarding his courses.
- And thirdly, the users' attendance set records the relation between users and taken courses. The included information here is the ratings and the scores needed for the RS system. Its initialization takes place also automatically when a user enrolls in a new course and updates due to the user's activity.

As for the technical aspect of this component, the database program chosen is MongoDB, a NoSQL database that communicates via JSON format documents with the other two components. In MongoDB language, the above collections are defined by declared schemas and have the following appearance when documented:

```
{ "_id" : ObjectId("5f65b31ca30faa0703e7098"), "courseid" : 14, "title" : "title", "coursefield" : "arts", "professor_involvement" : "medium", "types_of_content" : "scripts|extraliterature|exercises", "difficulty" : "medium" }
```

Figure 4.2: Courses' set example.

```
{ "_id" : ObjectId("5f65b4c86f1360249163f35d"), "courses" : [ 15, 54, 80, 49, 249, 65, 105, 13, 281, 25, 101, 146, 251, 117, 188, 82, 166, 46, 7, 225, 212, 107, 258, 113, 297, 3, 274, 77, 217, 22, 256, 276, 79, 57, 67, 21, 41, 201, 126, 262, 183, 18, 296, 144, 4, 176, 163, 25, 118, 175, 190, 120, 48, 157, 1, 38, 93, 36, 155, 69, 241, 266, 203, 119, 34, 190, 292, 44, 142, 12, 182, 170, 124, 6, 198, 136, 162, 24, 6, 154, 153, 88, 145, 87 ], "proposed" : [ 180, 246, 290, 279, 145 ], "email" : "user15@auth.gr", "password" : "123", "username" : "user15", "userid" : 15 }
```

Figure 4.3: User's information set example.

```
{ "_id" : ObjectId("5f65b234c2f47446944cf3ff"), "userid" : 3, "courseid" : 60, "score" : 93, "ratings" : 3 }
```

Figure 4.4: User's Attendance set example.

4.4 Recommendation system software

The Recommendation Systems are associated with prediction and classification techniques that are in close affiliation with the machine learning field. In addition, it is generally admitted that the Python programming language is currently a go-to language for related frameworks [22]. With those two things in consideration, research between Python RS

frameworks took place, and the result was to adopt the TensorRec framework. “TensorRec is a Python recommendation system that allows you to quickly develop recommendation algorithms and customize them using TensorFlow” [23]. It is open-sourced and utilizes TensorFlow, one of the leading software libraries when it comes to machine learning and modeling. Its inputs are the users’ features vector, the items’ features vector, and the interaction matrix.

For this study, the user’s features include students’ ids, the items’ features include the content type metadata for each course and the interactions matrix consists of the estimated score mentioned in chapter 3, related to the ratings and the scores of the users. In the first stage, the system transforms the features to representational graphs to start a training procedure for a given model. The programmer is called to adjust these representation graphs but just as well the loss and prediction graphs too. These last adjustments are responsible for the methods used to compute the loss of a set of recommendations and the recommendation score accordingly. The training that follows is a repeated process that aims to minimize the distance between the predicted ranks, which are a depiction of the recommendation scores, and the original interactions.

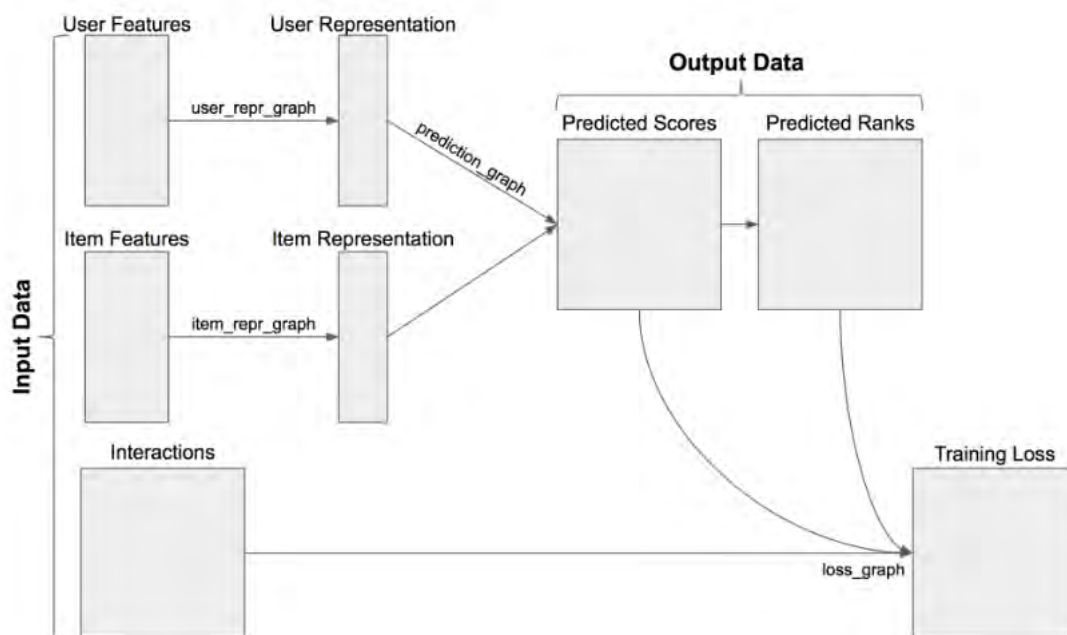


Figure 4.5: Logic of the TensorRec framework.

Source: <https://github.com/jfkirk/tensorrec>

After a trial and error process, the final adjustments chosen for this Thesis are the follow-

ing:

User representation graph: Normalized Linear Representation Graph, Calculates the representation by passing the features through a linear embedding where all latent representations have the same magnitude.

Item representation graph: Weighted Feature Pass Through Representation Graph, uses the features as representations and learns weights for each one of them.

Prediction graph: Cosine Similarity Prediction Graph, calculates the prediction as the cosine between the user and item representations.

Loss graph WMRB (Weighted Margin-Rank Batch) Loss Graph, takes account of positive interactions only and ignores magnitude. It implements the idea of ordered weighted average loss in a batch training algorithm. [24].

It should be noted that the values "Rating" and "Score" from chapter 2 are combined to a single value for every relation between user and course, to fit the requirements of the system.

$$FinalScore = 0.6 * Score + 0.4 * Rating \quad (4.1)$$

Regarding the performance of the model, the fitting needs about 20 minutes to be completed and produce the final results. Moreover, the qualitative effectiveness of this adaptation is discussed in chapter 5.

4.5 Web application software

As described earlier, the web application is a collaboration of the 'client' and the 'server' side, which translates to the front and back end of the platform. To achieve this inclusive development, the components of the MERN stack are deployed. This stack consists of the MongoDB program, the express JavaScript framework for the implementation of the server-side, the REACT JavaScript framework for the implementation of the user interface and the client-side, and the node JS environment. This allows the development of the platform solemnly with the JavaScript programming language.

4.5.1 User interface

The implementation of a MOOC platform is not a simple task, primarily because of the "massive" term. At this stage, due to the lack of content and for the sake of simplicity, the

developed interface delivers an environment only for the students. A user can log in and navigate to the “courses table” page where according to the tab he chooses, he can view his taken, available, or recommended courses. From within this page, he can enroll in new classes or cancel old ones by checking the corresponding box and submitting his changes. Another service of the platform is that the user can choose a specific course and view its material. In this “course” page, he can also rate the curriculum and submit his progress. This translates to updating the database and ultimately providing new data to the recommender. The figures below give a better picture of the interface.

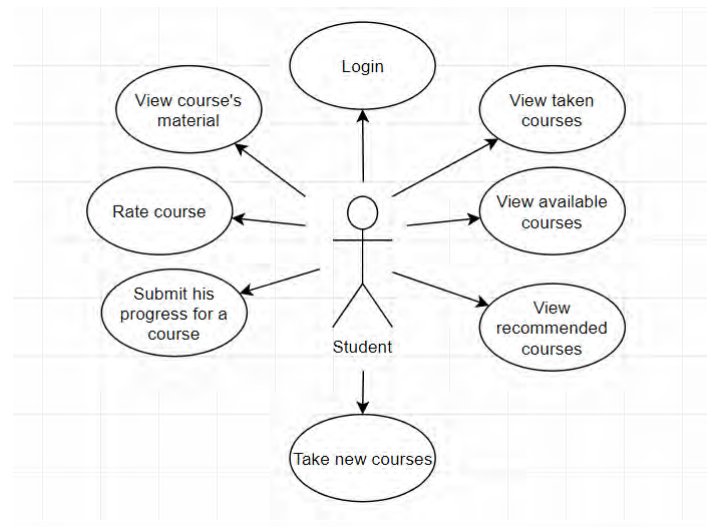


Figure 4.6: Use cases diagram.

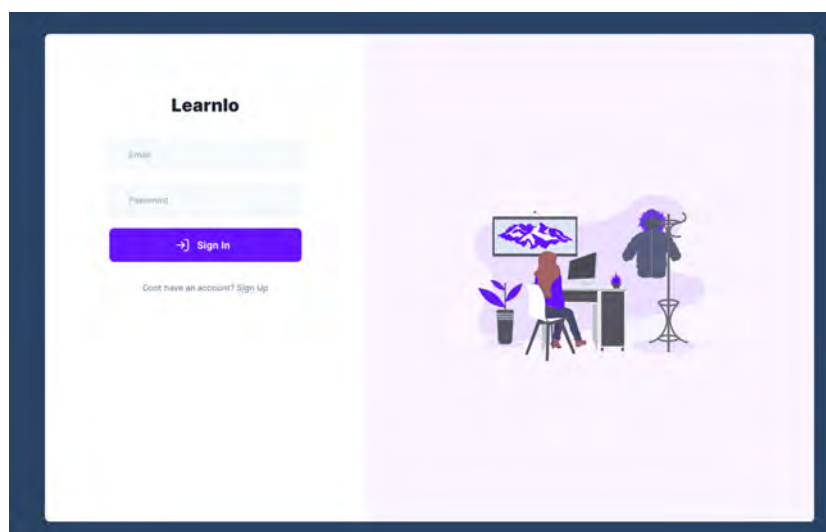
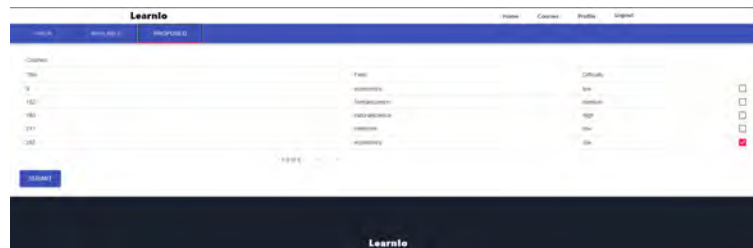


Figure 4.7: Login page.



The screenshot shows the 'Learnio' web application interface. At the top, there is a navigation bar with 'Home', 'Courses', 'Profile', and 'Logout' links. Below the navigation bar, there is a table of recommended courses. The table has columns for 'Course ID', 'Course Name', 'Difficulty', and 'Status'. The courses listed are:

Course ID	Course Name	Difficulty	Status
101	Introduction to Python	Beginner	Not Started
102	Advanced Python	Intermediate	Not Started
103	Python for Data Science	Advanced	Not Started
104	Python for Web Development	Intermediate	Not Started
105	Python for Automation	Beginner	Not Started

Figure 4.8: Table of recommended courses.



Figure 4.9: Rating functionality.

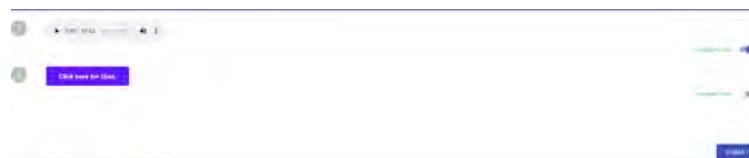


Figure 4.10: Progress score submission.

Chapter 5

Results and evaluation

5.1 Introduction

Evaluation of effectiveness is a critical part of the developing cycle of the RS. The final results ought to be qualitatively and quantitatively assessed. Beyond numerical predictions and algorithmic outcomes, there is a further need to understand users' satisfaction. One can easily find a plethora of approaches adjusted to different decisions of the developers, such as the previous data manipulation, parameters of the system, or the available means and audience.

First of all, there are mainly three methodologies from which a researcher may start his course of action. [25] [26]

1. This tool utilizes real-time recommendation options and feedback from real users. A user is called to select between two recommendations produced by different systems. The deviations between the systems may be specific parameters or the total structure of the model. This evaluation takes into account explicit or implicit signals of users' selection and suggests the system with the most preferred out-turns. Possible and mostly used metrics are click-through-rate.
2. User studies: Similar to the approach above, real users are provided with differentiated recommendations, but in this case, they are asked to give explicit ratings for the recommendations provided. In later time these ratings are evaluated by the developing team to conclude the best approach for their system.
3. Offline evaluation: Finally, Offline evaluations typically measure the accuracy of a

recommender system based on a ground-truth hypothesis and a more mathematical or statistical analysis [27]. Here, all the information needed is provided from the configured data and the results of the model. This method requires an effectiveness metric, a comparable numerical value, that assigns a numeric score to a ranking, as an assessment of its quality, or its accuracy. [28]

In the first two propositions, the objective is to decide between two unrelated recommender models, but in the last one, the objective is to improve the metrics variables by readjusting one existing model. In our study, the lack of real users to provide feedback and the predominance of offline evaluations in the recommender community [29] led to the adoption of the latter method.

5.2 The DCG algorithm

As mentioned before, offline evaluation comes hand to hand with a specific metric. A variety of different metrics can be categorized to belong to:

- accuracy and error base methods,
- decision support methods,
- and ranking based methods

respectively with what is considered to be more important by the programmers.

The first method aims to bring the final predicted scores of a test set closer to what it is known to be true. Usually, it utilizes popular metrics such as root mean square error. The second one aims to make a clear separation between “good” and “bad” items, assuring that only positively weighted items are recommended. Finally, the third option analyzes how the items were ordered, and assures the relevance of the recommendation

Ranking based evaluation methods assist us in understanding how suggested items are ordered in terms of their relevance for the users. This is the chosen method for this study not only because they help us to measure the ranking quality of the model [30], but also because the highly uncorrelated input data restricts us from safe use of the two first methods.

Discounted Cumulative Gain is a prominent metric for this category. In practice, it is a measure of ranking quality that is used often to measure the effectiveness of web search

engine algorithms or related applications [31]. In theory, every recommendation accumulates a graded score, which is computed by the relevance values extracted from the data history.

$$DCGp = \sum_{i=1}^p \frac{2_i^{rel} - 1}{\log_2(i + 1)} \quad (5.1)$$

In simple words, for a given set of values [1,2,3,4], the final score is the sum of these values, weighted accordingly to their position in the set. It is inferred that the first values of the set, are considered to be more pivotal for the recommendations.

$$\begin{aligned} \text{Cumulative Gain of [3,1,2,3,2,0]} &= 11 \\ \text{Discounted Cumulative Gain of [3,1,2,3,2,0]} &= 13.30 \end{aligned}$$

After the computation of the recommendations' score, one can proceed to analyze the results and make comparisons with the forenamed metric. A drawback to face is that while using DCG, there is not a clear picture of the ideal rankings, but previous studies have shown sufficient results [32] using normalized DCG.

5.3 Evaluation methodology

As mentioned before, the result of the RS is a ranked table of all available courses for each user. After sorting and selection, the database is updated with the best top five recommendations for each one. These top five recommendations will be evaluated by accumulating a score for each content type for some arbitrary total of users. The quoted DCG score is computed based on the input set:

$$\text{contentXvalues} = [\text{Score}, \text{Taken}]$$

The score value is the sum of the scores of courses, including the respective content type. Taken value is the counted appearances of the corresponded content type in all courses, taken and rated. So after the first step of accumulating the data for the contentXvalues set, the next step is to proceed to the computation of the DCG value. When the final results are in disposition for the observer, then follows the stage of comparison to weightiness if the observations are sufficient to be conclusions.

Examples and observations

Below there is a list of randomly selected users' metrics that contributed to the conclusions of the evaluation study.

user 456	SCORE	TAKEN	FINAL	DCG
VIDEOS	215	38	1	5,27E+80
AUDIOS	170	30	1	1,50E+67
SCRIPTS	175	33	0	4,79E+67
EXERCISES	206	34	2	1,03E+78
LITERATURE	222	37	1	6,74E+80

orderByFinalValue:[Exercises,Videos/Audios/Literature,Scripts]
oderByDCGValue:[Literature,Videos,Exercises,Scripts,Audios]

Figure 5.1: DCG results for the user 456.

user 645	SCORE	TAKEN	FINAL	DCG
VIDEOS	349,6	63	2	1,15E+121
AUDIOS	323,7	59	1	1,71E+112
SCRIPTS	323,1	62	1	1,71E+112
EXERCISES	303,1	59	0	1,63E+106
LITERATURE	348,6	63	3	5,73E+119

orderByFinalValue :[Literature,Videos,Audios/Scripts,Exercises]
orderByDCGValue :[Videos,Literature,Audios,Scripts,Exercises]

Figure 5.2: DCG results for the user 645.

user 3586	SCORE	TAKEN	FINAL	DCG
VIDEOS	95,3	15	2	4,88E+43
AUDIOS	92,3	14	1	6,10E+42
SCRIPTS	125	19	3	4,25E+52
EXERCISES	84,8	13	0	3,37E+41
LITERATURE	96,6	15	1	1,20E+44

orderByFinalValue:[Scripts,Videos,Audios,Literature,Exercises]
oderByDCGValue:[Literature,Videos,Exercises,Scripts,Audios]

Figure 5.3: DCG results for the user 3586.

user 7000	SCORE	TAKEN	FINAL	DCG
VIDEOS	215	38	3	5,79E+90
AUDIOS	170	30	0	3,53E+87
SCRIPTS	175	33	0	1,73E+84
EXERCISES	206	34	2	1,21E+99
LITERATURE	222	37	0	1,61E+76

orderByFinalValue:[Videos,Exercises,Audios/Scripts/Literature]
oderByDCGValue:[Exercises,Videos,Audios,Scripts,Literature]

Figure 5.4: DCG results for the user 7000.

user 8790	SCORE	TAKEN	FINAL	DCG
VIDEOS	368,4	61	2	6,01E+125
AUDIOS	310	54	0	2,09E+109
SCRIPTS	325,4	56	0	6,84E+112
EXERCISES	332,6	57	3	8,75E+114
LITERATURE	286	54	0	1,24E+102

orderByFinalValue	[Exercises, Videos, Audios/Scripts/Literature]
oderByDCGValue	[Videos, Exercises, Scripts, Audios, Literature]

Figure 5.5: DCG results for the user 8790.

user 13564	SCORE	TAKEN	FINAL	DCG
VIDEOS	177,2	28	1	2,96E+95
AUDIOS	141,9	26	1	1,45E+92
SCRIPTS	129,4	23	0	5,27E+80
EXERCISES	150,8	26	2	7,24E+90
LITERATURE	129	22	1	2,55E+105

orderByFinalValue	[Literature, Videos, Audios/Scripts, Exercises]
orderByDCGValue	[Videos, Literature, Audios, Scripts, Exercises]

Figure 5.6: DCG results for the user 13564.

user 18569	SCORE	TAKEN	FINAL	DCG
VIDEOS	262,4	52	1	7,41E+93
AUDIOS	234,9	45	0	2,76E+86
SCRIPTS	262	48	0	7,41E+93
EXERCISES	264,9	46	0	2,96E+95
LITERATURE	287,4	51	4	2,49E+102

orderByFinalValue	[Literature, Videos, Audios/Scripts/Exercises]
oderByDCGValue	[Literature, Exercises, Videos, Scripts, Audios]

Figure 5.7: DCG results for the user 18569.

1. It is improbable to achieve identification between the orderByFinalValue set and the orderByDCGValue set.
2. There is a high probability that content types that never appeared in the recommendations results will have the lowest metrics.
3. Additionally there is a high probability that the two most recommended types will remain in the first two positions of the proposition order.
4. Lower values for the SCORE and TAKEN column is noticed to contribute to more hazed and unordered propositions.(5.3)

5.4 Results overview

To summarize, the recommendations are judged to be adequate since there is noticeable convergence between the two sets of ordering. The RS model mainly counts on the score attribute but also considers content types of the courses as metadata, a supplementary source of information to be assisted. This association is reemerging in the evaluation method, verifying, simultaneously, the initial model, and the results. Even if there is no correlation proven, these conclusions are based upon the proportional nature of the measurements.

Chapter 6

Summary

6.1 Conclusion

This Thesis was an attempt to organize a fully functional smart MOOC platform. The original goal was the implementation of a hands-on interface for the user that incorporates a page of recommended courses based on the implemented recommender system mentioned in the above chapters.

The first step of our process was to study the bibliography for pedagogically based parameters to justify the reasonability of the recommendations. Consequently, the theoretical grounds of the system relied on three factors: the likeability of the courses, the level of progress of each individual, and the preferred type of content.

Secondly, an examination of available implemented recommender systems occurred, resulting in selecting a Python TensorFlow framework, named TensorRec. Reasonably, the next step was the configuration of the system and the trial and error stage of improving it. Finally, the DCG algorithm was applied to validate our hypothesis of qualitative consistency. The results were judged sufficient because of proportional measurements.

Alongside the RS development, took place the formation of a service-oriented web platform. The tools employed for this basic user interface were the JavaScript components of the MERN stack. At this stage, the platform meets limited functionalities but is capable of providing a contextual environment for the recommending results.

Overall, this Thesis was focused on the technical aspects of the system's development even though its starting point was of theoretical background.

6.2 Future work

It is acknowledged that the model at the current stage displays basic functionalities and that there is room for improvement. In respect to recommender results, our primary concern is the lack of real data, which would be vital for the learning analytics approach. Real life data would enable the researcher to perform real life experiments in the future and ensure a better evaluation of the system. Furthermore, there is a definite need to distinguish recommendations between fields, because presently there is no distinction between the enrolled courses. Finally, concerning the user's interface, it would be useful to develop a profile page to include the user's statistics, a feature once again helpful for analyzing his behavior.

Bibliography

- [1] Tiffany Ya Tang and Gordon McCalla. Smart recommendation for an evolving e-learning system. In *Workshop on Technologies for Electronic Documents for Supporting Learning, International Conference on Artificial Intelligence in Education*, pages 699–710, 2003.
- [2] Melanie Booth. Learning analytics: the new black. *Educause Review*, 47(4):52–53, 2012.
- [3] Scott A Schartel. Giving feedback—an integral part of education. *Best practice & research Clinical anaesthesiology*, 26(1):77–87, 2012.
- [4] Fatiha Bousbahi and Henda Chorfi. Mooc-rec: a case based recommender system for moocs. *Procedia-Social and Behavioral Sciences*, 195:1813–1822, 2015.
- [5] Soude Fazeli, Hendrik Drachsler, and Peter Sloep. Applying recommender systems for learning analytics: A tutorial. *Handbook of Learning Analytics*, page 235.
- [6] Fathi Essalmi, Leila Jemni Ben Ayed, Mohamed Jemni, Sabine Graf, et al. Generalized metrics for the analysis of e-learning personalization strategies. *Computers in Human Behavior*, 48:310–322, 2015.
- [7] Jan Renz, Daniel Hoffmann, Thomas Staubitz, and Christoph Meinel. Using a/b testing in mooc environments. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, pages 304–313, 2016.
- [8] Panagiotis Symeonidis and Dimitrios Malakoudis. Moocrec. com: Massive open online courses recommender system. In *nobookcollection*. CEUR-WS. org, 2016.

- [9] Zameer Gulzar, A Anny Leema, and Gerard Deepak. Pcrs: Personalized course recommender system based on hybrid approach. *Procedia Computer Science*, 125:518–524, 2018.
- [10] Jie Lu. A personalized e-learning material recommender system. In *International Conference on Information Technology and Applications*. Macquarie Scientific Publishing, 2004.
- [11] Daniel McFarland and Diane Hamilton. Factors affecting student performance and satisfaction: Online versus traditional course delivery. *Journal of Computer Information Systems*, 46(2):25–32, 2005.
- [12] Siu Keung Brian Wong, Thao Thi Nguyen, Elizabeth Chang, and Nimal Jayaratna. Usability metrics for e-learning. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*, pages 235–252. Springer, 2003.
- [13] Fathi Essalmi, Leila Jemni Ben Ayed, Mohamed Jemni, Sabine Graf, et al. A fully personalization strategy of e-learning scenarios. *Computers in Human Behavior*, 26(4):581–591, 2010.
- [14] Kalyan Veeramachaneni, Sherif Halawa, Franck Deroncourt, Una-May O’Reilly, Colin Taylor, and Chuong Do. Moocdb: Developing standards and systems to support mooc data science. *arXiv preprint arXiv:1406.2015*, 2014.
- [15] Barry Smyth, Maurice Coyle, and Peter Briggs. Communities, collaboration, and recommender systems in personalized web search. In *Recommender Systems Handbook*, pages 579–614. Springer, 2011.
- [16] Information filtering system. https://en.wikipedia.org/wiki/Information_filtering_system. Ημερομηνία πρόσβασης: 27-9-2020.
- [17] Recommender system wiki. https://en.wikipedia.org/wiki/Recommender_system. Ημερομηνία πρόσβασης: 27-9-2020.
- [18] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.
- [19] Jackson Wu. Types of recommender systems, May 2019.

- [20] Classifying different types of recommender systems, Nov 2015.
- [21] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21(6):1487–1524, 2017.
- [22] Dawid Karczewski. What is the best language for machine learning in 2020?
- [23] Jfkirk. jfkirk/tensorrec.
- [24] Kuan Liu and Prem Natarajan. Wmrb: Learning to rank in a scalable batch training approach. *arXiv preprint arXiv:1711.04015*, 2017.
- [25] Rocío Cañamares, Pablo Castells, and Alistair Moffat. Offline evaluation options for recommender systems. *Information Retrieval Journal*, pages 1–24, 2020.
- [26] Joeran Beel and Stefan Langer. A comparison of offline evaluations, online evaluations, and user studies in the context of research-paper recommender systems. In *International conference on theory and practice of digital libraries*, pages 153–168. Springer, 2015.
- [27] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 257–260, 2010.
- [28] Daniel Valcarce, Alejandro Bellogín, Javier Parapar, and Pablo Castells. On the robustness and discriminative power of information retrieval metrics for top-n recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 260–268, 2018.
- [29] Dietmar Jannach, Markus Zanker, Mouzhi Ge, and Marian Gröning. Recommender systems in computer science and information systems—a landscape of research. In *International Conference on Electronic Commerce and Web Technologies*, pages 76–87. Springer, 2012.
- [30] Muffaddal Qutbuddin. An exhaustive list of methods to evaluate recommender systems, May 2020.
- [31] Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.
- [32] Personalized web search challenge.

Appendix

Programming code

1 Recommender system programming code

1.1 Data generator code

Courses data

```
1 import csv
2 import random
3 fields=['physics','arts','formalscience','medicine','history','
         naturalscience','economics']
4 professor_involvement=['low','medium','high']
5 content=['scripts','videos','audios','extraliterature','exercises']
6 difficulty=['low','medium','high']
7 courses =[]
8
9 #This scripts creates a csv file where the item features of each course
   is stored "
10 #These features are determined by the instructor when the course is
   uploaded to the platform"
11 #Here the data is randomly produced to signify existing courses"
12 #They may be determined explicitly by the professor or taken from the
   web platform depending the professor's actions"
13
14 with open('courses_item_features.csv', 'w', newline='') as csvfile:
15     writer = csv.writer(csvfile, delimiter=',')
```

```
16 writer.writerow(['courseid','title','coursefield','
17 professor_involvement','types_of_content','difficulty'])
18 for i in range(1000):
19     course_string=[]
20     course_string.append(str(i))
21     course_string.append('title')
22     course_string.append(random.sample(fields,k=1)[0])
23     course_string.append(random.sample(professor_involvement,k=1)[0])
24
25     num_of_content_types = random.randint(1,5)
26     content_list= random.sample(content,k=num_of_content_types)[: (
27 num_of_content_types)]
28     course_string.append('|'.join(x for x in content_list))
29     course_string.append(random.sample(difficulty,k=1)[0])
30
31     writer.writerow(course_string)
```

User attendance data

```
1 import csv
2 import random
3 import pandas as pd
4
5 #This dataset is also randomly produced to represent the attendance of
6 #existing students
7 #Each student has a unique id and there should be only one unique
8 #interrelation between a student id and a course id
9 # The size of this dataset (= 10.000 samples) is determined by a quick
10 #estimation of the number of uth-eclass users
11
12 score = []
13 weights= []
14 for i in range(100):
15     score.append(i)
16     if i>50:
17         weights.append(0.66)
18     else:
19         weights.append(0.34)
```

```

19 with open('users_implicit_data.csv', 'w', newline='') as csvfile:
20     writer = csv.writer(csvfile, delimiter=',')
21     writer.writerow(['userid', 'courseid', 'score', 'ratings'])
22     for i in range(20000):
23         num_of_courses_taken = random.randint(5,100)
24         ids_of_courses = random.sample(range(0,300),num_of_courses_taken)
25
26         for j in range(num_of_courses_taken-1):
27             attendance_string = []
28             attendance_string.append(str(i))
29             attendance_string.append(str(ids_of_courses[j]))
30             attendance_string.append(random.choices(score,weights)[0])
31             attendance_string.append(random.randint(0,5))
32
33             writer.writerow(attendance_string)
34
35 df = pd.read_csv('users_implicit_data.csv')
36 print(df.info())

```

1.2 TensorRec code

```

1 #def function: Where we combine ratings and score
2 #via trial and error Multiplying the rating by 20
3 #is to bring in the same numerical
4 #scale as the score metric
5 # Weights were tried to achieve a reasonable outcome
6 def final_score(users_data,users_header):
7     for i in users_data:
8         i[2]=0.6 * i[2] + 0.4*20*i[3]
9         i = i.pop(3)
10        users_header.pop(3)
11
12 #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!CONNECTION WITH SERVER!!!!!!!!!!!!!!!!!!!!!!!!!!!!
13 print("@Connected to webappdbs db on mongodb://127.0.0.1:27017/")
14 client = MongoClient('mongodb://127.0.0.1:27017/')
15 db = client.webappdbs
16
17 ....
18

```

```
19 # Transform the content_types into binarized labels
20 #using scikit's MultiLabelBinarizer
21 courses_content_features = MultiLabelBinarizer().fit_transform(
    courses_content)
22 n_types_content = courses_content_features.shape[1]
23 print("@Binarized content types for course {}: {}".format(0,
    courses_content_features[0]))
24
25 ....
26
27 #!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
28 start_time = time.time()
29 print("@Training hybrid recommender")
30 hybrid_model = tensorrec.TensorRec(
31     n_components= 305,
32     user_repr_graph=tensorrec.representation_graphs.
    NormalizedLinearRepresentationGraph(),
33     item_repr_graph=tensorrec.representation_graphs.
    WeightedFeaturePassThroughRepresentationGraph(),
34     prediction_graph=tensorrec.prediction_graphs.
    CosineSimilarityPredictionGraph(),
35     loss_graph=tensorrec.loss_graphs.WMRBLLossGraph()
36 )
37 print("@Fitting the trained model")
38 hybrid_model.fit(interactions=users_data_train,
39     user_features=user_indicator_features,
40     item_features=full_item_features,
41     epochs = 100,
42     learning_rate = 0.1,
43     verbose = True,
44     n_sampled_items= int(n_items*0.1))
45 print("@Finished fitting in %s min ---" % ((time.time() - start_time)/60)
    )
46 print("@Hybrid recommender:")
47 predicted_ranks = hybrid_model.predict_rank(user_features=
    user_indicator_features,
48     item_features=
    full_item_features)
49
```



```
7   this.state = {courses :[],file:null,TabValue:'0',page:0};
8   this.handleTabChange = this.handleTabChange.bind(this);
9 }
10
11 handleTabChange (e, value) {
12   e.preventDefault();
13   axios.get ('http://localhost:4000/webappdbs/courses?ID='+localStorage.
14   id+'|'+value)
15     .then(response => {
16       this.setState({
17         courses: response.data,
18         TabValue: value
19       });
20     })
21     .catch(function (error ){
22       console.log(error);
23     })
24 }
25 componentWillMount () {
26   console.log("FirstMounting, should appear once")
27   axios.get ('http://localhost:4000/webappdbs/courses?ID='+localStorage.
28   id+'|0')
29     .then(response => {
30       response.data.forEach(element => {
31         selectedInit = selectedInit.concat(element.courseid)
32       });
33       this.setState({courses: response.data});
34     })
35     .catch(function (error ){
36       console.log(error);
37     })
38 }
39 render () {
40   return (
41     <Container>
42     <Header/>
43     <TabContext value={this.state.TabValue}>
44     <AppBar position="static">
```

```
44     <TabList onChange={this.handleTabChange} aria-label="simple
45     tabs example">
46       <Tab label="Taken" value="0"/>
47       <Tab label="Available" value="1"/>
48       <Tab label="Proposed" value="2"/>
49     </TabList>
50   </AppBar>
51   <TabPanel value="0">
52     <Table columns={columns} data={this.state.courses}/>
53   </TabPanel>
54   <TabPanel value="1">
55     <Table columns={columns} data={this.state.courses}/>
56   </TabPanel>
57   <TabPanel value="2">
58     <Table columns={columns} data={this.state.courses}/>
59   </TabPanel>
60 </TabContext>
61 </Container>
62 )
63 }
64 };
```

2.2 Server side

Extract from the server side

```
1 mongoose.connect('mongodb://127.0.0.1:27017/webappdbs', {useNewUrlParser:
2   true});
3
4 const connection = mongoose.connection;
5
6 connection.once('open', function() {
7   console.log("MongoDB database connection established successfully.");
8 })
9
10 .....
11 //Handling of the courses table request
12 Routes.route('/coursesubmit').post(function(req,res) {
```

```
12 let selectedNew = req.body.selected;
13 let id = Number(req.body.id);
14 let selectedPrev = [];
15 UserInfo.findOne({'userid': id}, (err, user) => {
16     selectedPrev = user.courses;
17     selectedNew.forEach((element) => {
18         if(!selectedPrev.includes(element)) {
19             UserAttend.create({'userid': id, 'courseid': Number(element)
20 , 'score': 0, 'ratings': 2})
21             .then(e => {
22                 console.log('added new subject')
23             });
24         })
25     selectedPrev.forEach(element => {
26         if(!selectedNew.includes(element)) {
27             UserAttend.findOneAndDelete({'userid': id, 'courseid':
28 Number(element)})
29             .then(e => {
30                 console.log('deleted old subject')
31             });
32         })
33     })
34     }).then(exit => {
35         UserInfo.findOneAndUpdate({'userid': id}, { $set: {"courses":
36 selectedNew}})
37         .then(res.json('user edited successfully'))
38         .catch(e => console.log(e));
39     })
40 });
41
42
43 app.use('/webappdbs', Routes);
44 app.listen(PORT, function() {
45     console.log("Server is running on Port "+PORT);
46 });
```