Electronic, didactic and innovative platform for learning based on multimedia assets

e-DIPLOMA

Funded by the European Union

D.3.4 AI Algorithms
Version No. V1.3
30 September 2024

HISTORY OF CHANGES			
Version*	**Publication date**	**Beneficiaries**	**Changes**
V1.0	15.09.2024	UJI	▪ Initial version of Deliverable Owner
V1.2	23.09.2024	UPV, TU Delft	▪ Pre-final version reviewed by Internal Reviewers
V1.3	25.09.2024	UJI	▪ Final version sent to the Project Coordinator

(*) According to the section “Review and Submission of Deliverables” of the Project Handbook

1. Technical References

Project Number	101061424
Project Acronym	e-diploma
Project Title	Electronic, Didactic and Innovative Platform for Learning based On Multimedia Assets
Granting Authority	European Research Executive Agency (REA)
Call	HORIZON-CL2-2021-TRANSFORMATIONS-01
Topic	HORIZON-CL2-2021-TRANSFORMATIONS-01-05
Type of the Action	HORIZON Research and Innovation Actions
Duration	1 September 2022 – 31 August 2025 (36 months)
Entry into force of the Grant	1 September 2022
Project Coordinator	Inmaculada Remolar Quintana

Deliverable No.	D3.4: AI Algorithms
Work Package	WP3: Piloting and testing of innovation procedures and technology enhancement
Task	T3.7: Definition of AI applied in the e-platform
Dissemination level*	PU- Public
Type of license:	CC-BY

Lead beneficiary	Universitat Jaume I (UJI)
PIC of the Lead beneficiary	999882985
Contributing beneficiary/ies	
PIC of the Contributing beneficiary/ies	
Author(s)	▪ Raúl Montoliu Colás (0000-0002-8467-391X) ▪ Indira Lázara Lanza Cruz (0000-0003-2413-2799)
Contributor(s)	
Due date of deliverable	30.09.2024
Actual submission date	30.09.2024

2. Table of Contents

3. Introduction

3.1. Executive Summary

This document D.3.4 of the e-DIPLOMA project gathers the architecture and artificial intelligence algorithms that will be used as part of the recommendation system associated with the e-learning platform. The algorithms, consisting of Deep Learning and Machine Learning models, can analyze the resources (Learning Objects) that users can interact with during a course and the patterns of those interactions. This analysis ensures that students' progression through the course is optimal.

Two approaches have been developed for the Recommender System (RS). The purpose of exploring different solutions is to explore the possibilities of the available data and their usage to help the students. For each of these algorithms, we will describe in detail how they have been developed and how they should be used, to obtain the resource recommendations.

This document summarises the results obtained from task T.3.7. Definition of AI applied in the e-platform.

3.2. Relation to Other Project Documents

This document describes the Artificial Intelligence algorithms that should be connected to the e-learning platform detailed in D4.1 E-platform specification and D4.2 E-platform software.

3.3. Abbreviation List

The following acronyms are frequently used in this document, listed in alphabetical order:

- AI: Artificial Intelligence
- CF: Collaborative Filtering
- ERS: educational recommendation systems
- KBF: Knowledge-based Filtering
- KG: Knowledge Graph
- LMS: Learning Management System
- LO: Learning Object
- RS: Recommender System
- RM: Rating Matrix (User-Interaction Matrix)

3.4. Reference Documents

For the comprehensive list of references, please refer to Section 9 of this document.

4. Recommendation Background

Most educational recommendation system solutions are focused on recommending a specific course or sequence of courses for a user to follow. However, to the best of our knowledge, few studies have been conducted related to content recommendation within the same course. This poses a different problem, since, although the user wants to receive a course with a pre-established structure, students often do not follow the official course sequence. Students tend to revisit resources that they consider necessary at a

specific moment or to exclude content that they already master. The designed learning path is not always the optimal for all students, so adapting the presentation of the content based on their needs can increase engagement and improve academic results. In this sense, we propose the study of two learning object recommendation models within a course. On the one hand, the first recommendation algorithm proposed is a hybrid system that combines CF and KBF. On the other hand, a second recommendation algorithm has been proposed, which combines user CF and behaviour paths as a knowledge graph. Each model explores different techniques, but they have a common objective, making recommendations adapted to the specific needs of the student. The purpose is to increase motivation and participation in the course by students, and thus optimize academic results.

Below a study of the state of the art, limitations and research opportunities are discussed. Subsequently, the following sections explain the recommendation systems proposed in this research project.

Recommender Systems (RS) are a subclass of information filtering techniques that can incorporate various AI algorithms designed to predict a user's preference for an item. RS helps users navigate large sets of choices by highlighting potentially interesting options. In commercial domains, Recommendation Systems are most used to increase sales of specific products and encourage greater consumption by individual users. In the educational sector, the objective of recommendation systems is to provide learning objects that fulfill educational objectives while adapting to the needs and characteristics of the end user.

In the educational field, there is a growing interest in developing digital technologies to enhance teaching processes. In this context, educational recommendation systems (ERS) are strategic solutions that allow for the personalization of the educational experience from both the teacher's and the student's perspectives. Various solutions are typically focused on the type of problem to be solved. For instance, a solution for the formal educational domain requires a set of considerations that may not be necessary in the informal domain. For example, in the formal academic environment, recommendations must align with curricular goals and structured learning sequences. This implies a more rigorous design and integration with learning management systems (LMS). Sequential recommendations based on prior knowledge or parameters allow students to progress adequately within a structured and time-bound academic framework. It is important that ERS consider a sequentially oriented approach towards adaptive learning, respecting the logical structure of courses (Nabizadeh, A. H. et al., 2020). The study in (Tarus et al, 2017) explores the use of sequential pattern mining to ensure that ERS respects the logical structure of university courses, which is crucial for student success in formal contexts. In contrast, in the informal domain, recommendations can be more flexible and adaptive, focusing on personal interests and the user's self-learning needs without the necessity to follow a strict content sequence (Rahman & Abdullah, 2018).

A systematic literature review in (da Silva, F.L. et al., 2023) showed that Various solution approaches have been explored in both formal and informal e-learning perspectives, with a growing emphasis on formal methods. Most e-learning recommendation strategies have focused more on the student perspective than the teacher perspective. Personalization has emerged as a key tool for addressing educational needs, aligning with user preferences, and enhancing satisfaction and motivation. However, challenges remain in aligning user expectations with recommendations. Modeling the individual factors that influence learning processes and generalizing them within a predictive framework continues to be a complex problem for effective resolution.

Next, a summary that groups the different approaches focused on developing recommendation systems in the educational field is presented. Figure 1 summarizes the different approaches that can be implemented in a recommendation system. Those approaches are categorized as “pure” and “hybrid” ones. The first group refers to methods relying on a single recommendation technique, such as collaborative filtering or content-based filtering. These methods utilize either user data or item characteristics to generate recommendations. The second group, hybrid approaches, combines multiple techniques to enhance the accuracy and diversity of recommendations, leveraging the strengths of each method to better cater to the educational needs of users.

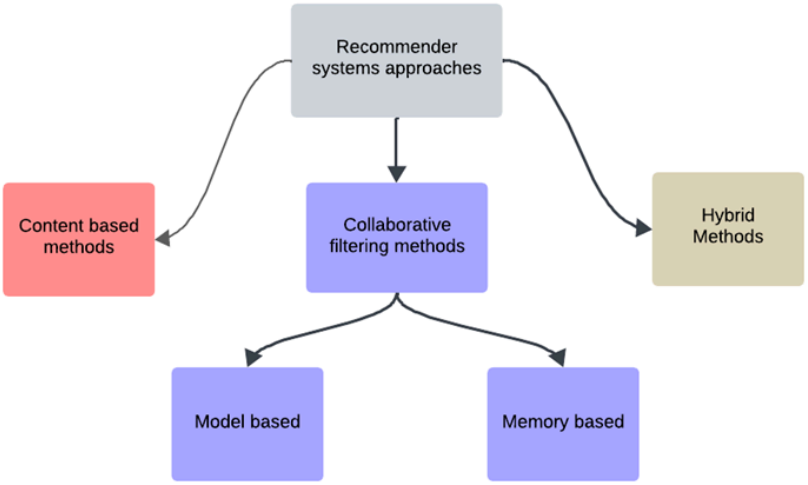

Figure 1. Recommender systems approaches (Roy & Dutta,2022).

4.1. Pure Approaches

Collaborative filtering is one of the most popular approaches in educational recommendation systems. It is based on the idea that users who have had similar behaviors in the past will have similar tastes in the future. Collaborative filtering methods can in turn be divided into two approaches: memory-based and model-based (Khanal, S.S. et al., 2020). Memory-based collaborative filtering relies on the direct use of historical user data to generate recommendations. This approach can be user-based or item-based. User-based methods recommend items to a user based on the similarity between users. A set of users with preferences similar to the target user is identified, and their interactions (e.g., ratings or access to educational materials) are used to recommend new items. Item-based methods recommend items based on the similarity between items. The interaction history of a user with various items is examined, and items similar to those the user has already shown interest in, are recommended. In each case, the similarity between explicit or implicit features of users/items is calculated using metrics such as Pearson correlation or cosine similarity. The advantages of this type of model include ease of implementation and understanding, as well as not requiring intensive training. However, its limitations lie in the limited scalability when dealing with large volumes of data.

On the other hand, model-based collaborative filtering uses machine learning techniques to build a predictive model from historical data. This approach aims to identify latent patterns in the data that can be used to make recommendations. Among the most common techniques are Singular Value Decomposition (SVD) and Matrix Factorization Models. SVD is a matrix decomposition technique that reduces the dimension of the rating matrix (users by items) into smaller matrices that capture the latent relationships between users and items. Similarly, Matrix Factorization Models aim to decompose the rating matrix into lower-dimensional matrices, capturing latent features of users and items. These models allow for the discovery of complex latent patterns in the data, facilitating the handling of large

volumes of data. However, they require greater computational capacity and a more intensive training process.

Content-based filtering is a recommendation technique that uses the intrinsic characteristics of items and user preferences to make personalized recommendations. This technique creates a user profile based on their past interactions with items, considering both explicit preferences (such as ratings and reviews) and implicit ones (such as browsing history and viewing time). The system then compares this profile with the features of other items to recommend those that are most similar to the ones the user has shown interest in (Khanal, S.S. et al., 2020). Unlike collaborative filtering, which relies on the interactions and preferences of multiple users to find common patterns and recommend items that other similar users have found useful, content-based filtering focuses solely on the features of the items and the individual user's history.

A content-based recommendation system is composed of three separate components, each with different functions. First, the content analyzer examines items using feature extraction techniques to obtain structured relevant information that serves as input for the other two components. The second is the profile learner, which is responsible for building the user's profile using generalization strategies to infer a model of the user's interests from items they have liked or disliked. Finally, the filtering component compares the user's profile representation with the items to be recommended, obtaining a relevance judgment (which can be binary or continuous) that determines whether an item should be recommended or not (Oliveira, R. J. M., 2016).

Content-based techniques have the advantage of being user-independent, as they only consider the ratings given by the user to build their profile and make recommendations. Additionally, recommendations are based on the characteristics of items that match the created profiles, allowing new items to be recommended without needing prior ratings. However, these techniques have limitations due to the available knowledge about items, as not all aspects that may interest a user can be described. They also suffer from the problem of over-specialization, as they cannot recommend items outside the user's profile. New users also pose challenges, as a stable number of ratings is needed for recommendations to become accurate.

4.2. Hybrid Approaches

Hybrid approaches are the most used as they improve the accuracy and diversity of recommendations. The solutions in the scientific literature combine two or more techniques to produce recommendations with better performance, overcoming some issues of individual techniques (da Silva, F.L. et al., 2023). They are common for addressing cold start or scalability problems. These techniques can be classified into the following four categories. The first is the "Weighted" approach. This uses the scores of the recommendation techniques used within a system to produce recommendations (for instance, a linear combination of all the different techniques' recommendation scores). It is easy to adjust but assumes that all techniques are equally valuable, which is not always true. Secondly, there is the "Switching" technique. This involves changing the recommendation technique based on certain criteria. For example, a system may start with content-based recommendations and switch to collaborative filtering if the former are not sufficiently accurate. It leverages the strengths of each technique but requires additional parameterization. Thirdly, we have "Mixed" hybrid techniques which present recommendations from multiple techniques simultaneously. An example is a solution that combines content-based techniques and collaborative information about the preferences of other users. Mixed hybrid solutions avoid the cold-start problem. On one hand, a content-based component can suggest new items based on their characteristics, even if they haven't been rated yet. On the other hand, the collaborative component can use popular information and content-based characteristics to recommend initial items and retrain based on user interactions. Finally, the "Cascade" hybridization method involves the sequential use of

techniques. The first technique produces a set of ranked candidates to be refined by a second technique (Oliveira, R. J. M., 2016).

4.3. Knowledge-Based Approaches

A distinct category in the realm of recommendation systems for e-learning is the knowledge-based approach. Unlike hybrid methods that combine various recommendation techniques, knowledge-based systems rely on a structured representation of information, such as ontologies or contextual rules, to deliver personalized recommendations. These systems leverage domain-specific knowledge to map learning resources to individual learner needs, ensuring that recommendations align with both the educational goals and the unique context of the learner. Given their emphasis on semantic reasoning and the precise matching of content to learner profiles, knowledge-based recommendation systems deserve separate attention as they offer a highly tailored and flexible solution, particularly suited for the complexities of educational environments.

Knowledge-based recommendation systems in e-learning often utilize semantics- or knowledge-based approaches, including context-based and ontology-based methods. These systems frame knowledge about content and stakeholders in the recommendation process through the use of ontology. In the context of e-learning, this means that such systems map learner-relevant learning resources by exploiting relational knowledge. By structuring and relating information about learning materials and user preferences, knowledge-based systems can provide more accurate and personalized recommendations tailored to individual learner needs (Khanal, S.S. et al.,2020).

4.4. ERS State of the Art

The following Table 1 summarizes the main techniques combining hybrid approaches applied to e-learning found in the scientific literature over the past five years. In general, most ERS in the literature focus on user-based CF, either pure or combined with other techniques. Authors such as (da Silva,F.L. et al., 2023) associate this trend with the growing perception within the educational domain of the importance of centering the learning process on the student.

Table 1. Summary of Main Hybrid Approaches in E-Learning from Scientific Literature (Last 5 Years).

Hybrid approach	**Brief description**	**References**
Collaborative Filtering (CF) + Content-Based Filtering (CBF)	This approach combines collaborative filtering with content-based filtering, using interaction data, user profiles, and item characteristics. This approach collects data both implicitly and explicitly from user actions.	(Widayanti, R. et al.,2023) (Li, L. et al., 2021)
Collaborative Filtering + Knowledge Representation + Other Techniques	This approach integrates collaborative data with knowledge bases, using ontologies and structured data. This combination allows for the incorporation of domain-specific knowledge and enhances the system's ability to provide relevant recommendations by leveraging structured information.	(Xu, G. et al., 2021) (Vedavathi, N., & Anil Kumar, K. M., 2022) (Tahir, S., 2022)

Content/Context Based Filtering + Knowledge Representation	This approach fuses content-based filtering with expert knowledge, using business rules and ontologies. The use of ontologies helps in structuring the content and enables semantic analysis, improving the recommendation quality by considering the relationships between concepts.	(Javed, U.,2021)
Knowledge Representation (Ontologies, Graphs, Fuzzy Sets) + Other Techniques	This approach combines knowledge representation methods with various other techniques, such as clustering or matrix factorization, to enhance the recommendation process. The integration of these methods allows for a more comprehensive analysis and better handling of complex data relationships.	(Nabizadeh, A. H. et al., 2020) (Ismail et al., 2019) (Joy, J., & Renumol, V.G., 2021)
Machine Learning / Deep Learning Algorithms	Machine learning algorithms and neural networks are increasingly being used in educational recommendation systems. These techniques leverage large datasets to learn complex patterns and make accurate predictions. Neural networks can model non-linear relationships and handle high-dimensional data, making them suitable for personalized recommendations.	(Mawane, J., Naji, A., Ramdani, M, 2020) (Khanal, S.S. et al., 2020)
Fuzzy Logic, Self-Organization, and Sequential Pattern Mining	Fuzzy logic, self-organization, and sequential pattern mining are used to deal with uncertainty, adapt to changes, and identify patterns in user behavior over time. These techniques provide flexibility and robustness, allowing for more adaptive and accurate recommendations.	(Wan, S., & Niu, Z., 2020)

4.5. Limitations and opportunities in developing ERS

From a technical perspective, ERS generally suffer from the cold start problem and data sparsity issues. There is still a lack of mature research addressing these known problems. Additionally, there is the issue of "overspecialization" of recommenders, meaning they focus on specific contexts, limiting the generalization of results to other educational environments(Javed, U.,2021). Another major limitation is the lack of public datasets, which restricts the reproducibility of experiments and the comparison between different approaches. Finally, most publications focus on evaluating the effectiveness of ERS based on precision metrics, neglecting important pedagogical aspects such as improvements in learning and student satisfaction.

An extensive review of the state of the art presented in (da Silva, F.L. et al., 2023) identifies several opportunities for future research in the field of ERS. Notably, there is a need for ERS to focus on organizing and sequencing recommendations to guide users' learning processes. Additionally, there is an opportunity to develop multidimensional evaluation frameworks that encompass critical aspects such as pedagogical effectiveness, novelty, and diversity of recommendations. This would provide a more comprehensive view of the impact of ERS on the teaching process.

One notable example that addresses these opportunities is the integration of recommender systems intelligent tutoring system. Muangprathub et al. (2020) demonstrated that their proposed system enhances personalized learning by tailoring content and resources to meet individual student needs and learning paths, leading to improved learning outcomes. Recommender systems also have the potential to boost student engagement by delivering personalized content, which makes learning experiences more relevant and enjoyable. This personalized approach can encourage students to explore new interests (Huang et al., 2023). Additionally, recommender systems have been shown to particularly benefit students with moderate levels of motivation, contributing to better learning outcomes overall.

The main objective of the AI algorithms to be implemented in the proposed e-learning platform is to allow users to receive recommendations on the next Learning Object they should visit. For this reason, various types of Recommender Systems have been investigated to carry out this task and their application within the education domain.

5. Algorithms Description

After careful consideration, it has been determined that the most effective Recommender Systems (RS) best suited to the needs of this project are systems that perform the recommendations by Collaborarive Filtering (CF) and Knowledge-Based Filtering (KBF). This decision was made based on the ability of both to capture user behavior and knowledge about course properties, to meet the ethical standards of e-DIPLOMA.

5.1. Collaborative Filtering

CF is one of the most common ways to perform recommendations, they form the baseline that huge online platforms like Netflix or Amazon use to bring their users movies or items they would like. CF works by finding users or items that are similar to the target user or item, and based on their data retrieving proper recommendations to the target audience. What determines the effectiveness of this recommender is how these similarities between users are obtained. Usually, these similarities are calculated using the User-Interaction Matrix or Rating Matrix. This RM gathers how users value each item of the recommendation corpus. The users are represented as the rows of the matrix while the items are represented as the columns of the matrix. These ratings are normally a Likert (e.g., from 1 to 5) or only indicate if the user has interacted with the items (e.g., 0 for no interaction and 1 for interaction). Figure 2 illustrates the relationship between users and items in an RM.

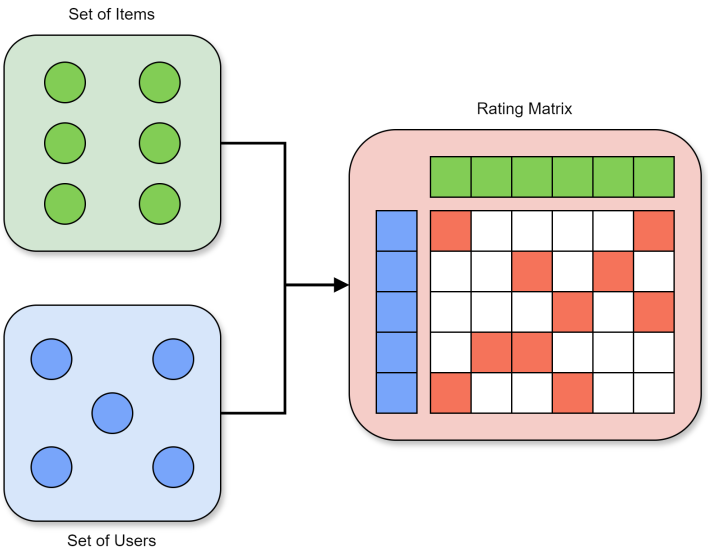

Figure 2. Example of how to create an RM from a set of users and items. A red square represents that the user has interacted with the item.

Based on the similarity target two types of CF algorithms surge. User-based CF focuses on obtaining users similar to a target user depending on how they behave with the different items. Once these similar users, also called neighbors, are identified, the system recommends items that these users have interacted with but the target user has not. On the contrary, Item-based CF focuses on finding items that are similar to the items the target user has interacted with (or rated highly). It leverages the similarity between items to make recommendations, based on the idea that users will like items similar to those they have liked in the past.

To obtain the similarities of the users or items their features must be translated into specific domains, which are called user and item domains. These domains are obtained from the respective target recommendation domain. For example, in user-based CF users are represented as lists of their interactions. Having 5 items to recommend, where the user can grade it between 1 and 5 once the user has interacted with them a possible representation could be [4, 3, 0, 5, 5]. Note the 0 grade in one item, this refers to the user not interacting with that item. Within this, the user space will be a 5-dimensional space where similar users will be close and different users will be apart. To obtain these similarities the distances between users must be computed. This distance can be calculated using any distance metric that works for measuring distances in a given space, such as the Euclidean Distance. Since distance and similarity are the opposite of each other, once the distance is obtained the similarity is also calculated. Once the similarity for each user is obtained, the neighborhood of the target user is defined. Instead of clustering users into exclusive groups, we define a neighborhood as a set of users whose similarity to the target user exceeds a predefined threshold. This means that a user can belong to multiple neighborhoods, depending on their proximity to different users.

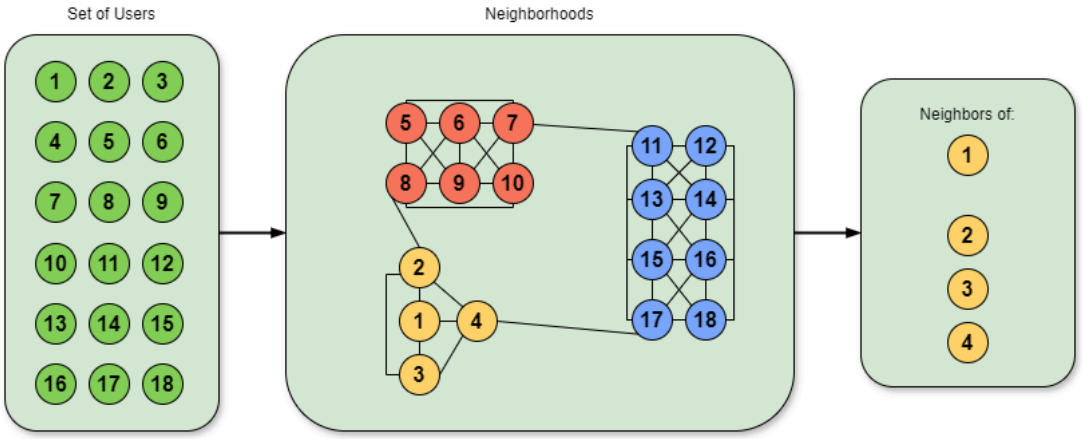

Figure 3. Graphical representation of how users are clustered in different neighborhoods.

Once the neighborhood of the target user is computed, as can be seen in Figure 3, it is used to predict the rating values the target user may give to the items with no interactions. The predicted value of a given item is obtained by a weighted sum of all the ratings of the users in the neighborhood, where the weights are the similarities of every user with the target user. Therefore the recommended items to the target user are the items where the predicted values are the highest. Figure 4 illustrates the process of obtaining the ratings of a user based on the weighted sum of the ratings of the rest of the users of the neighborhood.

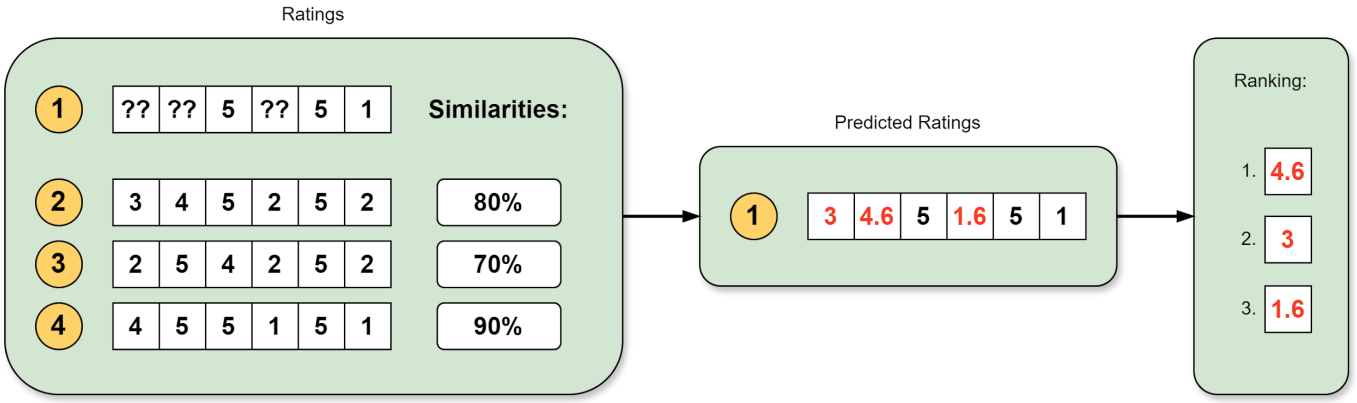

Figure 4. Process of ranking items based on the ratings of the neighborhood.

5.2. Knowledge-based Filtering

KBF, unlike CF, uses explicit domain knowledge to perform the recommendation. This knowledge can be obtained from different sources such as domain experts or databases. Once the knowledge is collected it must be first structured, normally by using Knowledge Graphs (KGs) or ontologies, to allow the system to process the information and understand how the items are related. These representations of information can capture the underlying information about the entities, relationships, and constraints that are relevant to the specific domain.

Both, ontologies and knowledge graphs, are structured sources of information that represent the entities by how they are related to each other. The entities are the objects, concepts, or things that exist within the domain where the recommendation is going to be applied (in this project, education). For this project, these entities correspond to the different LOs that will be part of the courses. Ontologies are usually represented using formal languages such as Resource Description Framework (RDF) or Web Ontology Language (OWL). KGs are designed to organize and connect information in a human-readable and

machine-understandable way. KGs are commonly represented as a set of triples where each triple is defined by a Subject (head) and an Object (tail) entity and the relation that unites them. Figure 5 illustrates an example of what is an item KG.

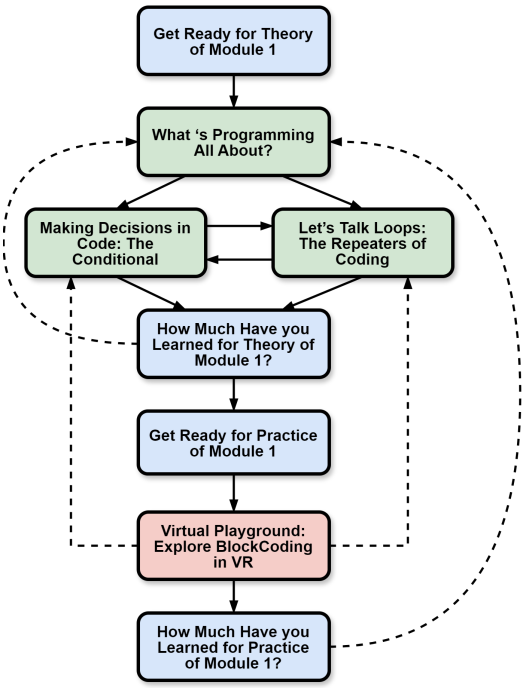

Figure 5. Example of an item Knowledge Graph based on the possible content of a course.

Although this data is designed to be understood by a machine, this raw data cannot be used as input to AI algorithms, it needs to be transformed first. This transformation seeks to translate raw data (text, numeric, etc.) into a data type that algorithms can use more appropriately. This translation, in a sense, extracts the original features from the data (in its own characteristic dimensional space) and translates them into a new feature space, with a new dimension (usually smaller than the original). These translated raw data are known as embeddings. These embeddings are numerical vectors that capture the inherent properties and relationships of the objects and are suitable for AI model training and analysis.

The process to be followed to obtain these embeddings may vary depending on several factors such as the order of the data, the relationships to be taken into account, etc. A common way to obtain them is by making use of some kind of neural network. For this option, the raw data will be the input data of the network. The embedding in this option could be calculated as the output obtained from the neural network without being processed, i.e. directly using the weights of the last layer of that neural network. Figure 6 illustrates the process for obtaining and embedding by using a neural network.

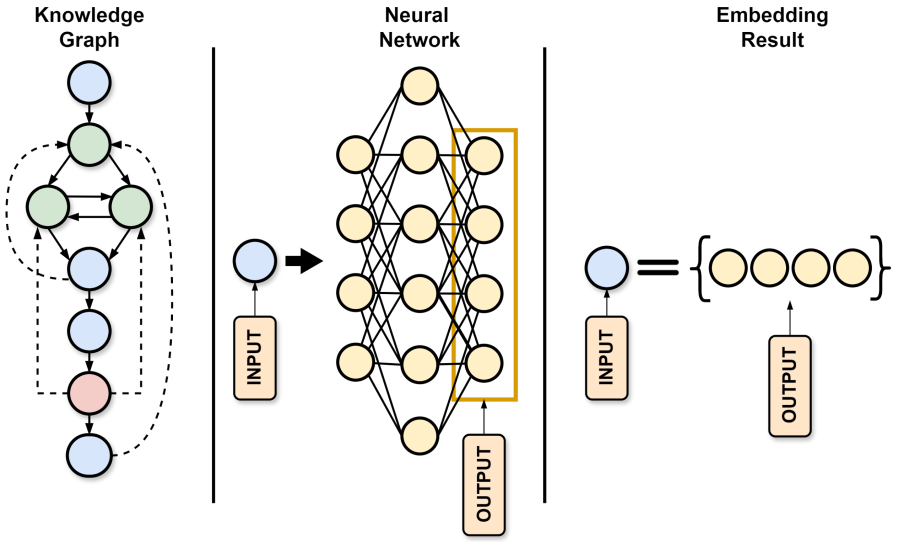

Figure 6. Workflow for obtaining the embedding of an item of a KG using a Neural Network.

If these embeddings correctly represent the data being translated, two completely different LOs, when viewed in their dimensional space, must be far apart. Conversely, two LOs that are very similar should be very close to each other. This allows the different LOs to be comparable to each other, which can be used to make recommendations based on resources similar to the last visited resource. These recommendations are completely independent of the user to whom they are being suggested, thus alleviating some of the drawbacks of collaborative filtering.

6. First Recommender Proposal

This section aims to describe how both RS, the CF and KBF algorithms, have been designed for this project and how they are hybridized together. After that, an explanation of how it can be used is provided. All the code is available at https://github.com/almtav08/recom_server. This main structure has been developed using Pytorch and is designed to be executed on both GPU and CPU.

6.1. Proposed Base Structure

The proposed RS has been developed using different modules, where each of them aims to lay the foundations of each of the developed RS. This modularity also allows for the flexibility and scalability of the RS, allowing it to be easily extended according to the needs to be met at any given time. This structure defines two main interfaces, and they serve as the core functionality of the RS workflow. Figure 7 illustrates the base structure of the different modules implemented for developing the hybrid RS.

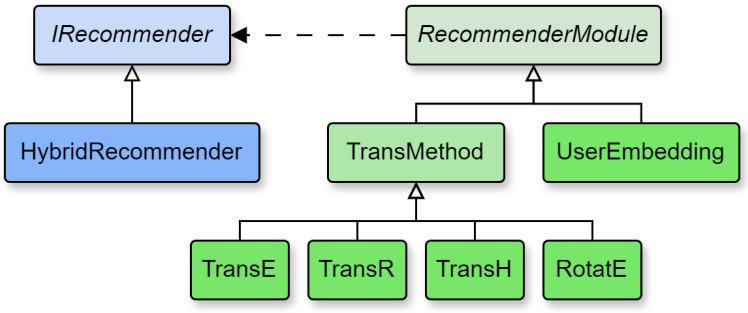

Figure 7. Proposed structure of the developed hybrid RS.

The *RecommenderModule* serves as the base to implement the different algorithms for preparing the data so the calculation of the similarities between the users or items and recommendations can be performed in the *IRecommender* interface. Each implemented module, that belongs to the *RecommenderModule* interface, must represent an algorithm for processing users' or items' data. This representation is pretended to be in the form of an embedding. Both users and items should be embedded to facilitate their comparison with different similarity metrics. For the implementation of CF and KBF two main modules have been defined:

- *TransMethod*: This serves as the base element to develop the algorithms that will be used in the KBF to process the item KG data. This model obtains the embeddings of the item KG entities and relations. Several modules have been developed using this *TransMethod* module.
- *UserEmbedding*: This serves as the base element to develop the algorithms that will be used in the CF to process the users' data. This model obtains the embeddings of the users based on their paths. Since calculating user embeddings may be a more flexible process, this is an open module that can be completely modified depending on the necessities the recommender system needs to address.

While the *TransMethod* module imposes more stringent structural requirements on its implementations, both modules are easily extensible with additional processing algorithms. This flexibility allows the system to adapt to diverse needs and advancements, irrespective of domain or constraints.

The *IRecommender* interface creates the base architecture to perform the recommendations calculation using the processed data obtained from the developed *RecommenderModule* implementations. Using the processed data of these modules, the similarities between the target user or item are performed and used for retrieving the individual recommendations of both modules. After obtaining these individual recommendations, the hybridization process is executed, thus obtaining the final recommendation set of items that will be given to the target user. This guarantees the enhancement of the quality of the overall recommendations assuring the possibility of using various combination strategies depending on the necessities of the RS.

6.2. Proposed Knowledge-based Algorithm

The KBF approach is developed using the *TransMethod* guidelines. The purpose of this algorithm is to analyze the data of the items of the KG for obtaining their embeddings. In the context of the project, this item KG where the entities are LOs, it is crucial to take the sequential order of the items into account for performing the recommendations. For this reason, defining the relations that will be present in the KG is a pivotal step for training the algorithm.

Taking Figure 5 as a basis, two types of relation between the LOs have been defined, the 'is previous of' (straight lines) and 'needs to repeat' (dotted lines) relations. These relationships compile some of the most common connections that occur in the education domain. The relation 'is previous of' indicates the ideal order that learners should follow at all times. The relation 'needs to repeat' indicates times when an LO may not be understood or failed, such as a quiz and the student must repeat an LO that reinforces their knowledge. These relations have to be defined in advance, usually by an expert.

For example, the triple ('Get Ready for Theory of Module 1', 'is previous of', 'What's Programming All About?') means that if students have visited the LO 'Get Ready for Theory of Module 1', their next step in the course should be to visit the LO 'What's Programming All About?'. On the contrary, the triple ('Virtual Playground: Explore BlockCoding in VR', 'needs to repeat', 'Making Decisions in Code: The Conditional') means that if the students fail or do not understand the LO 'Virtual Playground: Explore BlockCoding in VR' they should visit the LO 'Making Decisions in Code: The Conditional' to reinforce their knowledge about that topic. This seeks to serve as help for tutoring the students, as their path can be adjusted according to their performance. With the different types of connections between the LOs, it is possible to

determine if a learner struggles with a specific concept, and thus offering supplementary material or suggesting revisiting certain modules. This dynamic adaptation provides personalized tutoring, ensuring each learner receives the support they need at the right time, effectively functioning as a virtual tutor.

Several algorithms already exist for generating embeddings for the nodes and relations within a KG. The state-of-the-art methods for creating these embeddings have demonstrated their effectiveness in extracting relevant information from KGs. The proposed model is designed to be compatible with any of these embedding algorithms. The embedders that have been implemented for its use are the TransE, TransH, TransR, and RotatE (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015; Sun et al., 2019). Typically, these algorithms translate the entity using only an identifier or the entity name itself. Due to the possible limitations that this may bring, it has been decided to implement two different types of algorithms. Each of the knowledge embedders has two different implementations, which are as follows:

- Base Implementation: This represents the standard implementation of the algorithm, using only the identifier of the LO of the KG for obtaining the embeddings.
- Additional Information Implementation: This implementation integrates additional information about the items into the embedding process. This extra information, which can be a tag-like attribute, enhances the item's representation.

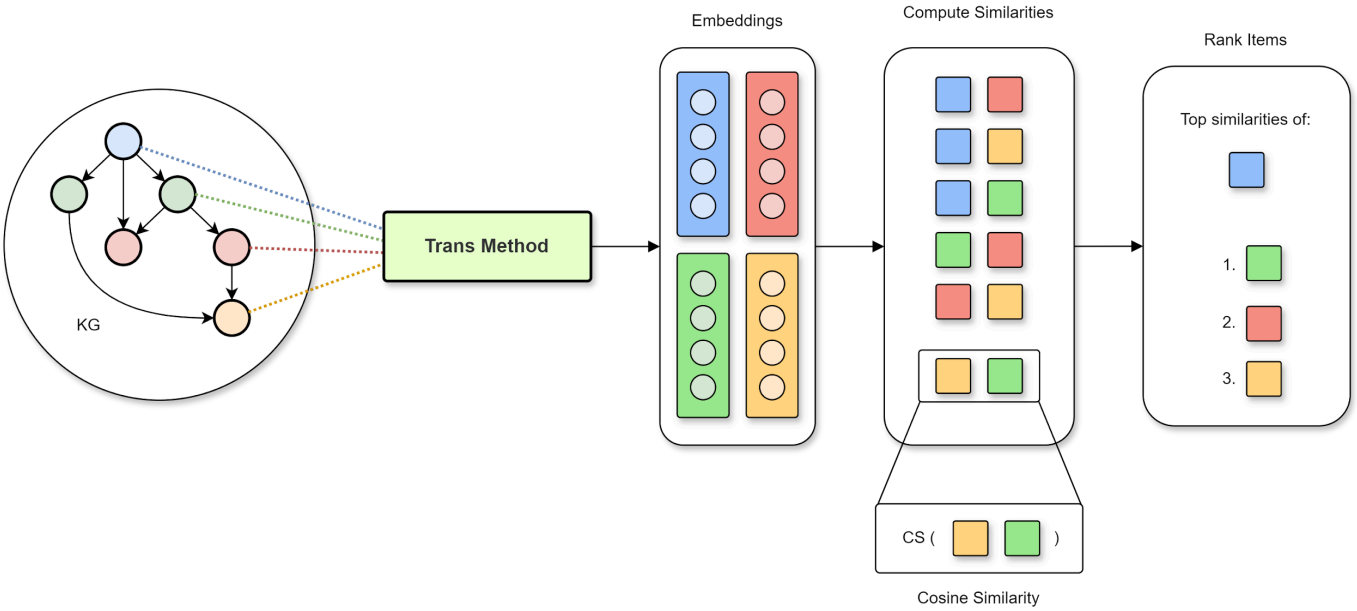

Figure 8. Proposed process for obtaining the recommendation list using the KBF technique.

Figure 8, illustrates the process for computing the similarities between the different LOs of the KG and therefore obtaining the recommendation list having a given LO as the target of the recommendation. Once the embeddings of the items are calculated with the desired *TransMethod*, in either of their implementations, the similarities between pairs of LO can be computed. This allows the system to select a LO as the target of the recommendation, compute the differences with every other LO, and obtain the top similar LOs that will be retrieved as the recommendations. The target LO is the last visited LO of the student that will receive the recommendations.

The recommendation process can be limited by defining different rules that somehow limit the LOs that are given as a recommendation. These rules can be defined according to the needs of the domain. For example, in the domain of education, one could limit certain recommendations based on competencies that the learner must visit to reach a particular LO or establish patterns of behavior in case he/she has failed a quiz.

6.3. Proposed Collaborative Algorithm

The CF is based on the usage of the *UserEmbedding* guidelines. The purpose of this algorithm is to analyze the data of the users' sequence of interactions for obtaining their embeddings. In the context of the project, clear ethical guidelines must be followed while treating users' data. For this reason, using the students' path, their privacy can be assured at any moment.we This decision implies that measures must be taken to be able to gather collaborative signals from users, and embedding their interaction paths is a technique that allows this to be achieved.

The user embedding model consists of a deep learning neural network with two layers. The first layer is an LSTM layer, which effectively handles the order in the input sequence. The second layer is a linear layer that generates the final embeddings. The path of the users will be given as an input to this network, and the weights of the final Linear Layer will be used as the users' embeddings. This technique allows the algorithm to translate any sequence of interactions, whatever the amount of interaction, into an embedding with the same dimension, thus allowing comparison between users.

As has been stated previously, this is an open module, compared with the *TransMethod* module. Several neural networks can retain sequential and temporal information such as RNNs, Transformers, etc. The internal structure defining the neural network of this module can vary in many ways and other methods of obtaining embeddings from user sequences can be explored. To ensure a minimum of reliability, the use of LSTMs has been chosen as they have proven their effectiveness in many of their applications.

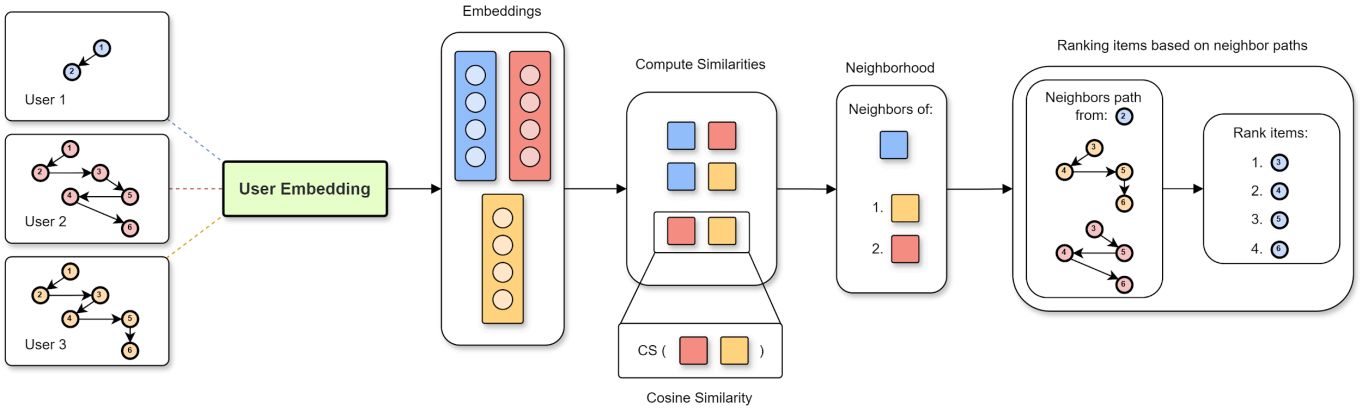

Figure 9. Proposed process for obtaining the recommendation list using the CF technique.

Figure 9 illustrates the process of computing similarities between different users' paths to generate a recommendation list for a target user. Once user embeddings are calculated using the desired UserEmbedding, the similarities between pairs of users can be determined. This enables the system to select a target user, compare them with other users in their neighborhood, and retrieve the top similar LOs as recommendations.

To recommend LOs after calculating user similarities, an additional step is necessary. Based on the paths of neighboring users, the next LO to visit is determined using the last LO (target LO) the target user visited. The system traverses all LOs that the neighbors visited after the target LO, and their association ratios, adjusted by the distance to the target LO, are used as scores to generate the recommendation list.

The usage of collaborative signals can have a critical impact in the engagement of the students. The system not only adapts students' path based on each learner's individual progress but also identifies patterns in the collective behavior of students. Understanding how students interact with the different LOs may give a glympse into which resources are more interesting for them and consequently adapt the recommendations based on this. This use of collective intelligence not only enhances personalization

but also optimizes learner engagement by surfacing content and activities that have a track record of boosting participation across the broader group.

Similar to the KBF algorithm, this recommendation process can be constrained by defining various rules that limit the LOs provided as recommendations. These rules can be tailored to the specific needs of the domain. For instance, in the education domain, recommendations could be restricted based on the competencies a learner has acquired to reach a particular LO or new interaction patterns could be established if the learner is at risk of failing the course.

6.4. Proposed Hybridization

Once all algorithms have been developed and trained, the final step to be able to provide recommendations to users is to combine all individual recommendation lists in a final list that will be given to the target user. Several strategies already exist for accomplishing this step and have been proven to work well in many RS regardless of the domain. Thanks to the modularity of the proposed system almost any of these strategies can be easily implemented. By default, the selected strategy is a weighting strategy.

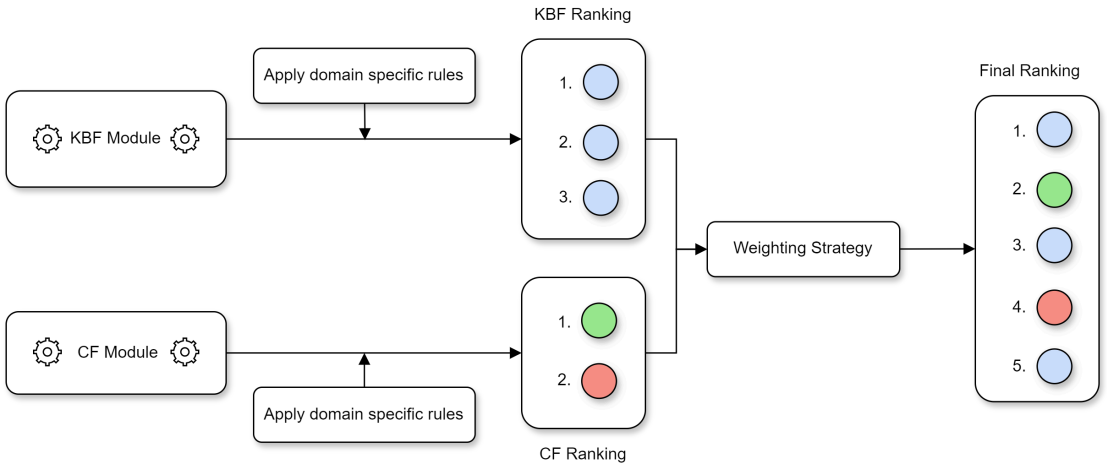

Figure 10. Proposed process for obtaining the final recommendation list combining the KBF and CF techniques.

Figure 10 illustrates the final process for generating the combined recommendation list. After both, the KBF and CF, modules have calculated the embeddings for items and users, their individual rankings are computed. Custom domain rules can be applied to each module to generate these rankings. Once the individual rankings are obtained, a weighting strategy fuses both sets of recommendations into a final list.

This weighting strategy assigns specific weights to each individual ranking, providing flexibility in generating recommendations. These weights can be adjusted based on the amount of data, the need to encourage students to visit essential LOs, or any other domain-specific requirements. The only stipulation is that all weights must sum to 100%. For example, a possible weighting might be 60% for the KBF module and 40% for the CF module.

These weights have been set as fixed values for the whole recommendation process, but there is also the possibility of creating a dynamic weighting system that varies at certain points in the course. These variations can be due to the needs of the students, the course, or any other type of need that may arise. In addition, it would also be possible to make a dynamic weighting system for specific students, which can further help them by better adjusting which LOs to visit.

One of the main purposes of combining these two algorithms is to be able to detect whether a student has difficulties with their learning journey an act in consequence. The system can monitor learner progression and adapt the assessments to match their current level of understanding. By analyzing both individual performance data and collaborative signals from other learners, the system can suggest appropriate assessments, such as quizzes or tasks, that target areas where the learner may be struggling or need reinforcement. This approach not only addresses immediate learning gaps but also works to continuously increase performance, helping students avoid the risk of failure. By ensuring that learners are tested on material they are prepared for, the system facilitates continuous improvement, providing fair evaluations and guiding them toward better results over time.

6.5. Algorithms Usage

The developed RS is intended to be integrated into the e-learning platform that has been created for the project, detailed in deliverables D4.1 and D4.2. Moodle has been established as a core module of this platform and for this reason, therefore, the decision was taken that RS should be connected to it to provide the recommendations to the students. Because of this decision, a communication system had to be created between these two elements. This communication has been carried out by using different requests through an API and by developing three important elements to complete the communication. The communication system has three main elements, which are the following:

- **Moodle Block e-Recommender**: This is a Moodle custom block that has been developed with the idea to ask the RS, through the API, the recommendations that must be given to the target user. It transforms the returned list of objects into an HTML list of items the student can interact with.
- **Moodle Plugin Log Event API**: This is a Moodle custom plugin with two important tasks to accomplish. The first one is to indicate the RS for all user interactions, allowing the adjustment of the RS and retrieving significant recommendations based on the user's last interacted LO. The second one is to create an API endpoint that allows the RS to obtain all interactions of a given course. This second task is supposed to be performed only for the training of the RS, facilitating access to all interaction data.
- **AI Server Hybrid Recommender**: This is the server where the RS is working and the main communication point between Moodle and the RS. The task of this element is to establish a series of endpoints that Moodle will use to communicate with the RS. A base structure of three endpoints has been established, but this can be extended according to specific needs, as long as the base architecture remains unchanged.

Once all these elements have been developed, Moodle and the RS can communicate bi-directionally to provide students with all the recommendations they need at all times. The AI server must be hosted so that Moodle can access it. The easiest way to achieve this is to host it on a publicly accessible server under the desired IP address. Both the block and the plugin can be configured to point to that IP address. Figure 11 illustrates how the recommendation list of a given user will be displayed in the Moodle course.

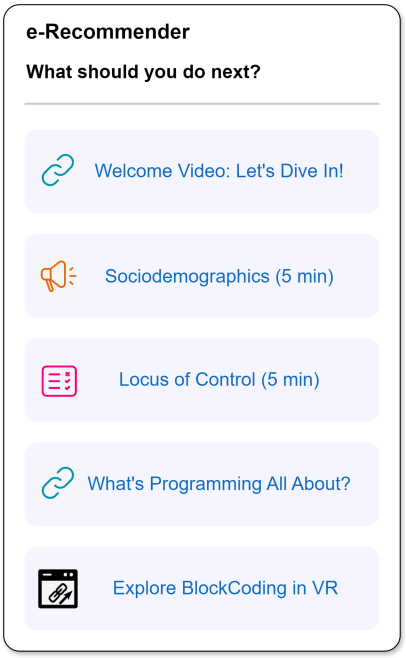

Figure 11. Final version of how the recommendation list is displayed in the Moodle course.

To start the server for the first time, train the different algorithms, and be able to carry out the recommendations, a series of scripts have been provided to simplify the initial data collection greatly. Details of how these should be executed are given at https://github.com/almtav08/recom_server. The code for the Moodle block can be found at https://github.com/almtav08/moodle_recommender_block and the code for the Moodle plugin at https://github.com/almtav08/moodle_recommender_plugin.

The following details the RS server API protocol that establishes the minimum communication pattern that has been developed.

6.5.1. The Retrain endpoint

This is a GET-type endpoint. The purpose of this endpoint is to adjust the recommendation model with the newest data available from the users. While the users interact with new data, their behavioral patterns can start mismatching the current ones, so it is crucial to adjust the users' embeddings at specific given times. The idea is to retrain the model regularly, either daily or weekly, to update user similarities and provide accurate recommendations over time.

6.5.2. The Append Log endpoint

This is a POST-type endpoint. The purpose of this endpoint is to update the last resource visited by the students. The body of this endpoint is composed of two elements:

- *userid*: The identifier of the user that has made the interaction (e.g., 1).
- *cmid*: The identifier of the Moodle course module the user has interacted with (e.g., 22). A Moodle course module is every element in the Moodle course that can be interacted with (e.g., quiz, file, url).

It is important to note that the logging will only work with course modules that are part of the recommendation corpus. It is common to see Moodle modules that are not used for the course itself but as a tool of communication between the teacher and the students. An example of this is the Announcements module.

6.5.3. The Recommendation endpoint

This is a GET-type endpoint. The purpose of this endpoint is to list the top recommendations of the given user. The parameters, that are used as path parameters, of this endpoint are composed by:

- *user*: The identifier of the user that has made the interaction (e.g., 1).
- *top*: The number of recommendations to be retrieved (e.g., 5).
- *message*: Boolean indicating if you want to retrieve the recommendation by a private message or by a regular API response.

To potentiate the flexibility of the RS, the possibility of making recommendations via direct messages on the Moodle platform has been added. This also helps users to decide how they prefer to interact with the RS and is more tailored to their needs.

7. Second Recommender Proposal

The method proposed in this section recommends personalised learning paths aimed at improving learning outcomes by promoting learning motivation and engagement. In educational recommender systems, the primary goal is often to optimize variables that impact learning outcomes rather than purely maximizing grades. Clustering students based on behaviors, interaction patterns, and learning success are key dimensions. The variables used for clustering can include *engagement metrics* (such as time spent on learning objects or interaction frequency) (Huang, A.Y.Q et a., 2023), *learning progress* (measured through assessments or task completion) (Nabizadeh,A. H. et al. 2020), and *cognitive profiles* based on knowledge or skill gaps (Oliveira, R. J. M., 2016)(George, G. & Lal, A.M., 2019). Recent research in the field highlights the growing use of hybrid approaches, combining collaborative filtering with additional layers of information to create more personalized and context-aware recommendations. Metrics related to user interaction within an LMS, such as active participation in discussions, completion rates, and content revisits, are frequently clustered to enhance personalization and increase engagement, which directly correlates to better learning outcomes.

In the context of the proposed algorithm, the path size refers to the number of LOs (Learning Objects) to recommend. The recommendation system is powered by information from two knowledge graphs. On one hand, a static knowledge graph with the course structure that relates the different LOs. On the other hand, a dynamically constructed graph based on user interactions within the course. The LO recommendation system is designed to utilise data generated by an LMS (Learning Management System) in a formal course. Specifically, for testing and validation, several LMS Moodle courses will be used.

To address the main objective, the problem has been divided into the following tasks:

1) Define a data model to represent the information provided by the LMS Moodle. (Define the knowledge graphs)

2) Build the knowledge graphs.

3) Adapt the recommendation based on users with similar behaviours. Analyse and select features to determine user behaviour.

4) Design a method to generate an optimised adaptive path of objects.

5) Recommend.

6) Evaluation.

7.1. User-based Collaborative Filtering

Our hypothesis for collaborative filtering suggests that recommending actions/interactions of "successful" users with behaviour patterns similar to the target user will result in more effective recommendations. This hypothesis is based on the idea that users who have demonstrated academic success have strategies and habits that other users can emulate to improve their own performance. Based on social learning theory, as well as empirical evidence demonstrated in previous studies, students tend to emulate those they consider models of success. By recommending the behaviour patterns of successful students, the system can promote practices that lead to better academic outcomes.

7.1.1. Feature Selection

The Moodle API offers, among several services, a web service to retrieve logs of system user activity. The goal is to recommend learning objects to a target user by emulating the past actions of users who were successful in the learning environment. To learn from user behaviour, it is necessary to do an exploratory analysis of the features and patterns collected in the Moodle logs. The aim is to check the significance of the variables collected in the logs to select those that allow predicting a student's success within the course. In the context of this study, student success refers to improvements in academic performance, engagement, and learning outcomes (Huang, A.Y.Q et a., 2023).

For the exploratory analysis of patterns in the data, firstly student logs data must be filtered. Subsequently, eliminate redundant and non-predictive features through data mining techniques as variables correlation analysis. In order to propose student behaviour features, we analyzed internal Moodle logs from three different completed courses. The number of records in the respective course logs were 5257, 128443, and 33314. However, the approach followed is just an example of feature selection, as it is possible to discover other types of patterns, as shown in (Huang, A.Y.Q. et al., 2023), to fine-tune the algorithm. The user activity variables selected for the correlation experiment were the frequency of access to LOs and the average visit time of an LO based on the final grades. In this way, the data can capture patterns that reflect engagement, actual evaluations, and the time factor. The grades were categorised as follows: failed, approved and outstanding. The study was carried out at different levels of the temporal dimension (day, month and year of the course). An analysis of two types of analysis was carried out, on the one hand a correlation analysis and on the other Multinomial Logistic Regression.

The correlation analysis demonstrated that the selected interaction characteristics are significantly associated with academic success, validating the potential effectiveness of recommendations based on these patterns. Results from the Multinomial Logistic Regression indicated that the type of component (module), total viewing time, and frequency of accesses are significant predictors of the final grade defined categories. The following aggregated variables best clustered the different data patterns for the selected user behaviour categories, with the first three variables being 'Quiz Frequency,' 'Total URL Time,' and 'Folder access Frequency,' in order of importance. This example highlights an approach to data analysis for determining the optimal number of clusters and applying collaborative filtering. Depending on the specific characteristics of each course—such as the type of learning objects involved—different features may emerge as more significant. The accuracy of collaborative filtering based on user behaviour is closely tied to the precision of the cluster analysis. While clustering is optional for the path recommendation algorithm proposed in this section, its implementation enhances recommendation personalization through user-based CF to target destination users.

7.1.2. User Clustering based on Graph Embeddings

In this work we represent user interactions with LOs as graph embeddings in order to capture the complex structural relationships between users and resources. Our goal in improving recommendations

is to create clusters for collaborative filtering based on users who have demonstrated success in a specific educational dimension of interest (for example, achieving a grade above a certain threshold, completing a module, or viewing a module, which can be considered as completion). In other words, reference users will be those whose module type access frequency and time patterns are most similar to the target user, and who are also having success. This type of monitoring should be done in real time to adapt the recommendations to the target user. Next, we detail how to implement the user clustering algorithm based on knowledge graph vectors.

For each student, a directed graph is constructed that relates the nodes traversed, which represents the path taken. The nodes represent the Learning Objects (LOs). The information in the graph represents all user interactions with the LOs aggregated at the user level up to the latest date data was collected. This graph is constructed dynamically, meaning it updates as new interactions between users and LOs occur. Therefore, the recommendation algorithm will be dynamic and the collaborative filtering mechanism will be updated based on historical interactions to date. Each edge represents the transition between two LOs; for example, the student visits LO2 after LO1. Each edge is assigned metadata indicating the number of times the LO2 node has been visited from LO1. Each LO node will have the following metadata: average time spent on the node, average visit frequency, type of LO (expository, evaluative), and the corresponding status according to the type of LO. For an expository LO, the status can be 'completed or not completed' (or seen or not seen), while for an evaluative LO, the status can take three values: 'fail, pass, or outstanding'. Figure 12 represents an example of the interactions graph of 5 students.

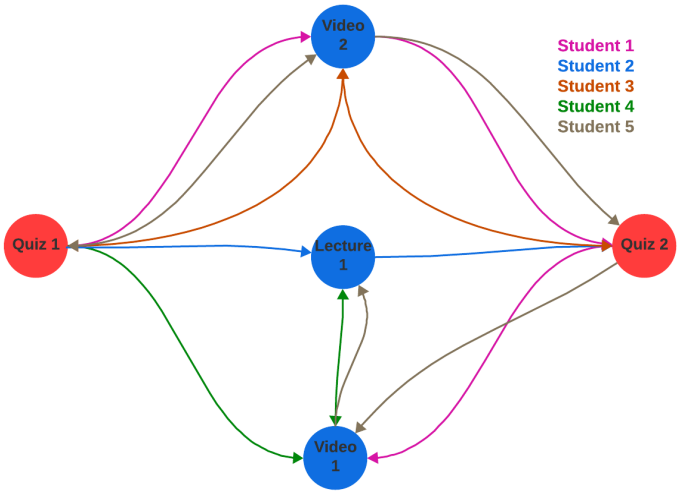

Figure 12. Example of the interaction graph of 5 students.

For creating the user graph embeddings, the *UserEmbeddings* method, discussed in sections 6.1 and 6.3, can be used. As mentioned in those sections, other graph embedding methods can be explored to achieve specific goals.

To determine the optimal number of user clusters for the dataset the Elbow Method can be used. The Elbow Method evaluates the sum of squared distances within clusters (inertia) for different values of k (number of clusters). The objective is to identify the value of k where the decrease in inertia begins to slow down (the 'elbow' in the graph), indicating the optimal number of clusters.

7.2. Path Recommender Algorithm

The aim of the recommender model is to provide personalised learning object (LO) recommendations on an e-learning platform. We use a graph-based approach and shortest paths, weighting user interactions, LO characteristics and teacher influence to identify the most effective and relevant learning sequences. The system aims to recommend a path of n ordered LOs tailored to the characteristics of the target learner. The purpose of the method is from a defined starting point, e.g. a given LO, to recommend an adapted path of LOs to follow that maximises the learner's grade.

In order to generate the optimised path we propose the following hypothesis: starting from a specific LO, recommending successful actions that have been performed by users similar to the target user will improve the academic performance of the target user. Successful actions are selected by a threshold defined on the basis of a metric that evaluates learning outcomes, such as 'grades' obtained, or the 'completed' or 'not completed' status of a LO. A collaborative filtering approach based on knowledge graphs is proposed for this purpose.

The proposed collaborative filtering method utilises the shortest path problem from graph theory (Madkour, A. et al, 2017). The shortest path problem involves finding a path between two vertices or nodes such that the sum of the weights of the edges constituting the path is minimised. Specifically, the proposed solution uses Dijkstra's shortest path algorithm (although other shortest path algorithms could be implemented) to find the fixed-length path of n with the minimum 'distance' (which represents the maximum score in our approach). In summary, the implemented algorithm weights the edge between two vertices of a directed graph, where the vertices represent the user's transition from $LO1$ to $LO2$, based on the 'importance' (which can be explicit or implicit) given to this transition by both students and teachers. The greater the number of students, similar to the target user, who have achieved academic success (either through exam scores or content viewing) and validate the transition between two vertices, the higher the edge's weighting. Then, the inverse of the scores is calculated as weights to minimise the edge lengths. This transformation is done to prioritise the shortest path. Thus, the system will recommend the shortest path for n elements that represent optimised transitions.

The implemented recommendation model considers both user interactions and the course design provided by the instructor. In the absence of sufficient user interactions, recommendations are based on the knowledge graph defined by the instructor, ensuring that relevant recommendations are always provided. When there is sufficient interaction data, the model integrates both the interactions of similar users and the instructor's recommendations to optimise the suggested learning sequence.

The representation of user interactions in a directed graph allows for the reflection of implicit data that are very important in the formal academic context, such as the sequential access to educational resources. Additionally, the adaptation of Dijkstra's shortest path search algorithm over the resulting graphs allows for finding an optimised path, maintaining the sequential order and giving greater relevance to paths validated by both the majority of students and the instructor. This not only ensures that the recommendations follow a coherent pedagogical logic but also align with proven study practices that have been successful for other students. Thus, the recommendation system can personalise learning paths to maximise the academic success of the target user, providing a structured and data-driven approach to the continuous improvement of educational performance.

7.2.1. Algorithm Summary

1. Automatic construction of both the course and user interactions Knowledge Graphs:

- The course knowledge graph construction can be very useful to solve the cold start problem when there are no user interactions yet. In order to create this graph, the approach mentioned in section 5.2 could be used by adjusting the data needed for the shortest path algorithm. This

graph is optional if only user interactions are considered; this can be parametrized in the final implementation.
- A directed graph is constructed where the nodes represent the LOs and the edges represent the transitions between LOs based on student interactions.
- Each edge has metadata that indicates the number of visits, and each node has metadata such as average time, average frequency, type and state of the LO.

2. Shortest Path Algorithm: Dijkstra's algorithm is adapted to find the shortest path based on the number of visits and other relevant weights in the educational field.

3. Generation of Recommendations:

- Similar users in the same cluster are identified.
- A subgraph is built with the interactions of similar users.
- The shortest path algorithm is used to find and recommend a path of n LO for the destination user.

Considerations:

1. Cold Start problem: If there are not enough interactions (less than the threshold), the system uses only the course knowledge graph provided by the teacher.

2. Initial interactions: If there are some interactions but they do not reach the minimum threshold, it is considered a single cluster with all users so far.

3. Sufficient interactions: Once the minimum thresholds are exceeded, clustering techniques are applied to group similar users and use their interactions to improve recommendations.

Next, the learning object recommendation model for a formal educational environment is described through formal definitions.

7.2.2. Formal description of the path recommendation method

1) Definition of the User Interaction Graph

A directed graph $G = (V, E)$ is defined, where:

- V is the set of nodes, representing the LO.
- E is the set of edges, representing the transitions between LOs based on user interactions

2) Weights estimation

Each User u and each LO lo have associated weights regarding different aspects:

- wu: Weight of the user u, calculated based on their interaction and overall ratings.
- wlo: Weight of the LO, based on its popularity and usage.
- wp, lo: Weight assigned by the teacher (the expert) to a specific LO.
- $wp, (lo1, lo2)$: Weight that the teacher gives to the access order between two specific LOs.

3) Node similarity estimation

The node similarity formula captures the relationship between two Learning Objects (LO) based on the ratings (implicit importance) attributed by users. In the context of the proposed recommendation system, for simplicity we define that the ratings are always 1, that is, each LO will have the same value among users (students). The proposed formula can be adapted and assigned a rating value per user based on implicit data such as the frequency of visits and other associated weights. In scenarios where

ratings are not uniform, the similarity formula captures how close the ratings are between two LOs. If the ratings are similar, the exponential of the difference is small, resulting in a higher similarity value. This indicates a strong relationship between the LOs.

To capture the relationship between two LOs $lo1$ and $lo2$, in addition to the user ratings, we added the importance assigned by the teacher. We define importance as the weight that the teacher assigns to that LO in the knowledge graph. The similarity formula between LO is as follows:

$$sim(lo1, lo2) = \frac{1}{e^{(|(rating(lo1)*wp,lo1)-(rating(lo2)*wp,lo2)|)}}$$

Using the exponential function penalises large differences in grades more severely than a linear function. This is useful to reduce the influence of LOs with disparate ratings on the recommendation. Maintaining the similarity formula allows flexibility to extend the model in the future. If it is decided to include different ratings for the LOs, the formula will automatically adjust the similarity, maintaining the relevance of the recommendations.

4) Visit order frequency

To capture the frequency with which users visit LOs pairs in a specific order, we define the visit order frequency $freq\_order(lo1, lo2)$ as:

$$freq\_order(lo1, lo2) = f_u(lo1 \rightarrow lo2)$$

$f_u(lo1 \rightarrow lo2)$ is the frequency with which user u has transitioned from $lo1$ to $lo2$. In other words, for each user the frequency with which they visit that targeted link is calculated.

5) Edge Weight

The weight of the edge between two nodes $lo1$ and $lo2$ in the graph G is calculated as:

$$w_{ij} = \sum_{u \in U} (sim * wu * wlo2 * freq\_order(lo1, lo2) * wp(lo1, lo2))$$

To adapt the edge weight for use with Dijkstra's algorithm (which requires smaller weights for stronger connections) the inverse of the edge weight is calculated. This ensures that paths with higher scores (more significant and common transitions) have smaller weights, allowing Dijkstra's algorithm to prioritise these paths.

$$inverse\_weight_{ij} = \frac{1}{w_{ij}}$$

Where:

U: Is the set of users.

sim: Similarity between LOs in an edge.

wu: User weight.

$wlo2$: Weight of second LO.

$freq\_order(lo1, lo2)$: Frequency of visits in specific order.

$wp(lo1, lo2)$: Weight that the teacher gives to the access order between $lo1$ and $lo2$.

This formula ensures that the weight of the edge wij is a sum of the resulting weights for each user who uses the path between $lo1$ and $lo2$.

6) Shortest Path Algorithm

To find the optimal path in the weighted graph G, Dijkstra's algorithm using a priority queue was employed, although other shortest path algorithms could be tested. This algorithm computes the shortest path from an initial node s to all other nodes in terms of the sum of the edge weights.

Dijkstra's algorithm is a classic graph theory method for finding the shortest path from a source node to all other nodes in a graph with non-negative weights. The algorithm starts by initialising the distance to the source node to zero and all other nodes to infinity. It then uses a priority queue (often implemented as a min-heap) to iteratively select the unvisited node with the shortest known distance. Upon selecting each node, the algorithm updates the distances to its neighbours if a shorter path is found. This process is repeated until all nodes have been visited or the shortest path to the destination node has been determined.

To sum up, this comprehensive approach combines graph theory, machine learning, and data-driven pedagogy, providing a robust and effective method for personalising learning in e-Learning environments.

7.3. Implementation in Moodle

The sequence diagram in figure 13 shows the intercommunication between the different modules of the proposed Recommender System and the end user (the student). Arrows with solid lines indicate a method call or message that waits for a response before continuing (synchronous message), while arrows with dashed lines indicate asynchronous communication, where an immediate response is not expected. The first module consists of a block-type plugin developed for Moodle. Each time the user interacts with a learning object, the recommendation block will show a recommended path to follow starting from the last visited element. Moodle will be installed on a web server with Apache, PHP, and MySQL. The second module is the Recommendation System, which will be hosted on a server we call the AI Server, with Python installed. This module is composed of the "Recommender" object, which is responsible for training the models and recommending possible learning paths to visit, and the FastAPI web server, which provides a series of API services necessary for communication with external systems, such as Moodle.

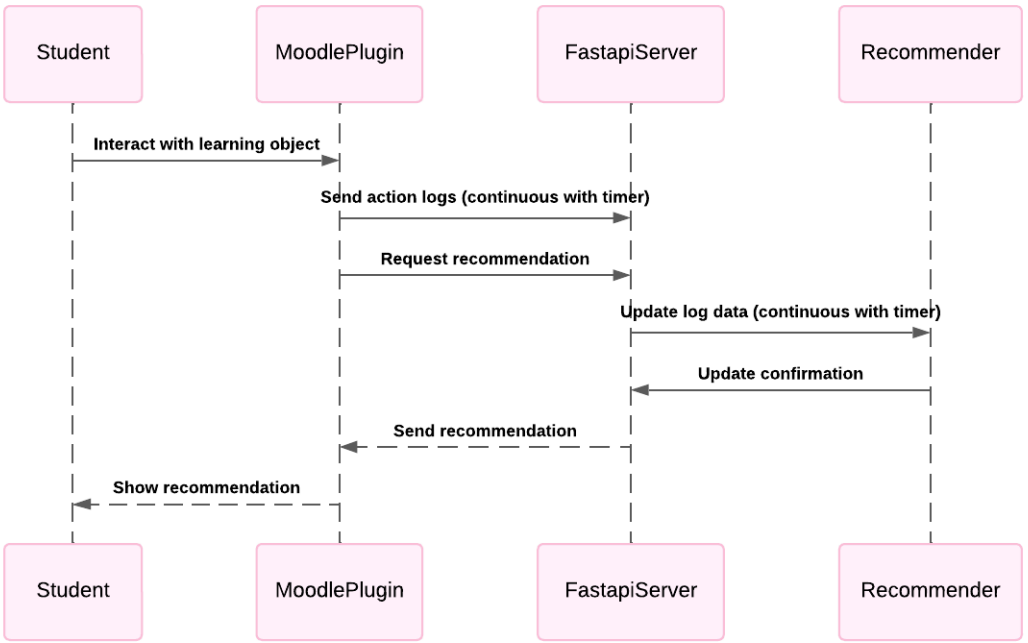

Figure 13. Sequence diagram of the Recommender System and the student.

The interaction between systems is described at a high level in the following sequence of steps. Initially, the student interacts with a learning object in a Moodle course through the recommendation block. The Moodle plugin calls a web service on the FastAPI server that allows sending the recent interaction logs of the students to the AI Server. The sending of the logs is continuous and regulated by a timer. Additionally, each time the user interacts with a learning object, the Moodle plugin makes a request to the FastAPI server to recommend new learning objects to visit. On the other hand, the FastAPI service continuously loads the logs of the students sent by Moodle, as defined by a timer. Once these data are read, the API service updates the "Recommender" object to train it with the new data. Once the "Recommender" object is updated, it sends a recommendation each time Moodle makes the corresponding request.

The block-type plugin implemented for Moodle is named 'Actions Recommender' and can be downloaded from the following repository published on GitHub: https://github.com/indilanza/actions_recommender. In the same repository you will find the instructions for its installation. Once installed in Moodle, you must access the plugin configuration options and set the IP address and port that hosts the recommendation system server. Figure 14 shows the screen in Moodle where these data are configured. Figure 15 shows the "Actions Recommender" block. Based on the requirements defined in the recommendation system for the e-Diploma project, specifically for the course related to learning programming using virtual reality and augmented reality techniques, the "Actions Recommender" block will only be shown to users belonging to the experimental group.

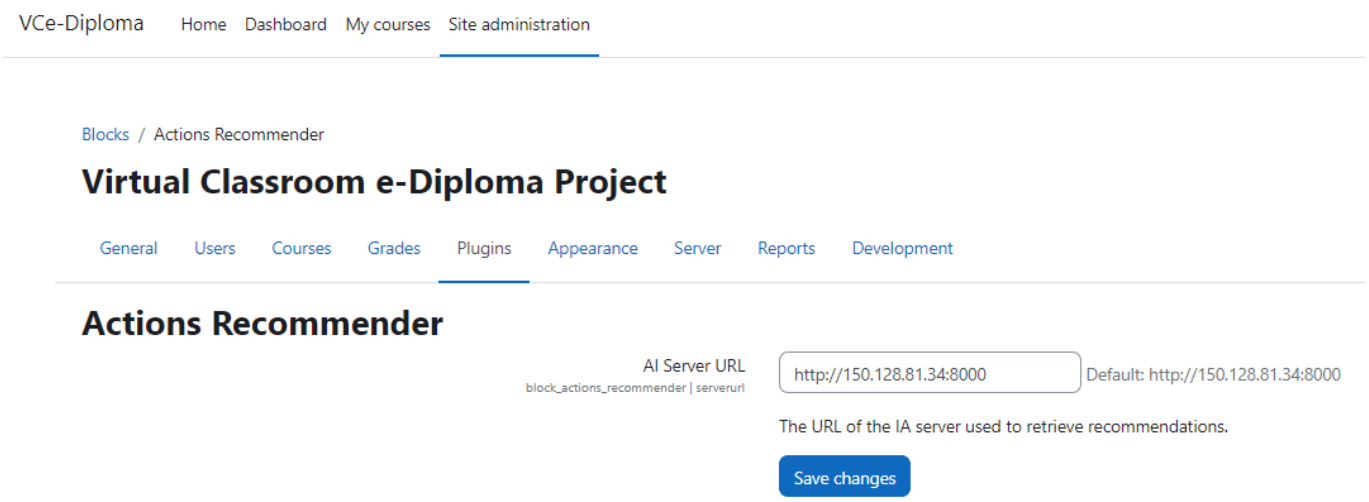

Figure 14. Settings for Actions Recommender block plugin.

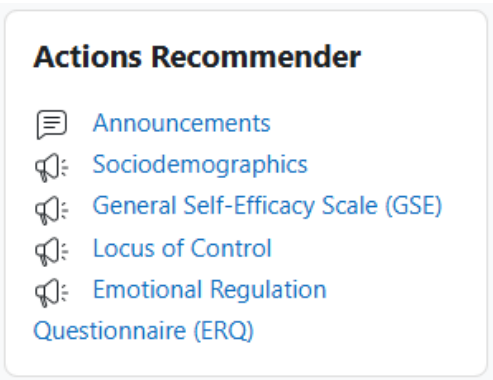

Figure 15. Settings for Actions Recommender block plugin.

The source code to deploy the Recommender System is published in the following directory: https://github.com/indilanza/LOPathRecomenderProject. In the same repository, it is documented how to set up the Recommender System service. Additionally, it documents how the different modules of the project communicate with each other, the relationships between the implemented classes, and their main functions.

8. Conclusion

The implementation of personalised recommendation systems in the educational context is crucial for adapting content to the individual needs of students. This personalization not only increases motivation and participation but also optimises academic results. The study of the state of the art in educational recommendation systems allowed us to identify complex issues, such as the need to generalise models so they can be used in various educational settings. Additionally, the proposed solutions focus on addressing the need to organise and sequence recommendations to guide users' learning processes. To ensure data protection policies and the ethical use of data in AI applications, models were developed that anonymize user data and use only activity traces. The primary conclusion is that the most effective Recommender System models within the scope of this project are those that utilise Collaborative Filtering (CF) and Knowledge-Based Filtering (KBF). This document proposes two systems to solve these needs and provides guidelines on how to integrate them into Moodle, offering a detailed framework for seamless implementation.

The first recommendation algorithm proposed is a hybrid system that combines CF and KBF. This proposal emphasizes the sequential nature of user interactions and the inherent order of the items to be recommended. By dynamically integrating the chronological sequence of user behavior and the sequential dependencies among items, the system ensures that recommendations are both contextually relevant and temporally aligned with user preferences. This approach recognizes the need for items to be interacted with in a specific order. By capturing temporal patterns and ordered relationships, the system can more effectively adapt to evolving user preferences. This adaptive capability is crucial for improving user satisfaction and engagement, as it ensures that recommendations remain pertinent over time.

A second recommendation algorithm has been proposed, which is user-based CF and the use of knowledge graphs. This algorithm can be parameterized with both static and dynamic information to construct personalised and ordered learning paths. On one hand, personalization is promoted through the use of user-based collaborative filtering techniques. In other words, learning paths of users with a certain level of academic success, whose behaviour patterns were most similar to the target user, will be recommended. On the other hand, the order of access to resources is a very important aspect in e-learning and has been little addressed in the scientific literature. Moreover, applying it at the level of a single course presents complexities due to the order constraints defined by the expert who designs the course within an LMS like Moodle. In this sense, in a second stage, the recommendation model makes use of graph theory and a shortest path search algorithm, predicting the most probable and relevant sequence. The relevance of the recommended paths will be a weighted combination of the importance assigned to each learning object by both the student and the teacher.

Both recommendation systems can be effectively adapted as an intelligent tutoring system (ITS) due to its ability to personalise learning paths This hybridization techniques mimics the behaviour of a human tutor by dynamically adapting the learning process to the student's needs, offering timely recommendations based on peer success and guiding the student toward optimal learning outcomes. The adaptive and data-driven nature of the presented systems aligns with the core principles of intelligent tutoring systems, providing continuous support and improving the learning experience.

This project highlights the importance of collaboration between technologists and educators to develop innovative solutions that improve the quality of education. The challenges addressed during implementation, and subsequently during the evaluation and commissioning of the systems, will serve as opportunities for learning and continuous improvement. The results of this project contribute to the development of more sophisticated and effective educational technologies.

9. References

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko O. (2013). Translating embeddings for modeling multi-relational data. *in Advances in Neural Information Processing Systems, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds*., vol. 26. Curran Associates, Inc.

da Silva, F.L., Slodkowski, B.K., da Silva, K.K.A. et al. (2023). A systematic literature review on educational recommender systems for teaching and learning: research trends, limitations and opportunities. *Educ Inf Technol* 28, 3289–3328 (2023). https://doi.org/10.1007/s10639-022-11341-9

George, G., & Lal, A.M. (2019). Review of ontology-based recommender systems in e-learning. Comput. Educ., 142. https://doi.org/10.1016/j.compedu.2019.103642

Huang, A. Y. Q., Lu, O. H. T., & Yang, S. J. H. (2023). Effects of artificial Intelligence–Enabled personalized recommendations on learners' learning engagement, motivation, and outcomes in a flipped classroom. Computers & Education, 194, 104684. https://doi.org/10.1016/j.compedu.2022.104684

Ismail, H. M., Belkhouche, B., & Harous, S. (2019). Framework for personalized content recommendations to support informal learning in massively diverse information Wikis. *IEEE Access*, 7, 172752–172773. https://doi.org/10.1109/ACCESS.2019.2956284

Javed, U., Shaukat, K., Hameed, I.A., Iqbal, F., Alam, T.M., & Luo, S. (2021). A Review of Content-Based and Context-Based Recommendation Systems. *Int. J. Emerg. Technol*. Learn., 16.

Joy, J., & Renumol, V.G. (2021). An ontology-based hybrid e-learning content recommender system for alleviating the cold-start problem. *Education and Information Technologies*, vol. 26, 4993 - 5022.

Khanal, S.S., Prasad, P., Alsadoon, A. et al. (2020). A systematic review: machine learning based recommendation systems for e-learning. *Educ. Inf. Technol.* vol. 25, 2635–2664. https://doi.org/10.1007/s10639-019-10063-9

Li, L., Zhang, Z., & Zhang, S. (2021). Hybrid algorithm based on content and collaborative filtering in recommendation system optimization and simulation. *Scientific Programming*. https://doi.org/10.1155/2021/7427409

Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, Z. (2015). Learning entity and relation embeddings for knowledge graph completion. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2 2015 https://doi.org/10.1609/aaai.v29i1.9491

Madkour, A., Aref, W.G., Rehman, F.U., Rahman, M.A., & Basalamah, S.M. (2017). A Survey of Shortest-Path Algorithms. ArXiv, abs/1705.02044.

Mawane, J., Naji, A., & Ramdani, M. (2020). Unsupervised Deep Collaborative Filtering Recommender System for E-Learning Platforms. *In: Hamlich, M., Bellatreche, L., Mondal, A., Ordonez, C. (eds) Smart Applications and Data Analysis.* SADASC 2020. Communications in Computer and Information Science, vol 1207. Springer, Cham. https://doi.org/10.1007/978-3-030-45183-7_11

Muangprathub, J., Boonjing, V., & Chamnongthai, K. (2020). Learning recommendation with formal concept analysis for intelligent tutoring system. Heliyon, 6(10). https://doi.org/10.1016/j.heliyon.2020.e05227

Nabizadeh, A. H., Gonçalves, D., Gama, S., Jorge, J., & Rafsanjani, H. N. (2020). Adaptive learning path recommender approach using auxiliary learning objects. Computers & Education, 147, 103777–103793. https://doi.org/10.1016/j.compedu.2019.103777

Oliveira, R.J.M. (2016). Recommender system for an e-learning platform (Master's thesis). Departamento de Ciência de Computadores, Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos.

Rahman, Mohammad Mustaneer and Abdullah, Nor Aniza (2018). A personalized group-based recommendation approach for web search in E-learning. *IEEE Access*, 6. pp. 34166-34178. ISSN 2169-3536, DOI https://doi.org/10.1109/ACCESS.2018.2850376

Roy, D., Dutta, M. (2022). A systematic review and research perspective on recommender systems. *J Big Data* 9, 59 . https://doi.org/10.1186/s40537-022-00592-5

Sun, Z., Deng, Z.-H., Nie, J.-Y., and Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space.

Tahir, S., Hafeez, Y., Abbas, M.A. et al. (2022). Smart Learning Objects Retrieval for E-Learning with Contextual Recommendation based on Collaborative Filtering. *Educ. Inf. Technol.* vol 27, 8631–8668. https://doi.org/10.1007/s10639-022-10966-0

Tarus, J. K., Niu, Z., & Yousif, A. (2017). A hybrid knowledge-based recommender system for e-learning

based on ontology and sequential pattern mining. Future Generation Computer Systems, vol. 72, 37–48.

https://doi.org/10.1016/j.future.2017.02.049

Vedavathi, N., & Anil Kumar, K. M. (2022). SentiWordNet ontology and deep neural network based collaborative filtering technique for course recommendation in an e-learning platform. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 30(4), 709-732. https://doi.org/10.1142/S0218488522500192

Wan, S., & Niu, Z. (2020). A hybrid E-Learning recommendation approach based on learners' influence propagation. *IEEE Transactions on Knowledge and Data Engineering*, vol 32(5), 827–840. https://doi.org/10.1109/TKDE.2019.2895033

Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge graph embedding by translating on hyperplanes. *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, Jun. 2014. https://doi.org/10.1609/aaai.v28i1.8870

Widayanti, R., Chakim, M., Lukita, C., Rahardja, U., & Lutfiani, N. (2023). Improving Recommender Systems using Hybrid Techniques of Collaborative Filtering and Content-Based Filtering. *Journal of Applied Data Sciences*, 4(3), 289-302. doi:https://doi.org/10.47738/jads.v4i3.115

Xu, G., Jia, G., Shi, L., & Zhang, Z. (2021). Personalized course recommendation system fusing with knowledge graph and collaborative filtering. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2021/9590502

e-DIPLOMA

Funded by
the European Union