UNIVERSIDAD DE LA REPÚBLICA URUGUAY

PEDECIBA

Collaborative Recommendations in Virtual Learning Environments

Daniel González Bernal

Master's Thesis

Programme for the Development of Basic Sciences (PEDECIBA)
University of the Republic

Montevideo, Uruguay
May 2019

Abstract

Recommender systems are software applications that aim to suggest meaningful and useful items to users when interacting with large volumes of data such as online multimedia content, news, products, among others. These systems generate personalized recommendations of items based on the explicit or implicit preferences expressed by users, as well as information about their collaborations and relationships. There are numerous fields of application for these systems such as e-commerce, social networking, online streaming, among others, playing a key role in virtual learning environments. Recommender systems assist students in finding appropriate educational items, enhancing their learning experience and academic results. Two previous research projects (QHIR LACCIR and UTU) with limited scope, provided experience in different aspects of the development of recommendation algorithms which process information about socially-connected users and their collaborations. Based on those positive results obtained, this thesis proposes mechanisms to generate personalized recommendations to students through the development of a complete recommender system, using information about their collaborations in virtual learning environments. The aim is to describe and specify in detail how different approaches, techniques, and recommendation algorithms are used to produce meaningful recommendations in real life educational scenarios. More specifically, it focuses on personalized recommendations for medical doctors and other health specialists enrolled in online Continuing Medical Education (CME) courses through a virtual learning environment called RedEMC. A general conclusion based on user feedback given by students, is that collaborative recommendations enhance each students' learning process and user's experience, stimulating further collaboration. It is worth to highlight that explicit feedback given by RedEMC's users stated that more than 90% of the recommendations of resources/activities and comments were rated positively.

Keywords: Recommender systems, recommendation algorithms, virtual learning environments.

Resumen

Los sistemas recomendadores son aplicaciones de software que tienen como objetivo sugerir elementos significativos y útiles para los usuarios cuando interactúan con grandes volúmenes de datos, como contenido multimedia en línea, noticias, productos, entre otros. Estos sistemas generan recomendaciones personalizadas de elementos basándose en las preferencias explícitas o implícitas expresadas por los usuarios, así como información sobre sus colaboraciones y relaciones. Existen numerosos campos de aplicación para estos sistemas, como el comercio electrónico, las redes sociales, la transmisión en línea, entre otros, desempeñando un papel clave en entornos de aprendizaje virtual. Los sistemas recomendadores ayudan a los estudiantes a encontrar elementos educativos apropiados, mejorando su experiencia de aprendizaje y resultados académicos. Dos proyectos de investigación anteriores (QHIR LACCIR y UTU) con alcance limitado, proporcionaron experiencia en diferentes aspectos del desarrollo de algoritmos de recomendación que procesan información sobre usuarios socialmente conectados y sus colaboraciones. Basados en esos resultados positivos obtenidos, esta tesis propone mecanismos para generar recomendaciones personalizadas para los estudiantes a través del desarrollo de un sistema recomendador completo, utilizando información sobre sus colaboraciones en entornos de aprendizaje virtual. El objetivo es describir y especificar en detalle cómo se pueden utilizar diferentes enfoques, técnicas y algoritmos de recomendación para producir recomendaciones significativas en escenarios educativos de la vida real. Más específicamente, se enfoca en recomendaciones personalizadas para médicos y otros especialistas de la salud inscriptos en cursos en línea de educación médica continua a través de un entorno de aprendizaje virtual denominado RedEMC. Una conclusión general basada en la retroalimentación dada por los usuarios es que las recomendaciones colaborativas en entornos virtuales de aprendizaje mejoran el proceso de aprendizaje de los estudiantes y la experiencia de usuario de cada estudiante, estimulando una mayor colaboración. Cabe destacar que la retroalimentación explicita dada por los usuarios de RedEMC afirmaron que más del 90% de las recomendaciones de recursos/actividades y comentarios fueron calificadas positivamente.

Palabras clave: Sistemas recomendadores, algoritmos de recomendación, entornos de aprendizaje virtual.

Related Papers

This thesis includes and expands the following papers:

- D. González, L. Tansini, "Collaborative Learning Recommendations for Continuing Medical Education in Virtual Learning Environments", presented at Co-creation of Curricula, Tools and Educational Scenarios for Building Soft Competences for Personal Development and Employability Conference (Co-Create! 2018), Tartu, Estonia, 17th September 2018.

- A. Margolis, A. López-Arredondo, S. García, N. Rubido, C. Caminada, D. González, L. Tansini, "Social Learning in large Online Audiences of Health Professionals: Improving Dialogue with Automated Tools", MedEdPublish, 2019. [Online] Available: https://www.mededpublish.org/manuscripts/2258

Acknowledgments

I would like to thank my master' thesis supervisor, Dr. Libertad Tansini, for her complete support, guidance, and immense knowledge.

A very special gratitude goes to Dr. Alvaro Margolis, Antonio López-Arredondo and Camilo Caminada, for their cooperation and feedback.

I would also like to thank my family for their continuous support and constant encouragement throughout the process of researching and writing this thesis.

Index

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents an introduction to the thesis and the motivation for this work. Then, the problem statement is described in order to explain the challenges addressed in this research project regarding the development of a complete recommender system. Finally, the outline of the thesis is presented.

1.1 Introduction

Recommender systems are software applications that aim to suggest meaningful and useful items to users when interacting with large volume of data such as online multimedia content, news, products, among others [1]. These systems generate personalized recommendations of items based on the explicit or implicit preferences expressed by users, as well as information about their collaborations and relationships. There are numerous fields of application for these systems such as e-commerce, social networking, online streaming, among others, playing a key role in virtual learning environments. Recommender systems assist students in finding appropriate educational items, enhancing their learning experience and academic results.

1.2 Motivation

Higher educational institutions generally employ virtual learning environments to offer educational items to students, as well as an environment for communication and collaboration between students and teachers within courses [2]. Some virtual learning environments such as *Moodle*, have the possibility to provide limited recommendations to students by installing specific plugins [3, 4, 5]. In general, these recommendations are non-personalized and based on popular preferences. The information contained in virtual learning environments could be used to generate useful suggestions of items [6]. Specifically, collaborations are valuable and rich sources of information which could be used to generate personalized recommendations of educational items. In that sense, collaborative recommendations take advantage of the interactions between people in a specific environment.

The main motivation of this thesis is to propose mechanisms to process information about collaborations in order to produce meaningful recommendations, focusing on the relationships between students in the virtual learning environment. This research work analyzes that these personalized suggestions of educational items enhance each students' learning process and user's experience.

1.3 Problem Statement

A local company called *Evimed* provides Continuing Medical Education (CME) courses to medical doctors and other health specialists, through its virtual learning environment called *RedEMC* [7]. In 2017, Evimed started working in a research project which aims to help students in finding meaningful and useful educational items and to assist students in discovering new CME courses of their interest [8]. In order to achieve these aims, Evimed intended to incorporate a recommender system which offers personalized recommendations through RedEMC.

The challenges and aims of this research work are:

- To understand the objectives of the research project and define functional and non-functional requirements according to Evimed's needs.
- To analyze the information stored in RedEMC, such as types of users, types of educational items, user's profile data, and relationships between users.
- To identify different types of user's collaborations including ratings of educational items, in order to process them when generating personalized suggestions of items.
- To decide which type of educational items will be selected when generating personalized recommendations of items.
- To determine which recommendation techniques and approaches are the most suitable, according to the identified requirements.
- To design and adapt recommendation algorithms that will be used in the recommender system for RedEMC, writing a technical specification.
- To describe different test cases that will be executed in order to verify the recommendation algorithms.
- To assist Evimed at the time of deploying the proposed recommender system in a production environment.
- To collect user feedback to determine the effectiveness of the recommender system for RedEMC and to adjust the recommendation algorithms whether necessary.

1.4 Outline of the Thesis

The remaining of this thesis is organized as follows:

- Chapter 2 introduces background information, including: an introduction to recommender systems, distinct fields of application for these systems, possible problems or issues of recommender systems, challenges for recommender systems, a description of the different recommendation techniques and approaches, an introduction to virtual learning environments, and collaborative recommendations in virtual learning environments for higher education.

- Chapter 3 presents a research project with a Uruguayan company named *Evimed*, about collaborative recommendations in virtual learning environments for continuing medical education. This chapter describes how to select, process and utilize information about collaborations in the context of continuing medical education, in order to produce meaningful and useful recommendations of educational items to medical doctors and other health specialists through a virtual learning environment called *RedEMC*. The requirements gathering and analysis, design, implementation and testing of the recommender system and its recommendation algorithms are explained, as well as the deployment in a production environment.

- Chapter 4 describes the user feedback analysis, including: parameter values and weights used in production, collecting user feedback for the recommendations of educational items, analysis of the implicit and explicit feedback, and origin of the recommendations of items.

- Chapter 5 presents the conclusions of the research project, regarding collaborative recommendations in virtual learning environments. Finally, several lines of future work are specified.

Chapter 2

Background

Recommender systems utilize different recommendation techniques which are methods that assists them in generating personalized recommendations of items. These recommendations techniques are based on recommendation approaches which focus on the sources of information that will be used to produce personalized suggestions of items.

This chapter introduces background information which presents relevant concepts for this thesis, including: an introduction to recommender systems, different fields of application for these systems, possible problems or issues of recommender systems, challenges for recommender systems, a description of the different recommendation techniques and approaches, an introduction to virtual learning environments, social-based recommendations used in the QHIR LACCIR research project, and collaborative recommendations in virtual learning environments for higher education used in the UTU research project.

2.1 Recommender Systems

In this section, recommender systems are described, explaining the advantages of using them, their fields of application, possible problems or issues of recommender systems, and challenges that they should address.

2.1.1 Introduction to Recommender Systems

Recommender systems are software tools and techniques that aim to provide useful items to a specific user, known as the *active user* [1]. An *item* refers to the type of object being recommended, and it depends on the field of application. The huge amount and variety of information available for example in the web, make finding meaningful items of interest a difficult task for users. Recommender systems help users in their decision-making, suggesting meaningful items to them.

Recommendations could be classified in two types [1]:

- <u>Personalized recommendations</u>: The suggestions are generated for a specific user, which implies different users may obtain distinct recommendations of items. This is the most common type of recommendations.

- Non-personalized recommendations: The same suggestions are generated for all users, which implies different users obtain the same recommendations of items. This type of recommendations may be useful in particular cases such as the most popular movies, news, etc.

Recommender systems collect different types of information to produce personalized recommendations of items [1]:

- User's profile: Information contained in user's profile such as demographic data is used to generate suggestions of items.

- User's preferences: Information about user's preferences such as ratings of items and website navigation is used to make meaningful recommendations of items.

Regarding transparency of recommender systems, in the beginning they acted like black boxes, not providing any explanations about the criteria used for generating recommendations of items [9]. At the present time, some recommender systems provide explanations about how the suggestions of items are produced.

2.1.2 Fields of Application

There are numerous fields of application for recommender systems, including the following ones [1]:

- E-commerce: These recommender systems suggest products and services to potential buyers, in the form of "Clients who watched this article X also watched or bought this article Y". For instance, Amazon and eBay are websites/applications which provide e-commerce recommendations.

- Social networking: These recommender systems suggest people as potential friends, followers, or colleagues who the active user may know. For instance, Facebook, Instagram, and LinkedIn are websites/applications which suggest social recommendations.

- Online streaming: These recommender systems suggest multimedia content to the active user, including recommendations of music, movies and videos. For instance, Spotify, Netflix and Youtube are websites/applications which suggest multimedia recommendations.

- E-learning: These recommender systems suggest educational items to students, such as resources and activities. For instance, Moodle is a virtual learning environment which suggest recommendations of educational items.

- Travel booking websites: These recommender systems suggest flights, accommodation, and car rental, among others. For instance, Booking is a well-known website/application which suggest destinations.

2.1.3 Problems of Recommender Systems

Recommender systems present several problems or issues, including the following ones [10, 11]:

- Sparsity: Recommender systems have difficulty in generating useful recommendations of items when there is a small number of ratings of items, since there is not enough information about user's preferences to infer items of interest for the active user.

- Scalability: Recommender systems process information about users and items to produce personalized recommendations of items. The number of users and items in the system increase over time, therefore recommender systems require more processing capacity.

- Cold-start problem: This is a well-known issue for recommender systems that occurs when a new user or a new item is added into the system. In the first case, there is not enough information to produce meaningful suggestions of items. In the second case, the new item cannot be recommended until users begin to rate it.

2.1.4 Challenges for Recommender Systems

There are several challenges that recommender systems have to address [10]:

- Changing user's preferences: As explained earlier in section 2.1.1, recommender systems utilizes user's preferences to generate suggestions of items. These preferences could change over time. Therefore, recommender systems should be able to consider this challenge when generating new recommendations.

- Privacy: Recommender systems collect information about users which could include sensitive data. This information should be carefully processed in order to protect user's privacy. It is also important to respect user's privacy when explaining the reasoning behind personalized recommendations of items.

2.2 Recommendation Techniques

In this section, different recommendation techniques used in recommender systems are described, explaining their characteristics, as well as their advantages and disadvantages.

A possible classification of recommender systems is based on the techniques employed, including [10, 12, 13]:

- Content-based filtering systems.
- Collaborative-filtering systems.
- Demographic filtering systems.
- Context-aware recommender systems.
- Social-based recommender systems.
- Hybrid recommender systems.

Fig. 1 depicts a possible classification of recommender systems which are described in the following sections.

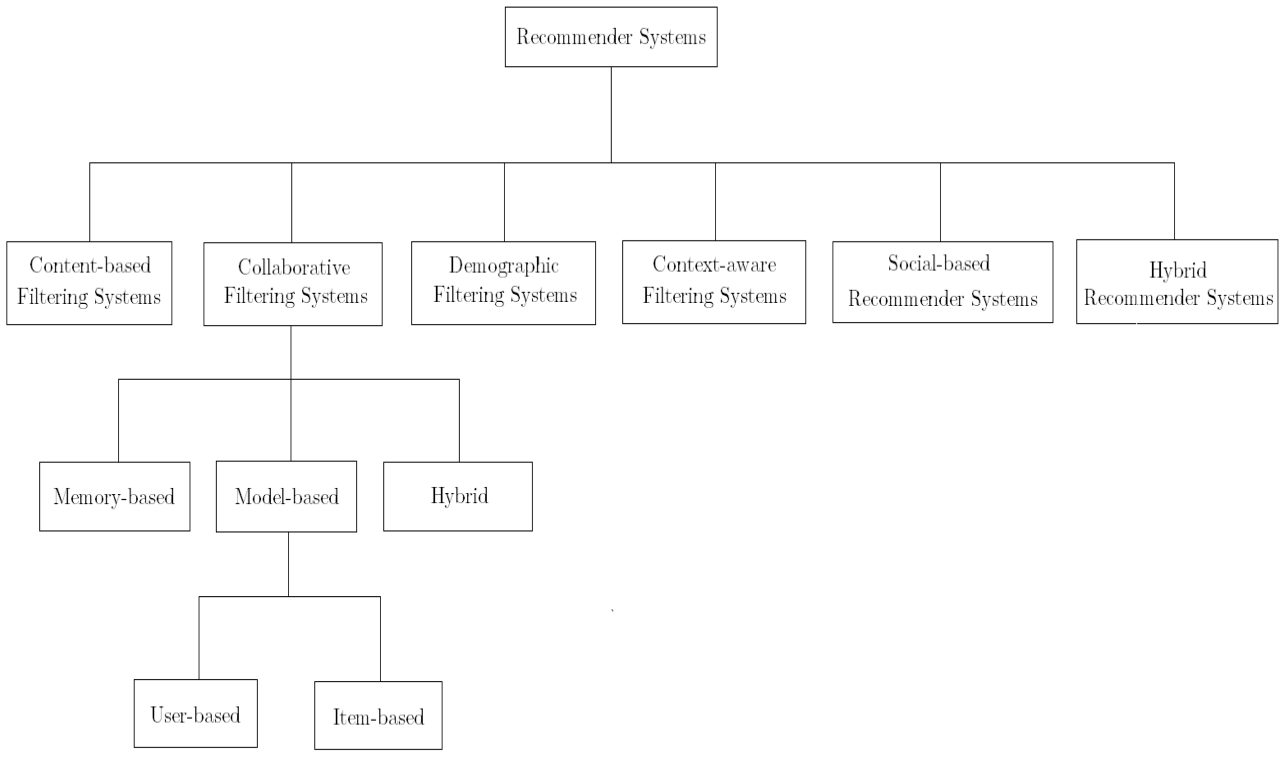

Recommender Systems
Content-based Filtering Systems
Collaborative Filtering Systems
Demographic Filtering Systems
Context-aware Filtering Systems
Social-based Recommender Systems
Hybrid Recommender Systems
Memory-based
Model-based
Hybrid
User-based
Item-based

Fig. 1 - Classification of Recommender Systems.

2.2.1 Content-based Filtering Systems

Content-based filtering systems use the content of items such as keywords in documents or texts, to make useful recommendations [12]. The recommendation process includes the following steps [10]:

1) Analyzing the content data of the items preferred by the active user in order to determine user's preferences, regarding content of items.
2) Saving the information about user's preferences in user's profile.
3) Comparing the content data of other items with the content data stored in user's profile.
4) Determining the items whose content data are similar to the content data stored in user's profile in order to generate recommendations to the active user.

The pros and cons of content-based filtering systems are listed below [12, 14]:

- Advantages: Since content-based filtering techniques consider item-to-item correlation, information about the preferences of other users is not needed, which implies it does not depend on other users' ratings of items to make suggestions. Another advantage is that users could discover items which have not been rated yet. This is beneficial to new users in the system.

- Disadvantages: Content-based filtering systems may experience difficulty in creating content data for the items. Besides that, these systems have to deal with the over-specialization issue.

2.2.2 Collaborative Filtering Systems

Collaborative filtering systems consider the preferences of other users who are similar to the active user, regarding ratings of similar items to those rated by the active user [10]. These systems could be classified in three types:

- Memory-based:

    - The first step of a collaborative filtering technique is to obtain groups of users who are similar to the active user. To do this task, ratings of users which give information about their preferences are considered. Then, based on these groups of users and their preferences, different recommendation algorithms are employed to make useful suggestions of items.

        - Different similarity metrics are used to determine similarity between two users, including:

            - Cosine similarity: This well-known similarity metric considers the cosine of the angle between two vectors which contain information about users' ratings of items [12].

            - Similarity in the relations of friendship: This similarity metric was used in the recommender system for the QHIR LACCIR research project, which considers information about friends in common between two users [15].

            - Similarity in the evaluations of items: This similarity metric was also utilized in the QHIR LACCIR research project, which regards to what degree two users evaluate items in the same manner [15].

            - Similarity in the activity of rating items: This similarity metric was also used in the QHIR LACCIR research project, which regards the level of activity of users, regarding the number rating of items [15].

        - Memory-based filtering techniques could be classified into two categories:

            - User-based: This technique calculates similarity between users regarding their ratings, to produce recommendations of items.

            - Item-based: This technique calculates similarity between items (instead of users), to generate recommendations of items. Items rated by users which are the most similar to the items highly rated by the active user, are selected to make suggestions.

    - Model-based:

        - These systems create recommendation models based on ratings of users to make recommendations of items.

        - Several techniques could be employed to obtain the models such as singular value decomposition, clustering, decision trees, among others.

- Hybrid:
    - These systems combine two or more collaborative filtering techniques into a hybrid one, minimizing the disadvantages of each individual recommendation technique, in order to produce more effective suggestions of items.

The pros and cons of collaborative filtering systems are listed below [10, 12]:

- Advantages: Collaborative-filtering techniques produce better recommendations than the ones generated by content-based filtering techniques, enhancing the performance of the recommender system. Another advantage is that these systems do not need content data for making suggestions of items.

- Disadvantages: These systems suffer from the cold-start problem when a new user enters the system, since there is not enough information to infer his/her preferences. Due to the amount of information to process to determine recommendations, these systems face the scalability issue. These systems also suffer from the sparsity issue, since users usually rate a small number of items, considering the whole set of items available in the system.

2.2.3 Demographic Filtering Systems

Demographic filtering systems use demographic information about users and their preferences to generate recommendations of items [12]. These systems assume that users who belong to specific demographic groups have similar preferences. Demographic information could include some of the following attributes: gender, age, nationality, occupation, education level, among others. In a previous research project for developing a music recommender system, Yapriadi et al. utilized demographic information about users, such as age, gender and nationality to make meaningful recommendations [16].

The pros and cons of demographic filtering systems are listed below [12]:

- Advantages: These systems do not need information about users' ratings of items to make recommendations and do not require content data as in content-based filtering systems.

- Disadvantages: It is difficult to gather demographic information of users, regarding user's satisfaction, privacy issues, time, among others.

2.2.4 Context-aware Recommender Systems

Context-aware recommender systems use contextual information about the user to produce meaningful recommendations [12]. Contextual information includes: location, date, season, among others.

There are three methods to obtain contextual information [13]:

- Explicitly: The information is directly collected by retrieving explicit data given by users, such as user's answers.

- Implicitly: The information is obtained by collecting implicit data, such as information about the environment and interactions.

- Inferring: The information is obtained by making inferences and deductions, using existing data.

The pros and cons of context-aware recommender systems are listed below [12, 13]:

- Advantages: These systems provide accurate suggestions of items.

- Disadvantages: It could be difficult to obtain relevant contextual information. For example, a mobile recommender system which uses information about users' location to produce suggestions of food, requires that the GPS location of the active user is turned on.

2.2.5 Social-based Recommender Systems

Social-based recommender systems utilize social information to produce recommendations of items [13]. Social information includes data about the relationships between users and social tags, among others.

The pros and cons of social-based recommender systems are listed below:

- Advantages: Social information could be used to improve the quality of suggestions.

- Disadvantages: These systems suffer from the cold-start problem when a new user enters the system, since there is not enough information to produce accurate recommendations, regarding relationships in the social network.

2.2.6 Hybrid Recommender Systems

Hybrid recommender systems combine two or more recommendations techniques into a hybrid one, in order to avoid the disadvantages of each recommender technique [17]. In general, content-based and collaborative filtering techniques are combined to obtain better recommendation techniques.

There are several hybrid methods to combine the recommendation techniques [17, 14]:

- Weighted: A weight is computed for recommended items, combining the results given by the recommendation techniques.

- Switching: This method is based on a function which switches between recommendation techniques. At a specific moment the system chooses a recommendation technique to provide suggestions of items.

- Mixed: This method merges and presents at the same moment, the recommended items given by the recommendation algorithms.

- Feature combination: This method combines the features of the results given by other recommendation techniques to produce suggestions of items.

- Feature augmentation: The features of the results given by a recommendation technique is utilized as the input of another technique.

The pros and cons of hybrid recommender systems are listed below [10]:

- Advantages: Hybrid techniques succeed in dealing with problems such as the sparsity and scalability issues.

- Disadvantages: The complexity of these recommender systems is considerable. These recommender systems are expensive to implement.

2.3 Recommendation Approaches

Recommendation approaches focus on the sources of information that will be used to make personalized suggestions of items. In this section, several recommendation approaches are described, including crowdsourcing, friendsourcing, and a hybrid approach.

2.3.1 Crowdsourcing

In crowdsourcing systems, a large number of people could make small contributions to solve an open task such as crowdsourcing systems succeeds in solving the following tasks [18]:

- Problem solving.
- Content generation:
- Knowledge aggregation.

A well-known example of crowdsourcing is *Wikipedia*, where users collaborate by adding information in wiki pages [19].

This approach is based on the "self-selection" principle which implies that tasks are published to general public from diverse backgrounds, allowing interested people to make small contributions in order to solve them. Crowdsourcing utilizes the "wisdom of crowds" to solve tasks from diverse groups of people [20].

Crowdsourcing systems present the following advantages [18]:

- Scalability: As mentioned earlier in section 2.1.3, the scalability issue is a common problem for recommender systems. Crowdsourcing systems scale well due to the large number of people who make contributions.

- Diversity: Since the tasks are published as open problems, contributions are made by people from diverse environments.

2.3.2 Friendsourcing

In friendsourcing systems, valuable information contained in a reduced group of socially-connected people is collected to do a particular task [21]. Friendsourcing is a type of crowdsourcing which focuses on a small group of individuals instead of considering a larger one.

In a previous research project, Bernstein et al. developed a social network application based on friendsourcing which collects accurate and valuable information about users' preferences by stimulating social tagging. This application produced good results in encouraging users to tag their friends.

Friendsourcing systems present the following advantages:

- Personalization: These systems are appropriate for personalization applications.
- Recommendation: Friendsourcing is suitable for recommender systems.
- Scalability: Friendsourcing overcomes the scalability issue since a reduced group of people is considered to generate personalized recommendations of items.

2.3.3 Hybrid

In hybrid systems, two or more recommendation approaches are combined [17]. This combination aims to take the advantages of each recommendation approach, minimizing their drawbacks.

In the UTU research project, González et al. proposed a hybrid recommender system for university students [6]. This hybrid recommender system utilizes friendsourcing and crowdsourcing approaches and different recommendation algorithms which process information about relationships between students as well as ratings to generate personalized recommendations of learning objects. By using a hybridization method which selects a particular recommendation algorithm to be executed at a particular moment, the system generates personalized recommendations of learning objects.

Preliminary results indicate that the hybrid recommender system produced meaningful and useful suggestions of learning objects.

2.4 Evaluation of Recommender Systems

Recommender systems aim to generate useful recommendations of items. In order to determine the performance of these systems, different evaluation metrics have been proposed.

In this section, different evaluation metrics used to measure the performance of recommender systems and different feedback techniques are described.

2.4.1 Evaluation Metrics

Some of the most used evaluation metrics for recommender systems include [22, 23]:

- Precision: This is a well-known evaluation metric which indicates the fraction of recommended items which are relevant. Precision is calculated as shown in (1):

$$\boxed{precision = \frac{|relevant\ recommended\ items|}{|recommended\ items|}} \tag{1}$$

- Recall: This is also a well-known evaluation metric which indicates the fraction of relevant items which are recommended. Recall is calculated as shown in (2):

$$\boxed{recall = \frac{|relevant\ recommended\ items|}{|relevant\ items|}} \tag{2}$$

- Mean Absolute Error (MAE): This is a frequently used evaluation metric which indicates the average of the difference between the predicted rating (r'_i) and the current user's rating of an item (r_i). MAE is calculated as shown in (3):

$$\boxed{MAE = \frac{\sum_{i=1}^{N} |r'_i - r_i|}{N}} \tag{3}$$

- Mean Square Error (MSE): This is also a frequently used evaluation metric which indicates the average of the squared difference between the predicted rating (r'_i) and the current user's rating of an item (r_i). MSE is calculated as shown in (4):

$$\boxed{MSE = \frac{\sum_{i=1}^{N} (r'_i - r_i)^2}{N}} \tag{4}$$

- Root Mean Square Error (RMSE): This is a common evaluation metric which indicate the root of the average of the squared difference between the predicted rating (*r'ᵢ*) and the current user's rating of an item (*rᵢ*). RMSE is calculated as shown in (5):

$$\boxed{RMSE = \sqrt{\frac{\sum_{i=1}^{N}(r'_i - r_i)^2}{N}}} \qquad (5)$$

2.4.2 Feedback Techniques

User feedback is relevant information that recommender systems should collect in order to provide meaningful and useful recommendations of items [23]. User feedback could also be used to determine the accuracy of the suggestions and to adjust the parameters of the recommendation algorithms [8].

There are three sorts of feedback techniques which allow recommender systems to collect information about users' preferences [23]:

- Implicit Feedback:
    - This technique collects information about the actions of the user, during the time he/she interacts with the system.
    - Users are not aware of the information implicitly provided to the system.
    - Several methods are employed to capture user feedback, such as the observation of users' behavior and examination of the actions performed in the system.
    - The advantages of this technique are:
        - Implicit user feedback could be collected at any time that the user interacts with the system.
        - The cost of collecting implicit user feedback is low.
        - Enhance user's satisfaction by not requiring extra time.

    - The disadvantages of this technique are:
        - If a user does not see a specific item, it cannot be concluded that it is not of his/her preference. No conclusion can be drawn about items that users have not seen.

- Explicit Feedback:
    - This technique collects information about the interests of the user, explicitly expressed by him/her.
    - Explicit feedback could be obtained in the following ways:
        - Like/dislike: Users express whether a specific item is of his/her preference, or not.
        - Ratings: Users evaluate items using a numeric rating scale.
        - Comments: Users express their preferences for a specific item by making textual comments.
    - The advantages of this technique are:
        - Users can directly express what items are of their preference and what others not.
        - This technique is more accurate than the implicit feedback technique.
    - The disadvantages of this technique are:
        - Evaluating items requires time that users have to spend on it.
        - Numeric rating scales could be difficult to understand if there are not clear explanations about the meaning of rating values.

- Hybrid Feedback:
    - This technique combines implicit and explicit feedback techniques.
    - Numeric ratings as well as user's behavior are considered to determine users' preferences and to make personalized suggestions of items.

- The advantages of this technique are:
    - It obtains more complete information about users' preferences.
    - This technique is more accurate than implicit and explicit feedback techniques separately.
- The disadvantages of this technique are:
    - It is expensive to process both implicit and explicit user feedback.

2.5 Virtual Learning Environments

In the last decade, many educational institutions chose to replace traditional websites and older discussion forums called "newsgroups" with virtual learning environments.

The courses offered through these platforms can be accessed from any host with Internet connection [2]. Virtual Learning Environments allow the management of resources, activities, discussion forums, questionnaires, and tests, among others.

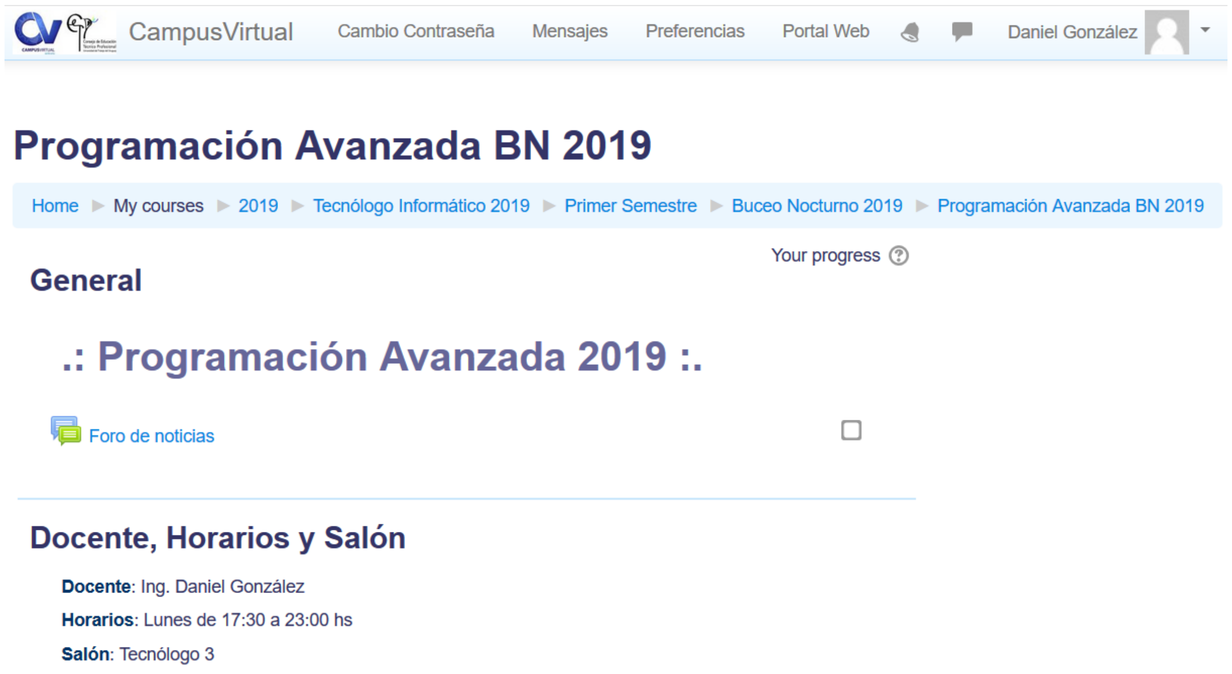
2.5.1 Virtual Learning Environments in Uruguay

In Uruguay, several virtual learning environments are utilized in different educational levels.

The Uruguayan government promoted the usage of virtual learning platforms such as CREA and PAM for high school students, through a socio-educational project called "Plan Ceibal" [24].

Moodle is a well-known open-source virtual learning environment which can be downloaded and used for free [3]. Some public and private universities utilize Moodle as its virtual learning platform, adapting the interface to their needs.

The University of Labor of Uruguay (UTU) uses a virtual learning environment called "Campus Virtual", adapting Moodle to fulfill its needs [25]. Fig. 2 shows the home page of a programming course named "Programación Avanzada BN 2019" in Campus Virtual.



Programación Avanzada BN 2019

Home ▶ My courses ▶ 2019 ▶ Tecnólogo Informático 2019 ▶ Primer Semestre ▶ Buceo Nocturno 2019 ▶ Programación Avanzada BN 2019

Your progress

General

.: Programación Avanzada 2019 :.

Foro de noticias

Docente, Horarios y Salón

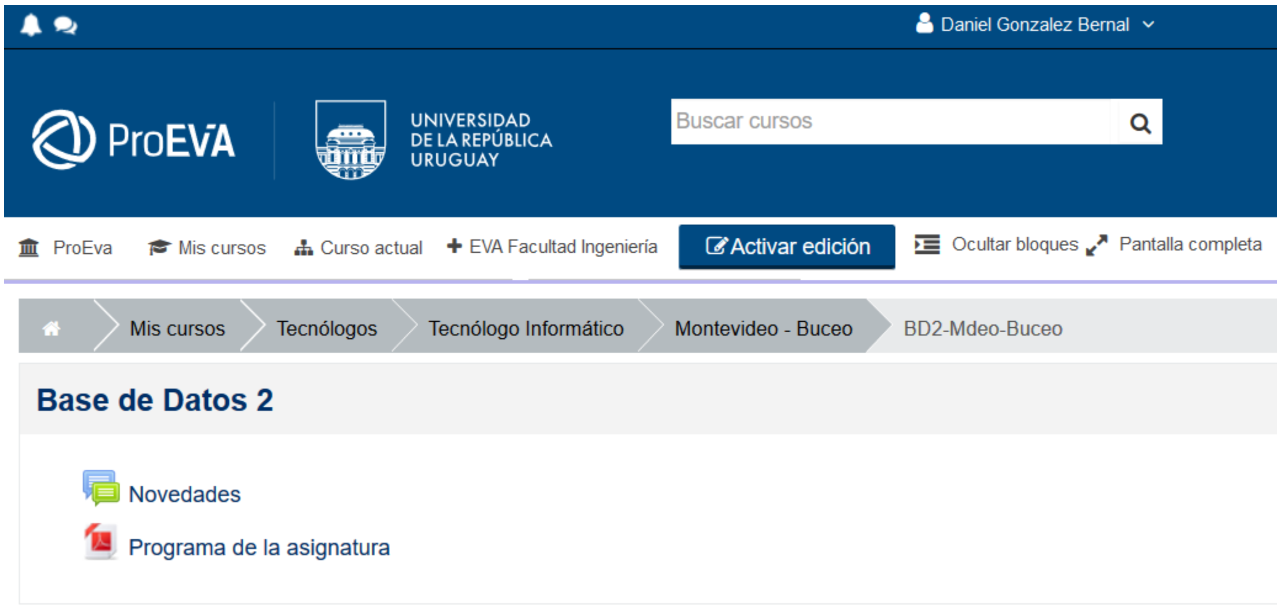
Docente: Ing. Daniel González

Horarios: Lunes de 17:30 a 23:00 hs

Salón: Tecnólogo 3

Fig. 2 - "Campus Virtual" Virtual Learning Environment.

Different faculties of the University of the Republic such as the Faculty of Engineering, also utilize an adapted Moodle called "Espacio Virtual de Aprendizaje" (EVA) [26]. Fig. 3 shows the home page of a database course named "Bases de Datos 2" in EVA.



Mis cursos > Tecnólogos > Tecnólogo Informático > Montevideo - Buceo > BD2-Mdeo-Buceo

Base de Datos 2

Novedades

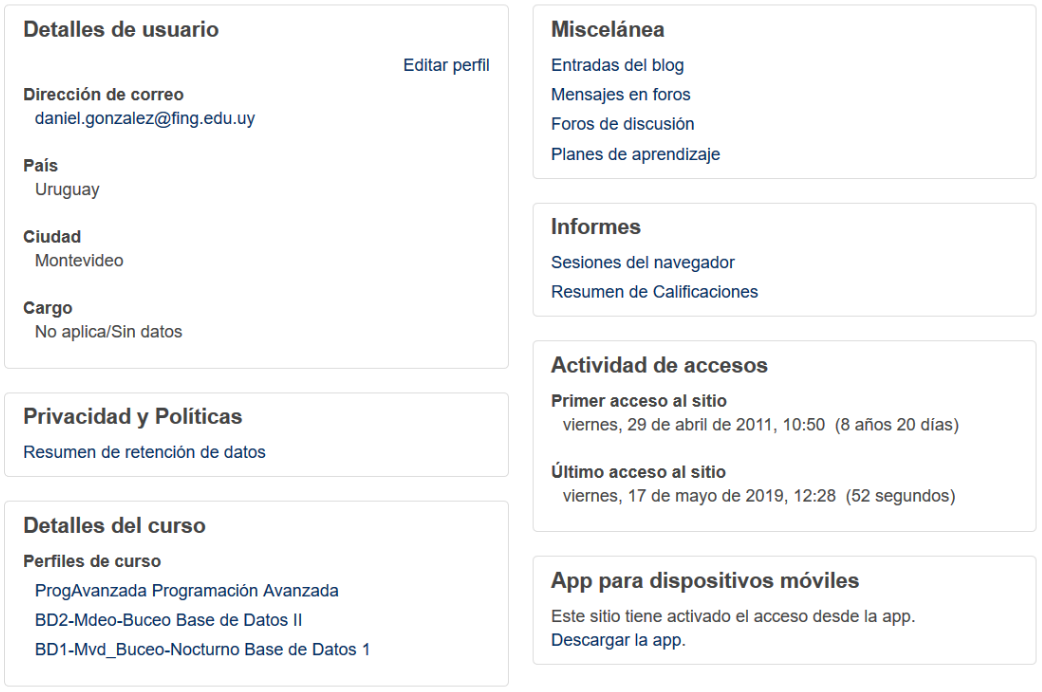
Programa de la asignatura

Fig. 3 - "EVA" Virtual Learning Environment.

EVA stores personal information about users as well as demographic data in their user's profiles, including:

- Name and Surnames: These are the name and surnames of the users of the virtual learning environment. This information cannot be updated by the user.
- Email address: This is an institutional address where the user receives notifications of enrolled courses, messages from students and teachers, among others. This information cannot be updated by the user.
- User's image: This is a picture of the user in order to be easily identified in the virtual learning environment.
- Country: This is the country where the user lives in.
- City: This is the city where the user lives in.
- Address: This is the address of the user.
- Telephones: These are the telephones numbers of the user (telephone and mobile phone).
- Position: This is the job occupied by the user in the educational institution.
- Enrolled courses: This is a list of the courses that the user enrolled in.
- Access dates: This information includes the dates of the first access and the last access to the platform.
- Interest: This is a set of tags which indicates the interests of the user.

Fig. 4 shows some of the information contained in user's profile.


Detalles de usuario

Editar perfil

Dirección de correo
daniel.gonzalez@fing.edu.uy

País
Uruguay

Ciudad
Montevideo

Cargo
No aplica/Sin datos

Privacidad y Políticas

Resumen de retención de datos

Detalles del curso

Perfiles de curso
ProgAvanzada Programación Avanzada
BD2-Mdeo-Buceo Base de Datos II
BD1-Mvd_Buceo-Nocturno Base de Datos 1

Miscelánea

Entradas del blog
Mensajes en foros
Foros de discusión
Planes de aprendizaje

Informes

Sesiones del navegador
Resumen de Calificaciones

Actividad de accesos

Primer acceso al sitio
viernes, 29 de abril de 2011, 10:50 (8 años 20 días)

Último acceso al sitio
viernes, 17 de mayo de 2019, 12:28 (52 segundos)

App para dispositivos móviles

Este sitio tiene activado el acceso desde la app.
Descargar la app.

Fig. 4 - User's profile data stored in EVA.

EVA allows teachers the possibility to:

- Add items:
  - Resources: Items of type *resource*, such as files, folders, tags, books, pages, URLs, among others, could be added into the course.
  - Activities: Items of type *activity*, such as questionnaires, surveys, discussion forums, among others, could be also added into the course.
- See statistical information:
  - Sessions: For each session, the virtual learning environment indicates date and time, IP address, type of device, visited page, and performed action. This information could be exported in processable files.
  - Users' activity: After selecting the type of activity (views, messages, or both) for a specific type of user and the duration of the period for a specific course, it is possible to see graphically and analytically the users' activity. This information could not be exported in any format. Fig. 5 depicts statistical information about users' activity in a programming course.

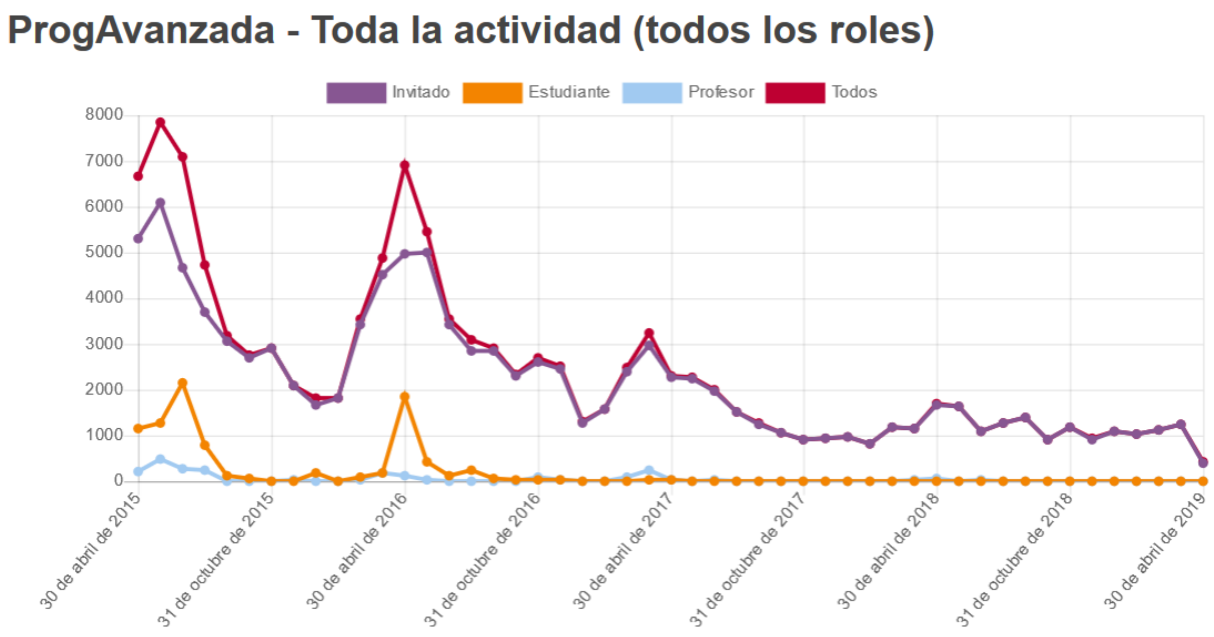


Fig. 5 - Statistical information about users' activity in a programming course.

2.5.2 Recommendations in Virtual Learning Environments

Some virtual learning environments provide recommendations of educational items.

A *Moodle block* is an item which could be added in a specific column position of a Moodle page (to the left, to the right or centered) [27]. Moodle offers recommendations by installing specific plugins in the form of blocks.

Some blocks which offer recommendations of educational items are:

- Analytics and Recommendations [4]:
    - Students can see their participation in different activities held in the course through colored charts, indicating low participation with red, medium participation with yellow, and high participation with green.
    - Students also receive recommendations of the activities that they should work harder in order to improve their academic results.
    - Teachers can see comparative and global analytics of the students' participation.
- Recommender [5]:
    - Users could receive as recommended items the ten most viewed resources or activities within a course.
    - Users could also receive as recommended items the three most popular courses that students of the same course also enrolled in.
    - Teachers have the possibility to recommend URLs.

Despite these recommendations could be useful for some students, they are limited and based on crowdsourcing information such as the most viewed resources or activities or the most popular courses.

Another disadvantage is that the suggestions do not consider valuable information contained in a reduced group of students. These recommendations could be improved if the recommender system utilize other sort of information to make the suggestions, such as demographic data.

2.6 Social-based Recommendations - QHIR LACCIR Research Project

In the Quality Health Information Retrieval (QHIR) LACCIR research project, a recommender system for users from a university hospital was designed and developed [15]. This recommender system utilizes a friendsourcing recommendation algorithm which processes information about socially-connected users.

The recommendation algorithm considers the friendship relationships between users in the social network to generate personalized recommendations of items. Firstly, a reduced group of friends is obtained. To do this task, different similarity metrics are proposed, including the similarity in friendship relationships, the similarity in the evaluations of items, and the similarity in the activity of rating items. After calculating these similarities between the active user and his/her friends, the friendsourcing recommendation algorithm selects the items which are the best rated by the most similar friends of the active user to make personalized suggestions.

The values of precision and recall obtained for the different components of similarity used in the friendsourcing recommendation algorithm were the following ones:

- Similarity in friendship relationships ⇒ precision = 0.65, recall = 0.72.
- Similarity in the evaluations of items ⇒ precision = 0.70, recall = 0.72.
- Similarity in the activity of rating items ⇒ precision = 0.75, recall = 0.78.

These preliminary results indicate that the proposed similarity metrics should be considered when selecting similar users. Notice that this recommendation technique could be combined with other recommendation techniques.

2.7 Collaborative Recommendations for Higher Education - UTU Research Project

In the University of Labor of Uruguay (UTU) research project, a hybrid recommender system which provides personalized recommendations of learning objects for university students was proposed [6]. This recommender system uses different recommendation approaches, such as friendsourcing and crowdsourcing, as well as distinct recommendation techniques including collaborative filtering and demographic techniques. The recommender system was proposed for Uruguayan students enrolled in a programming course. Students use a virtual learning environment called “Campus Virtual” where they have to do a sequence of mandatory group tasks in order to approve the course. As a consequence, each student belongs to a specific group of students. These groups define relationships between students in the virtual learning environment.

The recommender system considers the following information to make personalized recommendations of learning objects:

- Relationships between students: The membership of a particular group is the solely relationship regarded in the recommender system.

- Ratings of learning objects: Students have the possibility to rate distinct quality attributes of items ("intelligible", "complete", "agile to understand"), using a numeric scale from 1 (poor) to 5 (excellent). Users were aware of the meaning of the rating values.

- Course calendar: Indicates which learning objects are candidate to recommend in a specific period of time.

The recommender system also defines the following *states*:

- S_1: In this state, there are no ratings of learning objects in the system.

- S_2: In this state, there only exists ratings of items made by students of other groups (not from the group the active user belongs to).

- S_3: In this state, there only exists ratings of items made by students of the same group which the active user belongs to.

- S_4: In this state, there are ratings of learning objects made by students of all groups.

In each state, a specific recommendation algorithm is executed:

- In the state S_1, the recommendation algorithm named RA_1 considers the most important learning objects, according to the teacher's criterion and the course calendar.

- In the state S_2, a crowdsourcing recommendation algorithm named RA_2 considers the best ranked learning objects to make personalized suggestions of items.

- In the state S_3, a friendsourcing recommendation algorithm named RA_3 considers the similarity between the members of the group and the active user, based on demographic information such as age, gender and the distance existing between the university and his/her home.

- In the state S_4, a hybrid recommendation algorithm named RA_4 combines the recommended learning objects given by RA_2 and RA_3 recommendation algorithms.

The hybrid recommender system uses a switching method to change between the recommendation algorithms. A state machine indicates the current state for the active user, regarding the relationships between students and explicit user feedback, such as ratings of items.

The advantages of this hybrid recommender system are:

- It combines different recommendations approaches (friendsourcing, crowdsourcing, hybrid) and distinct recommendations techniques (collaborative filtering, demographic) in order to produce accurate suggestions of learning objects.

- It overcomes the cold-start problem, when there are no ratings of items in the system by recommending meaningful learning objects according to teacher's criterion.

The disadvantages of this hybrid recommender system are:

- The complexity of the recommender system is considerable.
- It is expensive in terms of implementation.

The values of precision and recall of the hybrid recommender system obtained for the different states were the following ones:

- $S_1 \Rightarrow$ precision = 0.75, recall = 0.68.

  This result indicates that the teacher criterion used in the RA_1 recommendation algorithm is suitable for generating recommendations of learning objects when there are no ratings of learning objects in the system.

- S_2, S_3, $S_4 \Rightarrow$ precision = 0.85, recall = 0.55.

  The second result indicates that the RA_2, RA_3, and RA_4 recommendation algorithms are more accurate than the RA_1 recommendation algorithm when there are ratings of learning objects made by students of any group. These algorithms consider the best ranked learning objects, similarity between the members of the group and the active user, demographic information, and a combination of these sources of information.

Preliminary results indicate that information about users' relationships in the virtual learning environment combined with demographic information, are useful for generating meaningful recommendations of learning objects.

Chapter 3

Collaborative Recommendations in VLE for Continuing Medical Education

This chapter presents a research project with a national company named *Evimed*, which consists in designing and developing a recommender system for a virtual learning environment called *RedEMC*, in the context of Continuing Medical Education (CME) [8]. Firstly, a standard definition of CME is introduced. Then, the different requirements are specified, and its virtual learning environment is described. Finally, a recommender system for RedEMC is proposed in order to accomplish the objectives, explaining the distinct phases of the software development process.

3.1 Continuing Medical Education

According to the Accreditation Council for Continuing Medical Education (ACCME), CME includes educational activities which help medical doctors and other health specialists in performing their professions in an effective and efficient manner. CME assists medical doctors in improving their knowledge and skills in their medical field [28].

3.2 Evimed

Evimed is a Uruguayan company which provides online CME courses to medical doctors and other health specialists, through its virtual learning environment called *RedEMC* [7]. In 2017, Evimed started working in a research project which pursued two main aims:

1) To help students in finding appropriate educational items: This is an *academic objective* which allows students to enhance their learning processes.

2) To assist students enroll in new CME courses of their interest: This is a *commercial objective* which allows an increase in the number of enrollments in RedEMC.

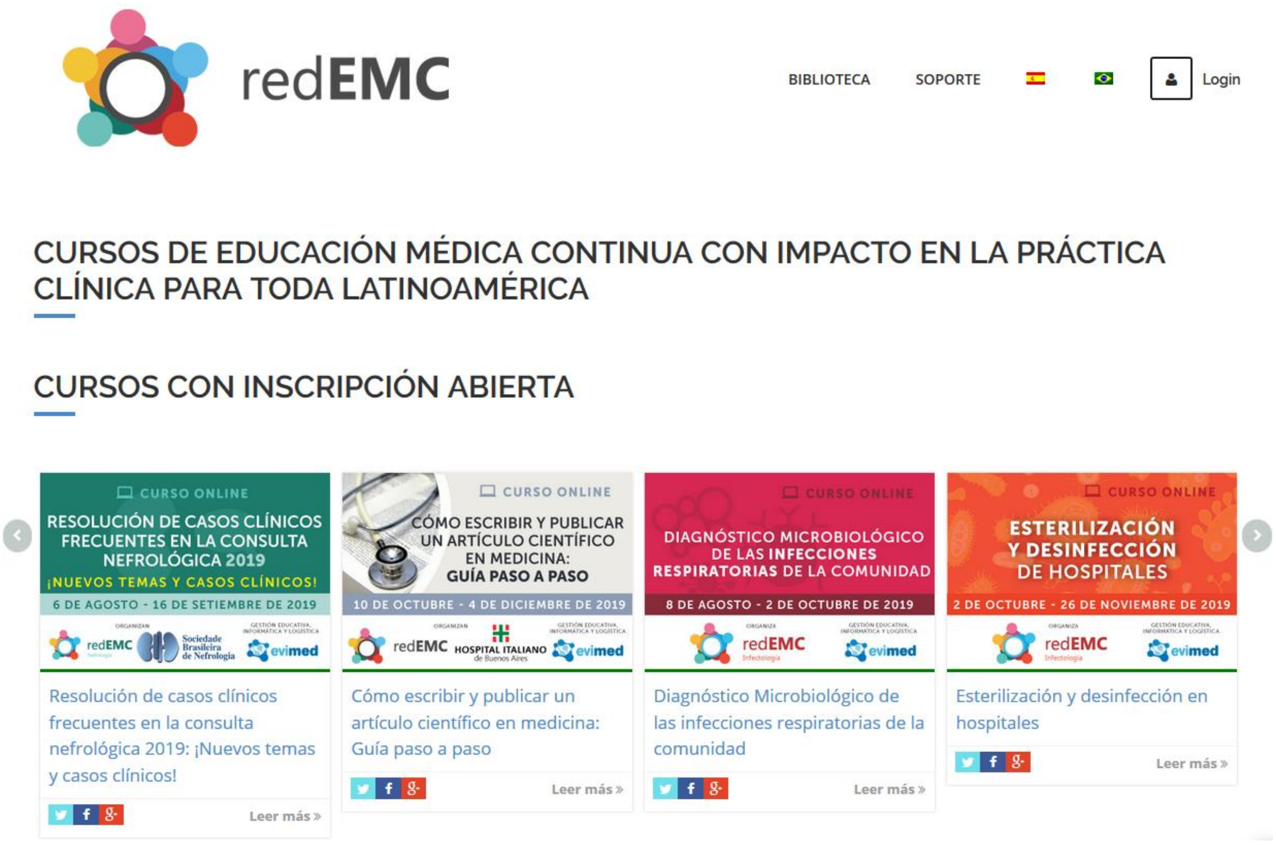
In order to achieve these aims, the development of a recommender system for RedEMC is proposed. Evimed searched for academic collaboration with the Institute of Computer Science of the Faculty of Engineering, University of the Republic (IN.CO. - Facultad de Ingeniería, UdelaR).

3.3 RedEMC Virtual Learning Environment

RedEMC is a virtual learning environment which offers one hundred and thirty online CME courses for thousands of health professionals across twenty-seven countries [7]. The wide range of CME courses include Anesthesiology, Nursing, Infectology, Intensive Care Medicine, Pediatrics, among others. There are three hundred and fifty teachers who educate medical doctors and other health specialists through RedEMC.

RedEMC is a platform which has the classic characteristics of a typical virtual learning environment combined with Facebook-like features such as the "like" button. An important feature is that RedEMC has its own academic network of colleagues which allows students to interact among each other and with teachers.

The virtual learning environment is available in: https://redemc.net/campus/. Fig. 6 shows the main page of RedEMC.



CURSOS DE EDUCACIÓN MÉDICA CONTINUA CON IMPACTO EN LA PRÁCTICA CLÍNICA PARA TODA LATINOAMÉRICA

CURSOS CON INSCRIPCIÓN ABIERTA

CURSO ONLINE
RESOLUCIÓN DE CASOS CLÍNICOS FRECUENTES EN LA CONSULTA NEFROLÓGICA 2019
¡NUEVOS TEMAS Y CASOS CLÍNICOS!
6 DE AGOSTO - 16 DE SETIEMBRE DE 2019

Resolución de casos clínicos frecuentes en la consulta nefrológica 2019: ¡Nuevos temas y casos clínicos!
Leer más »

CURSO ONLINE
CÓMO ESCRIBIR Y PUBLICAR UN ARTÍCULO CIENTÍFICO EN MEDICINA: GUÍA PASO A PASO
10 DE OCTUBRE - 4 DE DICIEMBRE DE 2019

Cómo escribir y publicar un artículo científico en medicina: Guía paso a paso
Leer más »

CURSO ONLINE
DIAGNÓSTICO MICROBIOLÓGICO DE LAS INFECCIONES RESPIRATORIAS DE LA COMUNIDAD
8 DE AGOSTO - 2 DE OCTUBRE DE 2019

Diagnóstico Microbiológico de las infecciones respiratorias de la comunidad
Leer más »

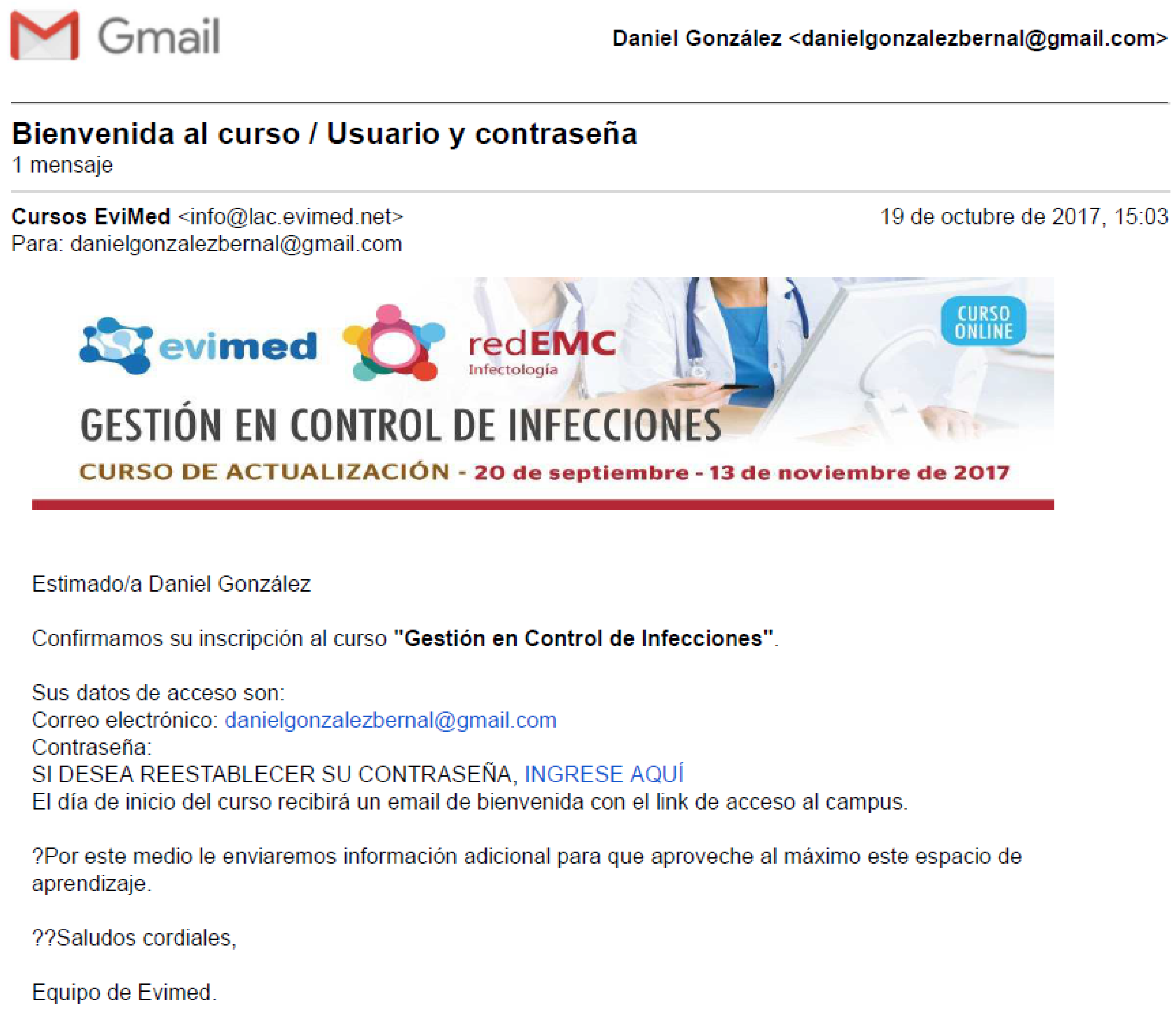
CURSO ONLINE
ESTERILIZACIÓN Y DESINFECCIÓN DE HOSPITALES
2 DE OCTUBRE - 26 DE NOVIEMBRE DE 2019

Esterilización y desinfección en hospitales
Leer más »

Fig. 6 - Main page of RedEMC.

3.3.1 Enrolling in an Online CME Course

When students enroll in an online CME course, they receive a welcome email which provides the credentials (username and password) to access to the virtual learning environment, as shown in Fig. 7.



Gmail

Daniel González <danielgonzalezbernal@gmail.com>

Bienvenida al curso / Usuario y contraseña
1 mensaje

Cursos EviMed <info@lac.evimed.net> 19 de octubre de 2017, 15:03
Para: danielgonzalezbernal@gmail.com

evimed
redEMC
Infectología
CURSO ONLINE
GESTIÓN EN CONTROL DE INFECCIONES
CURSO DE ACTUALIZACIÓN - 20 de septiembre - 13 de noviembre de 2017

Estimado/a Daniel González

Confirmamos su inscripción al curso **"Gestión en Control de Infecciones"**.

Sus datos de acceso son:
Correo electrónico: danielgonzalezbernal@gmail.com
Contraseña:
SI DESEA REESTABLECER SU CONTRASEÑA, INGRESE AQUÍ
El día de inicio del curso recibirá un email de bienvenida con el link de acceso al campus.

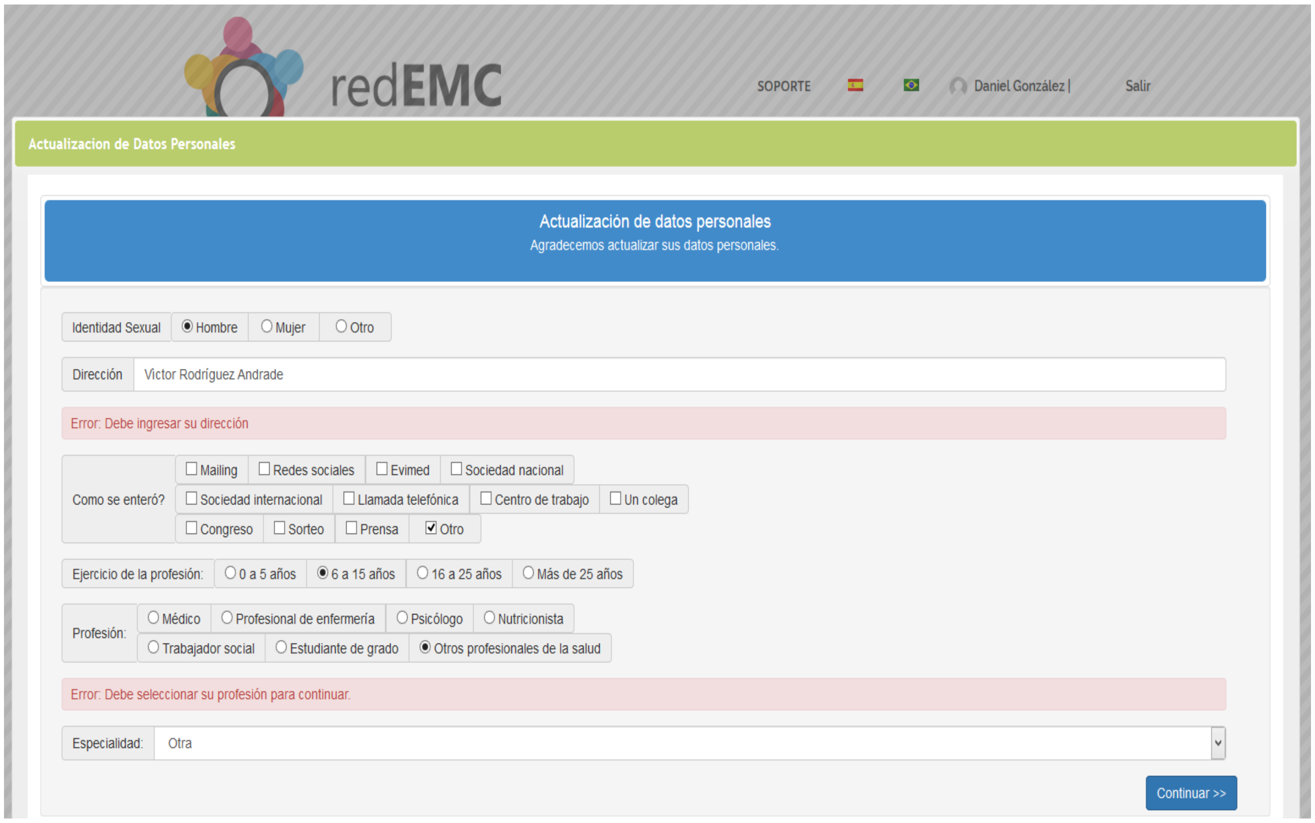
?Por este medio le enviaremos información adicional para que aproveche al máximo este espacio de aprendizaje.

??Saludos cordiales,

Equipo de Evimed.

Fig. 7 - Welcome email after enrolling in an online CME course.

After logging in into the virtual learning environment, RedEMC asks students to update their personal data such as gender, information about how they discovered the CME course, address, number of years of work experience, profession, and medical specialization, as shown in Fig. 8.



Actualizacion de Datos Personales

Actualización de datos personales

Agradecemos actualizar sus datos personales.

Identidad Sexual: [✓ Hombre] Mujer Otro

Dirección: Victor Rodríguez Andrade

Error: Debe ingresar su dirección

Como se enteró?: Mailing, Redes sociales, Evimed, Sociedad nacional, Sociedad internacional, Llamada telefónica, Centro de trabajo, Un colega, Congreso, Sorteo, Prensa, [✓ Otro]

Ejercicio de la profesión: 0 a 5 años [✓ 6 a 15 años] 16 a 25 años Más de 25 años

Profesión: Médico, Profesional de enfermería, Psicólogo, Nutricionista, Trabajador social, Estudiante de grado, [✓ Otros profesionales de la salud]

Error: Debe seleccionar su profesión para continuar.

Especialidad: Otra

Fig. 8 - Updating personal data in RedEMC.

As mentioned earlier in section 2.2.3, the information contained in user's profile is valuable and could be used to generate meaningful suggestions of items.

Users were notified that the information provided in RedEMC could be used for researching purposes and that their privacy will be protected as indicated by the Uruguayan Law 18331 [29].

In RedEMC, the information about the number of years of work experience, profession and medical specialization, could be used to groups of similar users with the same level of expertise in a particular medical field to be used in the recommendation process.

3.3.2 Collaborations in RedEMC

Once students enrolled in an online course in RedEMC, they have the possibility to participate and collaborate in diverse ways:

- Students create and maintain their academic network of colleagues by indicating explicitly whether they know a specific student enrolled in a CME course.

- RedEMC has a feature which allows students to watch their connections with other users, after indicating that the provided information about their links in the network will be used for scientific research, regarding privacy issues, as shown in Fig. 9.

Fig. 9 - “Watch your connections” feature.

Fig. 10 shows an example of an academic network of a real student in RedEMC, modelled by a directed graph where the nodes represent the colleagues of the student and the links represent the relationships between them. For privacy reasons, the images of the students were blurred, and their names were altered.

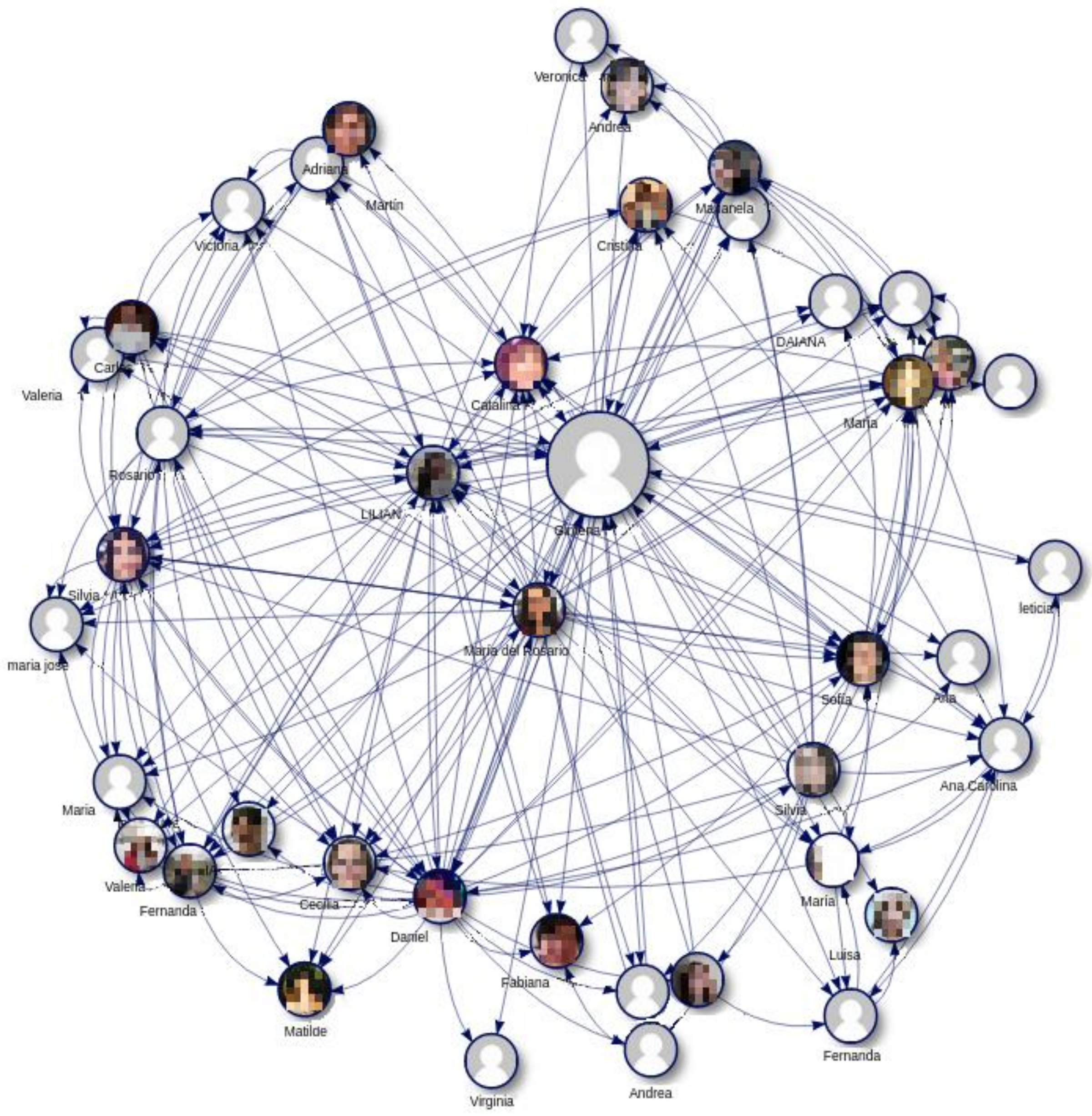
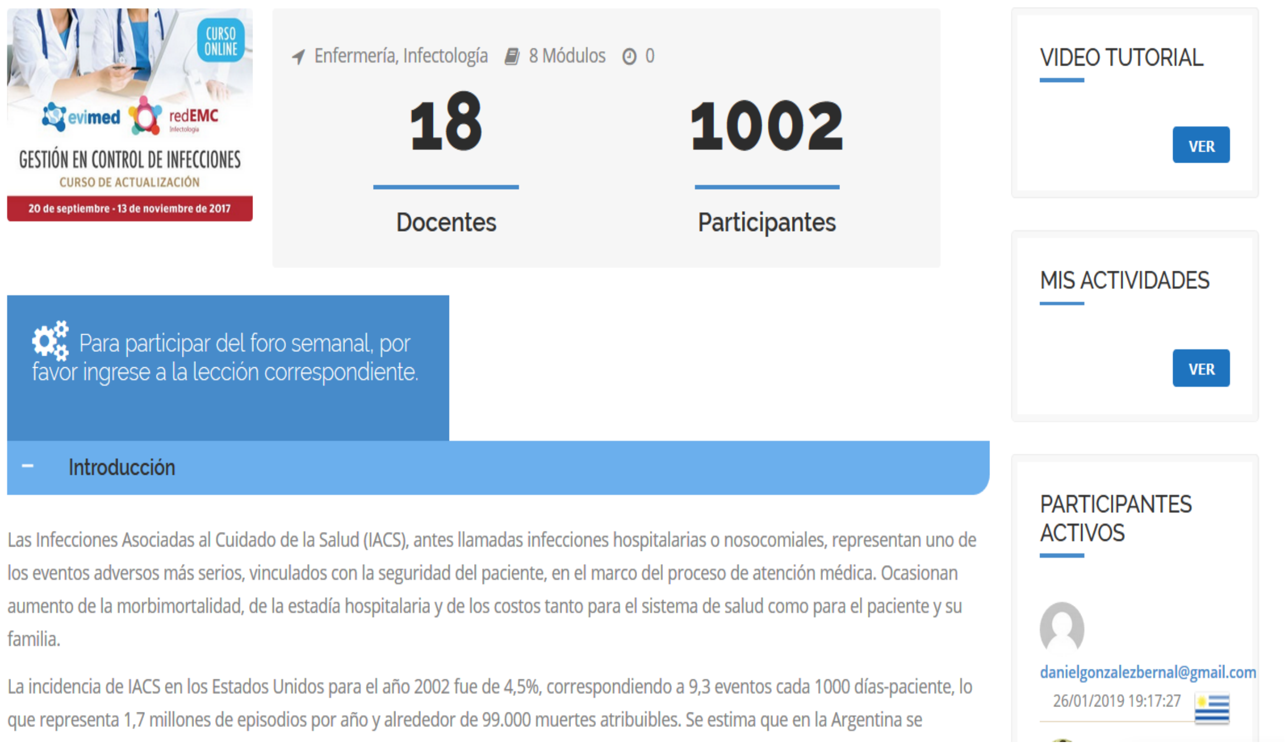


Fig. 10 - Academic network of a real student in RedEMC.

In RedEMC, each course has a sequence of modules which contain several *resources* (such as texts and videos) and *activities* (including group activities, discussion forums, tests, among others). In order to approve the courses, students have to do a series of mandatory activities during the course.

Fig. 11 shows an example of a resource of type *text* in the virtual learning environment.



Para participar del foro semanal, por favor ingrese a la lección correspondiente.

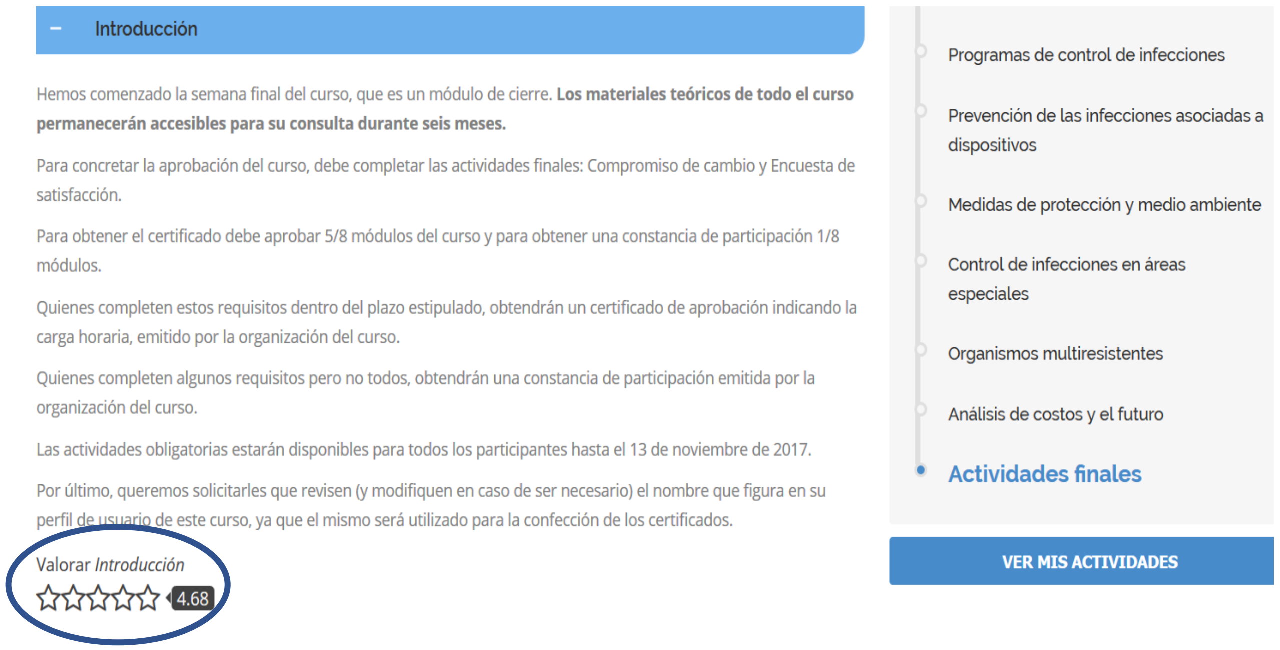
Las Infecciones Asociadas al Cuidado de la Salud (IACS), antes llamadas infecciones hospitalarias o nosocomiales, representan uno de los eventos adversos más serios, vinculados con la seguridad del paciente, en el marco del proceso de atención médica. Ocasionan aumento de la morbimortalidad, de la estadía hospitalaria y de los costos tanto para el sistema de salud como para el paciente y su familia.

La incidencia de IACS en los Estados Unidos para el año 2002 fue de 4,5%, correspondiendo a 9,3 eventos cada 1000 días-paciente, lo que representa 1,7 millones de episodios por año y alrededor de 99.000 muertes atribuibles. Se estima que en la Argentina se

Fig. 11 - Example of a resource in RedEMC.

Students also have the possibility to rate the distinct resources and activities of a module, giving from one to five stars to them, as shown in Fig. 12.

As mentioned earlier in section 2.4.2, explicit feedback information could be gathered through ratings of items. This type of information should be considered when generating personalized recommendations of educational items.



– Introducción

Hemos comenzado la semana final del curso, que es un módulo de cierre. **Los materiales teóricos de todo el curso permanecerán accesibles para su consulta durante seis meses.**

Para concretar la aprobación del curso, debe completar las actividades finales: Compromiso de cambio y Encuesta de satisfacción.

Para obtener el certificado debe aprobar 5/8 módulos del curso y para obtener una constancia de participación 1/8 módulos.

Quienes completen estos requisitos dentro del plazo estipulado, obtendrán un certificado de aprobación indicando la carga horaria, emitido por la organización del curso.

Quienes completen algunos requisitos pero no todos, obtendrán una constancia de participación emitida por la organización del curso.

Las actividades obligatorias estarán disponibles para todos los participantes hasta el 13 de noviembre de 2017.

Por último, queremos solicitarles que revisen (y modifiquen en caso de ser necesario) el nombre que figura en su perfil de usuario de este curso, ya que el mismo será utilizado para la confección de los certificados.

Valorar *Introducción*

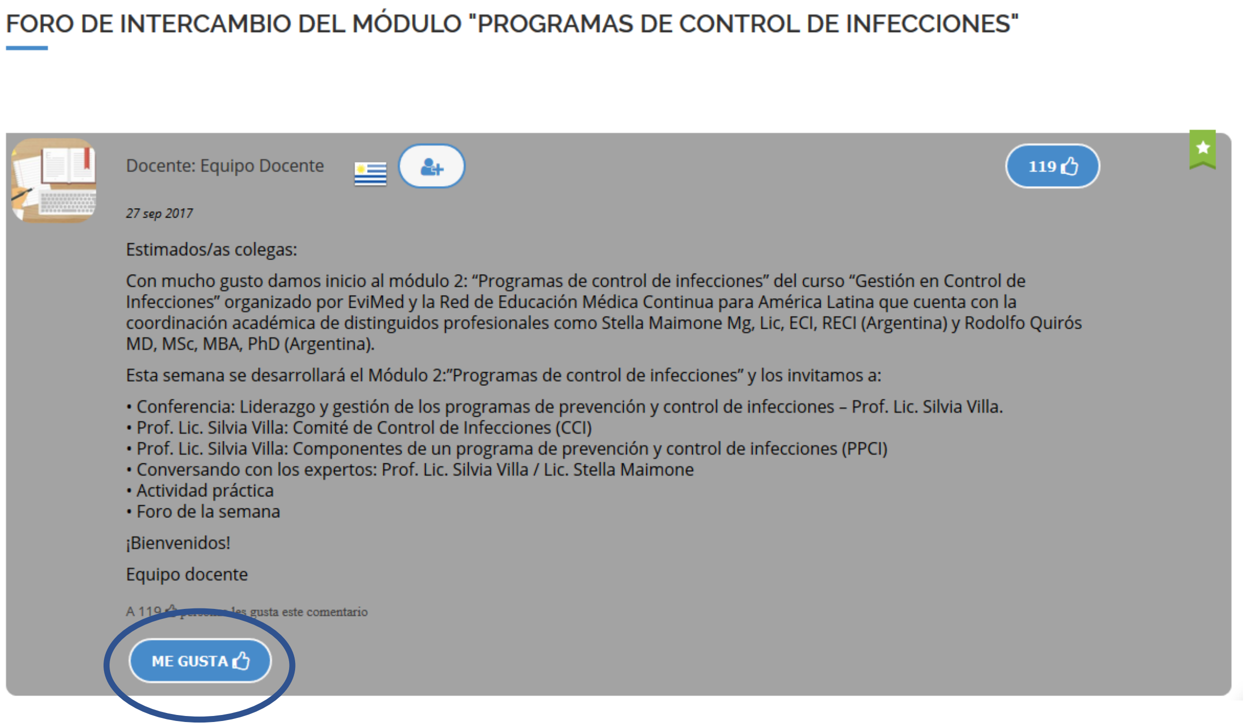
☆☆☆☆☆ 4.68

- Programas de control de infecciones
- Prevención de las infecciones asociadas a dispositivos
- Medidas de protección y medio ambiente
- Control de infecciones en áreas especiales
- Organismos multiresistentes
- Análisis de costos y el futuro
- Actividades finales

VER MIS ACTIVIDADES

Fig. 12 - “Five-stars” rating feature in RedEMC.

Students can use the “like” button to express that a comment included in a discussion forum is useful to them, as shown in Fig. 13.



FORO DE INTERCAMBIO DEL MÓDULO "PROGRAMAS DE CONTROL DE INFECCIONES"

Docente: Equipo Docente

119

27 sep 2017

Estimados/as colegas:

Con mucho gusto damos inicio al módulo 2: “Programas de control de infecciones” del curso “Gestión en Control de Infecciones” organizado por EviMed y la Red de Educación Médica Continua para América Latina que cuenta con la coordinación académica de distinguidos profesionales como Stella Maimone Mg, Lic, ECI, RECI (Argentina) y Rodolfo Quirós MD, MSc, MBA, PhD (Argentina).

Esta semana se desarrollará el Módulo 2:”Programas de control de infecciones” y los invitamos a:

- Conferencia: Liderazgo y gestión de los programas de prevención y control de infecciones – Prof. Lic. Silvia Villa.
- Prof. Lic. Silvia Villa: Comité de Control de Infecciones (CCI)
- Prof. Lic. Silvia Villa: Componentes de un programa de prevención y control de infecciones (PPCI)
- Conversando con los expertos: Prof. Lic. Silvia Villa / Lic. Stella Maimone
- Actividad práctica
- Foro de la semana

¡Bienvenidos!

Equipo docente

A 119 [illegible] les gusta este comentario

ME GUSTA

Fig. 13 - “Like-button” rating feature in RedEMC.

3.3.3 User relationships in RedEMC

RedEMC internally distinguishes three levels of user relationships which define three friendsourcing sets of users:

- $N_1(u)$: If user v belongs to $N_1(u)$, then user u knows user v and vice versa.
- $N_2(u)$: If user v belongs to $N_2(u)$, then user u knows user v.
- $N_3(u)$: If user v belongs to $N_3(u)$, then user u and v are acquaintances, which means neither user u and user v explicitly stated that they know each other but they have other sort of connection such as they enrolled together in the same course, made a group activity together, or they are Google contacts.

Therefore, N_1, N_2, and N_3, defines a high, a medium and a low level of relationship between users respectively.

3.3.4 Data Model

RedEMC uses a MySQL relational database to store information which has the following tables:

- *wp_users*:
  - Contain information about users of the virtual learning environment, such as identifier, email, name and avatar.
  - This table contains more than 17000 rows.

- *amigos_curso*:
  - Contain information about friends of a user in a specific course, according to the different friendsourcing relationships mentioned in the section 3.3.3.
  - This table contains more than 91000 rows.

- *formulario_inscripcion*:
  - Contain information about users when they registered in RedEMC, such as email, name, telephone, gender, address, country, profession, medical specialization, number of years of work experience, how they discovered the CME course, and date of update.
  - This table contains more than 9000 rows.

- *docentes_por_curso*:
  - Contain information about teachers of distinct courses.
  - This table contains more than 300 rows.

- *estadisticas_usuarios*:
    - Contain information about the activities of each user, such as identifier of the user, identifier of the course, identifier of the resource/activity, name and type of the resource/activity, and date.
    - This table contains more than 850000 rows.

- *wp_comments*:
    - Contain information about the comments published in the discussion forums, such as identifier of the comment, identifier of the user who published the comment, content of the comment, publication date, and the identifier of the parent comment.
    - This table contains more than 26000 rows.

- *posts_views*:
    - Contain information about the views of the resources and activities, such as identifier of the user who viewed the resource/activity, identifier of the resource/ activity, and date.
    - This table contains more than 400000 rows.

- *comments_views*:
    - Contain information about the views of the comments, such as identifier of the user who viewed the comment, identifier of the comment, and date.
    - This table contains more than 1000000 rows.

- *mg_comments*:
    - Contain information about the "likes" given to comments, such as identifier of the user who liked the comment, identifier of the comment, and date.
    - This table contains more than 10000 rows.

- *notificaciones*:
    - Contain information about the messages given between users, such as identifier of the message, identifier of the sender user, identifier of the receiver user, date, type of message, message.
    - This table contains more than 10000 rows.

- *wp_usermeta*:
    - Contain other information used by the virtual learning environment.

Fig. 14 shows a data model which contains the tables used by RedEMC.

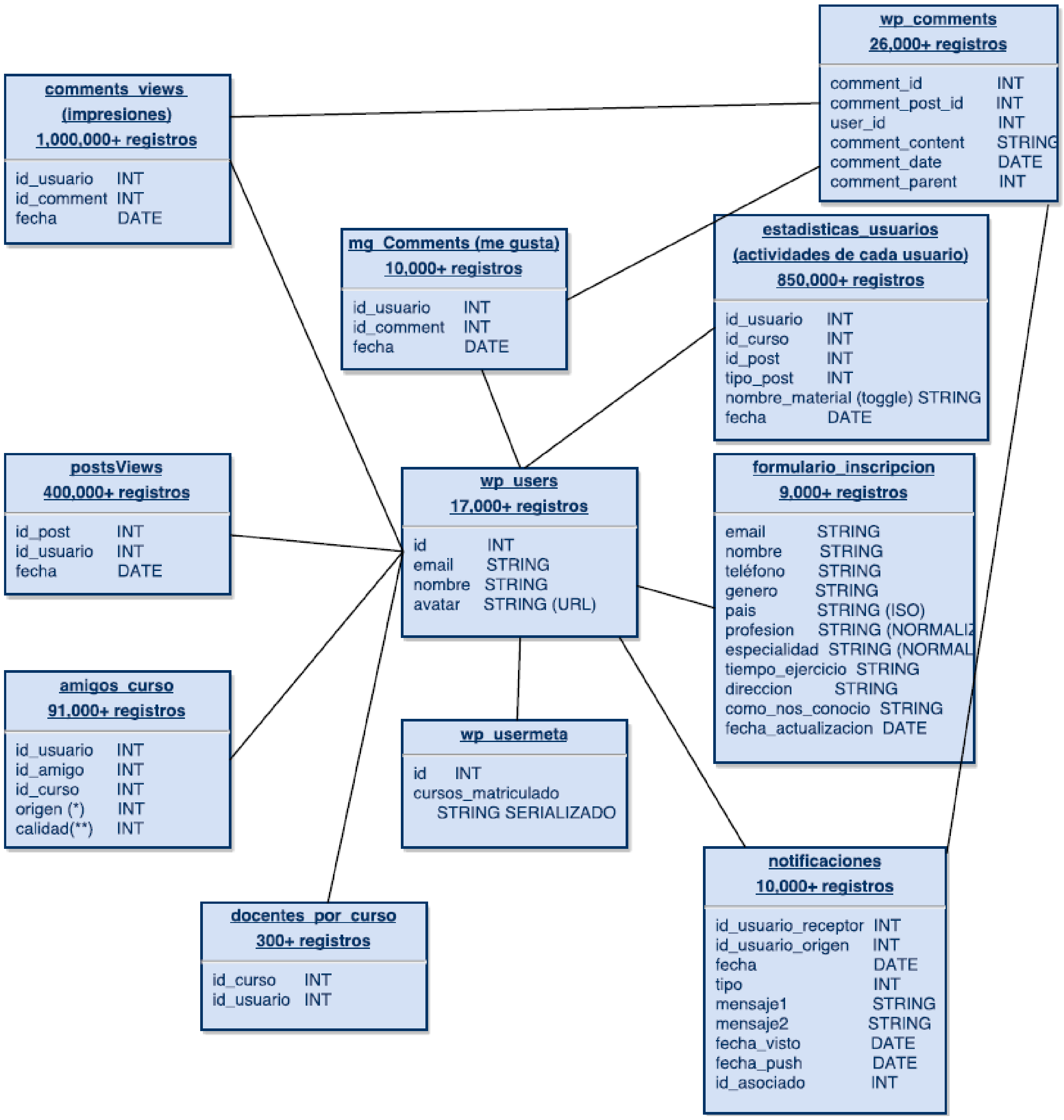


Fig. 14 - Data model of RedEMC.

3.4 Recommender System for RedEMC

A recommender system for RedEMC is proposed to satisfy the two main aims of the research project mentioned above.

The software development process for the recommender system includes the following phases:

- Requirements gathering and analysis: This phase consists in gathering the functional and non-functional requirements for the recommender system and analyzing them in order to determine the interfaces and the data structures that will be used.

- Design: This phase consists in designing the recommender system, including the presentation layer and the recommendation algorithms. Evimed's team is responsible for the design of the presentation layer.

- Implementation: This phase consists in constructing the recommender system and programming the recommendation algorithms previously designed. Evimed's team is responsible for these tasks.

- Testing: This phase consists in defining the test plan and executing its test cases.

- Deployment: This phase consists in deploying the recommender system in a production environment. Evimed's team is responsible for this task.

3.4.1 Requirements Gathering and Analysis

In order to gather the functional and non-functional requirements for the recommender system, several meetings with Evimed were held at the Faculty of Engineering, University of the Republic.

The identified functional requirements are listed below:

1) The recommender system should offer personalized recommendations of items to users.
2) The types of recommended items are: resources and activities, comments, and courses.
3) The number of recommended items should be parameterized.
4) The personalized recommendations of items should be traceable.

The only relevant identified non-functional requirement is listed below:

5) The recommendation algorithms should be implemented in stored procedures and functions.

These requirements are aligned with the academic and commercial objectives previously defined:

- To satisfy the first and second requirements, three recommendation algorithms which offer personalized recommendations of items are proposed. Each recommendation algorithm recommends a specific type of item: resource/activity, comment, or course.

- To satisfy the third requirement, different parameters are used to indicate the number of recommended items.

- To satisfy the fourth requirement, the friendsourcing sets N_1, N_2, N_3 could be utilized to trace the origin of the recommendations, explaining the criteria used to generate the recommendations.

- To satisfy the fifth requirement, the recommendation algorithms will be presented in pseudocode, in order to make it easier for Evimed's to implement them in store procedures and functions.

3.4.1.1 Algorithm Parameters

The algorithm parameters that are used by the recommendation algorithms are shown in Table 1.

Parameter	Description	Domain
λ	Indicates an acceptance threshold for a resource or activity	$0.0 \leq \lambda \leq 5.0$
σ	Indicates the minimum number of accepted ratings for a resource or activity	$0 \leq \sigma$
φ	Indicates the minimum number of likes for a comment	$0 \leq \varphi$
θ	Indicates the minimum number of views for a comment	$0 \leq \theta$
n	Indicates the maximum number of recommended items using a friendsourcing approach	$0 < n$
m	Indicates the maximum number of recommended items using a crowdsourcing approach	$0 < m$
u	Indicates the identifier of the active user	-
c	Indicates the identifier of the course	-
s	Indicates the subtype of item to be recommended	[resource, activity]
$nmod$	Indicates the number of modules backwards whose items could be recommended	$0 \leq nmod$

k	Indicates the number of days backwards whose comments could be recommended	$0 \leq k$

Table 1 - Algorithm parameters.

In the light of the positive results from the previous UTU research project, a hybrid recommender system which provides personalized recommendations of learning objects for university students was developed [6]. This recommender system used an acceptance threshold as a parameter to determine whether a learning object could be recommended to the active user.

3.4.1.2 Candidate Items

A *candidate item* is an item that could be recommended to the active user. The candidate items are specified in Table 2.

Candidate Item	Conditions
Candidate resource/activity	Its average rating is greater or equal to λ, its number of accepted ratings is greater or equal to σ, and the active user u has not viewed the resource or activity
Candidate comment	Its number of likes is greater or equal to φ and its number of views is greater or equal to θ
Candidate course	The active user u has not enrolled in the course

Table 2 - Candidate items for the recommender system.

3.4.2 Design of the Recommender System

This section describes the proposed recommendation algorithms for RedEMC:

- A recommendation algorithm which offers recommendations of resources or activities.
- A recommendation algorithm which offers recommendations of comments.
- A recommendation algorithm which offers recommendations of courses.

3.4.2.1 Recommendation Algorithm of Resources and Activities

The recommendation algorithm of resources and activities is a hybrid recommendation algorithm which uses:

- user-based and item-based collaborative filtering techniques,
- friendsourcing and crowdsourcing approaches.

This algorithm has a main function called *getRecommendationsRA* and several auxiliary functions, including *getFriendsourcingRA* and *getCrowdsourcingRA*.

3.4.2.1.1 Specification of getRecommendationsRA

The main function *getRecommendationsRA* takes as input parameters: the identifier of the active user u, the identifier of the course c, the subtype s of the item to be recommended (resource or activity), the number of modules backwards *nmod* whose items will be recommended, the acceptance threshold λ, the minimum number of accepted ratings σ, the maximum number n of items to be recommended by using a friendsourcing approach, and the maximum number m of items to be recommended by using a crowdsourcing approach. The output parameter of the main function is an ordered set which contains the recommended items for the active user u.

Firstly, the friendsourcing sets of users N_1, N_2 and N_3 are obtained to apply a friendsourcing approach. Then, the current module *cmod* of the course c is calculated considering the current system date. Next, four ordered lists L_1, L_2, L_3 y L_4 are defined. These lists contain pairs which associate a candidate item for the active user u with a relevance value. L_1 is obtained after calling the auxiliary function *getFriendsourcingRA*(u, c, s, *cmod*, *nmod*, λ, σ, N_1). Similarly, L_2 and L_3 are obtained after calling that auxiliary function, using the friendsourcing sets of users N_2 and N_3 respectively.

Notice that in some cases there might not be enough information about ratings of items in the virtual learning environment to generate recommendations through friendsourcing. To deal with the cold-start problem, the usage of recommendations of items obtained through crowdsourcing is proposed. Therefore, L_4 is obtained after calling the auxiliary function *getCrowdsourcingRA*(c, s, *cmod*, *nmod*, λ, σ).

R is defined as an ordered set which contains recommended items for the active user u. The candidate items obtained through friendsourcing are added to R in descending order by the relevance value without repeating, taking one recommended item at a time from each of the lists L_1, L_2 and L_3.

This process continues until reaching *n* recommended items or until L_1, L_2 and L_3 are empty. Then, a candidate item of L_4 is taken and added to *R* in descending order by the relevance value without repeating. This process continues until reaching *m* recommended items or until L_4 is empty. Finally, the ordered set *R* which contains recommended items for the active user *u* is returned.

3.4.2.1.2 Specification of getFriendsourcingRA

This auxiliary function *getFriendsourcingRA* takes as input parameters: the identifier of the active user *u*, the identifier of the course *c*, the subtype *s* of item to be recommended, the number of modules backwards *nmod*, the acceptance threshold λ, the minimum number of accepted ratings σ, and the friendsourcing set of users N_j. The output parameter of the auxiliary function is an ordered list that contains pairs which associate a candidate item for the active user *u* with a relevance value, obtained through a friendsourcing approach.

Initially, the ordered list *L* of pairs is defined. Then, the candidate items for each user *v* of the set N_j are taken. These candidate items are items of subtype *s* which were evaluated by the user *v* and belong to the current module or to *nmod* previous modules of the course *c*. For each candidate item, its relevance value is calculated and added to the ordered list *L*.

To calculate the relevance value of an item, similarity metrics are considered. Several standard similarity metrics presents limitations, such as the cosine similarity which not consider the difference in rating scale between distinct users [30].

Considering the positive results of the similarity in the evaluations of items in the QHIR LACCIR research project, they were proposed to be used in the RedEMC recommender system [15]. This similarity metric indicates how similar are two users *u* and *v* in evaluating items positively or negatively and is calculated as shown in (6), where *niric*(*u*, *v*) is the number of items rated in common by both users, *eval*(*u*, *i*) is the evaluation value given by the user *u* to the item *i*, and *eval*(*v*, *i*) is the evaluation value given by the user *v* to the item *i*.

$$\boxed{simEval(u, v) = \sum_{i=1}^{niric(u,v)} (1 - |eval(u, i) - eval(v, i)|)/niric(u, v)} \quad \textbf{(6)}$$

Regardless of the rating scale used, the evaluation values are normalized float values between 0.0 and 1.0. Sparsity issues can occur when there are only a few ratings in the virtual learning environment. A way to mitigate these problems is to employ Bayesian models [31].

The relevance value of an item *i* is calculated as shown in (7), where *maxMod*(*c*) is the maximum number of modules of the course *c* at the current date, *mod*(*i, c*) is the module of the course *c* which contains the item *i*, *eval*(*v*, *i*) is the evaluation value given by the user *v* to the item *i*, and *simEval*(*u*, *v*) is the value of similarity in the evaluations of items between users *u* and *v*.

$$\boxed{rv(u,\ v,\ i,\ c) = (1/(maxMod(c)\ \text{-}\ mod(i,\ c) + 1))\ *\ eval(v,\ i)\ *\ simEval(u,\ v)} \quad \textbf{(7)}$$

Notice that more than one user of the set N_j can evaluate the same candidate item. Therefore, there may be repeated items in the ordered list *L*. In order to process the repeated items in *L*, a repetition coefficient is calculated for each item *i* of the list *L* and its relevance value is adjusted.

The repetition coefficient is calculated as shown in (8), where *occurs*(*i*, *L*) is the number of occurrences of the item *i* in the list *L*, and *maxOccurs*(*L*) is the maximum number of repeated occurrences in the list *L*.

$$\boxed{rc(i,\ L) = (occurs(i,\ L)/maxOccurs(L))} \quad \textbf{(8)}$$

The relevance value for each item *i* is adjusted, multiplying the current relevance value by the repetition coefficient. Then, the repeated items of the list *L* are eliminated as going through the list in descending order by the adjusted relevance value, just keeping the occurrences associated with the maximum adjusted relevance values. Lastly, the list *L* which contains the recommended resources and activities for the active user *u* according to the friendsourcing set of users N_j is returned.

3.4.2.1.3 Specification of getCrowdsourcingRA

This auxiliary function *getCrowdsourcingRA* takes as input parameters: the identifier of the course *c*, the subtype *s* of item to be recommended, the number of modules backwards *nmod*, the acceptance threshold λ, and the minimum number of accepted ratings σ. The output parameter of the auxiliary function is an ordered list that contains pairs which associate a candidate item for the active user *u* with a relevance value, obtained through a crowdsourcing approach.

Considering the parameter values λ and σ, the auxiliary function takes the candidate items of subtype *s* which belong to the current module or to *nmod* previous modules of the course *c* and adds them to the ordered list by their relevance value. In this scenario, the relevance value of a candidate item is defined as the average rating value in the virtual learning environment.

3.4.2.1.4 Pseudocode of the Recommendation Algorithm of Resources and Activities

The pseudocode of the recommendation algorithm of resources and activities is presented below:

```
OrderedSet<Item> getRecommendationsRA(User u, Course c, Subtype s, int nmod, float $\lambda$, int $\sigma$, int n, int m) {
    Set<User> $N_1$ = get$N_1$(u);
    Set<User> $N_2$ = get$N_2$(u);
    Set<User> $N_3$ = get$N_3$(u);
    Module cmod = getCurrentModule(c, getCurrentDate());
    OrderedList<Item, float> $L_1$, $L_2$, $L_3$, $L_4$;
    OrderedSet<Item> R;
    $L_1$ = getFriendsourcingRA(u, c, s, cmod, nmod, $\lambda$, $\sigma$, $N_1$);
    $L_2$ = getFriendsourcingRA(u, c, s, cmod, nmod, $\lambda$, $\sigma$, $N_2$);
    $L_3$ = getFriendsourcingRA(u, c, s, cmod, nmod, $\lambda$, $\sigma$, $N_3$);
    $L_4$ = getCrowdsourcingRA(c, s, cmod, nmod, $\lambda$, $\sigma$);
    addFriendsourcingRA($L_1$, $L_2$, $L_3$, n, R);
    addCrowdsourcingRA($L_4$, m, R);
    return R;
}

OrderedList<Item, float> getFriendsourcingRA(User u, Course c, Subtype s, Module cmod, int nmod, float $\lambda$, int $\sigma$, Set<User> N) {
    OrderedList<Item, float> L;
    for each v in N {
        Set<Item> CI = getCandidateRA(v, c, s, cmod, nmod, $\lambda$, $\sigma$);
        for each i in CI {
            float rv = calculateRelevanceValueRA(u, v, i, c);
            addList((i, rv), L);
        }
    }
    for each i in L {
        float rc = calculateRepetitionCoefficient(i, L);
        adjustList(i, rc, L);
    }
    removeRepeatedItems(L);
    return L;
}
```

3.4.2.2 Recommendation Algorithm of Comments

The recommendation algorithm of comments is a hybrid algorithm which uses:

- item-based collaborative filtering techniques,
- friendsourcing and crowdsourcing approaches.

This algorithm has a main function called *getRecommendationsCM* and several auxiliary functions, including *getFriendsourcingCM* and *getCrowdsourcingCM.*

3.4.2.2.1 Specification of getRecommendationsCM

The main function *getRecommendationsCM* takes as input parameters: the identifier of the active user u, the identifier of the course c, the number of days k, the minimum value of likes φ, the minimum value of views θ, the maximum number n of comments to be recommended using a friendsourcing approach, and the maximum number m of comments to be recommended using a crowdsourcing approach. The output parameter of the main function is an ordered set which contains the recommended comments for the active user u.

Firstly, the friendsourcing sets of users N_1, N_2 and N_3 are obtained in the same manner as in the previous recommendation algorithm. Then, four ordered lists L_1, L_2, L_3 y L_4 are defined. These lists contain pairs which associate a candidate comment for the active user u with a relevance value. L_1 is obtained after calling the auxiliary function *getFriendsourcingCM*(c, k, φ, θ, N_1). Similarly, L_2 and L_3 are obtained after calling that auxiliary function using the friendsourcing sets of users N_2 and N_3 respectively. L_4 is obtained after calling the auxiliary function *getCrowdsourcingCM*(c, k, φ, θ).

R is defined as an ordered set which contains the recommended comments for the active user u. The candidate comments obtained through friendsourcing are added to R in descending order by the relevance value without repeating, taking one recommended comment at a time from each of the lists L_1, L_2 and L_3. Then, a candidate comment of L_4 is randomly taken and added to R in descending order by the relevance value without repeating. This process continues until reaching n recommended comments through L_1, L_2 or L_3 or until reaching m recommended comments through L_4 or until L_1, L_2, L_3 and L_4 are empty. Finally, the ordered set R which contains the recommended comments for the active user u is returned.

3.4.2.2.2 Specification of getFriendsourcingCM

This auxiliary function *getFriendsourcingCM* takes as input parameters: the identifier of the course c, the number of days k, the minimum value of likes φ, the minimum value of views θ, and the friendsourcing set of users N_j. The output parameter of the auxiliary function is an ordered list that contains pairs which associate a candidate comment for the active user u with a relevance value, obtained through a friendsourcing approach.

Initially, the ordered list L of pairs is defined. Then, the candidate comments for each user v of the set N_j are taken. These candidate comments are comments of the course c which were published in a discussion forum by the user v in the last k days. The parameter k could be set to the average duration (in number of days) of a module. Notice that Natural Language Processing techniques could be used to remove comments which are semantically duplicated [32].

For each candidate comment, its relevance value is calculated and added to the ordered list L. To calculate the relevance value of a comment, the number of likes and the number of views of the comment are considered. It is desirable to recommend comments which were recently published. Therefore, the publication time of the comment is also considered in the calculation of the relevance value. The relevance value of a comment cm is calculated as shown in (9), where $likes(cm)$ is the number of likes of the comment cm, $views(cm)$ is the number of views of the comment cm, and $publicationTime(cm)$ is the publication time of the comment cm.

$$\boxed{rv(cm) = (likes(cm)/views(cm)) * (1/publicationTime(cm))} \qquad \mathbf{(9)}$$

Lastly, the list L which contains the recommended comments for the active user u according to the friendsourcing set of users N_j is returned.

3.4.2.2.3 Specification of getCrowdsourcingCM

This auxiliary function *getCrowdsourcingCM* takes as input parameters: the identifier of the course c, the number of days k, the minimum value of likes φ, and the minimum value of views θ. The output parameter of the auxiliary function is an ordered list that contains pairs which associate a candidate comment for the active user u with a relevance value, obtained through a crowdsourcing approach.

Considering the parameter values φ and θ, the auxiliary function takes the candidate comments of the course c which were published in a discussion forum in the last k days and adds them to the ordered list by their relevance value. In this scenario, the relevance value of a candidate comment is defined as the number of likes of the comment in the virtual learning environment.

3.4.2.2.4 Pseudocode of the Recommendation Algorithm of Comments

The pseudocode of the recommendation algorithm of comments is presented below:

```
OrderedSet<Comment> getRecommendationsCM(User u, Course c, int k, int φ,
int θ, int n, int m) {
    Set<User> N₁ = getN₁(u);
    Set<User> N₂ = getN₂(u);
    Set<User> N₃ = getN₃(u);
    OrderedList<Comment, float> L₁, L₂, L₃, L₄;
    OrderedSet<Comment> R;
    L₁ = getFriendsourcingCM(c, k, φ, θ, N₁);
    L₂ = getFriendsourcingCM(c, k, φ, θ, N₂);
    L₃ = getFriendsourcingCM(c, k, φ, θ, N₃);
    L₄ = getCrowdsourcingCM(c, k, φ, θ);
    addFriendCrowdsourcingCM(L₁, L₂, L₃, L₄, n, m, R);
    return R;
}

OrderedList<Comment, float> getFriendsourcingCM(Course c, int k, int φ, int
θ, Set<User> N) {
    OrderedList<Comment, float> L;
    for each v in N {
        Set<Comment> CCM = getCandidateCM(v, c, k, φ, θ);
        for each cm in CCM {
            float rv = calculateRelevanceValueCM(cm);
            addList((cm, rv), L);
        }
    }
    return L;
}
```

3.4.2.3 Recommendation Algorithm of Courses

The recommendation algorithm of courses is a hybrid algorithm which uses:

- user-based collaborative filtering and demographic techniques,
- friendsourcing and crowdsourcing approaches.

This algorithm has a main function called *getRecommendationsCS* and several auxiliary functions, including *getFriendsourcingCS* and *getCrowdsourcingCS*.

3.4.2.3.1 Specification of getRecommendationsCS

The main function *getRecommendationsCS* takes as input parameters: the identifier of the active user u, the maximum number n of courses to be recommended using a friendsourcing approach, and the maximum number m of courses to be recommended using a crowdsourcing approach. The output parameter of the main function is an ordered set which contains the recommended courses for the active user u.

Firstly, the friendsourcing sets of users N_1, N_2 and N_3 are obtained in the same way as in the previous recommendation algorithms. Then, four ordered lists L_1, L_2, L_3 y L_4 are defined. These lists contain pairs which associate a candidate course for the active user u with a relevance value. L_1 is obtained after calling the auxiliary function *getFriendsourcingCS*(u, N_1). Similarly, L_2 and L_3 are obtained after calling that auxiliary function using the friendsourcing sets of users N_2 and N_3 respectively. L_4 is obtained after calling the auxiliary function *getCrowdsourcingCS*(u).

R is defined as an ordered set which contains the recommended courses for the active user u. The candidate courses obtained through friendsourcing are added to R in descending order by the relevance value without repeating, taking one recommended course at a time from each of the lists L_1, L_2 and L_3. This process continues until reaching n recommended courses or until L_1, L_2 and L_3 are empty. Then, a candidate course of L_4 is taken and added to R in descending order by the relevance value without repeating. This process continues until reaching m recommended courses or until L_4 is empty. Finally, the ordered set R which contains the recommended courses for the active user u is returned.

3.4.2.3.2 Specification of getFriendsourcingCS

This auxiliary function *getFriendsourcingCS* takes as input parameters: the identifier of the active user u and the friendsourcing set of users N_j. The output parameter of the auxiliary function is an ordered list that contains pairs which associate a candidate course for the active user u with a relevance value, obtained through a friendsourcing approach.

Initially, the ordered list L of pairs is defined. Then, the candidate courses for each user v of the set N_j are taken. These candidate courses are courses which the user v enrolled in. For each candidate course, its relevance value is calculated and added to the ordered list L.

The similarity metric used in a previous research project for a recommender system [16], considered some demographic attributes such as age, gender and nationality of users. In this project, an adaptation to this demographic similarity is utilized in order to calculate the relevance value of a specific course.

In order to obtain the demographic attributes that will be used in this research project, the information available in RedEMC is analyzed. The sources of information are the data model and a testing database provided by Evimed.

The gender could be considered when generating recommendations of courses. The similarity in gender between users u and v is calculated as shown in (10), where $gender(u)$ and $gender(v)$ are the genders of users u and v respectively.

$$\boxed{simG(u,\ v) = \begin{cases} 1: & gender(u) = gender(v) \\ 0: & gender(u) \neq gender(v) \end{cases}} \quad \textbf{(10)}$$

The country and city could also be considered in the recommendation process, since it might be of interest for the active user to enroll in courses that other users from the same country and/or city registered in. The similarity in nationality between users u and v is calculated as shown in (11), where $country(u)$ and $country(v)$ are the countries of users u and v respectively, and $city(u)$ and $city(v)$ are the cities of users u and v respectively.

$$\boxed{simN(u,\ v) = \begin{cases} 1: & country(u) = country(v) \wedge city(u) = city(v) \\ ½: & country(u) = country(v) \wedge city(u) \neq city(v) \\ 0: & in\ other\ case \end{cases}} \quad \textbf{(11)}$$

The work experience is of interest to be considered in the recommendation process. Therefore, this attribute could be discretized in two ranges: users with fifteen or less years of work experience and users with more than fifteen years of work experience. The similarity in experience is calculated as shown in (12), where $exp(u)$ and $exp(v)$ are the years of work experience of users u and v respectively.

$$\boxed{simE(u,\ v) = \begin{cases} 1: & 0 \leq exp(u), exp(v) \leq 15 \vee 15 < exp(u), exp(v) \\ 0: & in\ other\ case \end{cases}} \quad \textbf{(12)}$$

The academic profile is also of great interest to be considered in the recommendation process, since it could be attractive for the active user to enroll in courses that other users with the same profession and/or specialization registered in.

The similarity in academic profile is calculated as shown in (13), where *prof*(u) and *prof*(v) are the professions of users u and v respectively, and *mspec*(u) and *mspec*(v) are the medical specializations of users u and v respectively.

$$\boxed{simA(u, v) = \begin{cases} 1: & prof(u) = prof(v) \wedge mspec(u) = mspec(v) \\ ½: & prof(u) = prof(v) \vee mspec(u) = mspec(v) \\ 0: & in\ other\ case \end{cases}} \qquad \textbf{(13)}$$

The selected demographic attributes to be used in the calculation of the demographic similarity are: gender, country, city, work experience, profession and medical specialization.

The demographic similarity is calculated as shown in (14), where *simG*(u, v) is the similarity in gender between users u and v, *simN*(u, v) is the similarity in nationality between users u and v, *simE*(u, v) is the similarity in experience between users u and v, *simA*(u, v) is the similarity in academic profile between users u and v, and α, β, χ, δ are the weights for each of these similarity metrics.

To determine the best parameter values of the weights, several machine learning approaches could be utilized. Notice that $\alpha + \beta + \chi + \delta = 1$ and $\alpha < \beta < \chi < \delta$.

$$\boxed{simD(u, v) = (\alpha * simG(u, v) + \beta * simN(u, v) + \chi * simE(u, v) + \delta * simA(u, v)) /4} \qquad \textbf{(14)}$$

The relevance value of a course *cs* is calculated as shown in (15), where *simD*(u, v) is the demographic similarity between users u and v.

$$\boxed{rv(cs) = simD(u, v)} \qquad \textbf{(15)}$$

Notice that more than one user of the set N_j can enroll the same candidate course. Therefore, there may be repeated courses in the ordered list L. In order to process the repeated courses in L, a repetition coefficient is calculated for each course *cs* of the list L and its relevance value is adjusted.

The repetition coefficient is calculated as shown in (16), where *occurs*(*cs*, *L*) is the number of occurrences of the course *cs* in the list *L*, and *maxOccurs*(*L*) is the maximum number of repeated occurrences in the list *L*.

$$\boxed{rc(cs,\ L) = (occurs(cs,\ L)/maxOccurs(L))} \qquad \textbf{(16)}$$

The relevance value for each course *cs* is adjusted, multiplying the current relevance value by the repetition coefficient.

Then, the repeated courses of the list *L* are eliminated as going through the list in descending order by the adjusted relevance value, just keeping the occurrences associated with the maximum adjusted relevance values.

Lastly, the list *L* which contains the recommended courses for the active user *u* according to the friendsourcing set of users N_j is returned.

3.4.2.3.3 Specification of getCrowdsourcingCS

This auxiliary function *getCrowdsourcingCS* takes the active user *u* as the solely input parameter.

The output parameter of the auxiliary function is an ordered list that contains pairs which associate a candidate course for the active user *u* with a relevance value, obtained through a crowdsourcing approach.

Considering the statistics of the network, the auxiliary function takes the most popular candidate courses which users from the same country and with the same profession and/or medical specialization as those of the active user *u* enrolled in and adds them to the ordered list by their relevance value.

In this context, the relevance value of a candidate course is defined as the percentage of students enrolled in that course regarding the population of the country.

3.4.2.3.4 Pseudocode of the Recommendation Algorithm of Courses

The pseudocode of the recommendation algorithm of courses is presented below:

```
OrderedSet<Course> getRecommendationsCS(User u, int n, int m) {
    Set<User> $N_1$ = getN$_1$(u);
    Set<User> $N_2$ = getN$_2$(u);
    Set<User> $N_3$ = getN$_3$(u);
    OrderedList<Course, float> $L_1$, $L_2$, $L_3$, $L_4$;
    OrderedSet<Course> R;
    $L_1$ = getFriendsourcingCS(u, $N_1$);
    $L_2$ = getFriendsourcingCS(u, $N_2$);
    $L_3$ = getFriendsourcingCS(u, $N_3$);
    $L_4$ = getCrowdsourcingCS(u);
    addFriendsourcingCS($L_1$, $L_2$, $L_3$, n, R);
    addCrowdsourcingCS($L_4$, m, R);
    return R;
}

OrderedList<Course, float> getFriendsourcingCS(User u, Set<User> N) {
    OrderedList<Course, float> L;
    for each v in N {
        Set<Course> CCS = getCandidateCS(u, v);
        for each cs in CCS {
            float rv = calculateRelevanceValueCS (u, v, cs);
            addList((cs, rv), L);
        }
    }
    for each cs in L {
        float rc = calculateRepetitionCoefficient(cs, L);
        adjustList(cs, rc, L);
    }
    removeRepeatedCourses(L);
    return L;
}
```

3.4.3 Implementation of the Recommender System

The proposed recommendation algorithms are implemented in MySQL stored procedures and functions, using the presented pseudocodes. Note that the following codes were provided by Evimed.

3.4.3.1 Stored Procedures

The implementation of the main stored procedure *getResourcesActivitiesRecommendations* is presented below:

```
/* getResourcesActivitiesRecommendations
inputs:
int User -> identifier of the user
int Curso -> identifier of the course
int Minval -> minimum average ranking for items to be recommended
int Minvot -> minimum number of ratings for items to be recommended
int N -> number of items to be recommended using friendsourcing
int M -> number of items to be recommended using crowdsourcing
*/

CREATE PROCEDURE getResourcesActivitiesRecommendations(in User int, in
Curso int, in Minval int, in Minvot int, in N int, in M int)
BEGIN
        CALL getN(User);
        CALL getCurrentModule(Curso, @currentModule);

        DELETE FROM 'auxiliar_items_friendsourcing' WHERE 'user' = User
        AND 'id_post' = @currentModule;

        CALL getFriendsourcingResourcesActivities(User, @currentModule,
        Minval, Minvot, 1);
        CALL getFriendsourcingResourcesActivities(User, @currentModule,
        Minval, Minvot, 2);
        CALL getFriendsourcingResourcesActivities(User, @currentModule,
        Minval, Minvot, 3);
        CALL getCrowdsourcingResourcesActivities(User, @currentModule,
        Minval, Minvot);
```

```
    INSERT INTO ‘auxiliar2_items_friendsourcing’
    SELECT ‘user’, ‘id_post’, ‘titulo’, ‘RV’, ‘nivel’
    FROM (SELECT *, FIND_IN_SET (‘RV’, (SELECT GROUP_CONCAT (‘RV’
    ORDER BY ‘RV’ DESC) FROM auxiliar_items_friendsourcing a2 WHERE
    a2.‘nivel’ = a1.‘nivel’ AND a2.‘user’ = a1.‘user’ AND a2.‘id_post’ =
    a1.‘id_post’)) AS rank FROM auxiliar_items_friendsourcing a1 WHERE
    a1.‘nivel’ <> 4 AND User = a1.‘user’ AND @currentModule = a1.‘id_post’
    ORDER BY rank, ‘nivel’) as t
    WHERE 1
    GROUP BY ‘titulo’
    ORDER BY rank, ‘nivel’
    LIMIT 0, N;

    INSERT INTO ‘auxiliar2_items_friendsourcing’
    SELECT ‘user’, ‘id_post’, ‘titulo’, ‘RV’, ‘nivel’
    FROM ‘auxiliar_items_friendsourcing’
    WHERE ‘nivel’ = 4 AND ‘id_post’ = @currentModule AND ‘titulo’ NOT
    IN (SELECT ‘titulo’
        FROM ‘auxiliar2_items_friendsourcing’
        WHERE ‘user’ = User AND ‘id_post’ = @currentModule)
    ORDER BY ‘RV’ DESC
    LIMIT 0, M;

    DELETE FROM ‘auxiliar_items_friendsourcing’ WHERE ‘user’ = User
    AND ‘id_post’ = @currentModule;

    INSERT INTO ‘auxiliar_items_friendsourcing’
    SELECT * FROM ‘auxiliar2_items_friendsourcing’
    WHERE ‘user’ = User AND ‘id_post’ = @currentModule;

    DELETE FROM ‘auxiliar2_items_friendsourcing’ WHERE ‘user’ = User
    AND ‘id_post’ = @currentModule;

    SELECT * FROM ‘auxiliar_items_friendsourcing’ WHERE ‘user’ = User
    AND ‘id_post’ = @currentModule;
END
```

The auxiliary stored procedure *getN* searches the three levels of user relationships for a user and inserts this information in the auxiliary table *auxiliar_amigos*. An implementation of this procedure is presented below:

```
/* getN
inputs:
int _User -> identifier of the user
*/

CREATE PROCEDURE getN(in _User int)
BEGIN
        DELETE FROM ‘auxiliar_amigos’ WHERE ‘user’ = _User;

        INSERT INTO ‘auxiliar_amigos’ (‘user’, ‘amigo’, ‘nivel’)
        SELECT _User, u1.‘IDuser_conocido’, 1
        FROM ‘user_relaciones’ u1
        WHERE u1. ‘IDuser’ = _User AND u1. ‘IDuser_conocido’ IN
                (SELECT u2. ‘IDuser’
                 FROM ‘user_relaciones’ u2
                 WHERE u2.‘IDuser_conocido’ = u1.‘IDuser’) AND
                 u1.‘IDuser_conocido’ > 0
        GROUP BY u1.‘IDuser’, u1.‘IDuser_conocido’;

        INSERT INTO ‘auxiliar_amigos’ (‘user’, ‘amigo’, ‘nivel’)
        SELECT _User, u1.‘IDuser_conocido’, 2
        FROM ‘user_relaciones’ u1
        WHERE u1.‘IDuser’ = _User AND u1.‘IDuser_conocido’ NOT IN
                (SELECT u2.‘IDuser’
                 FROM ‘user_relaciones’ u2
                 WHERE u2.‘IDuser_conocido’ = u1.‘IDuser’) AND
                 u1.‘IDuser_conocido’ > 0
        GROUP BY u1.‘IDuser’, u1.‘IDuser_conocido’;

        INSERT INTO ‘auxiliar_amigos’ (‘user’, ‘amigo’, ‘nivel’)
        SELECT _User, a1.‘ID_amigo’, 3
        FROM ‘amigos_cursos’ a1
        WHERE a1.‘ID_usuario’ = _User AND a1.‘ID_amigo’ NOT IN
                (SELECT u1.‘IDuser_conocido’
                 FROM ‘user_relaciones’ u1
                 WHERE u1.‘IDuser’ = a1.‘ID_usuario’) AND a1.‘ID_amigo’ > 0
        GROUP BY a1.‘ID_usuario’ , a1.‘ID_amigo’;
END
```

The auxiliary stored procedure *getCurrentModule* returns the current module for a course. An implementation of this procedure is shown below:

```
/* getCurrentModule
inputs:
int Curso -> Identifier of the course
*/

CREATE PROCEDURE getCurrentModule(in Curso int, out varSalida int)
BEGIN
        SET @consulta = CONCAT(
                              'SELECT pm1.‘post_id’ as post
                              INTO @resultado
                              FROM ‘wp_postmeta’ pm1, ‘wp_postmeta’ pm2
                              WHERE pm1.‘meta_key’ = “fecha_fin_actividad” AND
                              pm1.‘meta_value’ > “2017-11-15” AND pm1.‘post_id’ =
                              pm2.‘post_id’        AND        pm2.‘meta_key’        =
                              “dt_lesson_course” AND pm2.‘meta_value’ =
                              ',Curso,'
                              ORDER BY pm1.‘meta_value’
                              LIMIT 0, 1');
        PREPARE aux FROM @consulta;
        EXECUTE aux;
        IF @resultado is null
        THEN set varSalida = -1;
        ELSE set varSalida = @resultado;
        END if;
END
```

The auxiliary stored procedure *getFriendsourcingResourcesActivities* inserts the candidate items in the auxiliary table auxiliar_items_friendsourcing. The implementation of this procedure is shown below:

```
/* getFriendsourcingResourcesActivities
inputs:
int User -> identifier of the user
int Modulo -> identifier of the module
int Minval -> minimum average ranking for items to be recommended
int Minvot -> minimum number of ratings for items to be recommended
int Nivel -> level of user relationship
*/
```

```
CREATE PROCEDURE getFriendsourcingResourcesActivities(in User int, in
Modulo int, in Minval int, in Minvot int, in Nivel int)
BEGIN
        INSERT INTO ‘auxiliar_items_friendsourcing’ (‘user’, ‘id_post’, ‘titulo’,
        ‘RV’, ‘nivel’)
        SELECT User, Modulo, v.‘nombre_seccion’, relevanceValueRA(User,
        v.‘id_usuario’, Modulo, v.‘nombre_seccion’) , 2
        FROM ‘valores_toggle’ v WHERE v.‘id_post’ = Modulo AND
        v.‘nombre_seccion’ IN (SELECT v2.‘nombre_seccion’ FROM
        ‘valores_toggle’ v2 WHERE v2.‘id_post’ = v.‘id_post’ GROUP BY
        v2.‘id_post’ , v2.‘nombre_seccion’
        HAVING AVG(v2.‘valor’) > Minval AND COUNT(v2.‘ID’) > Minvot AND
        SUM(IF(v2.‘id_usuario’ IN (SELECT a.‘amigo’ FROM ‘auxiliar_amigos’ a
        WHERE a.‘user’ = User AND a.‘nivel’ = Nivel), 1, 0)) > 0) AND
        v.‘id_usuario’ IN (SELECT a.‘amigo’ FROM ‘auxiliar_amigos’ a WHERE
        a.‘user’ = User AND a.‘nivel’ = Nivel);

        UPDATE ‘auxiliar_items_friendsourcing’
        SET ‘RV’ = (‘RV’ * repetitionCoefficient(‘user’, ‘id_post’, ‘titulo’,
        ‘nivel’))
        WHERE ‘user’ = User AND ‘id_post’ = Modulo AND ‘nivel’ = Nivel;

        INSERT INTO ‘auxiliar2_items_friendsourcing’
        SELECT *
        FROM ‘auxiliar_items_friendsourcing’ a1
        WHERE a1.‘RV’ = (SELECT MAX(a2.‘RV’) FROM
        ‘auxiliar_items_friendsourcing’ a2 WHERE a1.‘user’ = a2.‘user’ AND
        a1.‘id_post’ = a2.‘id_post’ AND a1.‘titulo’ = a2.‘titulo’ AND a1.‘nivel’ =
        a2.‘nivel’);

        DELETE FROM ‘auxiliar_items_friendsourcing’ WHERE ‘user’ = User
        AND ‘id_post’ = Modulo AND ‘nivel’ = Nivel;

        INSERT INTO ‘auxiliar_items_friendsourcing’
        SELECT * FROM ‘auxiliar2_items_friendsourcing’
        WHERE ‘user’ = User AND ‘id_post’ = Modulo AND ‘nivel’ = Nivel;

        DELETE FROM ‘auxiliar2_items_friendsourcing’
        WHERE ‘user’ = User AND ‘id_post’ = Modulo AND ‘nivel’ = Nivel;
END
```

The auxiliary stored procedure *getCrowdsourcingResourcesActivities* inserts in the candidate items in the auxiliary table auxiliar_items_friendsourcing with level 4 (crowdsourcing). The implementation of this procedure is shown below:

```
/* getCrowdsourcingResourcesActivities
inputs:
int User -> identifier of the user
int Modulo -> identifier of the module
int Minval -> minimum average ranking for items to be recommended
int Minvot -> minimum number of ratings for items to be recommended
*/

CREATE PROCEDURE getCrowdsourcingResourcesActivities(in User int, in
Modulo int, in Minval Int, in Minvot int)
BEGIN
    INSERT INTO 'auxiliar_items_friendsourcing' ('user', 'id_post', 'titulo',
    'RV', 'nivel')
    SELECT User, Modulo, v.'nombre_seccion', AVG(v.'valor'), 4
    FROM 'valores_toggle' v
    WHERE v.'id_post' = Modulo
    GROUP BY v.'id_post' , v.'nombre_seccion'
    HAVING AVG(v. 'valor') > MINVAL AND COUNT(v.'ID') > Minvot;
END
```

3.4.3.2 Functions

The main stored procedure *getResourcesActivitiesRecommendations* uses several functions, including *relevanceValueRA*, which calculates the relevance value of an item, and *repetitionCoefficient*, which calculates the value of the repetition coefficient. The implementation of these functions is shown below:

```
/* relevanceValueRA
inputs:
int User -> Identifier of the user
int Amigo -> identifier of the friend
int Modulo -> Identifier of the module
varchar(200) Seccion -> title of the section
outputs:
double
*/
```

```
CREATE FUNCTION relevanceValueRA(User int, Amigo int, Modulo int, Seccion
varchar(200)) RETURNS double
BEGIN
        DECLARE RV DOUBLE;
        DECLARE EVAL INT;
        DECLARE SIME DOUBLE;
        SELECT ‘valor’ INTO EVAL
        FROM ‘valores_toggle’ WHERE ‘id_post’ = Modulo AND
        ‘nombre_seccion’ = Seccion AND ‘id_usuario’ = Amigo;
        SELECT (SUM(5 - ABS(v1.‘valor’ - v2.‘valor’)) / COUNT(v1.‘ID’)) INTO
        SIME FROM ‘valores_toggle’ v1,
        ‘valores_toggle’ v2
        WHERE v1.‘id_usuario’ = User AND v2.‘id_usuario’ = Amigo AND
        v1.‘id_post’ = v2.‘id_post’ AND v1.‘nombre_seccion’ =
        v2.‘nombre_seccion’;
        SET RV = (EVAL * SIME) / 5;
        RETURN RV;
END

/* repetitionCoefficient
inputs:
int User -> Identifier of the user
int Modulo -> Identifier of the module
varchar(200) Seccion -> Title of the section
int Nivel -> level of user relationship
output:
double
*/

CREATE FUNCTION repetitionCoefficient(User int, Modulo int, Seccion
varchar(200), Nivel int) RETURNS double
BEGIN
        DECLARE RC DOUBLE;
        DECLARE cantREP INT;
        DECLARE cantMaxRep INT;

        SELECT COUNT(*) INTO cantREP
        FROM ‘auxiliar_items_friendsourcing’
        WHERE ‘user’ = User AND ‘id_post’ = Modulo AND ‘titulo’ = Seccion
        AND ‘nivel’ = Nivel;
        SELECT COUNT(*) INTO cantMaxRep
        FROM 'auxiliar__items__friendsourcing'
        WHERE 'user' = User AND 'id__post' = Modulo AND 'nivel' = Nivel
        GROUP BY 'titulo'
        ORDER BY count(*)
        DESC LIMIT 0, 1;
        SET RC = (cantREP/cantMaxRep);
        RETURN RC;
END
```

3.4.4 Testing of the Recommender System

The main aim of the testing is to find defects in order to improve the quality of the proposed recommender system. In this section, test cases which allow to verify the different recommendation algorithms are defined.

Firstly, the sets of test users, test items, and system operations are specified. Then, an example of interaction diagram which shows the interchange of messages in form of system operations between test users are presented.

Finally, the test cases mentioned in the interaction diagrams are specified, indicating the state of the sets N_1, N_2, N_3, and the parameter values.

3.4.4.1 Test Users, Test Items and System Operations

In this section, test users, test items and systems operations used in the different test cases are specified. Table 3 shows the different test users considered in test cases.

Test User Identifier	Test User Name
A	Antonio López
C	Camilo Caminada
D	Daniel González
L	Libertad Tansini

Table 3 - Test users.

Then, resources, activities and comments of a course C are specified in Table 4.

Test Item Identifier	Test Item Name	Notes
R_i	Resource_i, $i = 1..4$	It is a resource of the current module
R_j	Resource_j, $j = 5..8$	It is a resource of the immediate previous module
A_i	Activity_i, $i = 1..4$	It is an activity of the current module
A_j	Activity_j, $j = 5..8$	It is an activity of the immediate previous module
C_i	Comment_i, $i = 1..4$	It is a comment published in the last day
C_5	Comment_5	It is a comment published in the last three days
C_6	Comment_6	It is a comment published in the last five days
C_7	Comment_7	It is a comment published in the last six days
C_8	Comment_8	It is a comment published in the last seven days

Table 4 - Test items.

The system operations considered in the testing are listed below:

- *esConocido*(U_1, U_2): User U_1 explicitly states that he knows user U_2, which means $U_2 \in N_2(U_1)$. If it is also true that *esConocido*(U_2, U_1) then $U_2 \in N_2(U_1)$.

- *esContacto*(U_1, U_2): User U_1 indicates that user U_2 is an acquaintance, which means $U_2 \in N_3(U_1)$

- *evaluarItem*(U, I, V): User U rates item I (resource (R) or activity (A)) with value V.

- *visualizar*(U, C): User U visualizes comment C.

- *meGusta*(U, C): User U likes comment C. This operation implies visualizing the comment first.

- *obtenerRec*(U, I, CP): User U obtains personalized recommendations of items of type I (resources (R), activity (A) or comment(C), through the test case CP.

3.4.4.2 Interaction Diagrams

Fig. 15 and Fig. 16 depict examples of interaction diagrams which shows the interchange of messages in form of system operations between test users. Initially, the friendsourcing sets N_1, N_2 and N_3 are empty.

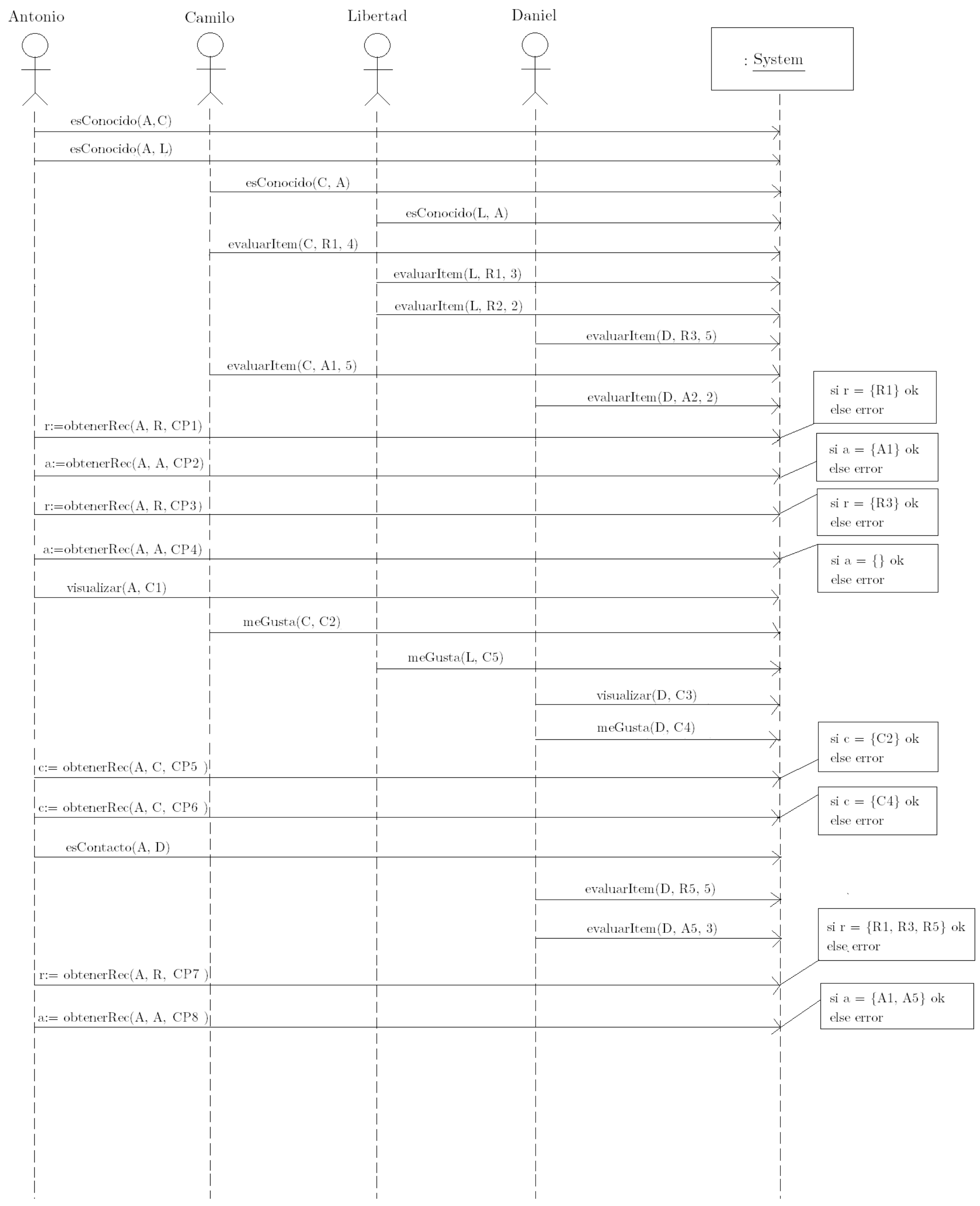


Fig. 15 - First example of interaction diagram between users.

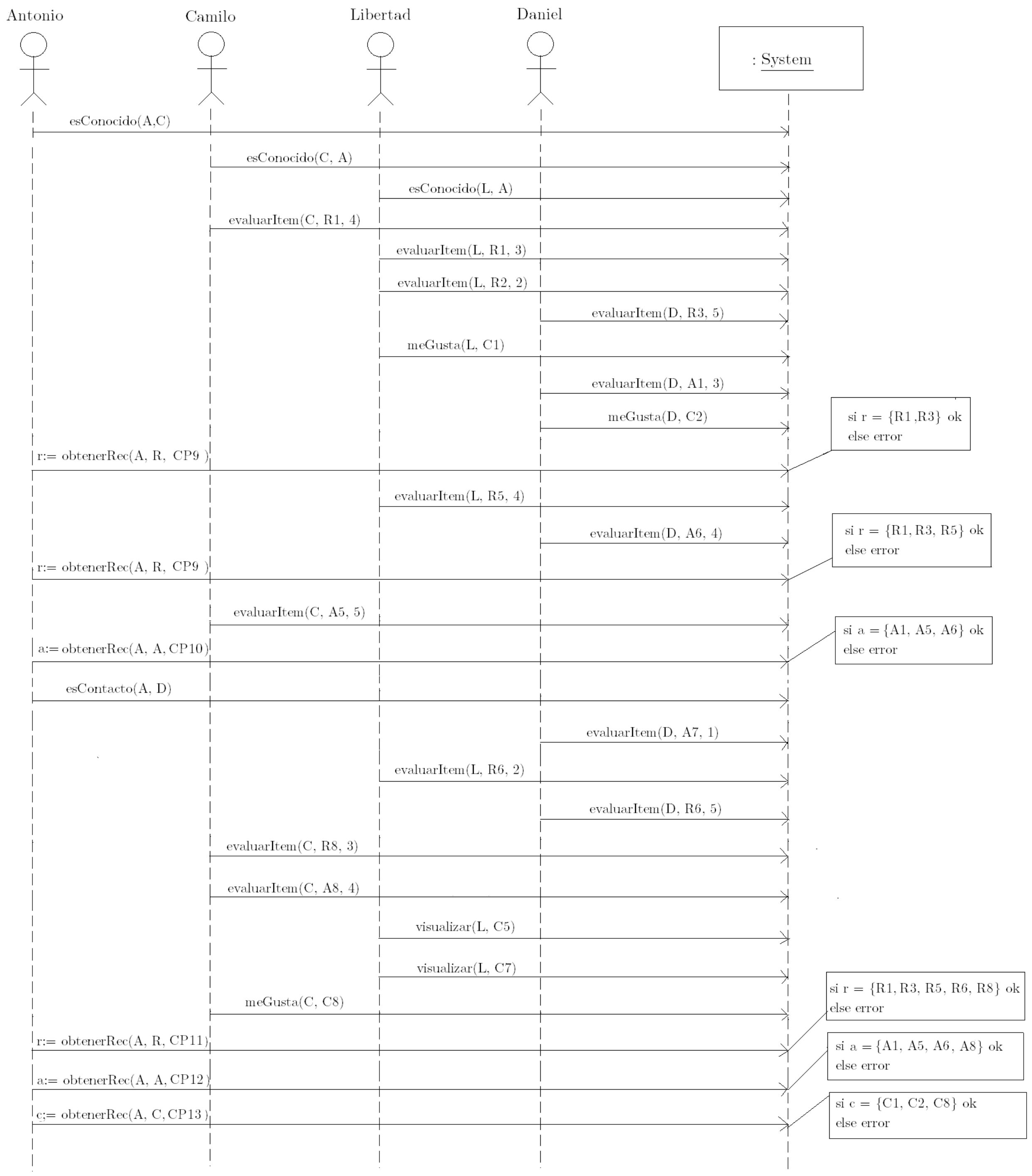


Fig. 16 - Second example of interaction diagram between users.

3.4.4.3 Test Cases

Table 5 shows the different test cases for the three recommendation algorithms. For each test case, the following information is specified:

- Identifier: Indicates the identifier of the test case.
- Type of recommended item: Indicates the sort of the recommended item. The values are "Resource", "Activity", or "Comment".
- Pre-conditions: Indicate the conditions before generating personalized recommendations, regarding the friendsourcing sets of users (N_1, N_2, N_3) and the parameter values (u, c, s, $nmod$, λ, σ, k, φ, θ, n, m), mentioned earlier in section 3.4.1.1.
- Expected results: Indicate the set of recommended items and the origin of the recommendations regarding the lists L_1, L_2, L_3 or L_4, specified earlier in section 3.4.2.

Test Case Identifier	Type of Recommended Item	Pre-Conditions	Expected Results
CP_1	Resource	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) = \emptyset$, $c = C$, $s =$ 'R', $nmod = 0$, $\lambda = 3.0$, $\sigma = 1$, $n = 5$, $m = 0$	Active user u receives from 0 to 5 recommended resources of the current module of the course C, whose average rating is greater or equal to 3.0 and have one or more views. Recommendations came from L_1
CP_2	Activity	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) = \emptyset$, $c = C$, $s =$ 'A', $nmod = 0$, $\lambda = 3.0$, $\sigma = 1$, $n = 5$, $m = 0$	Active user u receives from 0 to 5 recommended activities of the current module of the course C, whose average rating is greater or equal to 3.0 and have one or more views. Recommendations came from L_1
CP_3	Resource	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) = \emptyset$, $c = C$, $s =$ 'R', $nmod = 0$, $\lambda = 3.0$, $\sigma = 1$, $n = 0$, $m = 5$	Active user u receives from 0 to 5 recommended resources of the current module of the course C, whose average rating is greater or equal to 3.0 and have one or

			more views. Recommendations came from L_4
CP_4	Activity	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) = \emptyset$, $c = C$, $s =$ 'A', $nmod = 0$, $\lambda = 3.0$, $\sigma = 1$, $n = 0$, $m = 5$	Active user u receives from 0 to 5 recommended activities of the current module of the course C, whose average rating is greater or equal to 3.0 and have one or more views. Recommendations came from L_4
CP_5	Comment	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) = \emptyset$, $c = C$, $k = 1$, $\varphi = 1$, $\theta = 1$, $n = 5$, $m = 0$	Active user u receives from 0 to 5 recommended comments of the course C published in the last day which have at least one "like" and one view. Recommendations came from L_1
CP_6	Comment	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) = \emptyset$, $c = C$, $k = 1$, $\varphi = 1$, $\theta = 1$, $n = 0$, $m = 5$	Active user u receives from 0 to 5 recommended comments of the course C, published in the last day which have at least one "like" and one view. Recommendations came from L_4
CP_7	Resource	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) \neq \emptyset$, $c = C$, $s =$ 'R', $nmod = 1$, $\lambda = 3.0$, $\sigma = 1$, $n = 3$, $m = 2$	Active user u receives from 0 to 5 recommended resources of the current and immediate previous module of the course C, whose average rating is greater or equal to 3.0 and have one or more ratings. Recommendations came from L_1, L_3 o L_4
CP_8	Activity	$N_1(u) \neq \emptyset$, $N_2(u) = \emptyset$, $N_3(u) \neq \emptyset$, $c = C$, $s =$ 'A', $nmod = 1$, $\lambda = 3.0$, $\sigma = 1$, $n = 3$, $m = 2$	Active user u receives from 0 to 5 recommended activities of the current and immediate previous module of the course C, whose average rating is greater or equal to 3.0 and have one or more ratings. Recommendations came from L_1, L_3, o L_4
CP_9	Resource	$N_1(u) \neq \emptyset$, $N_2(u) \neq \emptyset$, $N_3(u) = \emptyset$, $c = C$, $s =$ 'R', $nmod = 1$, $\lambda = 3.0$, $\sigma = 1$, $n = 3$, $m = 2$	Active user u receives from 0 to 5 recommended resources of the current and immediate previous module of the course C, whose

			average rating is greater or equal to 3.0 and have one or more ratings. Recommendations came from L_1, L_2, o L_4
CP_{10}	Activity	$N_1(u) \neq \emptyset$, $N_2(u) \neq \emptyset$, $N_3(u) = \emptyset$, $c = C$, $s =$ *'A'*, $nmod = 1$, $\lambda = 3.0$, $\sigma = 1$, $n = 3$, $m = 2$	Active user u receives from 0 to 5 recommended activities of the current and immediate previous module of the course C, whose average rating is greater or equal to 3.0 and have one or more ratings. Recommendations came from L_1, L_2, o L_4
CP_{11}	Resource	$N_1(u) \neq \emptyset$, $N_2(u) \neq \emptyset$, $N_3(u) \neq \emptyset$, $c = C$, $s =$ *'R'*, $nmod = 1$, $\lambda = 3.0$, $\sigma = 1$, $n = 3$, $m = 2$	Active user u receives from 0 to 5 recommended resources of the current and immediate previous module of the course C, whose average rating is greater or equal to 3.0 and have one or more ratings. Recommendations came from L_1, L_2, L_3, o L_4
CP_{12}	Activity	$N_1(u) \neq \emptyset$, $N_2(u) \neq \emptyset$, $N_3(u) \neq \emptyset$, $c = C$, $s =$ *'A'*, $nmod = 1$, $\lambda = 3.0$, $\sigma = 1$, $n = 3$, $m = 2$	Active user u receives from 0 to 5 recommended activities of the current and immediate previous module of the course C, whose average rating is greater or equal to 3.0 and have one or more ratings. Recommendations came from L_1, L_2, L_3, o L_4
CP_{13}	Comment	$N_1(u) \neq \emptyset$, $N_2(u) \neq \emptyset$, $N_3(u) \neq \emptyset$, $c = C$, $k = 7$, $\varphi = 1$, $\theta = 1$, $n = 3$, $m = 2$	Active user u receives from 0 to 5 recommended comments of the course C, published in the last seven days which have at least one "like" and one view. Recommendations came from L_1, L_2, L_3, o L_4

Table 5 - Test cases for the personalized recommendation algorithms.

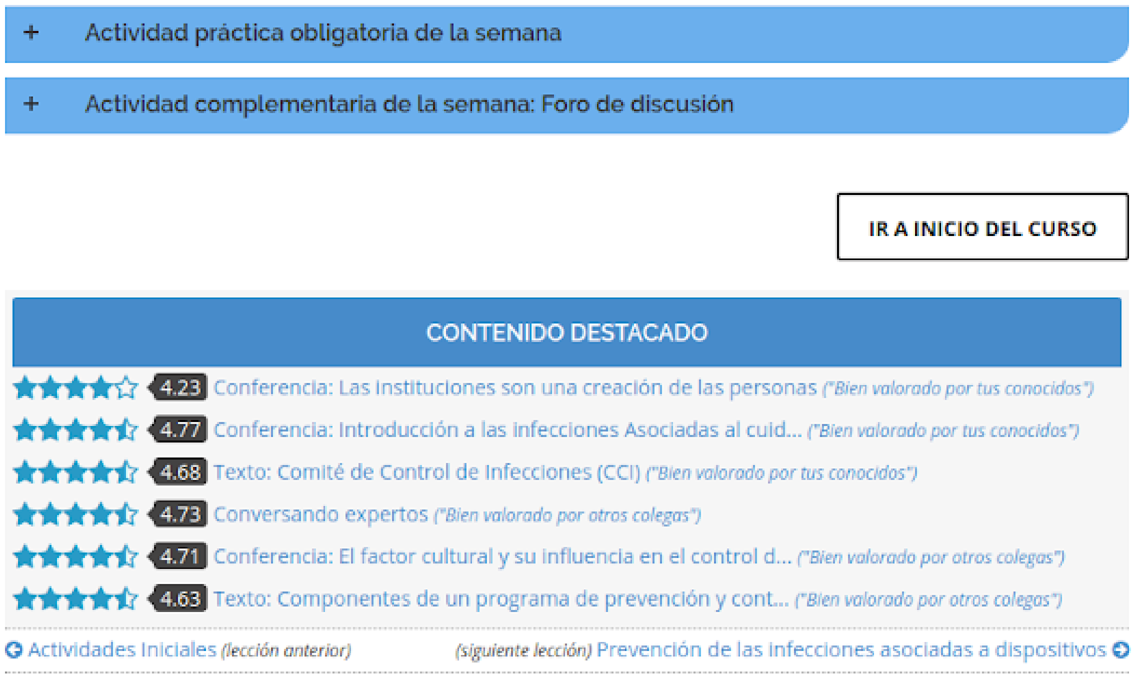
3.4.5 Deployment in Production Environment

After the testing phase, the deployment of the recommender system for RedEMC is done in a production environment. In this section, it is described how to present personalized recommendations to users in RedEMC. Then, the performance of the recommender system is analyzed.

3.4.5.1 Personalized Recommendations in RedEMC

Fig. 17 depicts six personalized recommendations of resources and activities for the active user in RedEMC.

The average rating in stars and in numeric form are presented for each recommended resource/activity, next to the name of the resources/activities, as well as an explanation of the recommendations.

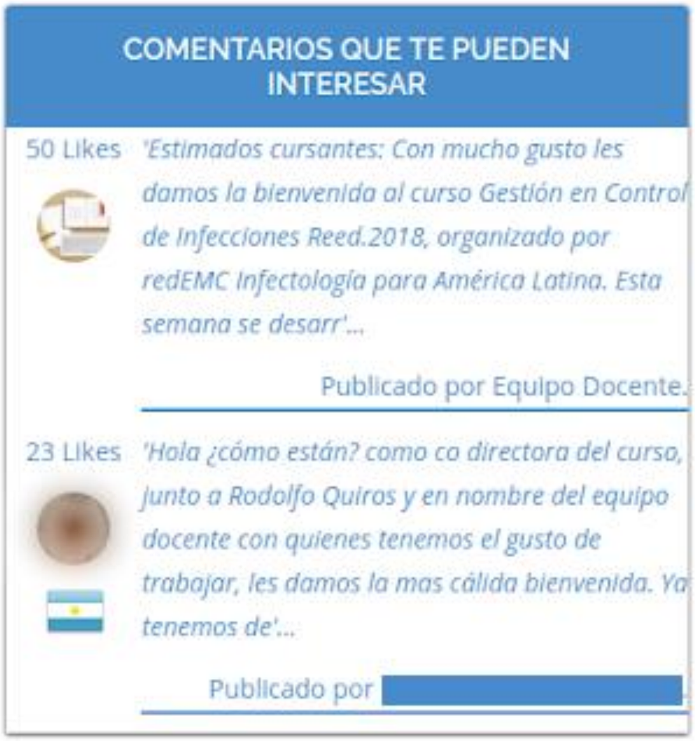

+ Actividad práctica obligatoria de la semana
+ Actividad complementaria de la semana: Foro de discusión
IR A INICIO DEL CURSO
CONTENIDO DESTACADO

4.23 Conferencia: Las instituciones son una creación de las personas ("Bien valorado por tus conocidos")
4.77 Conferencia: Introducción a las infecciones Asociadas al cuid... ("Bien valorado por tus conocidos")
4.68 Texto: Comité de Control de Infecciones (CCI) ("Bien valorado por tus conocidos")
4.73 Conversando expertos ("Bien valorado por otros colegas")
4.71 Conferencia: El factor cultural y su influencia en el control d... ("Bien valorado por otros colegas")
4.63 Texto: Componentes de un programa de prevención y cont... ("Bien valorado por otros colegas")

Actividades Iniciales (lección anterior) (siguiente lección) Prevención de las infecciones asociadas a dispositivos

Fig. 17 - Personalized recommendations of resources and activities in RedEMC.

Fig. 18 depicts two recommendations of comments for the active user in RedEMC. The number of likes and the user who published the comment is presented next to the text of the comment.



COMENTARIOS QUE TE PUEDEN INTERESAR

50 Likes 'Estimados cursantes: Con mucho gusto les damos la bienvenida al curso Gestión en Control de Infecciones Reed.2018, organizado por redEMC Infectología para América Latina. Esta semana se desarr'...

Publicado por Equipo Docente.

23 Likes 'Hola ¿cómo están? como co directora del curso, junto a Rodolfo Quiros y en nombre del equipo docente con quienes tenemos el gusto de trabajar, les damos la mas cálida bienvenida. Ya tenemos de'...

Publicado por [redacted]

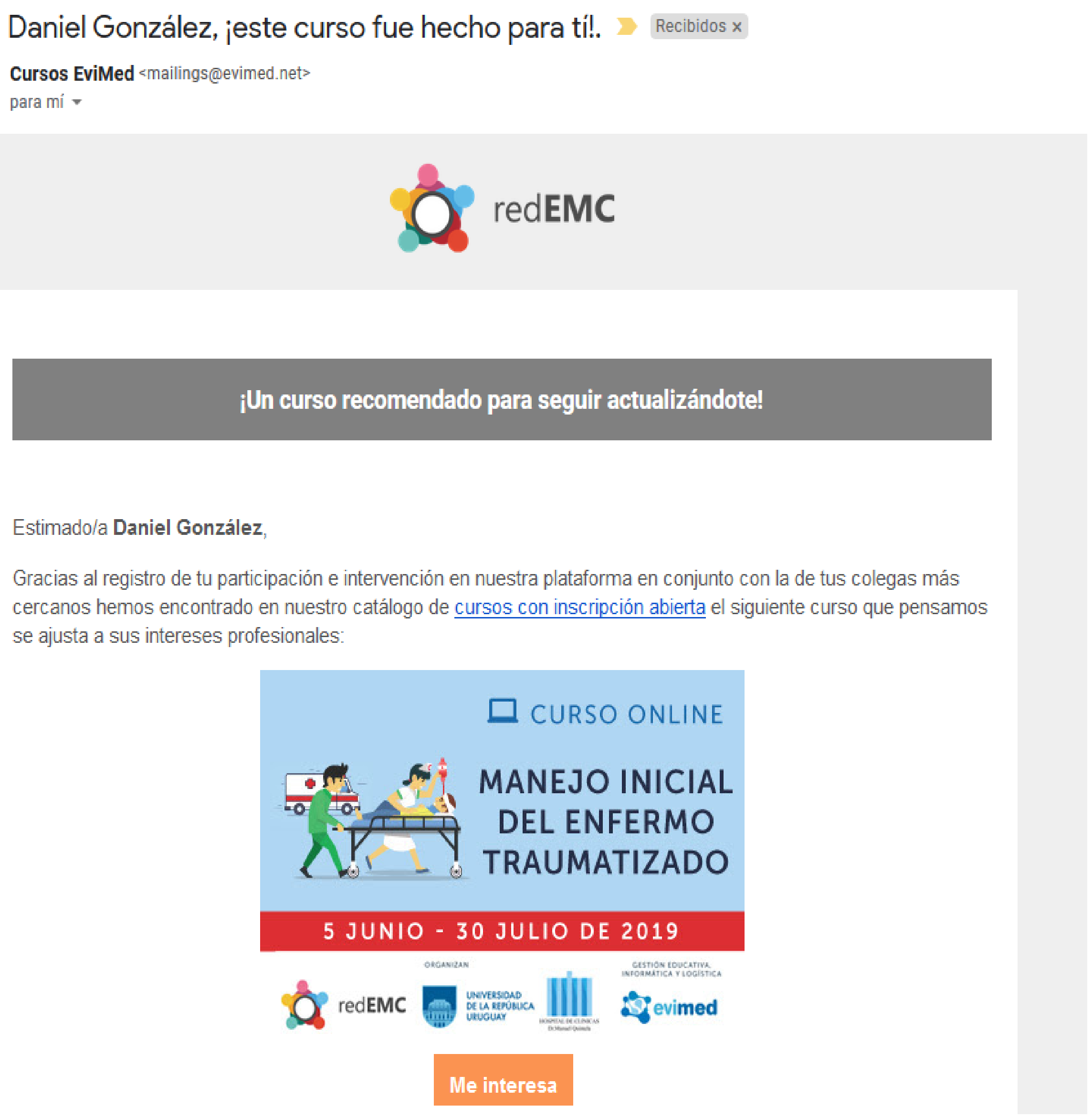
Fig. 18 - Personalized recommendations of comments in RedEMC.

After visualizing a recommended item (resource/activity or comment), the recommender system offers the possibility to indicate whether the recommendation has been useful. This information is relevant to analyze the efficiency of the recommendation process.

Fig. 19 shows a personalized recommendation of courses for the active user via email.

The subject of the email has the form: "<*User Name*>, this course was done for you!". Students have the possibility to indicate that the recommended course is of his/her interest.

The number of recommended courses is determined by the parameters which indicate the maximum number of comments to be recommended using friendsourcing and crowdsourcing approaches.



Daniel González, ¡este curso fue hecho para tí!.

Cursos EviMed <mailings@evimed.net>

para mí

¡Un curso recomendado para seguir actualizándote!

Estimado/a **Daniel González**,

Gracias al registro de tu participación e intervención en nuestra plataforma en conjunto con la de tus colegas más cercanos hemos encontrado en nuestro catálogo de cursos con inscripción abierta el siguiente curso que pensamos se ajusta a sus intereses profesionales:

CURSO ONLINE
MANEJO INICIAL DEL ENFERMO TRAUMATIZADO
5 JUNIO - 30 JULIO DE 2019
ORGANIZAN
redEMC
UNIVERSIDAD DE LA REPÚBLICA URUGUAY
GESTIÓN EDUCATIVA, INFORMÁTICA Y LOGÍSTICA
evimed
Me interesa

Fig. 19 - Personalized recommendation of courses via email.

Giving feedback about personalized recommendations is another way to collaborate in the virtual learning environment. After receiving a recommendation of a resource or activity, users could indicate whether the recommended resource/activity is useful and after receiving a recommendation of a course, users could express whether the recommended course is of his/her interest.

3.4.5.2 Performance of the Recommender System

Recommendation algorithms are executed in different moments, depending on the type of recommended item.

For each recommendation algorithm, its average execution time is calculated:

- The recommendation algorithm of resources/activities and comments are executed when users enter the virtual learning environment. The average execution time is approximately 1.0 second per user.

- The recommendation algorithm of courses is executed once for all the users of the virtual learning environment. This process lasts four hours approximately for fifteen thousand students. Therefore, the average execution time is approximately 0.96 seconds per user.

Table 6 shows the average execution times for recommended items.

Recommended Item	Average Execution Time (in seconds)
Resource	1.0
Activities	1.0
Courses	0.96

Table 6 - Average execution times for recommended items.

With these average execution times it is possible to indicate that the proposed recommendation algorithms have good performances. This is due to the implementation of the algorithms in stored procedures and functions.

Chapter 4

User Feedback Analysis

This chapter presents the analysis of the user feedback considering the responses given by students, after receiving personalized recommendations of items in a production environment. The proposed analysis aims to indicate the usefulness of the recommendations and their impact in the learning process.

4.1 Parameter Values and Weights used in Production

In this section, the parameter values and weights used in production are specified below:

- For the recommendation algorithm of resources and activities:

  getResourcesActivitiesRecommendations(in User int, in Curso int, in Nmod int, in Minval int, in minvot int, in N int, in M int)

  The call of this stored procedure passed the following values:

  CALL getResourcesActivitiesRecommendations($userid, $curso_inscripto, 5, 1, 1, 3, 3)

- For the recommendation algorithm of comments:

  getCommentsRecommendations(in User int, in Curso int, in K int, in Minval int, in Minvis int, in N int, in M int)

  The call of this stored procedure passed the following values:

  CALL getCommentsRecommendations(get_current_user_id(), get_course_id(), 15, 3, 5, 2, 2)

- For the recommendation algorithm of courses:

  getCoursesRecommendations(in User int, in N int, in M int)

The call of this stored procedure passed the following values:

CALL getCoursesRecommendations ($userid, 2, 2)

The weights used when calculating demographic similarity between users are the following:

$\alpha = 0.1$, $\beta = 0.15$, $\chi = 0.25$, $\delta = 0.5$

4.2 Collecting User Feedback

The user feedback was collected in different periods of time. Notice that the recommendation algorithms of resources/activities and comments were the former two algorithms implemented and put into production. The latter was the recommendation algorithm of courses. Note that the following information about user feedback was provided by Evimed.

4.2.1 User Feedback for the Recommendations of Resources and Activities

In June 2018, the proposed recommender system for RedEMC was put into a production environment. The following user feedback was collected in the first fortnight of June. In this period of time, 266 resources and activities were rated by users, 538 users have received personalized recommendations and 77 recommended resources or activities were clicked [8].

Table 7 depicts the origin of the feedback for the recommendations of resources and activities, regarding the friendsourcing and crowdsourcing sets of users, and the number of positive explicit feedback obtained.

Origin	Number of positive explicit feedback
Friendsourcing (N_1)	4
Friendsourcing (N_2)	3
Friendsourcing (N_3)	0
Crowdsourcing	8

Table 7 - Positive explicit feedback for the recommendations of resources and activities.

Table 8 depicts the origin of the feedback for the recommendations of resources and activities, regarding the friendsourcing and crowdsourcing sets of users, and the number of negative explicit feedback obtained.

Origin	Number of negative explicit feedback
Friendsourcing (N_1)	0
Friendsourcing (N_2)	0
Friendsourcing (N_3)	0
Crowdsourcing	1

Table 8 - Negative explicit feedback for the recommendations of resources and activities.

Considering the implicit feedback, which means that the active user did not indicate explicitly whether recommended items were useful, 14% of the recommendations were viewed and 86% of the recommendations were not viewed. Fig. 20 shows a pie chart which indicates the percentages of recommendations of resources and activities viewed and not viewed by the active user, regarding implicit feedback.

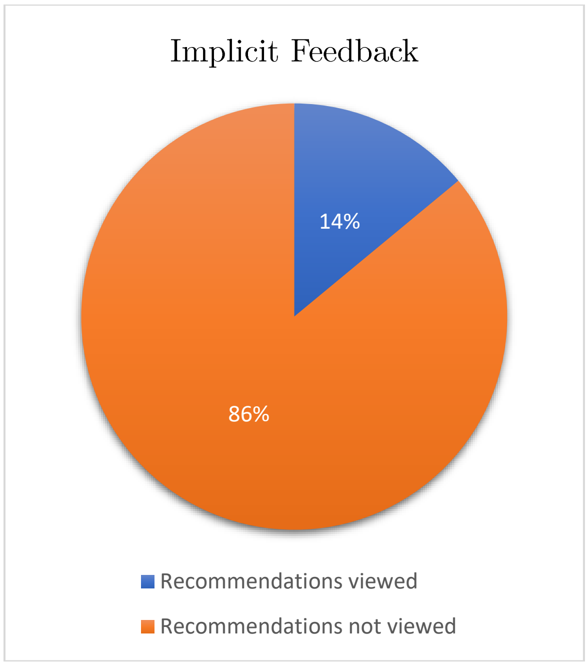


Fig. 20 - Implicit feedback for the recommendations of resources and activities.

Most of the recommendations of resources and activities generated by the recommender system were not viewed. There are several explanations to justify the students' behavior, including the lack of interest for discovering new learning objects.

Considering the explicit feedback, which means that the active user indicates explicitly whether recommended resources and activities were useful, 94% of the recommendations were rated positively and 6% of the recommendations were rated negatively.

Fig. 21 shows a pie chart which indicates the percentages of recommendations of resources and activities rated positively and negatively by active users, regarding the explicit feedback.

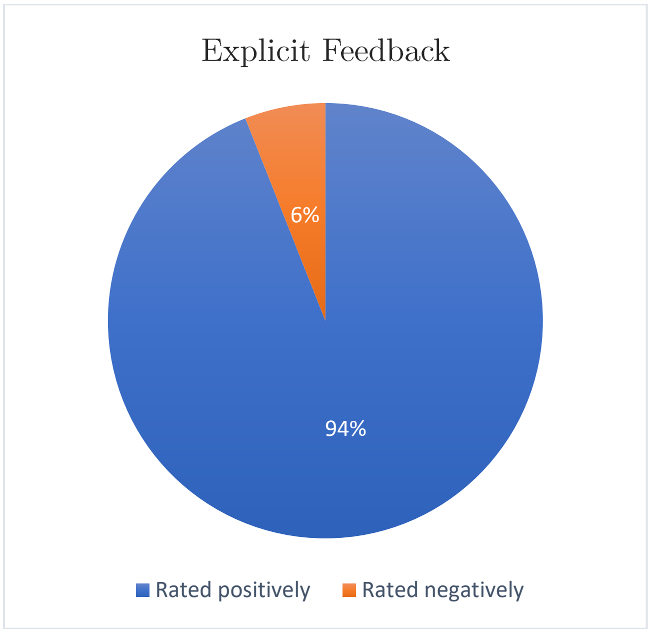


Fig. 21 - Explicit feedback for the recommendations of resources and activities.

Regardless of the low percentage of students who gave explicit feedback, it is possible to conclude that personalized recommendations of resources and activities were useful and significant to students. These recommendations allow students to discover new resources and activities which contribute to enhancing their learning process, since they can be more aware of the most relevant learning objects in RedEMC.

A relevant functional requirement in this research project implied tracing the origin of personalized recommendations in order to explain users the reasoning behind the suggestions.

Fig. 22 shows the origin of the recommendations of resources and activities, according to the friendsourcing sets of users (N_1, N_2, N_3) and the crowdsourcing set of users (N_4). These results indicate the source of the suggestions, regarding the users and their relationships with the active user.

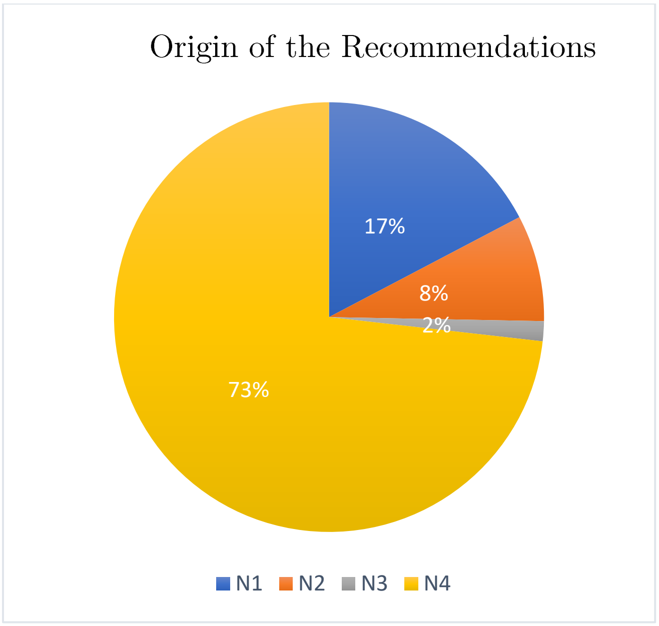


Fig. 22 - Origin of the recommendations of resources and activities.

Notice that the majority of the recommendations (73%) were originated from the crowdsourcing set of users N_4, which implies that personalized suggestions were generated considering the whole set of users of the course, instead of considering the information provided from a reduced group of users.

This result is probably due to the cold-start problem, since there may not be enough friendsourcing relationships between users or there may not be enough ratings of items at the time of generating recommendations.

4.2.2 User Feedback for the Recommendations of Comments

The following user feedback was also collected in the first fortnight of June. In this period of time, 2635 comments were published in discussion forums, 2649 users have received personalized recommendations and 700 recommended comments were clicked [8].

Table 9 depicts the origin of the feedback for the recommendations of comments, regarding the friendsourcing and crowdsourcing sets of users, and the number of positive explicit feedback obtained.

Origin	Number of positive explicit feedback
Friendsourcing (N_1)	0
Friendsourcing (N_2)	4
Friendsourcing (N_3)	0
Crowdsourcing	13

Table 9 - Positive explicit feedback for the recommendations of comments.

Table 10 depicts the origin of the feedback for the recommendations of comments, regarding the friendsourcing and crowdsourcing sets of users, and the number of negative explicit feedback obtained.

Origin	Number of negative explicit feedback
Friendsourcing (N_1)	0
Friendsourcing (N_2)	0
Friendsourcing (N_3)	0
Crowdsourcing	0

Table 10 - Negative explicit feedback for the recommendations of comments.

Considering the implicit feedback, which means that the active user did not indicate explicitly whether recommended items were useful, 26% of the recommendations were viewed and 74% of the recommendations were not viewed. Fig. 23 shows a pie chart which indicates the percentages of recommendations of comments viewed and not viewed by active users, regarding implicit feedback. As in the previous case, most of the recommendations of comments generated by the recommender system were not viewed.

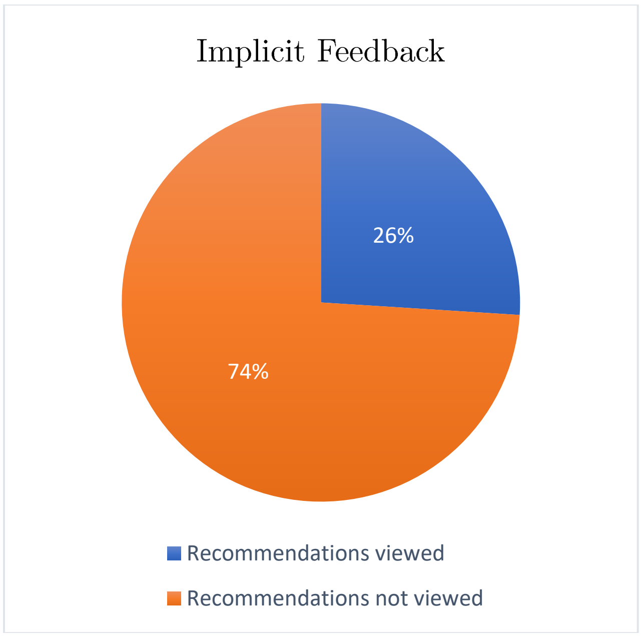


Fig. 23 - Implicit feedback for the recommendations of comments.

Considering the explicit feedback, which means that the active user indicates explicitly whether recommended comments were useful, 100% of the recommendations were rated positively and 0% of the recommendations were rated negatively. Fig. 24 shows a pie chart which indicates the percentages of recommendations of comments rated positively and negatively by the active user, regarding the explicit feedback.

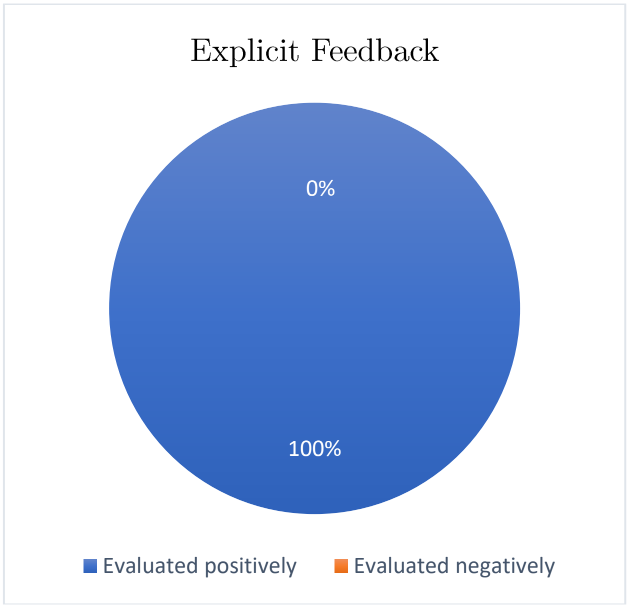


Fig. 24 - Explicit feedback for the recommendations of comments.

Despite the low percentage of students who gave explicit feedback, it is possible to conclude that personalized recommendations of comments were meaningful to students and contribute to enhancing the learning process of the students. These recommendations of comments allow students to be more aware of interesting exchanges among colleagues and teachers in different discussion forums, which helps to fulfill the academic objective mentioned earlier in Chapter 3.

Regarding the traceability of the recommendations, Fig. 25 shows the origin of the recommendations of comments, according to the friendsourcing and crowdsourcing sets of users.

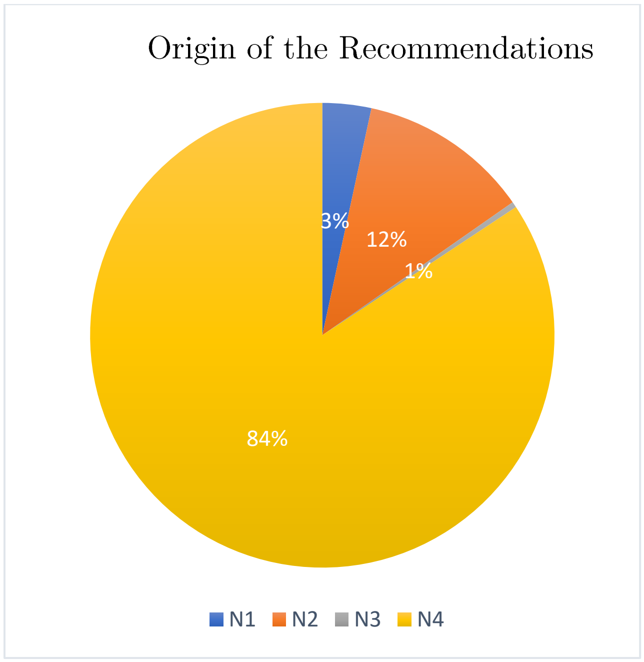


Fig. 25 - Origin of the recommendations of comments.

The majority of the recommendations (84%) were also originated from the crowdsourcing set of users N_4, which implies that personalized suggestions were generated considering the whole set of users of the course, instead of considering the information provided from a reduced group of users. This result is also probably due to the cold-start problem, since there may not be enough friendsourcing relationships between users at the time of generating recommendations.

4.2.3 User Feedback for the Recommendations of Courses

Evimed sends emails to users including: newsletters, invitations to courses and webinars, and course announcements among others [33]. After implementing the recommendation algorithm of courses, a new type of email is considered, which provides personalized recommendations of courses using the suggestions provided by the proposed algorithm.

In a period of three months approximately (from September to November 2018), the number of emails, the percentages of opened emails, and the percentages of clicked emails were registered. Table 11 presents these data about some types of emails sent by Evimed during the trimester.

Date	Type of Email	Sent	Opened (%)	Clicked (%)
2018-11-27	Personalized recommendations of courses	12932	23.65	1.38
2018-11-26	Get discounts on your courses	15974	6.45	0.58
2018-11-20	First-semester online courses	27991	12.28	2.08
2018-10-29	[One week free] Access to the course	16224	14.46	0.68
2018-10-24	[Two weeks free] Access to the course	16290	13.77	0.86
2018-10-04	Watch a free webinar	13766	14.45	2.04
2018-09-10	I invite you to be part of this course	21565	25.64	1.89
2018-09-05	Your opinion matters to us	17025	10.88	1.47
2018-09-03	Meet the international teachers of this course	16095	13.53	0.77

Table 11 - Types of email sent by Evimed.

Almost 24% of the emails which contain personalized recommendations of courses were opened by the active user. Regarding Evimed's expertise, the percentage of opened emails is high for this type of email.

Notice that the percentage of clicked emails shows the interest in the recommended courses and therefore, it may imply possible enrollments which helps to fulfill the commercial objective mentioned earlier in Chapter 3.

Chapter 5

Conclusions and Future Work

This chapter presents the conclusions of the research project, regarding collaborative recommendations in virtual learning environments. Then, several lines of future work are specified.

5.1 Conclusions

In this research project, it is explained how to process valuable information about collaborations in virtual learning environments, to generate meaningful recommendations of educational items.

Evimed encouraged students to participate and collaborate in its virtual learning environment by creating and maintaining their academic network, rating different types of items, and giving explicit and implicit feedback, in order to produce meaningful and personalized recommendations of items in RedEMC [8].

In this research project, a real recommender system for a virtual learning environment in the context of CME was proposed, designed, developed and put into production, fulfilling the academic and commercial objectives specified by Evimed. Three distinct recommendation algorithms which produce personalized recommendations of resources and activities, comments, and courses were designed and implemented. These algorithms utilized information about different sorts of collaborations in the virtual learning environment, relationships between users, as well as information about user's profile such as demographic data. Different collaborative filtering techniques were employed including user-based and item-based collaborative filtering, as well as demographic techniques.

Recommendation algorithms utilized distinct friendsourcing and crowdsourcing sets of users to generate meaningful suggestions of items. These sets of users determined the origin of personalized recommendations. To calculate how similar is a specific student to the active user, several similarity metrics were used. After implementing the recommendation algorithms, a set of test cases were defined to verify the proposed recommender system.

User feedback was collected during different periods of time. According to the explicit feedback, more than 90% of the recommendations of resources/activities and comments were rated positively by RedEMC's users. These results indicate that collaborative recommendations in virtual learning environments enhance each students' learning process and user's experience and could stimulate further collaboration.

5.2 Future Work

In this section, several lines of future work are described in relation to collaborative recommendations.

Due to the limited time of the research project, there was not enough time to adequately test the recommender system for RedEMC [8]. Despite this fact, Evimed's team executed some of the proposed test cases. It is desirable to make extensive tests to detect possible improvements, and therefore to enhance recommender system's quality.

The different recommendation algorithms use parameters and weights to determine personalized recommendations of items. Initially, Evimed considered our suggestions about parameter values and weights. These parameters values and weights were set manually by a system administrator. It could be interesting to consider several machine learning techniques in order to determine and set optimal parameter values and weights, enhancing the efficiency of the recommender system.

Regarding relationships between users, the proposed recommender system only considered one-hop relationships, including the students who knew the active user and the ones who were acquaintances of him/her to determine personalized suggestions of items.

An interesting line of future work is to consider other types of relationships in the virtual learning environment, such as multi-hop relationships.

In relation to the recommendations of comments, the proposed recommendation algorithm did not regard comments which were semantically duplicated. It is desirable to discard duplicated comments which have the same meaning. To achieve this goal, Natural Language Processing techniques could be used [32].

One important requirement in this research project was to trace the origin of the recommendations [8]. Defining a general model of traceability which utilizes recommendation approaches and techniques, relationships between users and recommended items, could be considered as future work. This model could be used to indicate how personalized recommendations were generated for the active user.

Another interesting line of future work is to create appropriate justifications for the recommendations, regarding information about the relationships between users in the virtual learning environment. These justifications explain the criteria used to produce meaningful suggestions, enabling users to gain confidence and transparency in the recommender system [34].

It will be interesting to employ collaborative recommendations in the context of local institutional repositories, in order to assist users in discovering academic documents that may be of interest to them, considering their relationships and collaborations in the platform.

References

[1] F. Ricci, L. Rokach, B. Shapira, “Recommender Systems: Introduction and Challenges”. In: F. Ricci, L. Rokach, B. Shapira (eds), Recommender Systems Handbook, Springer, Boston, MA, 2015.

[2] W. F. McComas, “Virtual Learning Environment”. In: W. F. McComas (eds) The Language of Science Education. SensePublishers, Rotterdam, 2014.

[3] Moodle. [Online] Available: https://moodle.org [Accessed May 29, 2019]

[4] Moodle plugins directory: Analytics and Recommendations. [Online] Available: https://moodle.org/plugins/block_analytics_recommendations [Accessed May 29, 2019]

[5] Moodle plugins directory: Recommender. [Online] Available: https://moodle.org/plugins/block_recommender [Accessed May 29, 2019]

[6] D. González, R. Motz, L. Tansini, “Improving Social Collaborations in Virtual Learning Environments”. In: I. Ciuciu et al. (eds), On the Move to Meaningful Internet Systems: OTM 2015 Workshops, OTM 2015, Lecture Notes in Computer Science, vol 9416, Springer, Cham, 2015.

[7] RedEMC. [Online] Available: https://redemc.net [Accessed May 29, 2019]

[8] D. González, L. Tansini, “Collaborative Learning Recommendations for Continuing Medical Education in Virtual Learning Environments”, presented at Co-creation of Curricula, Tools and Educational Scenarios for Building Soft Competences for Personal Development and Employability Conference (Co-Create! 2018), Tartu, Estonia, 17th September 2018.

[9] R. Sinha, K. Swearingen, “The Role of Transparency in Recommender Systems”. In: Conference on Human Factors in Computing Systems, pp. 830-831, ACM Press, 2002.

[10] B. Patel, P. Desai, U. Panchal, “Methods of Recommender System: A Review”. In: 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017.

[11] M. Braunhofer, “Hybrid Solution of the Cold-Start Problem in Context-Aware Recommender Systems”. In: V. Dimitrova, T. Kuflik, D. Chin, F. Ricci, P. Dolog, G. J. Houben (eds) User Modeling, Adaptation, and Personalization. UMAP 2014. Lecture Notes in Computer Science, vol 8538. Springer, Cham, 2014.

[12] S. Jain, A. Grover, P. S. Thakur, S. K. Choudhary, “Trends, problems and solutions of recommender system”. In: International Conference on Computing, Communication & Automation, 2015.

[13] S. M. Mahdi, K. Nobuko, J. A. Bowllan, A. O. Smith, “A Review on Recommendation Systems: Context-aware to Social-based”. [Online] Available: https://arxiv.org/pdf/1811.11866.pdf [Accessed May 29, 2019]

[14] K. Shah, A. Salunke, S. Dongare, K. Antala, “Recommender systems: an overview of different approaches to recommendations”. In: 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2017.

[15] D. González, R. Motz, L. Tansini, “Recommendations Given from Socially-Connected People”. In: Y. T. Demey, H. Panetto (eds) On the Move to Meaningful Internet Systems: OTM 2013 Workshops, OTM 2013, Lecture Notes in Computer Science, vol 8186. Springer, Berlin, Heidelberg, 2013.

[16] B. Yapriady, A. L. Uitdenbogerd, “Combining Demographic Data with Collaborative Filtering for Automatic Music Recommendation”. In: R. Khosla, R. J. Howlett, L. C. Jain (eds) Knowledge-Based Intelligent Information and Engineering Systems, KES 2005, Lecture Notes in Computer Science, vol. 3684, Springer, Berlin, Heidelberg, 2005.

[17] R. Burke, “Hybrid Recommender Systems: Survey and Experiments”. In: User Modeling and User-Adapted Interaction, vol. 12, pp. 331-370, 2002.

[18] D. Geiger, M. Schader, “Personalized Task Recommendation in Crowdsourcing Information Systems - Current State of the Art”. In: Decision Support Systems, vol. 65, pp. 3-16, 2014.

[19] Wikipedia. [Online] Available: https://www.wikipedia.org/ [Accessed May 29, 2019]

[20] J. Surowiecki, “The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations”, Doubleday, 2004.

[21] M. S. Bernstein, D. Tan, G. Smith, M. Czerwinski, E. Horvitz, “Personalization via Friendsourcing,” In: ACM Transactions on Computer-Human Interaction (TOCHI), vol. 17, no. 2, 2010.

[22] C. Manning, P. Raghavan, H. Schütze, “Introduction to Information Retrieval”, Cambridge University Press, 2008.

[23] M. Lerato, O. A. Esan, A. Ebunoluwa, N. SM, T. Zuva, “A Survey of Recommender System Feedback Techniques, Comparison and Evaluation Metrics”. In: 2015 International Conference on Computing, Communication and Security (ICCCS), 2015.

[24] Plan Ceibal. [Online] Available: https://www.ceibal.edu.uy [Accessed May 29, 2019]

[25] Campus Virtual. [Online] Available: https://cv.utu.edu.uy/moodle/ [Accessed May 29, 2019]

[26] EVA - Facultad de Ingeniería. [Online] Available: https://eva.fing.edu.uy [Accessed May 29, 2019]

[27] Blocks - MoodleDocs. [Online] Available: https://docs.moodle.org/36/en/Blocks [Accessed May 29, 2019]

[28] Accreditation Council for Continuing Medical Education (ACCME), “CME Content: Definition and Examples”. [Online] Available: https://www.accme.org/accreditation-rules/policies/cme-content-definition-and-examples [Accessed May 29, 2019]

[29] IMPO - Ley N° 18331 - Ley de Protección de Datos Personales. [Online] Available: https://www.impo.com.uy/bases/leyes/18331-2008 [Accessed May 29, 2019]

[30] B. M. Sarwar, G. Karypis, J. A. Konstan, J. Riedl, “Item-Based Collaborative Filtering Recommendation Algorithms”. In: Proceedings of the 10th International World Wide Web Conference, WWW 10, Hong Kong, China, pp. 285–295, ACM, 2001.

[31] K. Miyahara, M. J. Pazzani, “Collaborative Filtering with the Simple Bayesian Classifier”. In: Proceedings of the 6th Pacific rim International Conference on Artificial Intelligence, pp. 679–689, 2000.

[32] G. Majumder, P. Pakray, A. Gelbukh, D. Pinto, “Semantic Textual Similarity Methods, Tools, and Applications: A Survey”, Computación y Sistemas, pp. 647-665, 2016.

[33] A. Margolis, A. López-Arredondo, S. García, N. Rubido, C. Caminada, D. González, L. Tansini, “Social Learning in large Online Audiences of Health Professionals: Improving Dialogue with Automated Tools”, MedEdPublish, 2019. [Online] Available: https://www.mededpublish.org/manuscripts/2258 [Accessed May 29, 2019]

[34] N. Tintarev, J. Masthoff, “A survey of explanations in recommender systems”. In: Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop (ICDEW ’07), IEEE Computer Society, pp. 801-810, 2007.