Educational Resource Recommender Systems Using Python and Moodle

Alicia Andrade Vera(✉) and Carlos Mendoza González

Universidad Estatal Península de Santa Elena, La Libertad, Ecuador
aandrade@upse.edu.ec

Abstract. This paper presents the implementation in Phyton of an algorithm for recommending educational resources based on the user's needs, where these resources will be obtained through the use of the YouTube API that suggests videos of educational materials focused on the student's level of knowledge to customize the recommendations, taking into consideration data obtained from a knowledge survey and the academic grades obtained from different resources made from the Moodle learning platform. The emphasis is on building an RS with the capacity to assist in educational settings. The RSs applied in education allow students to find materials that fit their needs and preferences, and the recommended materials are adapted to the pedagogical objectives of the teachers. It should be noted that the implementation of the project was carried out in a controlled environment, with the interaction of a group of students of a particular subject and semester.

Keywords: Recommendation system · Collaborative filtering · Moodle · Python

1 Introduction

The use of recommender systems as information retrieval techniques and strategies aims to solve the problem of data overload. They help filter the information available on the web and find the information that is most interesting and valuable to users, allowing them to discover new content more quickly and efficiently.

Recommender systems are used in various applications, such as recommending books, music, movies or news. The task of RS is to automatically select the most appropriate items based on each user's personal interests and preferences. Typically, RS focuses on a specific type of item to recommend, such as in travel e-commerce or an educational repository, and generates personalized recommendations that provide useful and effective recommendations [1].

Nowadays, the adoption of personalized learning approaches, especially RS, is justified due to the high demand for interpretation of data stored in educational institutions and in various virtual environments or services. Almost all student behaviors and actions are stored on the servers of educational institutions [2].

In addition, learning activities take place in a virtual environment that includes a variety of tools and systems. For example, virtual teaching and learning environments (VLEEs) provide access to learning resources, but do not guarantee that a teacher or

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13380, pp. 15–30, 2022.
https://doi.org/10.1007/978-3-031-10542-5_2

learner will use them in the course. Students often use additional tools to collaborate or find resources, as if the educational materials presented in a virtual environment are insufficient. Learning situations are also becoming increasingly complex due to the different pedagogical approaches between formal and informal learning, which have different requirements for the learning environment and, as such, for reinforcement in learning [3].

In terms of the recommendation techniques used, RSs are generally based on collaborative filtering, content-based filtering, knowledge-based filtering or link recommendation algorithms. These algorithms use the user's information and resources to generate recommendations.

In general, recommender systems are evaluated using three types of tests driven by protocol evaluation in areas such as information retrieval and machine learning [4].

The aim of this work is to search for resources with the help of the YouTube API, and suggest those that best suit not only the search, but also the educational needs of the student, so the first part of this work focuses on the recommendation of videos based on a survey of knowledge that the student has previously assessed. In addition, we work with the results obtained from the lessons, workshops and practices carried out on the MOODLE platform.

2 Theoretical Background

This section presents the theoretical and technological bases on which the proposed recommender system is based. First, the concept of the Moodle learning platform, the environment on which this work is focused, is described. Secondly, the types of recommender systems are presented.

2.1 E-Learning Platforms

The e-learning platform is an online learning management system (LMS) that allows you to manage, deliver, monitor, evaluate and support various pre-planned and designed activities in the process of fully virtual training (e-learning) [5].

The use of LMS platforms offers many advantages to support direct instruction, leading to better results than through traditional teaching methods.

The correct use of the LMS allows access to information and communication, expands learning strategies and offers the possibility of improving certain cognitive skills that depend directly on the specific stimuli of each person and the creative techniques used in learning. In the publication and creation of content.

Of all the information platforms that exist in the market, this paper focuses on Moodle because it is the one used in this work.

The Moodle distance learning platform is an LMS, it allows you to keep track of the content and the different users that interact within it. In this case, Moodle has most of the tools to communicate and monitor the activities of the participants [6].

Moodle facilitates the teaching and learning process, teamwork activities and collaboration in projects, and because of the interaction it provides it promotes the existence of three models: (teaching model in transmitting knowledge, teaching model in acquiring,

compiling and accumulating knowledge and the teaching model for developing, inventing and creating knowledge), and the five different types of content management systems with educational value (pure CMS system, weblog content management systems, CMS systems oriented to collaboration, community and collaborative content management systems, and wiki systems) [7].

2.2 Recommender Systems

A recommender system is defined as a tool, platform or software module for the purpose of providing recommendations that are useful to users [8].

Most experts are thinking of four architectures of recommender systems: content-based systems, knowledge-based systems, collaborative filtering systems, and hybrid systems. All four architectures are often used, and the data mappings refer to user choices for specific factors. Each quantity within the matrix shows a pair of user agents, which serves this information on multiple supports to create recommendations [9].

Of all the recommender system architectures, this paper will focus on collaborative filtering because it is the one used in this work.

Collaborative filtering (CF) systems their make their recommendations based on the relationships between users and objects. RS recommends an active user, and items that it wants other users to have the same options. The similarity between users is calculated, taking into account the similarity between their levels [10]. Within CF, nearest neighbor methods have high popularity, efficiency and capabilities to create accurate and personal recommendations. In this approach: (1) the utility matrix is completed for different users and (2) the similarity between users is determined between users from the Jaccard index or the degree of cosine [11].

Memory-based collaborative filtering algorithms, or nearest-neighbor algorithms they use a comprehensive database of items and users to create forecasts [12]. First, they used statistical techniques to find neighbors, using items with similar rating histories to existing users [13]. Once a list of neighbors is created, their choices are combined to create a list containing the N most recommended items for the current user. Among its limitations is the need to have a minimum number of users with a minimum number of predictions for each person, including which users the recommendation is targeted [14].

In addition to the above systems, other authors consider other recommender systems, such as those based on demographics or utility, to be particularly relevant. These systems are actually subcategories of the aforementioned recommender system architecture [15].

3 Python Recommender System

This section describes the Phyton implementation of a user-based collaborative filtering recommendation system that connects to the YouTube API, whose main objective is to suggest educational resources to users based on the score that will be a discrete numerical value between a maximum and minimum that each student placed in a survey generated at the end of the academic cycle on the knowledge they consider to have in the different topics covered during the academic period, using the nearest neighbor

algorithm. In addition, based on the lowest overall average grades of the lessons, practices and workshops carried out within the Moodle platform, videos will be suggested.

Because the proposed RS focuses on fostering learning, it must have access to: (1) student information, (2) lesson information, (3) workshop and practice information, and (4) knowledge survey information.

3.1 Development Environment

Open-source development tools were used for this work. First, the Visual Studio Code source code editor was installed, where the repository containing the Python files with the algorithm was created. Secondly, the Python programming language was installed, which was used due to the number of libraries that the developer community has created, among them: NumPy, which was used to work with the data in a multidimensional way; Pandas was used to read and manage the dataset, creating the DataFrame with integrated indexing; Strealim, which allowed converting the data scripts into web applications and due to the great flexibility of this language to integrate with the YouTube API. In addition, a Python-mysql-connector module was installed, used in the Windows operating system; in order to install the module, it is necessary to have Python and PIP pre-installed in the system.

3.2 API YouTube

The YouTube website plays a relevant role, since the information of the educational elements is obtained through its API. To make the recommendation, the collaborative filtering technique is used, implementing an algorithm that recommends "elements" to the user based on the grades obtained according to the knowledge survey executed. Additionally, using the most popular recommendation technique, the student will be able to have recommendations based on the grades obtained in lessons, workshops and practices.

To make use of the resources of the YouTube platform using other software such as Python, it is necessary to make use of the YouTube resources Data API which will allow us to access the resources and related information of the entire platform such as channels, playlists, comments, subscriptions, thumbnails, categories and videos.

An API key is required to make use of the Data API, so it is necessary to have a Google account to access the Google API Console where the API key is requested, and the application that will make use of the resources is registered through requests.

The main resources of the YouTube API are: activity, channel, channeBanner, guideCategory, playlist, playlistItem, search result, subscription, thumbnail, video and videoCategory. Of these the RS shows only videos of the topics that the algorithm determines that the student should reinforce, through the following request: `GET` https://www.googleapis.com/youtube/v3/search, this request requires a parameter "q" of type string that specifies the query term to be searched, the same that will be determined by the RS.

3.3 Moodle Virtual Platform

This system is based on the results obtained in a knowledge survey, considering student preferences, the recommendations in this case are made taking into account the three

most similar profiles based on student responses and thus recommending different topics seen throughout the academic period [16]. Additionally, recommendations are also made considering the subjects with the lowest overall average and lowest individual grades.

This system is mainly aimed at students, obtains automatic feedback information and can be applied to any domain, as it is web-based and recommends resources. The Moodle recommendation system was implemented in a sixteen-week course.

The knowledge survey is elaborated with twenty-five (25) questions in relation to the topics contained in the syllabus of the Fundamentals of Redes course and the following Likert scale was used to rate: (5) Very well, (4) well, (3) Fair, (2) Little, (1) Not at all.

The Moodle gradebook configuration is composed of two parent categories: cycle I and cycle II that contain the subcategories assessment strategies and final exam. The evaluative strategies subcategory is composed of the components teaching, practice and experimentation and autonomous work as shown in Table 1.

Table 1. Moodle grade chart.

<table>
<tr><td rowspan="10">Assessment strategies</td><td rowspan="3">Teaching component</td><td>Lesson 1</td></tr>
<tr><td>Lesson 2</td></tr>
<tr><td>Lesson n</td></tr>
<tr><td colspan="2">Total teaching component</td></tr>
<tr><td rowspan="3">Practice and experimentation component</td><td>Practice 1</td></tr>
<tr><td>Practice 2</td></tr>
<tr><td>Practice n</td></tr>
<tr><td colspan="2">Total practice and experimentation component</td></tr>
<tr><td rowspan="3">Autonomous work component</td><td>Task 1</td></tr>
<tr><td>Task 2</td></tr>
<tr><td>Task n</td></tr>
<tr><td colspan="2">Total autonomous work component</td></tr>
<tr><td colspan="3">Total assessment strategies</td></tr>
<tr><td colspan="3">Final exam</td></tr>
<tr><td colspan="3">Total course = Assessment strategies + Final exam</td></tr>
</table>

3.4 Connecting to the Moodle Database

To connect the MySQL database with Python is done in two ways:

1. The MySQL Connector Python module is used to execute the queries and the following steps were performed.

   a. Using the pip command to install the MySQL Python connector.
   ```
   pip install mysql-connector-python
   ```
   b. Import using a statement so that you can use the methods of this module to communicate with the MySQL database.
   ```
   import mysql.connector
   ```

c. Use the connect () method with the necessary arguments to connect to MySQL.

```
db = mysql.connect(
host="localhost",
user="root",
passwd="",
database="Moodle
)
```

2. To convert a table in MySQL to Pandas DataFrame the connection is as follows.

a. Connection to the database.

```
db_connection_str = "mysql+pymysql://"+UserName+ ":"
+Password +"@localhost/"+ DatabaseName
```

3.5 Algorithm Creation and Implementation of the Recommender System

The recommendation system will show information to students and teachers of a particular course regarding general topics such as: (1) the most failed questions in each of the tests (lessons and exams), (2) the in-class workshop with the lowest overall average, (3) the practice performed with the lowest overall average, and specifically (4) information on a knowledge survey of topics covered throughout the semester and the scale of difficulty it represented for each student [17].

For the first three cases the mechanism of the most popular without using collaborative filtering will be used, this is a basic way without using machine learning to offer recommendations in more general and not personalized aspects being the same for all users, which is accurate for these cases where it is expected that both the teacher and students are shown this general information about the course activities, despite this for the first case on the questions that most students failed in the tests; students will be shown information only of the tests they took, and as for the teacher if you can see all existing tests.

The last recommendation topic on the knowledge evaluation survey, being of a personalized nature, will make use of the user-based collaborative filtering technique with the nearest neighbor algorithm, so this section will cover the detail of the implementation of this resource to be recommended [18].

The nearest neighbor method essentially consists of determining similar users within a set through a quantifiable element that relates them on a topic or preference. In this case there is a set of students where the preferences of other similar students will be taken into account, i.e., the "neighbor" that most resembles him/her with respect to a preference, in this case the scale of values of the questions of the knowledge survey [19].

This process is defined by the following Eq. (1).

$$r_{ij} = \frac{\sum_k Similaries(u_i, u_k) r_{kj}}{numberofratings} \tag{1}$$

There are two ways to calculate preferences: (1) user-based collaborative filtering and (2) item-based collaborative filtering.

[20]The recommendation is to show a student the topics that are most difficult for the general population to understand considering the weaknesses of users whose scores are similar to the first student. Considering these aspects, the following resources will be available:

– Matrix of Survey Items (Items Feedback): Preferences.
– Matrix of ratings by user and topic (Values Item Feedback): Set of elements that represent the preferences of users regarding topics.

From these resources the goal is to calculate a student's preferences through the preferences of similar students or "near neighbors" on the topics he or she may be interested in reinforcing. This process will be detailed as a series of steps for better explanation:

1. Creation of the resource sets (Matrixes of survey topics and ratings by user and topic).
2. Find similarity among students.
3. Develop a recommendation mechanism.

Creation of Resource Sets. The necessary matrixes are created by querying the Moodle database hosted in MYSQL through sqlalchemy to be converted into a DataFrame for better handling, as follows:

```
db_connection = create_engine(db_connection_str)
```

Survey topic matrix.

```
df_items_feedback = pd.read_sql(listFeedbackItems(),
con=db_connection_str)
```

Matrix of ratings by user and topic.

```
df_values_items_feedback = pd.read_sql(listItemsVal-
uesByUser(), con=db_connection_str)
```

This process must be carried out with due analysis since the information provided to the recommender system is essential if reliable information is to be obtained. That is why at this point it is necessary to limit the margin of error of the records, i.e., redundant or unnecessary data. In this case the data shown are filtered through the queries directly and not at the DataFrame level with the panda library.

The data obtained in the subject matrix of the survey show a total of 25 subjects and about the grades per user and subject there are a total of 975 records regarding 39 students with 25 grades each.

Find Similarities Among Students. For this purpose, a function for calculating nearest neighbors must be created. Similarity can be calculated in two ways: Pearson correlation and cosine similarity [21]. In this case, the second option has been used to find three students with similar grades.

Cosine similarity Eq. (2).

$$\cos(x_1, x_2) = \frac{\vec{x}_1 . \vec{x}_2}{[[\vec{x}_1 \Delta \vec{x}_2]]} \tag{2}$$

The function will receive three parameters the student's unique user identifier (user_id), the pivot matrix of grades per user and subject, and the number of similar users required. The first parameter `user_id` will be entered through the web interface after typing the user's name and retrieved by a database query. The second parameter matrix will be obtained from the matrix ratings per user and topic using the `pivot_table` statement in Python. Finally, the third parameter is an integer previously defined in the programming of the interface, which will represent a parameter "k" that will vary depending on what is required.

Function for generation of similar users:

```
def generateSimilarUsers(user_id, matrix, k=3):
    user = matrix[matrix.index == user_id]
    other_users = matrix[matrix.index != user_id]
    similarities = co-
sine_similarity(user,other_users)[0].tolist()
    indices = other_users.index.tolist()
    index_similarity = dict(zip(indices, similarities))
    index_similarity_sorted = sort-
ed(index_similarity.items(), key=operator.itemgetter(1))
    index_similarity_sorted.reverse()
    top_users_similarities = index_similarity_sorted[:k]
    users = [u[0] for u in top_users_similarities]
    return users
```

Second function parameter:

```
grade_matrix = df_values_items_feedback.pivot_table(in-
dex='user_id', columns='item_feedback_id', val-
ues='grade')
```

The function will first create a DataFrame with the current user that has been provided, and then another one with the remaining users. Using the "cosine_similarity" function imported from the sklearn module, the similarity between students is calculated, and then a list is created with the indexes of these students, then key/value pairs are created for each user index and its similarity, where they are ordered to finally show the number of similar neighbors according to the number that has been defined.

Import skleam module.

```
from sklearn.metrics.pairwise import cosine_similarity
```

The Python sklearn module that integrates classical machine learning algorithms into the tightly coupled world of Python science packages. Its goal is to provide simple and efficient solutions to learning problems that are accessible to all and reusable in various contexts: machine learning as a versatile tool for science and engineering.

Develop Recommendation Mechanism. The recommendations listed will also be provided by a function that receives 4 parameters: the unique identifier of the student's user (user_id), the matrix of similar users obtained previously, again the pivot matrix of the grades per user and subject, and the number of recommendations to display. The first parameter `User_id` is obtained in the same way as the previous function. The second parameter matrix of similar users will be obtained by the function `GenerateSimilarUsers` and, finally, the fourth parameter is an integer previously defined in a global variable.

Function to generate personalized recommendations:

```
def recommend_item(user_index, similar_user_indices, matrix, items=max_item_recomended):
    similar_users = matrix[matrix.index.isin(similar_user_indices)]
    similar_users = similar_users.mean(axis=0)
    similar_users_df = pd.DataFrame(similar_users, columns=['mean'])
    similar_users_df.head()
    user_df = matrix[matrix.index == user_index]
    user_df_transposed = user_df.transpose()
    user_df_transposed.columns = ['grade']
    user_df_transposed = user_df_transposed[user_df_transposed['grade']==0]
    animes_unseen = user_df_transposed.index.tolist()
    similar_users_df_filtered = similar_users_df[similar_users_df.index.isin(animes_unseen)]
    similar_users_df_ordered = similar_users_df.sort_values(by=['mean'], ascending=True)
    top_n_item_feedback = similar_users_df_ordered.head(items)
    top_n_items_feedback_indices = top_n_item_feedback.index.tolist()
    items_feedback_information = df_items_feedback[df_items_feedback['item_feedback_id'].isin(top_n_items_feedback_indices)]
    return items_feedback_information
```

The function will initially create a matrix of the grades per topic only considering the list of similar users, then it calculates the average grade of all the questions on the matrix, which is converted into a DataFrame for better use, it creates a vector with the

matrix of grades only with the current user entered, it reduces redundant data if any. After that, a list of the topics that are difficult for the current student is created, and the average grades of similar users are filtered, the DataFrame is sorted to list the items with lower scores that represent the recommendations.

After obtaining the personalized recommendations, the suggestions are supplemented by displaying a number of YouTube videos. This process is best shown in the graphical interface to be used by students and teachers, which is described below.

4 Results

The generated recommendations are displayed in a graphical interface through a web application. There are several ways to interpret the native Pyhton code in the browser, in this particular case the use of Streamlit has been chosen, which is an extremely pow-erful tool that provides the user with an extensive set of text components, data visualization, and data visualization, graphics, multimedia and many others, thus facilitating the creation and dynamism of web applications. In this particular case the following was used:

Streamlit library import:

```
import streamlit as st
```

- st.title: title creation; st.image: image display.;st.text_input: data input; st.header y st.subheader: for texts in header and subheader formats; st.table: Display a static table; st.write: display any text string; st.checkbox: checkbox usage; st.error: display error messages; st.warning: display warning messages; st.success: display a success message.

In terms of interface construction, this is done in a very similar way to traditional web applications, only that instead of using hmtl elements in a file of the same extension, the view is coded directly in the Python file with.py extension making use of the Streamlit library, and using the elements that it provides and are previously detailed. The structure of the extension then results in a combination of graphical elements of Streamlit calling functions implemented through Python code such as the nearest neighbor algorithm and many others, and which in turn makes use of the necessary information extracted from the database queries. These resources and their interaction will result in a list of recommendations suggesting content from the YouTube platform. The structure of the application can best be described by the following graph presented in Fig. 1.

4.1 Display of Recommendations

In general terms, the recommendation system will show the information in a web application, which can be accessed from a link published in the Moodle virtual environment. The application will ask for the user's name as a parameter as a validation mechanism. Once the user is verified, four types of recommendations will be displayed; three general

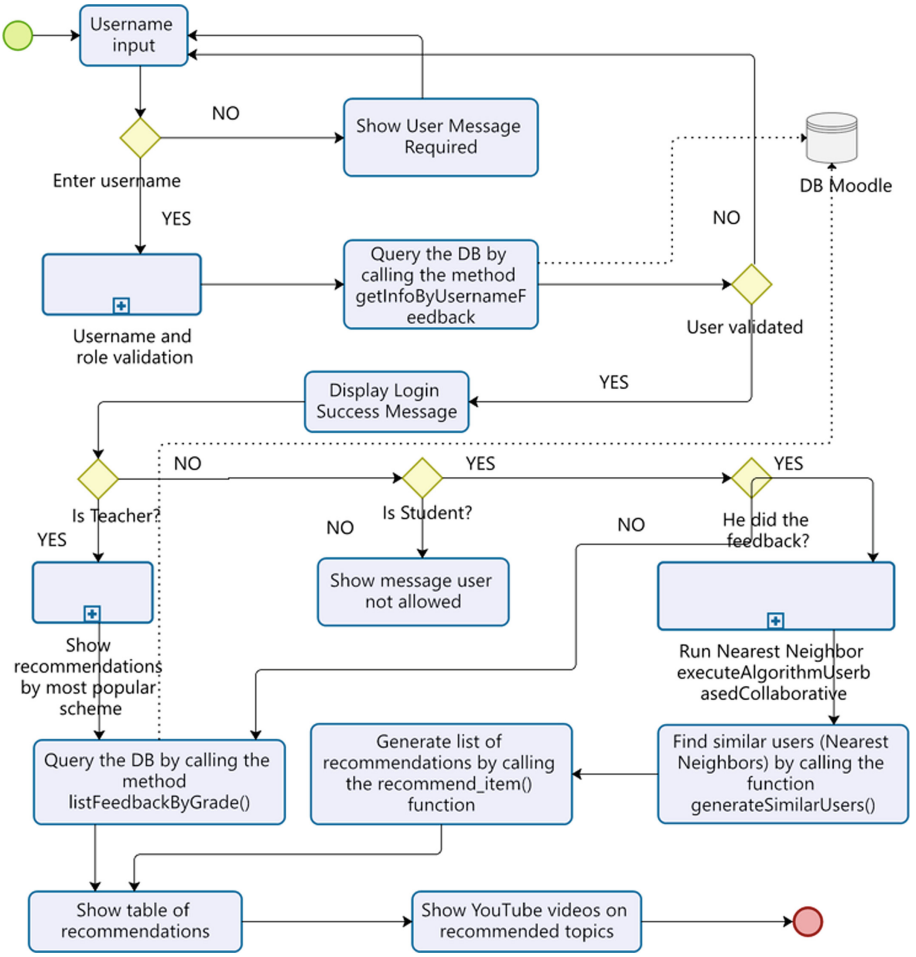

Fig. 1. Process diagram of the knowledge survey recommendation process.

ones for any user using the most popular scheme without collaborative filtering and one specific recommendation using the nearest neighbor algorithm previously developed and explained.

The information that will be displayed to the student and teacher user of a course is: (1) the most failed questions in each of the tests (lessons and exams), (2) the workshop in class with the lowest overall average, (3) the practice performed with the lowest overall average, (4) recommendations on topics covered throughout the semester through a knowledge survey, to evaluate the difficulty that represented for each student a particular topic.

The results obtained from all types of recommendations will be complemented by showing a certain number of YouTube videos regarding the suggestions obtained so that students have support material to reinforce knowledge in those topics, and in the case of the teacher to implement new teaching strategies in those areas where students have difficulties. The web application once the link is accessed from the virtual classroom is dis-played as shown in Fig. 2.



Fig. 2. Main recommendation interface.

To access the first three recommendations, it will be necessary to enter the user's name and select the desired item to display the respective recommendations, these are general, but despite this only the information related to the student will be displayed. For example, in the first case of the most failed items, only the information of the evaluations that the student has taken will be shown, as shown in Fig. 3.

Figure 4 show the recommendations according to the three workshops and three practices with the lowest overall average, since the same content is shown to all students when determined by the lowest average score.

Regarding the last recommendation on the knowledge survey, this will be personalized and will make use of the nearest neighbor algorithm, and in the same way as the previous cases will require the entry of the user's name.

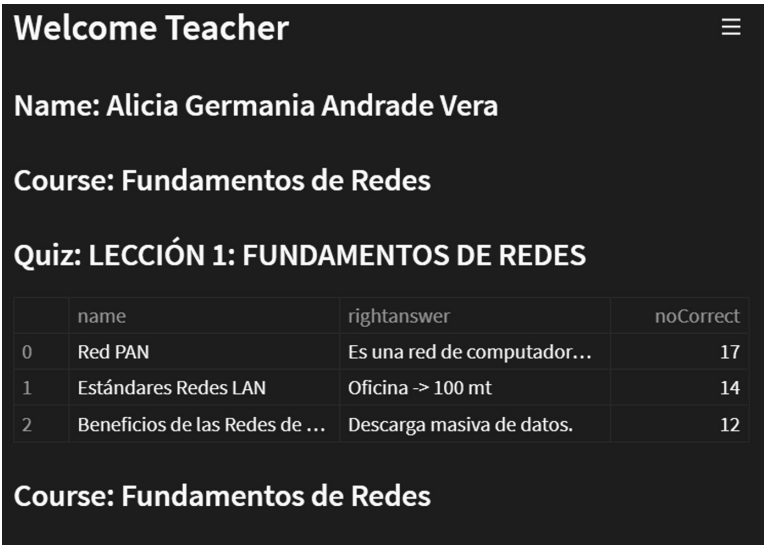

Fig. 3. Items most failed by quizzes.

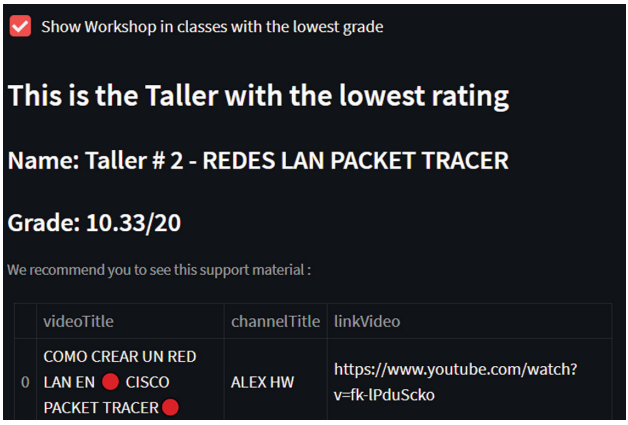

Fig. 4. Workshop in classes with the lowest grade.

Figure 5 shows the results when the student takes the survey, but what happens when the student using the recommendation system did not take the survey based on the theory, the collaborative filtering technique is based on the user's interactions with respect to a preference, and when the user does not have interactions, it is simply not possible to make use of this method.

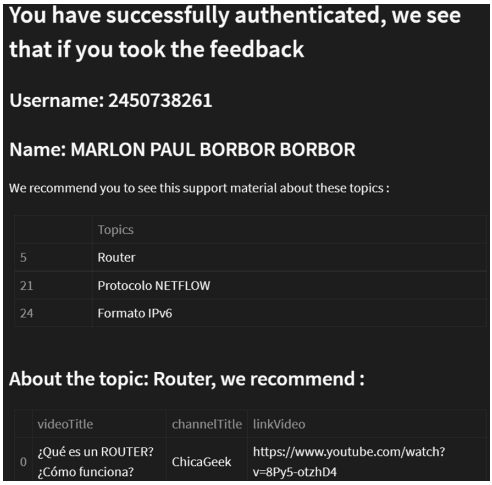

Fig. 5. Rendered the survey.

The solution to this situation is to make random recommendations to the user, but in this case the most popular mechanism will be used to show the most difficult topics for the general student population as shown in Fig. 6.

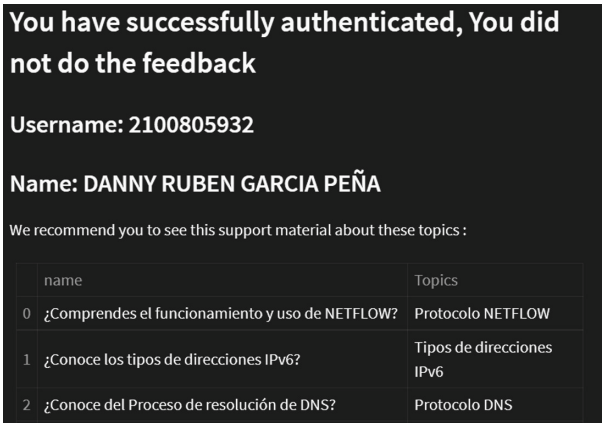

Fig. 6. Did not submit the survey.

5 Conclusions

The application of RS in education can benefit both educators and students. On the one hand, they reduce the need for teachers to search for and select educational materials; on the other hand, they improve students' decision-making process because they have access to high-quality materials and accurate, personalized information.

The application of this type of systems is extremely useful and practical by globally providing all users of a course with recommendations on specific topics by providing multimedia support material from platforms such as YouTube. In addition to this fact, all the tools used in the development of the system were open source, so it does not represent economic costs but only time investment in the development process.

Finally, it should be noted that the implementation of the project was carried out in a controlled environment, with the interaction of a group of students of a particular subject and semester. From the results obtained in the present work, it was possible to show how the Collaborative Filtering approaches offer good results. However, it is important to keep in mind that the results presented serve only as a first step towards understanding and appropriate specialization for recommendations.

This work should be complemented with experiments in a production environment, in cloud servers maximizing the scope of the work by employing it in the subjects of a semester or in all subjects of an institution, considering the independent characteristics, needs, level of knowledge, and expectations of each student who is part of the RS users, as well as their information search tasks and how to use the recommended resources in their learning activities, by using new rule patterns and extending the recommendation criteria designed to increase the type and number of recommendations.

References

1. Ricci, F., Rokach, L., Shapira, B.: Recommender systems: introduction and challenges. In: Recommender systems handbook. Springer, pp. 1–34 (2015). https://doi.org/10.1007/978-1-4899-7637-6_1
2. Durovic, G., Holenko Dlab, M., Hoic-Bozic, N.: Educational recommender systems: An overview and guidelines for further research and development. Sveucilista u Zagrebu, **20** (2018)
3. El Guabass, I., Al Achhab, M., Jellouli, I., Mohajir, B.: Context-aware recommender systems for learning. Int. J. Inf. Sci. **1**(1), 17–25 (2018)
4. Guy Shani, Asela Gunawardana,: Evaluating recommendation systems. In: Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B Kantor, (ed.) Recommender systems handbook, pp. 257–297. Springer, Boston, MA (2011). https://doi.org/10.1007/978-0-387-85820-3_8
5. Simanullang, N.H.S., Rajagukguk, J.: Learning Management System (LMS) Based On Moodle To Improve Students Learning Activity. J. Physics: Conference Series **1462**(1), 012067 (2020)
6. Harrathi, M., Touzani, N., Braham, R.: Toward a personalized recommender system for learning activities in the context of MOOCs. In: De Pietro, G., Gallo, L., Howlett, R.J., Jain, L.C. (eds.) KES-IIMSS 2017. SIST, vol. 76, pp. 575–583. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-59480-4_57
7. Gamage, S.H.P.W., Ayres, J.R., Behrend, M.B.: A systematic review on trends in using Moodle for teaching and learning. Int. J. STEM Educ. **9**(1), 1-24 (2022)
8. Rickard, P., Stiles, R.: Comprehensive adult student assessment system (casas) design for effective assessment in correctional educational programs, Journal of Correctional, **32**, 51–53 (1985)
9. Ullman, J., Leskovec, J., Rajaraman, A.: Mining of Massive Datasets (2011). http://www.mmds.org/
10. Yago Corral, H.: Dialnet (2019). https://dialnet.unirioja.es/servlet/tesis?codigo=251131. [Último acceso: 22 octubre 2021]

11. Sapountzi, A., Psannis, K.E.: Social networking data analysis tools & challenges. ScienceDirect, **86**, 893–913 (2018)
12. Cunha, T., Soares, C., De Carvalho, A.C.: Metalearning and Recommender Systems: A literature review and empirical study on the algorithm selection problem for Collaborative Filtering. ScienceDirect **423**, 128–144 (2018)
13. Tarus, J., Niu, Z., Mustafa, G.: Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. Artif Intell Rev **50**, 21–48 (2018)
14. Galán Nieto, S.M.: UC3M Research Portal. (2007). https://www.it.uc3m.es/jvillena/irc/practicas/06-07/31.pdf. [Último acceso: 23 noviembre 2021]
15. Li, H., Shi, J., Zhang, S., Yun, H.: Implementation of intelligent recommendation system for learning resources. In: 12th International Conference on Computer Science and Education (ICCSE), pp. 139–144 (2017)
16. Kulkarn, P.V., Rohini Kale, S.R.: Recommender System in eLearning: A Survey. In: Proceeding of International Conference on Computational Science and Applications , pp. 119–126 (2020)
17. Bansal, S., Baliyan, N.: A Study of Recent Recommender System Techniques. International Journal of Knowledge and Systems Science, **10**(2), 13–41 (2019)
18. Liang, D., Krishnan, R.G., Hoffman, M.D., Jebara, T.: Variational Autoencoders for Collaborative Filtering. Advancing Computing as a Science & Profession, 689–698 (2018)
19. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Comput Surv, **52** (2019)
20. Kharita, M.K., Kumar, A., Singh, P.: Item-Based Collaborative Filtering in Movie Recommendation in Real time. In: 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC). pp. 340–342. IEEE (2018)
21. Shu, J., Shen, X., Liu, H., Yi, B., Zhang, Z.: A content-based recommendation algorithm for learning resources. Multimedia Syst. **24**(2), 163–173 (2017). https://doi.org/10.1007/s00530-017-0539-8