Research Article

A System to Search and Recommend Learning Courses Sequences

Saidi Imène[*], Aymen Ali Taleb, Mahammed Nadir, Klouche Badia, Oumrani Abdelkader and Boukhobza Sofiane.

Higher School of computer science, Sidi Bel Abbes, Algeria.
**e-mail: i.saidi@esi-sba.dz*

Article History: *Received: 9 October 2022, Accepted: 17 November 2022, Published: 24 November 2022.*

Abstract: Traditional recommender systems provide the user with a list of items supposed to be of interest to the user. Each item is a single independent object and the entire result represents alternatives that match user's preferences. Another category of recommender systems provides recommendations as collections of items. For this type of systems the recommended items are not alternatives but items to be taken in a "certain order". In this work, we propose a system that recommends learning courses sequences. Another objective of our system is to enable efficient courses search by proposing an approach based on a multi-criteria weighting method. Our general purpose is to recommend sequences of learning courses using the user's profile and also to search specific courses by keywords and to suggest related courses. Our goal is to facilitate the learning process and satisfy the user's need.

Keywords: Recommender Systems, Search Systems, Sequence Recommendation, Collaborative Filtering, Process Mining, Learning Courses, Document Retrieval.

1. Introduction

Recommender Systems (RS) have become an emerging technology in several application fields such as: travel, e-commerce, entertainment, etc. Traditional recommender systems provide the user with a list of items supposed to be of interest to the user. Each item is a single independent object and the entire result represents alternatives that match user's preferences. Another category of recommender systems provides recommendations as collections of items [1]. For this type of systems the recommended items are not alternatives but items to be taken in a "certain order". Examples of this type of systems are: recommendation of sequences of activities during events (cruises, conferences, etc.) [2], [3], recommendation of points of interest in a city [4], [5], recommendation of music tracks [6], [7], or the recommendation of learning courses [8], [9], [10] which is the context of our work. The main idea of our work is to disable searching of courses in an efficient and optimal way that takes into account certain constraints (level of the student, course already acquired by the student, time, etc.) in order to allow the user (in our case: the student) to find courses and to learn them in the most optimal way. The purpose is to allow user to accomplish his learning easily and/or quickly. Indeed, the process of learning and after having read a few courses, different alternatives are available to the student but the choice of the "next course" to read is not always obvious and the concepts to be acquired before reading a course are not known to the student. A large number of works deal with the recommendation of educational resources [11], [12], [13], but few works exploit the notion of order. In addition, existing works only apply content-based [14] or collaborative filtering [15] approaches, and only few works have considered exploiting the order of how students take courses [16], [17].

The objective of this work is to emphasize the order of courses to be followed and also to present a search method that allows finding courses in an efficient way. The rest of the paper is organized as follows: In section 2 we present our system architecture and then we detail the proposed approach. In section 3, we will present some experiments. Section 4 concludes the paper.

2. A System to Search and Recommend Learning Courses

In this section, we present our approach to search and recommend learning courses. We will start by describing the architecture of our system. We will then detail the steps that the construction of our system must follow. We finish this section by the presentation of the ranking function used to weight results.

A. System Overview

In this section, we describe the main parts of our system. Figure 1 presents the global system architecture.

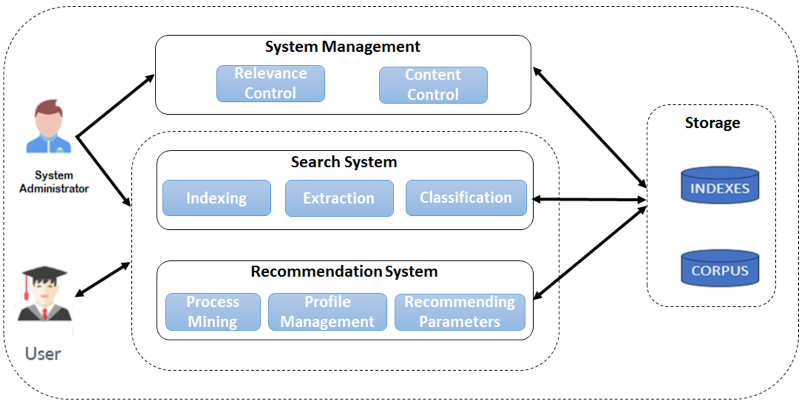

Fig. 1. Global system architecture

As shown in Figure 1, our system is made up of three main modules:

- **System management:** this module is dedicated to the hosting of the application so that it can control the relevance of the search, manage the content of the corpus and the users.
- **Search system:** this module is composed of three modules: indexing, extraction and classification.
  1) Indexing: it is the process of representing documents by keywords (tokens) and storing them in an index file which will be requested in online processing.
  2) Extraction: it is the process of extracting information from the document such as the category (the theme or the subject of the document) and the topics (an idea discussed in the document represented by a keyword). This is made by extracting semantics between the words of the document and the categories and topics predefined in the system.
  3) Classification: it is the process of assigning new documents to the collection. This process is used in case the user wants to add documents to the system.
- **Recommendation system:** this system is responsible of recommending learning courses based both on the process mining of logs of courses history and on the profile of the user. Modules of this system are:
  1) Process Mining: this module extracts dependencies between courses from logs of history courses.
  2) Profile Management: this module manages the profile of a user, its informations and background.
  3) Recommending Parameters: this module fixes different parameters to be taken into account in the recommending process.

B. Proposed method

An offline processing is executed to prepare the system for the online processing.

1) Offline Processing:

Different processes are executed as an offline processing.

- Indexing: We consider a corpus D of documents (learning courses). Our first step is to make an offline processing to prepare ad-hoc indexes for online processing. Before starting indexing, the text of the document is extracted[1]. The stemming[2] and the elimination of stop words[3] are done afterwards. Finally, indexing is done after to store the index terms in the index file.
- Classification: Another process will be executed if users decide to add new learning resources, this process will be the classification according to the category of the document. We used a machine learning model (logistic regression (LR)[4] to classify new added documents based on categories.
- Extracting dependencies of courses: logs files of courses history are used to find all dependencies of courses. These logs are history of all past successful students. We assume success is measured based on final GPA[5] of the student.

[1] Tokenization is the act of breaking up a sequence of strings into pieces such as words, keywords, phrases.
[2] Stemming is the process of reducing inflected or derived words to their word stem.
[3] Stop-words: are the most frequent words appearing generally in all texts and has little contribution to score the importance of a sentence.
[4] We obtained an accuracy of 73%, so we didn't try other classifier since classification is an optional function of our system. System administrator can restart the learning by adding new documents added to have better results.
[5] Grade Point Average: A number that indicates how high a student scored in courses on average.

2) **Online Processing:**
In the online processing the user can search a specific course using ”Search system” or explore sequences of courses using ”Recommendation system”.

- Searching system
Users can search documents by expressing their need in the form of a keyword query, the client application will therefore transmit this query to the server application to process it. The server application receives the request, transforms it into tokens and begins searching the index. The results retrieved will be ranked according to the following formula:

Score = B_score + C_score + D_score. (1)

– B_score (Bayesian Score):

Suppose that we are looking to buy a book on “information retrieval”. To have the best one, we can sort results using the ratings of other users. With a standard ranking we will get these results:

- o 1st book: 1 rating, average note: 5.
- o 2nd book: 50 ratings, average note: 4.5.

The first book have a higher note but the second book should be displayed first. Books with many ratings must be considered instead of books that only few people had rate.
To solve this problem, we propose to calculate probabilities and more precisely the Bayesian estimation (Bayesian inference)[6].

B_score = (r ∗ v + c ∗ m)/ (v + m) (2)

- o r (the note of the document): r is the average of the votes of the document. For example, if an item has no rating, its r is 0. If someone gives it 5 stars, r becomes 5. If someone else gives it 1 star, r becomes 3, the average of [1; 5].
- o v (the number of votes for a document): if 5 people gave notes, v is 5.
- o c (the average of all the notes): suppose that there are 4 documents in our corpus and that their notes are [2; 3; 5;5] c is therefore 3,75, the average of these numbers.
- o m (imaginary number of notes): The authors of the IMDB[7] site published in 2015 the formula used to rank the top 250 movies, they used the Bayesian estimator and they specified m as 250,000. So m represents the minimum number of ratings for the document to be classified in the list [18]. And since we don’t have any notes when starting the system, we initialize m with 10. This m can be modified at any time.

– C_score (Correction Score):

We score candidate documents using the Okapi BM25 weighting function. We introduce the position of the documents in our ranking formula (1). This parameter named “correction” is calculated as follows:

C_score = N – P/N (3)

- o N: the total number of documents returned.
- o P: the position of the document in the list.

– D_score (Download Score):

The number of downloads of the document also plays a role in the ranking, so we had to think of a function to give a score that represents the confidence of this number. For example, if one document is downloaded by 2 people and another document is downloaded by 50 people, it is clear that the 2nd is more likely to be relevant than the 1st, but if the 1st document is downloaded 20, 000 times and the 2nd is downloaded 30, 000 times, there is no point in differentiating between them, since both have exceeded 10, 000 downloads and it’s already a lot. We propose a function which gives for all x ∈ [0, +∞] a result close to the maximum value of note, in our case it’s 5, and this function is the natural logarithm.
The natural logarithm function perfectly answers our problem.
We have log(100000) = 5, log(50000) = 4.69, log(2) = 0.30, log(50) = 1.69.
The only problem is log(0) which is undefined, so we define our function as follows:

$$D\_score(x) = \begin{cases} 0 & if\ x = 0 \\ log(x) & if\ x > 0 \end{cases}$$

[6] Bayesian inference is a method of inference by which the probabilities of various hypothetical causes are calculated from the observation of known events. It is mainly based on Bayes’theorem.
[7] https ://www.imdb.com

- Recommendation system

The steps of recommendation are as follows (Figure 2):

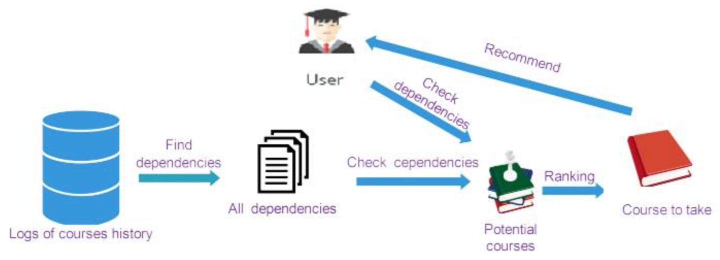

Fig. 2. Recommendation Process

The idea is to recommend the path that successful students take. So as shown in Figure 2, we exploit logs of courses history to recommend courses taken by students who are successful and similar to the student (user of our system). Steps are detailed in Algorithm 1.

Algorithm 1: Selection of best courses sequences

```
Input:
Logs C of all courses history;
Successful Students profiles S;
Student profile P;
Execute: set candidates courses sequences cc=0;
E ← Select_Students (Similarity(P, S)) ;
For e in E do
Get best courses bc from C;
Add bc to cc
End
Rank cc (by Grade of corresponding student);
Recommend cc to student;
```

Explanation of the previous Algorithm is as follows: First we find all course histories of finished students. We then select the most similar students among successful students using profiles. This similarity calculates associations between items of the profiles. A value is calculated between each of the items based on their properties. We the select best courses that similar successful students have taken. We finally rank best courses sequences using the GPA of the corresponding student.

3. Experiments

Our system is based on a 3-tier architecture. In this architecture, there is an intermediate level that is to say that we generally have an architecture shared between: the client, the web application and the database. In order to make requests to the server, the service REST[8] is generally used. REST is an architecture style that relies on the HTTP protocol.

We implemented a python prototype reflecting the processing of our system. The system can handle user queries (keyword queries). We didn't found a suitable dataset of documents and logs files to test our approach of searching and recommending learning courses. We conducted a set of experiments using data collected from students of our computer science department[9] to determine the performance of our approach. We are interested in whether the ranking of the returned results (of search and recommendation) retained good accuracy. A suitable measure to investigate the quality of results are precision and recall.

Table I summarizes some results of recall and precision for the searching task. We used a corpus that contains our university resources "computer science courses". We test different queries chosen randomly.

Query	**Precision**	**Recall**
human-machine interaction	0.54	0.76
networks	0.22	1
semantic web	0.64	0.84
internet of things	0.20	0.80
Interoperability	0.46	1
machine learning	0.75	0.54

TABLE I. Recall and Precision of Document Search

[8] Representational State Transfer
[9] Higher School of Computer Science - ESI SBA: www.esi-sba.dz

For some queries, for example, “networks” and “interoperability” we notice that the system was able to retrieve all the relevant documents (Recall = 1), but also retrieved a lot of documents that are not relevant (very low precision).Another example, for query “internet of things”, the system retrieved 80% of the relevant documents but retrieved other irrelevant documents (Precision = 20%). For other queries, (e.g. “Machine learning”, “human machine-interaction” and “semantic web”), results were acceptable since the recall was good and the system did not return many irrelevant results. We think that the low precision problem could be improved by integrating a parameter that fixes a stricter threshold on the returned results. This will be done in the future.

Table II summarizes some results of recall and precision for the recommendation task. We used a corpus that contains our university resources “computer science courses” and we test different user profiles chosen randomly.

User profile	**Precision**	**Recall**
Profile 1	0.66	0.40
Profile 2	1	0.20
Profile 3	0.55	0.15
Profile 4	0.90	0.30

TABLE II. Recall and Precision of Recommendations

For the task of recommendation, precision is high for all users profiles (more than 55%) this means that at least half of the recommended courses were relevant. For profile 2, all the recommender courses were relevant. For all profiles, the recall is lower than precision which means that some relevant recommendations were not found by our system which is acceptable since our selection of recommendation were based on similarity of students and ranking was made using GPA. A manual investigation will provide insights as to what might have caused such a low overall recall. We finally conclude that overall results are encouraging.

4. Conclusion

We proposed a course recommender system that assist students choosing suitable courses for an optimal learning. Our approach is based on process mining and users profiles to recommend courses taken by the students who are both successful and similar to the user of our system. We also proposed a method to search relevant documents, the method is based on a multi-criteria weighting function. We conducted some experiments on a dataset collected from our computer science department resources. Results proved that our approach is promising. Due to time constraints, there are still many aspects of our work that can be improved upon in the future. We will conduct several experiments to evaluate our system and we will test other recommendation approaches to find the best approach.

REFERENCES

[1] Melville, P., Sindhwani, V. (2011). Recommender Systems. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA.

[2] Diana Nurbakova. (2018). Recommendation of Activity Sequences during Distributed Events. In Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization (UMAP ’18). Association for Computing Machinery, New York, NY, USA, 261–264.

[3] Augusto Q. Macedo, Leandro B. Marinho, and Rodrygo L.T. Santos. (2015). Context-Aware Event Recommendation in Event-based Social Networks. In RecSys ’15. 123–130.

[4] Chenyi Zhang, Hongwei Liang, Ke Wang, and Jianling Sun. (2015). Personalized Trip Recommendation with POI Availability and Uncertain Traveling Time. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM ’15). 911–920.

[5] Taylor, Kendall & Lim, Kwan Hui & Chan, Jeffrey. (2018). Travel Itinerary Recommendations with Must-see Points-of-Interest. 1198-1205.

[6] Kai-Chun Hsu, Szu-Yu Chou, Yi-Hsuan Yang, and Tai-Shih Chi. (2016). Neural network based next-song recommendation. CoRR,abs/1606.07722.

[7] Binbin Hu, Chuan Shi, and Jian Liu. (2017). Playlist Recommendation Based on Reinforcement Learning, pages 172–182. Springer International Publishing, Cham.

[8] AlTwijri, Mohammed & Luna, Jos´e Mar´ıa & Herrera, Francisco& Ventura, Sebatian. (2022). Course Recommendation based on Sequences: An Evolutionary Search of Emerging Sequential Patterns. Cognitive Computation.

[9] Sankhe V, Shah J, Paranjape T, Shankarmani R. Skill Based Course Recommendation System. (2020). IEEE International Conference on Computing, Power and Communication Technologies (GUCON). p.573-6.

[10] S. Reddy, I. Labutov, and T. Joachims (2017). Learning student and content embeddings for personalized lesson sequence recommendation. In ACM Learning Scale 16, pages 93-96.

[11] Wu, W.; Wang, B.; Liu, Y.; Yin, L.; Zheng, W. (2020). Higher Education Online Courses Personalized Recommendation Algorithm Based on Score and Attributes. J. Phys. Conf. Ser., 1673, 012025.

[12] Adam, N.L.; Zulkafli, M.A.; Soh, S.C.; Kamal, N.A.M. (2017). Preliminary study on educational recommender system. In Proceedings of the 2017 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), Kuching, Malaysia, 15–17 November 2017; pp.97–101.
[13] Urdaneta-Ponte, M.C.; Mendez-Zorrilla, A.; Oleagordia-Ruiz, I. (2021). Recommendation Systems for Education: Systematic Review. Electronics, 10, 1611.
[14] R. Burke. Hybrid web recommender systems (2007). In The adaptive web, pages 377-408. Springer.
[15] Afoudi, Y., Lazaar, M., Al Achhab, M. (2019): Collaborative filtering recommender system. In: Ezziyyani, M. (ed.) AI2SD. AISC, vol. 915, pp. 332345. Springer, Cham.
[16] Ren Wang, Osmar R. Za¨ıane: (2018). Sequence-Based Approaches to Course Recommender Systems. DEXA (1): 35-50.
[17] J. Xu, T. Xing, and M. van der Schaar (2016). Personalized course sequence recommendations. IEEE Trans. Signal Process. 64(20):5340-5352.
[18] Das, Palash Ranjan and Govindasamy, Gopal. (2019). On the application of Bayesian credibility theory in movie rankings. International Journal of Operational Research, 36(2), 254-269.