Intelligent Digital Learning: Agent-Based Recommender System

Imène Brigui-Chtioui
Emlyon business school
23 Avenue Guy de Collongue
69130 Écully France
brigui-chtioui@em-lyon.com

Philippe Caillou
Université Paris Sud, LRI
INRIA, TAO Team
Bat 490, Paris
caillou@lri.fr

Elsa Negre
Paris-Dauphine University,
PSL Research University
CNRS UMR 7243, LAMSADE
Data Science Team, 75016 Paris
elsa.negre@dauphine.fr

ABSTRACT
In the context of intelligent digital learning, we propose an agent-based recommender system that aims to help learners overcome their gaps by suggesting relevant learning resources. The main idea is to provide them with appropriate support in order to make their learning experience more effective. To this end we design an agent-based cooperative system where autonomous agents are able to update recommendation data and to improve the recommender outcome on behalf of their past experiences in the learning platform.

CCS Concepts
• **Information systems→Information retrieval →Retrieval tasks and goals →Recommender systems.**

Keywords
Recommender systems; multiagent systems; digital learning.

1. INTRODUCTION
Massive open online course (MOOC), serious game, social learning, flash and mobile learning, no doubt, education and learning do not escape from the digital revolution that overwhelms our way of learning, working and thinking. Exit the endless and boring slideshows face to face with your PC screen, now everyone learns on the internet by following MOOC or playing videos posted by experts from around the world, participating in forums via collaborative networks or by practicing using serious games. E-learning exists since twenty years and his form has evolved: it moved from CD-ROMs to the Internet and fom slide shows to serious games. Henceforth we talk about digital learning, and corresponding platforms/systems (also known as Learning Management System or Learning Support System) support and manage a learning process or a learning path. Usually, the components of a digital learning system are: a community of learners; a learning platform; tutors / facilitators; text or multimedia teaching content (resources); a teaching and tutoring strategy; and validation activities.

ICMLC 2017, February 24-26, 2017, Singapore, Singapore.

DOI: http://dx.doi.org/10.1145/3055635.3056592

In this paper, we focus on platforms such as MOOC (based on Open edX [1]) and propose an agent-based recommender system of relevant learning resources for learners to help them to overcome their shortcomings/gaps. To this end, we propose a multi-agent system with a set of autonomous agents acting on behalf of their beliefs, preferences and goals. Agents belong to a cooperative system that aims to be attentive to the learning path of each learner and to ensure him a good support by providing appropriate resources in each unique situation. Agents are also able to learn using their past experiences and thus to improve their decision-making process by adjusting their strategies. This is particularly suitable in the context of recommender systems that will be able to improve recommendations over time.

This article is organized as follows: The next section presents the related works in digital learning, recommender systems and recommender systems for digital learning; Section 3 describes our proposal of agent-based recommender system for digital learning; Section 4 illustrates this proposal and Section 5 concludes this paper and presents some future works.

2. RELATED WORKS
Widely used in e-commerce sites since 1990s, recommender systems popularity increased during the last decade due to the continuous growth of e-learning systems and environments. Recommender systems require input information to properly operate and deliver content or behavior suggestions to end-users. E-learning scenarios are no exception [2]. Indeed, recommender systems are one type of filtering system to provide advice and support to users by providing information that are likely of interest to them [3]. To this end, multiple works are proposed in order to assist learners in the choice of the good training or course by means of recommender systems [4] for e-learning systems [5, 6], as well as mobile learning [7]. Other works focus on the teacher side and propose recommendation services that enable them to choose properly an engineering education e-learning system [8].

The key problem addressed by recommendation may be summarized as an estimation of scores for items that have not yet been seen by a given user [9]. The number of items and the number of system users may be very high, making it difficult for each user to view each item or for each item to be evaluated by all users. Traditionally, recommender systems are classified according to the scores, which have already been evaluated, used to estimate the missing scores [10, 11, 12]: (i) Content-based method: the user receives recommendations for items that are similar (in terms of a measure of similarity between the two items) to those which he/she has given high scores previously, (ii) Collaborative filtering method: the user receives

recommendations for items that have received high ratings from other users with similar tastes and preferences (in terms of a measure of similarity between users and items), (iii) Hybrid method: a combination of the earlier-described two methods. For example, collaborative filtering recommender systems are extremely varied and may be based on several techniques including, for example, Pearson correlation coefficient [13] or neighborhood selection (e.g. kNN [14]) for similarity between users; Cosine similarity [15] for similarity between items; Principal Component Analysis [16], matrix factorization (e.g. Singular Value Decomposition - SVD [17]) or Bayesian approaches for score prediction.

The recommender system proposed by Hsu [18] combines content-based analysis, collaborative filtering, and data mining techniques. It makes analysis of students' reading data and generates scores to assist students in the selection of relevant lessons. Lu [19] develops a framework for personalized learning recommender systems based on recommendation procedure that identifies the student's learning requirement and then use matching rules to generate recommendations of learning materials. The study refers to multicriteria decision models and fuzzy sets technology to deal with learners' preferences and requirement analysis.

Another way to find learner's profile to organize and recommend course content is to exploit semantic net [20] or content ontology [21, 22]. Concerning the technological aspect, Web service technology may constitute an operational solution for implementing personalized learning approach and for the interoperability with other e-learning personalization systems [23]. Bousbahi and Chorfia [24] propose a web-based application that provides suitable learning resources among MOOCs providers based on the learner's expressed interests. The system is similar to the generic Case-Based Reasoning problem solving system including the four steps: retrieve, reuse, adapt and retain [25]. The process starts with a problem and tries to find similar cases from the case base to suggest relevant solutions or adapt solutions to better solve the new problem and terminates by retaining the new case. In Verbert and al. [26], authors discuss the importance of contextual information that refers to the learner's environment. They construct a classification of context information in technology enhanced learning (TEL) by combining existing context definitions and adapting them to TEL. They outline 8 dimensions: computing, location, time, physical conditions, activity, resource, user, social relations. In Draschsler and al. [27], authors propose a complete classification of recommender systems supporting technology-enhanced learning (TEL RecSys). They suggest 7 exclusive clusters: (i) TEL RecSys following collaborative filtering approaches; (ii) TEL RecSys that propose improvements to collaborative filtering approaches to consider particularities of the TEL domain; (iii) TEL RecSys that consider explicitly educational constraints as a source of information for the recommendation process; (iv) TEL RecSys that explore other alternatives to collaborative filtering approaches; (v) TEL RecSys that consider contextual information within TEL scenarios to improve the recommendation process; (vi) TEL RecSys that assess the educational impact of the recommendations delivered; and (vii) TEL RecSys that focus on recommending courses.

To the best of our knowledge, there does not exist an agent-based recommender system following collaborative filtering approaches and supporting technology-enhanced learning (first cluster of the [27]'s classification). Thus, in this paper, we introduce such a system that recommends relevant learning resources to learners in order to help them overcome their shortcomings/gaps.

3. AGENT-BASED RECOMMENDER SYSTEM

The agent-based recommender system we propose is supported by several types of agents represented on Figure 1. The recommendation process is initiated by the recommender agent that has access to multiple tracking logs stored in JSON documents and containing reference information about the event data packages. Events are emitted by the server, the browser or the mobile device to capture information about interactions with the courseware and the instructor dashboard [1]. These data are not all relevant. The recommender agent is able to judge the event's relevance.

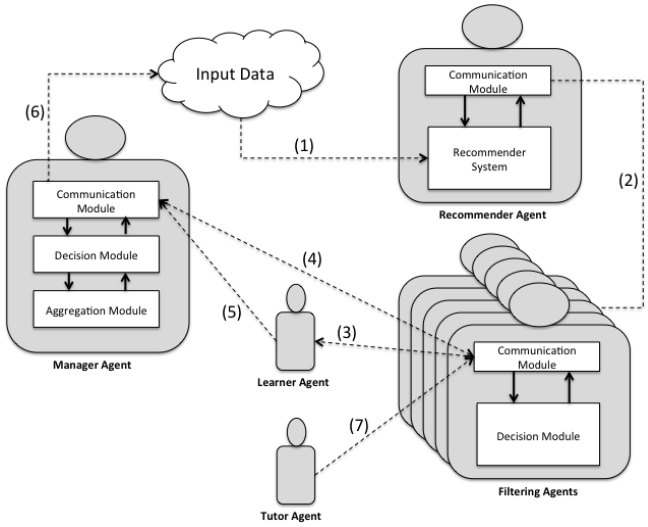

Figure 1. Agent-based recommender system.

Each agent has a communication module that is implemented to allow message exchange between agents with respect to the communication protocol. In addition, it allows message interpretation and construction. The communication protocol specifies the actions that the agents are authorized to take during the recommendation process. Table 1 presents a partial view of the primitives used by agents to communicate.

Figure 2 shows interactions between agents in order to assist the learners in their learning experience by suggesting useful resources. Based on the input data, the recommender agent provides a ranking of potentially interesting recommendations (resources) to the concerned filtering agent. Filtering agents choose the appropriate one, send it to the learner agent and communicate this decision for information to the manager agent. Each filtering agent is associated with a course and could be personalized by the concerned tutor agent which has the ability to update ranking according to his preferences. At the end of the recommendation process, the learner agent sends its feedback to evaluate the recommended resource to the manager agent that centralizes all feedbacks and consequently updates its decision rules (if necessary). New facts and updated rules are added to the input data in order to actualize it.

Table 1: Communication Protocol.

Primitive	Comment	Context
UpdateData(ma, ra, <data>)	The manager agent *ma* sends a request to the recommender agent *ra* to update data	The manager agent *ma* initiates the process by sending the input data to the recommender agent *ra*
UpdateRanking(ta, fa, <resource, ranking>)	The tutor agent *ta* orders the filtering agent *fa* to update a resource ranking	According to its local preferences, *ta* makes the decision to update a resource ranking
information(fa, ma, <info>)	The filtering agent sends back information about the recommendation to the manager agent *ma*	After choosing a given resource, fa communicates this decision for information to the manager agent *ma*
feedback(la, ma, <resource, score>)	The learner agent *la* sends a feedback to the manager agent *ma* concerning a resource	After assessing the usefulness of a resource, *la* sends back a score to the manager agent *ma*

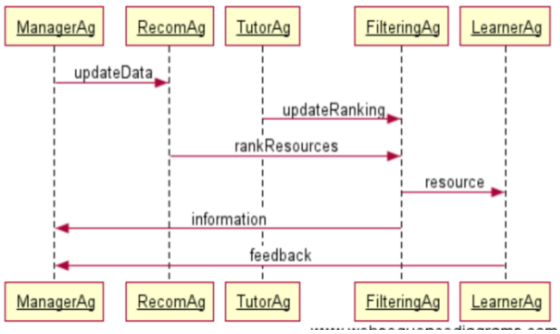

Figure 2. Recommendation process (UML).

3.1 Recommender Agent

Recommendation can be seen as an estimation of scores for non-viewed items. In more formal terms, by analogy with [9], the recommendation problem in the intelligent digital learning context can be given by the following definition.

Definition 1: Let L be the set of all learners (users), I the set of all possible learning resources that may be recommended and u a function that measures the utility of a resource i to a learner l, that is $u: L \times I \rightarrow \mathbb{R}$. Thus, for each learner $l \in L$, we wish to select the learning resource $i' \in I$ that has the maximum utility for the learner: $\forall l \in L, i'_c = argmax_{i \in I} u(l, i)$.

In recommender systems, the utility of an item (here a learning resource) is generally represented by a score, which indicates a specific user's (here a learner) appreciation of a specific item. Utilities fill a utility matrix M_u: Learners × Resources. In the multi-user intelligent digital learning context, the more appropriated approach is the collaborative filtering. Systems based on collaborative filtering produce recommendations by calculating the similarity between the preferences of different users. These systems do not attempt to analyze or understand the content of recommended items but suggest new items to the users based on the opinions of the users with similar preferences. The method consists of making automatic predictions (filtering) regarding the interests of a given user by collecting the opinions of a large number of users. The hypothesis underpinning this type of approach is that those who liked a particular item in the past tend to continue to like this specific item (or very similar items). Collaborative approaches attempt to predict the opinion a user will have about different items and to recommend the "best" item to each user in relation with their previous tastes/opinions and the opinions of other similar users [28].

In this paper, we propose the function $RecoS\ (I, L, l_c, M_u)$: S_r where I is the set of learning resources, L the set of learners, l_c the "current" learner i.e. the learner seeking for a recommended learning resource and M_u is the utility matrix. This function returns S_r, the ordered set of learning resources that can be recommended to the current learner l_c. In fact, *RecoS* is decomposed into three steps: (i) Preprocessing (matrix normalization and/or reduction); (ii) Processing (score predictions); (iii) Ranking of recommendations.

Note that a well-known drawback of collaborative filtering recommender systems is the cold-start problem for new users and new items. To overcome this problem, when the system is unable to calculate a recommendation according to the *RecoS* function, it returns a default recommendation. More formally, the *RecoS* function could be seen as the following algorithm:

***RecoS* (*I*, *L*, l_c, M_u, t_1, *preprocessing*, *predict*, *ranking*, *default*, $\prec$)**
Require: *I*: the set of learning resources
L: the set of learners
l_c: the "current" learner i.e. the learner seeking for a recommended learning resource
M_u: the utility matrix (Learners×Resources)
t_1: a threshold
preprocessing: *a preprocessing function of Mu*
predict: a recommendation (resource) prediction function
ranking: a resource ranking function
default: a function returning a default recommendation
$\prec$: a ranking of resources
Ensure: an ordered set of recommended learning resources
MP ← *preprocessing* (M_u)
SetCandResource ← *predict* (*I*, *L*, l_c, *MP*, t_1)
If *SetCandResource* = ∅
return *ranking* (*SetCandResource*, $\prec$)
else return *default* (*I*)
endIf

Due to its scalability, we propose to use an item-to-item approach using SVD (known to be easy to use) for score prediction, and default recommendations are the most "popular" and the most recent learning resource. This is particularly suitable to our context (intelligent digital learning) and corresponding data.
RecoS becomes *RecoS* (*I*, *L*, l_c, M_u, t_1, *calcAverage*, *calcSVD*, *rankResource*, *popNew*, $\leq_{resource}$).

calcAverage calculates the average ratings for each resource (item) in M_u and returns a new matrix *MP* by filling M_u ratings matrix (with the obtained average values). Note that M_u can be very sparse, thus, to capture meaningful latent relationship we start by removing sparsity using the average ratings for an item, according to [29]. The algorithm of *calcAverage* may be:

***calcAverage* (M_u)**
Require: M_u: the utility matrix (Learners×Resources)
Ensure: *MP*: a matrix
AvgResource ← ∅
For each resource i of M_u
AvgResource[i] ← AVG(l_i)
endFor
MP ← Fill each empty cell (u, i) of M_u with AvgResource[i]
return *MP*

calcSVD (*I*, *L*, l_c, *MP*, t_1) calculates the singular value decomposition of *MP* (obtained at the preprocessing step) and returns the set of non-ordered resources *SetCandResource* (subset of I) that can be recommended to the current learner *lc* and the corresponding prediction value. These candidates recommended

resources correspond to the t_1 best predictions obtained through the singular values of the SVD.

rankResource (*SetCandResource*, $\leq_{resource}$) ranks the candidate recommended resources of *SetCandResource* according to the decreasing order ($\leq_{resource}$) of the prediction values.

popNew(I) returns the set $\{s_{pop}, s_{new}\}$ where s_{pop} is the most popular resource, i.e. the one having, in average, the best rating among all the learners (set *L*) in M_u; and s_{new} is the most recent resource in the set of resources *I*.

As a synthesis, the recommender agent returns an ordered set of learning resources that can help a given learner to overcome his/her gaps.

3.2 Filtering Agent

Each filtering agent is associated with a unique course and can be managed by a tutor agent. In addition to the communication module, a decision module is implemented based on the local knowledge base that is real-time actualized on behalf of learners' feedback via the manager agent. Decision rules are established by the tutor agent in order to associate priorities to resources that may be recommended to learners. We consider like in [30] that the system deals with several types of resources: learning, support and communication resources.

Definition 2: Learning resources are texts, videos, interviews, blogs… that are generally defined before the starting of a MOOC. They are materials constituting the main structure of a course.

Definition 3: Support resources are videos, useful links, tutorials, texts... implemented to assist and to guide learners in a personalized way. These resources give details about some points of the course in order to achieve better knowledge transfer.

Definition 4: Communication resources are chat forum, message system, exchange platform that allow communication between MOOC participants (leaners, tutors...).

All these resources could be recommended as a support for learners on behalf of their gaps and difficulties.

3.3 Manager Agent

Based on its communication module, the manager agent is related to all agents evolved in the learning platform. It has a key role in the recommendation process. It centralizes crucial information at two main levels: (i) Information on decisions taken by the filtering agents; (ii) Information about the feedbacks of learner agents to assess the relevance and quality of recommended resources. This information is analyzed by the aggregation module able to make a multicriteria evaluation of each recommended resource and to transmit it to the decision module to infer rules about it and finally decide as to whether, consequently, there should be an update of the input data. As an example, a decision rule could have the following structure: *If* $(1/n \sum_{i=1..n} B((l_i, R))$

As we can see, the rule establishes that "if the feedback average of a resource *R* is greater than a value *v* and the number of learners that have assessed R is significant (greater than 5) then the score associated with the concerned resource R is updated with the value of 10".

4. ILLUSTRATION

For an easier understanding, we illustrate our model on a toy example. We first describe the data, then the basic recommendation process, the filtering agents' behaviors, and the manager agent behavior.

4.1 Illustrative Example Overview

We consider a MOOC example with 2 Java and 2 Python classes (same class level, C1…C4). Each class is followed by 100 learners (STi) and managed by one tutor, represented by his tutor agent T1…T4. As resources proposed to learners, we consider two video lessons AV1, AV2 and two exercises AE1, AE2, in both cases one for Python and one for Java. Once achieved, the learners give feedback on their activity item (score between 1 and 10). The past evaluations are summarized in the utility matrix which is provided to the recommender system.

Table 2: Initial evaluations used to train the recommender system. Grey cells represent the utility matrix M_u.

	Resource items (I)	AV1	AV2	AE1	AE2
Learner (L)	Tags	[Video] [Python]	[Video] [Java]	[Ex] [Python]	[Ex] [Java]
St1		8	7	2	
St2		2		8	
St3		2			
St4			6	8	8

The process can be described as follows:

- The recommender system is trained on the M_u matrix to build the *RecoS* function. Once trained, the system can score the activities for any new/old learner profile (see Section 4.2);
- The filtering agents manage the results obtained from the recommender system to fit their teaching preferences and present the final propositions to the learner (see Section 4.3);
- The manager agent gets the feedbacks from the learners and manages the recommender system update mechanism (see Section 4.4).

4.2 Initial Recommendation Using Collaborative Filtering

The recommender system is trained from past learners' feedback (see Section 4.4 for update illustration). The SVD dimensionality reduction is performed on the M_u matrix to build learner and item profiles in the SVD latent space. The advantage of using SVD rather than a raw similarity is that it will both generalize and remove repeated information. Both items and learners can be projected in the k-dimensions space generated by the SVD. We will consider here a simple 2-dimension SVD (dimensions D1 and D2). The learner values are given in Table 3, the items (resources) values in Table 4 and the singular values in Table 5.

Table 3. Latent space values learned from the utility matrix for learners.

	D1	D2
St1	0,483	-0,863
St2	0,512	0,412
St3	0,474	0,21
St4	0,526	0,201

Table 4. Latent space values learned from the utility matrix for resource items.

	D1	D2
AV1	0,316	-0,731
AV2	0,514	-0,119
AE1	0,482	-0,669
AE2	0,635	-0,047

Table 5. First singular values learned from the utility matrix.

	D1	D2
Sigma	25,18	6,64

When the system should provide an activity to a learner, he will compute the score of the items using these dimensions. The score can be computed for any learner l_c, even if it was not in the original matrix (you just should project the new learner in the latent space using the existing projection matrix). For example, if we consider the Learner St3 (or a profile like St3), the score will be the scalar product of the learner coordinates and the item coordinates (weighted by the singular values).

Table 6. Initial scores and rank from the recommender system for Learner St3.

	AV1	AV2	AE1	AE2
Score		5,986	6,694	7,518
Rank		3	2	1

These are the scores provided to the filtering agent.

4.3 Filtering Agents and the Class Tutors

A filtering agent represents a class managed by a tutor agent. We consider here the T3 agent (a Java class). It will apply mainly three types of filters:

Local filters, dependent on the course, the learner state and eventually the tutor, to fit the course and manage the course progression. For example, to ensure that only java items are proposed, the T3 Filtering agent will always have the filter :

$$\text{Score(I)=0 if } [Java] \notin Tags(I)$$

- If the course has several steps/chapters that should be followed in a specific order, this is where the prerequisite rules will be set (no Chapter 2 videos before a minimal score at Chapter 1)

Preference filters. The (human) tutor can personalize the filters to fit its teaching preferences/expertise. For example, the T3 tutor may consider that videos are much more useful than exercises (for whatever reason). It could also be preferences on type of exercises, on some video providers … Anything that appears in the activity description can be used to personalize the result. The preference filters are however capped biases. If a tutor wants more strict rules (like remove entirely a type of exercise he doesn't consider useful), he should send a request to the manager agent to be able to integrate it into its filter rules. We will here consider a simple video-preference rule (set very high for the sake of the example).

$$\text{Score(I)+=3 if } [Video] \in Tags(I)$$

Global filters (mainly provided by the manager agent), for example to select the activities the learners can select (see Section 4.4 below). For now, they are empty.

The result of the filters applied on our previous example to propose a new activity to St3 in the Java course T3 is:

Table 7. Final scores and ranks from the Filtering agent for Learner St3.

	AV1	AV2	AE1	AE2
Score	-	8,986	-	7,518
Rank		1		2

AV1 was already used by St3, AE1 was removed by the global filter, and AV2 was much increased because of the tutor preferences.

The learner St3 will be presented with first AV2, then AE2. Thus, the result presented to the learner follows his/her deduced preferences (by the recommender system), but also follows the class constraints and the tutor preferences.

At the end of the selected activity (say AV2), the learner gives a feedback (here 1.0 since he doesn't like videos…). This feedback is sent by the learner agent to the manager agent to update the recommender system.

4.4 Manager agent and the System Management

First, the manager agent updates the recommender system periodically (each night/week depending on the computation power, the amount of feedbacks and the biases introduced in the recommender system). For example, the new feedback of learner St3 is included in the utility matrix that will be used for the next recommender system training (it will change both the SVD space definition and the utility matrix used to compute the nearest neighbors' preferences).

Second, he can apply global rules that will be sent to the filtering agents. For example, the rule presented in Section 4.3 (change the score if enough feedback are positives). We can also apply rules for specific kind of learners (learners have access to limited type of resources depending on what they paid).

Third, he manages and controls the filtering agents. Filtering agents are mostly free to define their preference filter rules, but some have to be validated (the most restrictive ones), and they are all sent to the manager agents. The manager agent has an overview of all tutor policies and results. Rules can be set to alert in case of problems. For example, if the proportion of bad evaluations for a specific tag is too high. In our example, if too many learners put 2 or bellow to the AV2 video just because the tutor prefers videos, an alert will be raised. The manager can react by changing the maximum bias this Tutor can apply (max. bias to 1 point or 10%). The T3 tutor won't be able to impose videos to learners who don't like them even if he thinks it's best for them.

Finally, the manager agent will manage newcomers and new activities (cold start case). For platform like ours, even newcomers have some data (initial evaluations). But in a more general case, they could be completely unknown. In this case, the manager can let the recommender system recommend the most generally liked activities (by default), and/or set some newcomers filters sent to the filtering agents to impose some initial activities (sponsored activities or introduction activities especially developed to help newcomers), and/or let the tutor agent define the starting scores. The result of these initial activities will help the recommender system to learn the learner preferences. Similarly, new activities scores can be set with filter rules until some feedback allow them to be integrated into the recommender system.

5. CONCLUSION AND FUTURE WORKS

In this paper, we have presented an agent-based recommender system that aims to assist leaners in their learning process by suggesting relevant learning resources on behalf of their detected gaps or shortcomings.

To this end we have proposed a cooperative multi-agent system made up of a set of autonomous cognitive agents to recommend useful resources to learners. Our recommender agent algorithm is based on an item-to-item approach using SVD for score prediction. To illustrate our recommendation process, we have proposed a toy example.

We argue that more research must be done in three main directions: (i) validating our proposition: we plan to compare our multiagent approach to other methods using real data, (ii)

conducting simulations considering different learners' behavior to better adapt recommendations to different learning styles, (iii) generalizing our results to other types of learning platforms.

6. REFERENCES

[1] edX Research Guide, http://edx.readthedocs.org/en/latest/, 2015

[2] Corbi, A., and Burgos, D., "Review of Current Student-Monitoring Techniques used in eLearning-Focused recommender Systems and Learning analytics. The Experience API & LIME model Case Study", Int. *Journal of Interactive Multimedia and Artificial Intelligence*, 2(7), 2014

[3] Burke, R. "Knowledge-based Recommender Systems", In: *A. Kent, Encyclopedia of Library and Information Systems*, 69(32), 2000.

[4] Farzan, R., and Brusilovsky, P., "Social navigation support in a course recommendation system. In. " *Adaptive Hypermedia and Adaptive Web-Based Systems*", Vol.4018 of the series Lecture Notes in Computer Science, pp 91-100, 2006

[5] Guzman Apaza, R. V., Vera Cervantes, E., Cruz Quispe, L., and Ochoa Luna, J., "*Online Courses Recommendation based on LDA*", SIMBIG 2014

[6] Onah, D.F.O and Sinclair, J.E., "Collaborative filtering recommendation system : a framework in massive open online courses", *9th International Technology, Education and Development Conference*, pp. 1249-1257, Madrid, Spain, 2015

[7] Andronico, A., Carbonaro, A., Casadei, G., Colazzo, L., Molinari, A. and Ronchetti, M., " Integrating a multi-agent recommendation system into a Mobile Learning Management System", *Proceedings of Artificial intelligence in Mobile System,* Seattle, USA, 2003.

[8] Sommer, T., Bach, U., Richert, A. and Jeschke, S., "*A Web-Based Recommendation System for Engineering Education E-Learning Solutions*", 9th International Conference on e-Learning, Valparaiso, 2014

[9] Adomavicius, G. and Tuzhilin, A., "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions", *IEEE Transactions on Knowledge and Data Engineering* 17(6), 734-749, 2005.

[10] Balabanovic, M. and Shoham, Y., "Fab: content-based, collaborative recommendation". *Communications of the ACM*, 40(3), 66-72, 1997.

[11] Hill, W., Stead, L., Rosenstein, M. and Furnas, G., "Recommending and evaluating choices in a virtual community of use", *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '95), Irvin R. Katz, Robert Mack, Linn Marks, Mary Beth Rosson, and Jakob Nielsen (Eds.). ACM Press/Addison-Wesley Publishing Co., NY, USA, 194-201, 1995.

[12] Kazienko, P., Kolodziejski P., *Personalized integration of recommendation methods for e-commerce*, IJCSA, vol.3, no.3, pp.12-26, 2006.

[13] Rodgers J.L., Nicewander A.W., "Thirteen ways to look at the correlation coefficient", *The American Statistician*, vol.42, no.1, pp. 59-66, 1988

[14] Adeniyi, D.A. Wei, Z. Yongquan, Y., "*Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method, Applied Computing and Informatics*", Volume 12, Issue 1, pp. 90-108, 2016

[15] Qamar, A.M.,. "*Generalized Cosine and Similarity Metrics: A Supervised Learning Approach based on Nearest Neighbors*.", Thesis. Computer Science. Université de Grenoble, 2010.

[16] Pearson, K., "'*On Lines and Planes of Closest Fit to Systems of Points in Space*", Philosophical Mag. 2 , 559-572, 1901

[17] Golub, G. and Kahan, W., « Calculating the Singular Values and Pseudo-Inverse of a Matrix», *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, 1965, Vol. 2, No. 2, pp. 205-224, 1965

[18] Hsu, M.H. "A personalized English learning recommender system for ESL students", *Expert Systems with Applications*, Vol. 34, pp.683–688, 2008

[19] Lu, J., "A personalized e-learning material recommender system," In *Proceedings of International Conferene On International Conference Information Technology for Application*, 2004, pp. 374-379, Sydney.

[20] Senthil kumaran, V. and Sankar, A., "Recommendation System for Adaptive E-learning using Semantic Net", *International Journal of Computer Applications*, 63(7), 2013.

[21] Shen, L. and Shen, R., "Ontology-based Learning Content Recommendation", *International Journal of Engineering Education and Lifelong Learning*, 15(3-6), 2005.

[22] Sunil, L., Saini, D. K., "Design of a Recommender System for Web Based Learning", *Proceedings of the World Congress on Engineering*, Vol I, 2013, London, U.K.

[23] Essalmi, F., Jemni Ben Ayed, L., Jemni, M., Kinshuk, M. Graf, S., "*A fully personalization strategy of E-learning scenarios*", Computers in Human Behavior, 26, pp. 581–591, 2010.

[24] Bousbahia, F. and Chorfia, H., " MOOC-Rec: A Case Based Recommender System for MOOCs", *Procedia - Social and Behavioral Sciences 195* (2015), pp 1813-1822, 2015.

[25] Wess S, Althoff K-D, Derwand G. Using kd-trees to improve the retrieval step in case based reasoning, In: Wess Zemmouri, A., Benslimane, M. and khaldi, M., "*Hybrid approach into the design of a user model for adaptive recommendation system in MOOCs*".

[26] Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., Duval, E., "Context-aware Recommender Systems for Learning: a Survey and Future Challenges", *IEEE Journal Of Latex Class Files*, Vol. 6-1, 2007.

[27] Drachsler, H., Hummel, H. and Koper, R., "Recommendations for learners are different: Applying memory-based recommender system techniques to lifelong learning", *Proceedings of the 1st Workshop on Social Information Retrieval for Technology-Enhanced Learning & Exchange*, 2007

[28] Negre, E., "*Information and Recommender System*", Wiley, 2015.

[29] Sarwar, B.; Karypis, G.; Konstan, J.; Riedl, J. (2000) *Application of dimensionality reduction in recommender system-a case study*,ACM WebKDD 2000 Workshop,ACM SIGKDD

[30] Brigui-Chtioui, I. and Caillou, P., "Multidimensional Decision Model for Classifying Learners: the case of Massive Online Open Courses (MOOCs)", *Journal of Decision Systems*, to appear.