Methods for building course recommendation systems

Conference Paper · October 2016

DOI: 10.1109/KSE.2016.7758047

3 authors:

Thanh-Nhan Huynh-Ly
An Giang University
8 PUBLICATIONS 43 CITATIONS
SEE PROFILE

Huu-Hoa Nguyen
Can Tho University
7 PUBLICATIONS 69 CITATIONS
SEE PROFILE

Nguyen Thai-Nghe
Can Tho University
91 PUBLICATIONS 1,359 CITATIONS
SEE PROFILE

Methods for building course recommendation systems

Huynh-Ly Thanh-Nhan[1,2]
[1]Lac Hong University
Bien Hoa City, Viet Nam
[2]An Giang University
Long Xuyen City, Viet Nam
hltnhan@gmail.com

Huu-Hoa Nguyen
Can Tho University
Can Tho City, Viet Nam
nhhoa@ctu.edu.vn

Nguyen Thai-Nghe
Can Tho University
Can Tho City, Viet Nam
ntnghe@ctu.edu.vn

Abstract—**This work introduces several methods which can be used for building the course recommendation systems. By using course recommendation system, students can early predict their learning results as well as select appropriate courses so that they can have better studying plans. After introducing the methods, this study also compares and analyzes their performance by using a real educational data set. Next, we select the best model for building the course recommendation system. Initial results show that the proposed course recommendation system can be applied in practice.**

Keywords— Recommender systems, course selection, biased matrix factorization, educational data mining

I. INTRODUCTION

In credit system at the university, many courses are opened for selection including of elective courses and mandatory courses. Elective course is the one chosen by the students from a number of optional subjects or course groups in a curriculum while the mandatory course must be taken by the students. The mandatory courses are essential for an academic degree while the elective courses tend to be more specialized. Each course has with or without previous courses, which are called pre-requisite courses or co-requisite courses that they must be learnt before going to select the following courses. In many universities, the students have to establish their learning plan for the whole four years. Although this plan can be updated every semester, the students should know or predict their performance for each course before making selection. Thus, they really need an academic advisor or an automatic course recommendation system.

Indeed, recommender systems (RS) are widely used in many fields such as e-commerce, entertainment, health-care, etc. Recently, researches have been conducted such technology for education. In practice, the universities have their own grading management systems, however, most of these systems are not still mined. From data mining point of view, we can extract the knowledge from those systems and use them for academic course advising. This is exactly what we would like to do – building the course recommender system for automatic predicting and recommending courses to the students.

This work introduces several methods which can be used for building the course recommendation systems. First, we present an overview of different methods in recommender systems which can be used for predicting student's results such as k-nearest neighbors collaborative filtering, matrix factorization, and biased-matrix factorization [8, 2]. Then, a comparison of their effectiveness is conducted. Next, the best method is selected for building the system. This study also propose a framework for building the course recommendation systems which can be personalized for each student.

II. RELATED WORK

There are many researches about educational data mining, they have been used common data mining algorithms such as decision trees, support vector machines, neural networks. For examples, [3] compared between two methods Decision Tree and Bayesian Network for predicting the academic performance of undergraduate and postgraduate students at two different academic institutes. In their intelligent course recommendation system, [7] used association rules that can recommend courses to student by using common rules, however, this system was not personalized for each student. To address the problem of predicting student performance by using RS, [4] proposed several RS algorithms which can be applied for predicting student performance. However, they have not yet built specific application for course recommendation.

In [5], they used an advising model of the Moodle. This is a web environment for student to establish learning plan, but it does not supply an intelligent advising. [6] proposed a system for academic advising using case based reasoning (CBR) that recommends to the student the most suitable majors in his case, after comparing the historical cases by using the current case.

Moreover, to address the problem of predicting student performance, many papers have been published but most of them are based on traditional classification/regression techniques [12]. Many other works can be found in [3][13]. Recently, [2][4][14] have proposed using recommendation techniques, e.g. matrix factorization [8], for predicting student performance [2,4,16,17].

However, these works just proposed the models for prediction the student performance but not for building the applications.

III. METHODS FOR BUILDING COURSE RECOMMENDATION SYSTEMS

A. *Problem formulation*

Recommender systems typically produce a list of recommendations by using collaborative filtering or content-based filtering, or hybrid approach. Collaborative filtering methods are classified as memory-based and model-based collaborative filtering. A well-known example of memory-based approaches is user-based algorithm and that of model-based approaches is the latent factor models.

In recommender systems, there are three main terms which are *user, item* and *rating*. The recommendation task is to predict the rating that the user would give for all un-rated items, then recommending top-N highest predicted score to the user. Similarly, in the grading management system (GMS) which contains three essential objects: *student, course* and *mark/grade*. In this setting, the task is predicting the course's mark that students have not learnt. As presented in Figure 1, there is a similar mapping between student modeling in a GMS and recommender systems where student, course, and mark/grade would become user, item, and rating, respectively {Student → User; Course → Item; Grading → Ratings}.

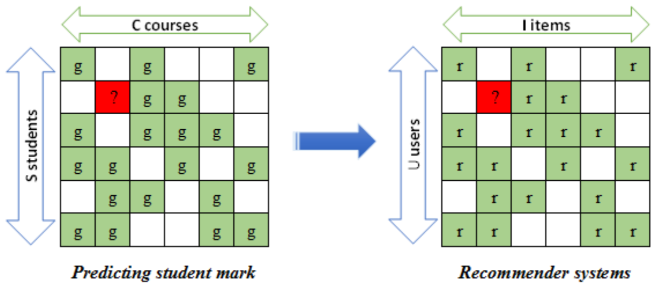
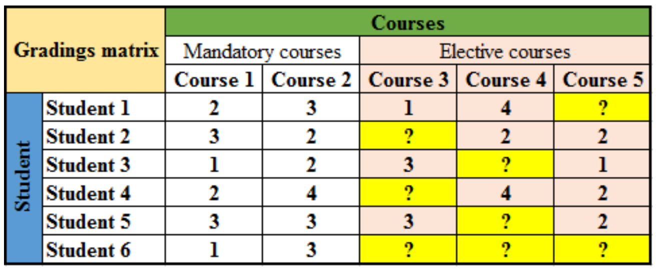


Figure 1: Mapping student model into recommender systems

After mapping student, course and mark into user, item and ratings in RS, we have a grading matrix/table as presented in Figure 2. Each cell in this matrix is a mark that the student got on a course. It is a numeric value. The cells with "?" are those students have not yet learnt.



Gradings matrix		Courses				
		Mandatory courses		Elective courses		
		Course 1	Course 2	Course 3	Course 4	Course 5
Student	Student 1	2	3	1	4	?
	Student 2	3	2	?	2	2
	Student 3	1	2	3	?	1
	Student 4	2	4	?	4	2
	Student 5	3	3	3	?	2
	Student 6	1	3	?	?	?

Figure 2: The grading matrix

Our purpose is to predict all of those missing cells ("?" values) in the matrix and using predicted results for course recommendation. However, these recommendations depend on prerequisite conditions and curriculum of a university.

B. Methods for predicting student mark/grade

In this part, we present several methods that can be used for predicting student mark in the course recommendation systems.

1) Baseline predictors

The simple methods are the baselines such as global average and student/course average [4]. Precisely, for the global average, we simply compute an average mark on the training set, then, using this number as the predicted values for all instances in the test set.

The student/course average is similar to the global average but averaging for each individual student/course, as in the following equation, where g, s, and c denote for grade/mark, student, and course, respectively (we also use these notations for all of the following formulas).

$$\hat{g}_s = \frac{\sum_{(s',c,g)\in D^{train}|s'=s}(g)}{\left|\{(s',c,g)\in D^{train}|s'=s\}\right|} \qquad (1)$$

2) Student/Course k-NNs collaborative filtering

In recommender system context, we usually assume that "similar users" may like "similar items" and vice versa. Similarly, in educational domain, we also assume that "similar students" may have similar performances on "similar courses" [4]. Thus, user-based/item-based collaborative filtering would be a choice for taking into account correlations between the students and the courses in predicting student performance. We will briefly describe how to use the k-nearest neighbors collaborative filtering in the following. This method called "Student-kNNs".

In this method, the predicted mark $\hat{g}_{sc}$ of student s on the course c is based on the mark of its nearest neighbors (students) on that course. The prediction function is determined by:

$$\hat{g}_{si} = \frac{\sum_{s'\in K_s} sim(s,s')p_{s'i}}{\sum_{s'\in K_s}|sim(s,s')|} \qquad (2)$$

Where K_s is the set of k nearest neighbors of student s, and $sim(s,s')$ is the similarity between student s and s' which can be computed by using the Cosine similarity or Pearson similarity:

$$sim_{cosine}(s,s') = \frac{\sum_{c\in C_{ss'}} g_{s'i}\cdot g_{si}}{\sqrt{\sum_{c\in C_{ss'}} g_{sc}^2 \sum_{c\in C_{ss'}} g_{s'c}^2}} \qquad (3)$$

$$sim_{pearson}(s,s') = \frac{\sum_{c\in C_{ss'}}(g_{si}-\bar{g}_s)(g_{s'i}-\bar{g}_{s'})}{\sqrt{\sum_{c\in C_{ss'}}(g_{si}-\bar{g}_s)^2 \sum_{c\in C_{ss'}}(g_{s'i}-\bar{g}_{s'})^2}} \qquad (4)$$

Where $c_{ss'}$ is a set of courses performed by both student s and student s'; $\bar{g}_s$ and $\bar{g}_{s'}$ are the mean (average) performance over all the courses of student s and s', respectively. Another prediction approach, instead of using the weighted sum, one could also use the prediction using deviations from the user (student) mean. Using deviation, the performance of student s on courses c is now determined by:

$$\hat{g}_{sc} = \bar{g}_s + \frac{.\sum_{s'\in K_s} sim(s,s')(g_{s'i}-\bar{g}_s)}{\sum_{s'\in K_s}|sim(s,s')|} \qquad (5)$$

3) Latent factor models - Matrix factorization (MF)

The latent factor models, especially matrix factorization [8] model, are flexibility in dealing with various data aspects and applications. The idea of matrix factorization method is approximating a matrix $X \in R^{|S| \times |C|}$ by a product of two smaller matrices W and H, i.e., $X \approx WH^T$; $W \in R^{|S| \times |K|}$ is a matrix where each row s is a vector containing k latent factors describing the student s, and $H \in R^{|C| \times |K|}$ is a matrix where each row c is a vector containing k latent factors describing the course c. Let w_{sk} and h_{ck} be the elements, w_s and h_c be the vectors of W and H, respectively, then the grading g given by a student s to a course c is predicted by:

$$\hat{g}_{sc} = \sum_{k=1}^{K} w_{sk} h_{ck} = w_s h_c^T \quad (6)$$

The main issue of this technique is how to find optimal values for the parameters W and H given a criterion such as Root Mean Squared Error (RMSE), which is determined by

$$RMSE = \sqrt{\frac{1}{|D^{test}|} \sum_{s,c,g \in D^{test}} (g_{si} - \hat{g}_{si})^2} \quad (7)$$

Using matrix factorization, training the model is to find the optimal parameters W and H. One approach is that we first initialize these two matrices with some random values, e.g., from the normal distribution $N(0, \sigma^2)$ with mean = 0 and standard deviation $\sigma^2 = 0.01$ and compute the error (objective) function, for example

$$O^{MF} = \sum_{(s,c,g) \in D^{train}} e_{si}^2 \quad (8)$$

Where $e_{sc}^2 = (g_{sc} - \hat{g}_{si})^2 = (g_{sc} - \sum_{k=1}^{K} w_{sk} h_{ck})^2$ (9)

Then try to minimize this error function by updating the values of W and H iteratively, e.g., using gradient descent [15]. To minimize the error function in equation (3), we need to know for each data point in which direction to update the value of w_{sk} and h_{ck}. Thus, we compute the gradient of the function (4):

$$\frac{\partial}{\partial w_{sk}} e_{sc}^2 = -2 e_{sc} h_{ck} = -2(g_{sc} - \hat{g}_{sc}) h_{ck} \quad (10)$$

$$\frac{\partial}{\partial h_{ck}} e_{sc}^2 = -2 e_{sc} w_{sk} = -2(g_{sc} - \hat{g}_{sc}) w_{sk} \quad (11)$$

After having the gradients, we update the values of w_{sk} and h_{ck} in the direction opposite to the gradient:

$$w'_{sk} = w_{sk} - \beta \frac{\partial}{\partial w_{sk}} e_{sc}^2 = w_{sk} + 2\beta e_{sc} h_{ck} = w_{sk} + 2\beta (g_{sc} - \hat{g}_{sc}) h_{ck} \quad (12)$$

$$h'_{ck} = h_{ck} - \beta \frac{\partial}{\partial h_{ck}} e_{sc}^2 = h_{ck} + 2\beta e_{sc} w_{sk} = h_{ck} + 2\beta (g_{sc} - \hat{g}_{sc}) w_{sk} \quad (13)$$

Where β is the learning rate. We iteratively update the values of W and H until the error converges to its minimum $(O_{n-1}^{MF} - O_n^{MF} < \varepsilon)$ or reaching a predefined number of iterations.

To prevent over-fitting, the error function (8) is modified by adding a term what controls the magnitudes of the factor vectors such that W and H would give a good approximation of X without having to contain large numbers. The error function now becomes:

$$O^{MF} = \sum_{(s,c,g) \in D^{train}} (g_{sc} - \sum_{k=1}^{K} w_{sk} h_{ck})^2 + \lambda \left(\|W\|_F^2 + \|H\|_F^2 \right) \quad (14)$$

Where $\|\cdot\|_F$ is a Frobenius norm and λ is a regularization term (regularization weight). With this new error function, the values of w_{sk} and h_{ck} are updated by

$$w'_{sk} = w_{sk} + \beta (2 e_{sc} h_{ck} - \lambda w_{sk}) \quad (15)$$

$$h'_{ck} = h_{ck} + \beta (2 e_{sc} w_{sk} - \lambda h_{ck}) \quad (16)$$

4) Latent factor models - Biased MF (BMF)

We have presented the standard matrix factorization to encode the student/course latent factors. Now, we introduce how to use the biased matrix factorization (BMF) to deal with the problem of “user effect” (“user bias”) and “item effect” (“item bias”) [8]. On the educational setting, the user and item biases are, respectively, the student and course biases/effects. The student effect (student bias) models how good/clever/bad a student is (i.e., how likely is the student to perform a course correctly), and the course effect (course bias) models how difficult/easy the course is (i.e., how likely is the course to be performed correctly) [4]. With these biases, the prediction function for student s on course c is presented by

$$\hat{g}_{sc} = \mu + b_s + b_c + \sum_{k=1}^{K} w_{sk} h_{ck} \quad (17)$$

$$\mu = \frac{\sum (s,c,g) \in D^{trainP}}{|D^{train}|} \quad (18)$$

$$b_s = \frac{\sum_{(s',c,g) \in D^{train} | s' = s} (g - \mu)}{|\{(s',c,g) \in D^{train} | s' = s\}|} \quad (19)$$

$$b_c = \frac{\sum_{(s,c',g) \in D^{train} | c' = c} (g - \mu)}{|\{(s,c',g) \in D^{train} | c' = c\}|} \quad (20)$$

Moreover, the error function is also changed by adding these two biases to the regularization:

$$O^{BMF} = \sum_{(s,c,g) \in D^{train}} (g_{sc} - \mu - b_s - b_c - \sum_{k=1}^{K} w_{sk} h_{ck})^2 + \lambda \left(\|W\|_F^2 + \|H\|_F^2 + b_s^2 + b_c^2 \right) \quad (21)$$

Details of this method is presented in Procedure “Student-Course-Biased-Matrix-Factorization” which learns a biased matrix factorization for factorizing student and course using stochastic gradient descent with K latent factors, β learning rate, λ regularization weight, and stopping condition. The parameter W and H are initialized randomly from the normal distribution $N(0, \sigma^2)$ with mean is zero and standard deviation $\sigma^2 = 0.01$ as in lines 9-10. While the stopping condition is not met. E.g., reaching the maximum number of predefined iterations or converging $(o_{n-1}^{BMF} - o_n^{BMF} < \varepsilon)$, the latent factors are updated iteratively. For example, in each iteration, we randomly select an instance $\{s,c,g\}$ in the train set, then compute the prediction as in lines 11-17. We then estimate the error in this iteration and update the values of W and H as in lines 18-20.

```
1.  Procedure Student-Course-Biased-Matrix-Factorization ($D^{train}$, K, β, λ, stopping condition)
    Let $s \in S$ be a student, $c \in C$ a course, $g \in G$ a mark
    Let $W[|S|][K]$ and $H[|I|][K]$ be latent factors of students, courses
    Let $b_s[|S|]$ and $b_i[|I|]$ be students-bias and courses-bias
2.    $\mu \leftarrow \frac{\sum_{g \in D^{train}} g}{|D^{train}|}$
3.    for each student $s$ do
4.      $b_s[s] \leftarrow \frac{\sum_c (g_{sc} - \mu)}{|D_s^{train}|}$
5.    end for
6.    for each course $c$ do
7.      $b_c[C] \leftarrow \frac{\sum_s (g_{sc} - \mu)}{|D_c^{train}|}$
8.    end for
9.    $W \leftarrow N(0, \sigma^2)$
10.   $H \leftarrow N(0, \sigma^2)$
11.   while (Stopping criterion is NOT met) do
12.     Draw randomly $(s, i, p_{si})$ from $D^{train}$
13.     $\hat{p}_{si} \leftarrow \mu + b_s[s] + b_i[i] + \sum_k^K (W[s][k] * H[i][k])$
14.     $e_{si} = p_{si} - \hat{p}_{si}$
15.     $\mu \leftarrow \mu + \beta * e_{si}$
16.     $b_s[s] \leftarrow b_s[s] + \beta * (e_{si} - \lambda * b_s[s])$
17.     $b_i[i] \leftarrow b_i[i] + \beta * (e_{si} - \lambda * b_i[i])$
18.     for $k \leftarrow 1, \ldots, K$ do
19.       $W[s][k] \leftarrow W[s][k] + \beta * (2e_{si} * H[i][k] - \lambda * W[s][k])$
20.       $H[i][k] \leftarrow H[i][k] + \beta * (2e_{si} * W[s][k] - \lambda * H[i][k])$
21.     end for
22.   end while
23. return $\{W, H, b_s, b_i, \mu\}$
24. end procedure
```

Parameters in this algorithm are *k* for latent factors, *beta* for learning rate, *lambda* for regularization weight, *iteration* for stopping condition. Setting these parameters is very important and difficult which effect on the accuracy of predicting. Therefore, the hyper-parameter search methods will address this problem to find the best parameters with the highest accuracy.

After the training phase, we have the two optimal latent factors W and H. The performance of a student s in a given course c is predicted by equation 17.

C. *System Architechture and Design*

The proposed system has three main feature groups: grading prediction, transferring data, and course recommendation.

- In the first group: Training/Predicting application was implemented in term of desktop application since this task would take a lot of time depending on how large the data sets. This module also includes pre-processing the missing data features/values.
- In the second group: After predicting, all grades are stored in the grading-matrix and they are transferred to web application for course recommendation.
- In the third group: This is a web application supporting for three actors: student, advisor and administrator. Figure 3 below presents the proposed system architecture.

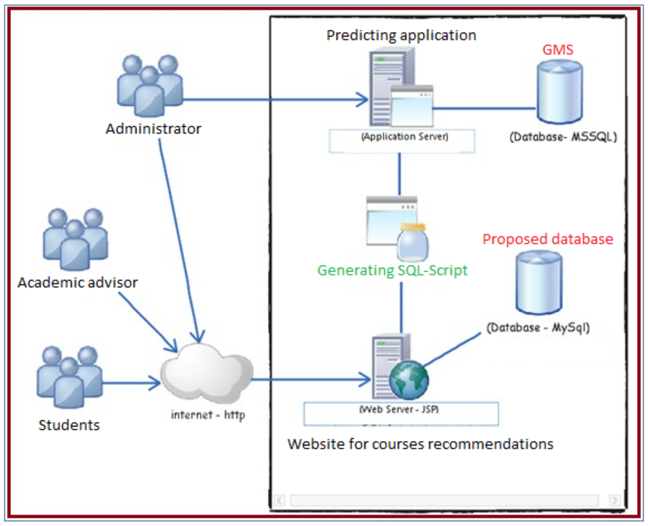


Figure 3: System Architecture

The integrated model was designed as in figure 4. They have two main components which are pre-process and process. Fortunately, there are many open source libraries that implemented several algorithms of the RS such as MyMediaLite (mymedialite.net) [11] or LibRec (librec.net) which can be used for integrating to the application.

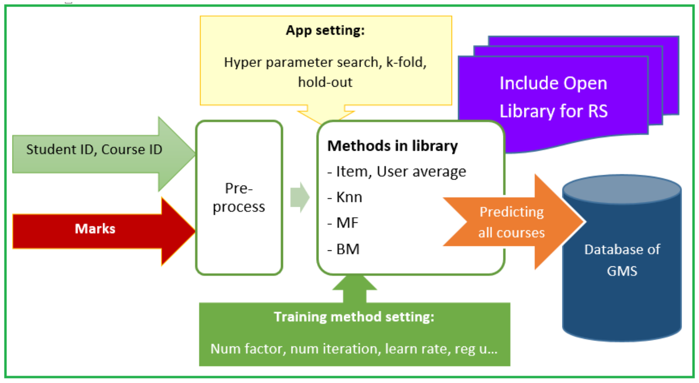


Figure 4: Integrated model for Course recommendation

An overview of the class diagram for the proposed system is presented in Figure 5. Here, we ignore the details of entity classes as well as ignore other designed models because of the paper page limitation.

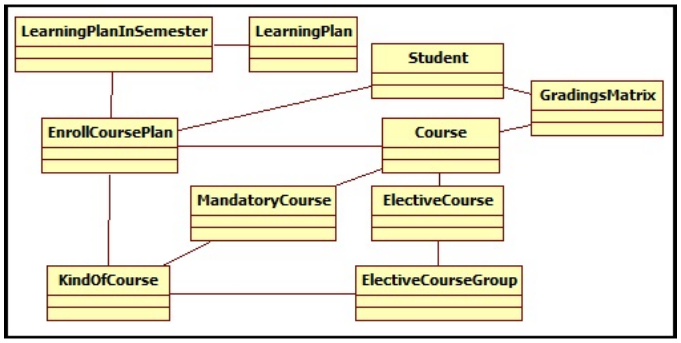


Figure 5: Class Diagram

IV. SYSTEM EVALUATION

A. *Data pre-processing*

In this work, we use a real data set which is collected from the grading management system at Can Tho University (www.ctu.edu.vn) from 1994 to 2004 in the field study of Information & Communication Technology. This data set contains 4017 students (users), 353 courses (items) and 279536 grading (ratings). In order to use these data, we have pre-processed them to match the format of recommendation algorithms as well as to remove the noises, redundant, and missing values.

B. *Evaluation Measure and Model Setting*

There are several measures which can be used to evaluate the models of the recommender systems depending on the type of problems [9]. When evaluating the models, we need to choose appropriate measure for both the algorithm and data. In this work, predicting student mark is the task of rating prediction (explicit feedback), so we use the popular measure which is Root Mean Squared Error (RMSE) for model evaluation. We have used the hold out approach (2/3 of data is used for training and 1/3 of data is used for testing) for evaluating the models.

Moreover, hyper-parameter search is applied for searching all of the hyper-parameters of the models. The hyper-parameter search has two phases such as raw search (for the long segments) and smooth search (for the short segments). First, a raw search is applied to find the best hyper-parameters in the long segments. Then, we use a smooth search to find the nearby hyper-parameters. Using RMSE as a criterion, the hyper-parameter search results for the models are presented in Table 1.

Table 1: Hyper-parameters and running-time (seconds)

Methods	Hyper Parameter	Train time
Student KNNs	k=5, simMeasure=Cosine	7680
Course KNNs	k=5, simMeasure=Cosine	450
MF	β=0.01, #iter=30, K=10, λ=0.049	10
BMF	β=0.01, #iter=35, K=10, λ=0.042	12

After having the best hyper-parameters, we use them for training and testing the final models.

C. *RMSE results*

Experimental results are displayed in Figure 6 (please note that we interchangeably call the user as the student, and the item as the course). Comparing with others, RMSE of the BMF algorithm is the smallest one (0.831) so it is used for building the course recommendation system later.

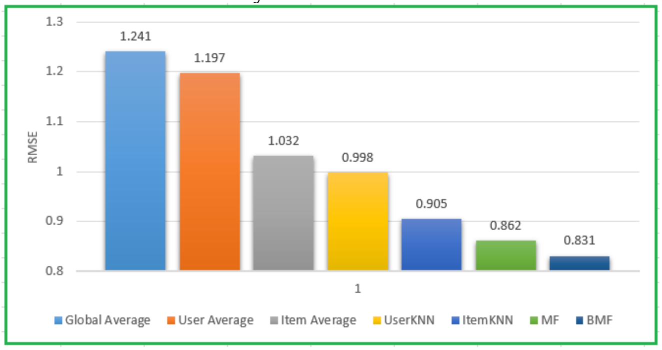
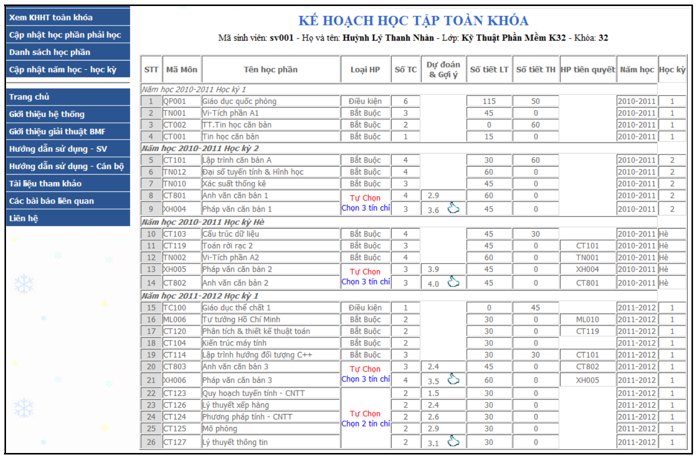


Figure 6: RMSE results on the real data set

D. *Application Snapshots*

Each student has to build his/her own learning plan, so he/she has to select the elective courses. For all of the elective courses in the selected semester, the system automatically generates recommendation courses (e.g., the courses with the "like/hand signal" ☝ in Figure 7). These courses are recommended based on predicted results, i.e., the courses with highest predicted scores are recommended (please note that since this system is designed for Vietnamese students/advisors, its interface is still in Vietnamese language).



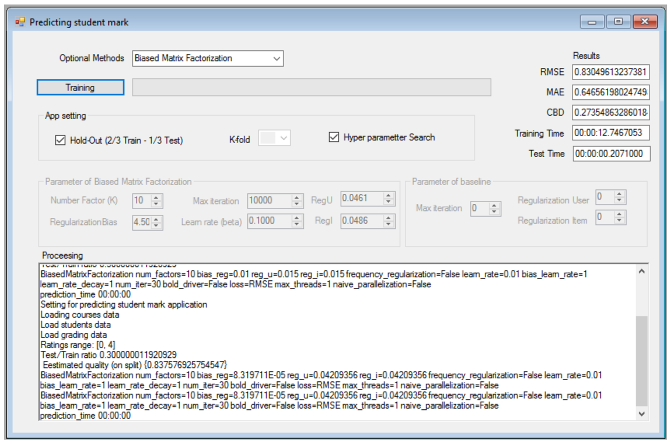
KẾ HOẠCH HỌC TẬP TOÀN KHÓA

Mã sinh viên: **sv001** - Họ và tên: **Huỳnh Lý Thanh Nhàn** - Lớp: **Kỹ Thuật Phần Mềm K32** - Khóa: **32**

STT	Mã Môn	Tên học phần	Loại HP	Số TC	Dự đoán & Gợi ý	Số tiết LT	Số tiết TH	HP tiên quyết	Năm học	Học kỳ
Năm học 2010-2011 Học kỳ 1										
1	QP001	Giáo dục quốc phòng	Điều kiện	6		115	50		2010-2011	1
2	TN001	Vi-Tích phân A1	Bắt Buộc	3		45	0		2010-2011	1
3	CT002	TT.Tin học căn bản	Bắt Buộc	2		0	60		2010-2011	1
4	CT001	Tin học căn bản	Bắt Buộc	1		15	0		2010-2011	1
Năm học 2010-2011 Học kỳ 2										
5	CT101	Lập trình căn bản A	Bắt Buộc	4		30	60		2010-2011	2
6	TN012	Đại số tuyến tính & Hình học	Bắt Buộc	4		60	0		2010-2011	2
7	TN010	Xác suất thống kê	Bắt Buộc	3		45	0		2010-2011	2
8	CT801	Anh văn căn bản 1	Tự Chọn Chọn 3 tín chỉ	4	2.9	60	0		2010-2011	2
9	XH004	Pháp văn căn bản 1		3	3.6 ☝	45	0		2010-2011	2
Năm học 2010-2011 Học kỳ Hè										
10	CT103	Cấu trúc dữ liệu	Bắt Buộc	4		45	30		2010-2011	Hè
11	CT119	Toán rời rạc 2	Bắt Buộc	3		45	0	CT101	2010-2011	Hè
12	TN002	Vi-Tích phân A2	Bắt Buộc	4		60	0	TN001	2010-2011	Hè
13	XH005	Pháp văn căn bản 2	Tự Chọn Chọn 3 tín chỉ	3	3.9	45	0	XH004	2010-2011	Hè
14	CT802	Anh văn căn bản 2		3	4.0 ☝	45	0	CT801	2010-2011	Hè
Năm học 2011-2012 Học kỳ 1										
15	TC100	Giáo dục thể chất 1	Điều kiện	1		0	45		2011-2012	1
16	ML006	Tư tưởng Hồ Chí Minh	Bắt Buộc	2		30	0	ML010	2011-2012	1
17	CT120	Phân tích & thiết kế thuật toán	Bắt Buộc	2		30	0	CT119	2011-2012	1
18	CT104	Kiến trúc máy tính	Bắt Buộc	2		30	0		2011-2012	1
19	CT114	Lập trình hướng đối tượng C++	Bắt Buộc	3		30	30	CT101	2011-2012	1
20	CT803	Anh văn căn bản 3	Tự Chọn Chọn 3 tín chỉ	3	2.4	45	0	CT802	2011-2012	1
21	XH006	Pháp văn căn bản 3		4	3.5 ☝	60	0	XH005	2011-2012	1
22	CT123	Quy hoạch tuyến tính - CNTT	Tự Chọn Chọn 2 tín chỉ	2	1.5	30	0		2011-2012	1
23	CT126	Lý thuyết xếp hàng		2	2.4	30	0		2011-2012	1
24	CT124	Phương pháp tính - CNTT		2	2.6	30	0		2011-2012	1
25	CT125	Mô phỏng		2	2.9	30	0		2011-2012	1
26	CT127	Lý thuyết thông tin		2	3.1 ☝	30	0		2011-2012	1

Figure 7: Course recommendation page

For the system administrator, he/she can use the system to re-train the models, to find the hyper-parameters such as number of iterations, number of factors, learning rate, and regularization and so on, which is presented in Figure 8. After training, the system transfers the grading matrix table from app-server to web-server. Please note that, unlike in e-commerce application where the models have to be retrained continuously to recommend suitable products to the users, in this educational domain, where the mark/grade only changes at the end of semester (after grading), thus, the system does not need to be retrained instantly.



Optional Methods: Biased Matrix Factorization

Results
RMSE 0.83049613237381
MAE 0.64656198024749
CBD 0.27354863286018
Training Time 00:00:12.7467053
Test Time 00:00:00.2071000

☑ Hold-Out (2/3 Train - 1/3 Test)   K-fold   ☑ Hyper parametter Search

Number Factor (K) 10   Max iteration 10000   RegU 0.0461
RegularizationBias 4.50   Learn rate (beta) 0.1000   RegI 0.0486

Max iteration 0   Regularization User 0   Regularization Item 0

```
BiasedMatrixFactorization num_factors=10 bias_reg=0.01 reg_u=0.015 reg_i=0.015 frequency_regularization=False learn_rate=0.01 bias_learn_rate=1
learn_rate_decay=1 num_iter=30 bold_driver=False loss=RMSE max_threads=1 naive_parallelization=False
prediction_time 00:00:00
Setting for predicting student mark application
Loading courses data
Load students data
Load grading data
Ratings range: [0, 4]
Test/Train ratio 0.300000011920929
 Eestimated quality (on split) {0.837576925754547}
BiasedMatrixFactorization num_factors=10 bias_reg=8.319711E-05 reg_u=0.04209356 reg_i=0.04209356 frequency_regularization=False learn_rate=0.01
bias_learn_rate=1 learn_rate_decay=1 num_iter=30 bold_driver=False loss=RMSE max_threads=1 naive_parallelization=False
BiasedMatrixFactorization num_factors=10 bias_reg=8.319711E-05 reg_u=0.04209356 reg_i=0.04209356 frequency_regularization=False learn_rate=0.01
bias_learn_rate=1 learn_rate_decay=1 num_iter=30 bold_driver=False loss=RMSE max_threads=1 naive_parallelization=False
prediction_time 00:00:00
```

Figure 8: Model training interface

Moreover, similar to any other information systems, the proposed system also has many management features such as course management, student management, etc.

V. CONCLUSION

This work presents several methods which can be used for building course recommendation systems. These methods are analyzed and validated by using a real data set. After selecting appropriate method, we present the framework for building the course recommendation system. This kind of system can help the students as well as their advisors in preparing their study plan.

In future work, we continue to update the models to get better prediction accuracy. For example, we can use multi-relational approach to utilize more relational information or train the models with constraints such as pre-requisite courses or non-negative contents in the matrices.

REFERENCES

[1] Feghali, T., Zbib, I., & Hallal, S. (2011). A Web-based Decision Support Tool for Academic Advising. Educational Technology & Society, 14 (1), 82–94

[2] Thai-Nghe, N., Drumond, L., Krohn-Grimberghe, A., and Schmidt-Thieme, L. 2010. Recommender system for predicting student performance. In Proceedings of the 1st Workshop on Recommender Systems for Technology Enhanced Learning (RecSysTEL 2010). Vol. 1. Elsevier's Procedia CS, 2811-2819.

[3] Nguyen Thai-Nghe, Paul Janecek, and Peter Haddawy. 2007. A comparative analysis of techniques for predicting academic performance, in Proceedings of the 37th ASEE/IEEE Frontiers in Education (FIE 2007), pp. T2G-7-T2G-12. ISSN: 0190-5848. E-ISBN: 978-1-4244-1084-2. Print ISBN: 978-1-4244-1083-5. IEEE Xplore

[4] Nguyen Thai-Nghe, Tomáš Horváth, and Lars Schmidt-Thieme. 2011. Factorization Models for Forecasting Student Performance, in Pechenizkiy, M., Calders, T., Conati, C., Ventura, S., Romero , C., and Stamper, J. (Eds.) Proceedings of the 4th International Conference on Educational Data Mining (EDM 2011). ISBN 978-90-386-2537-9

[5] N. Binh,H. Duong, T. Hieu, N. Nhuan, N. Son,” An integrated approach for an academic advising system in adaptive credit-based learning environment”, *VNU Journal of Science, Natural Sciences and Technology,vol.* 24 ,pp. 110-121, 2008

[6] Lamiaa Mostafa, Giles Oately, Nermin Khalifa, and Walid Rabie. 2014 A Case based Reasoning System for Academic Advising in Egyptian Educational Institutions. 2nd International Conference on Research in science, Engineering and Technology (ICRSET'2014)

[7] Mohammed Al-Sarem, 2015. Building a Decision Tree Model for Academic Advising Affairs Based on the Algorithm C4. 5. IJCSI International Journal of Computer Science Issues, Volume 12, Issue 5, September 2015 ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784.

[8] Yehuda Koren, Robert Bell and Chris Volinsky, 2009. Matrix Factorization Techniques for Recommender Systems.

[9] Guy Shani and Asela Gunawardana. Evaluating Recommendation Systems. November 2009.

[10] Nikos Manouselis, Hendrik Drachsler, Katrien Verbert and Erik Duval. 2012. Recommender Systems for Learning. March 23, 2012

[11] Zeno Gantner et al, 2011. MyMediaLite: A Free Recommender System Library.

[12] M. Feng, N. Heffernan, and K. Koedinger, "Addressing the assessment challenge with an online system that tutors as it assesses," User Modeling and User-Adapted Interaction, vol. 19, no. 3, pp. 243–266, 2009

[13] C. Romero, S. Ventura, M. Pechenizkiy, and R. S. Baker, Handbook of Educational Data Mining. Chapman and Hall/CRC, October 2010

[14] Toscher, A. and Jahrer, M. 2010. Collaborative ltering applied to educational data mining. KDD Cup 2010: Improving Cognitive Models with Educational Data Mining.

[15] Takacs, G., Pil´aszy, I., N´emeth, B. & Tikk, D. (2007). On the Gravity recommendation system. In Proceddings of KDD Cup Workshop at SIGKDD'07, 13th 175 REFERENCES ACM Int. Conf. on Knowledge Discovery and Data Mining, 22–30, San Jose, CA, USA. 42

[16] Nguyen Thai-Nghe, Tomáš Horváth, and Lars Schmidt-Thieme. 2011. Personalized Forecasting Student Performance, in Proceedings of the 11th IEEE International Conference on Advanced Learning Technologies (ICALT 2011). pp. 412 - 414. IEEE Xplore

[17] Nguyen Thai-Nghe, Lucas Drumond, Tomáš Horváth, and Lars Schmidt-Thieme. 2012. Using Factorization Machines for Student Modeling. Proceedings of FactMod 2012 at the UMAP 2012. Vol. 872, CEUR-WS, ISSN:1613-0073