EasyChair Preprint

№ 529

# Taking e-Assessment Quizzes - A Case Study with an SVD Based Recommender System

Oana Maria Teodorescu, Paul Stefan Popescu and
Marian Cristian Mihaescu



September 26, 2018

# Taking e-Assessment Quizzes - A Case Study with an SVD Based Recommender System

Oana Maria Teodorescu, Paul Stefan Popescu, and Marian Cristian Mihaescu

University of Craiova, Romania,
teodorescu_oanamaria@yahoo.com, {popescu, mihaescu}@software.ucv.ro

**Abstract.** Recommending learning assets in e-Learning systems represents a key aspect. Among many available assets there are quizzes that validate and also evaluate learner's knowledge level. This paper presents a recommender system based on SVD algorithm that is able to properly recommend quizzes such that learner's knowledge level is evaluated and displayed in real time by means of a custom designed concept map for *graphs* algorithms within the *Data Structures* course. A preliminary case study presents a comparative analysis between a group a learners that received random quizzes and a group of learners that received recommended questions. The visual analytics and interpretation of two representative cases show a clear advantage of the students received recommended questions over the other ones.

**Keywords:** SVD,· recommender systems,· quizzes, · e-Learning.

## 1 Introduction

E-assessments are an important part of any e-learning platform, as they test the learners accomplishment of meeting the learning objectives. They are useful in two directions, for both the learner and the professor. One of them is for strengthening ones knowledge (involving primarily the learners objective when studying a course) and the other is evaluating the learners comprehension of the course (involving both the learner and the professor, as a means of self-evaluation and progress tracking versus progress evaluation and defectiveness of learning materials).

Therefore, development of knowledge evaluating and tracking methods is an issue that gets continuous attention in educational data mining. One of the paper's purpose is to investigate a visualization approach of the students knowledge by means concept maps and question coverage. Thus, several visualization methods accompanied by a SVD based recommendation algorithm were proposed. Firstly, for each student, we design a student's concept map which presents concepts and their coverage in percentages. Based on available weighted concept maps, a students time-line is constructed for one or more tests (i.e, pool of quizzes), such that the evolution in time of the concepts coverage in percentages may be easily visualized.

The scope of this paper is the evaluation of the effectiveness of the test question recommender system implemented in the Tesys platform for multiple types of questions. This will be done by studying and comparing the results and knowledge gained by learners taking tests using the random vs recommender strategy. Furthermore, an objective function will be defined for each learner to visualize the unique path taken by each individual when learning new concepts. This function will also be used to validate the recommenders effectiveness against the random-pick strategy. The function will result from creating a concept map for the concepts of a particular subject and by mapping them, in a many-to-many relationship, to the questions designed for the self-testing purpose of the learner for each chapter of that subject. This also allows for the tracking and evaluation of a students progress by both learner and professor.

## 2 Related Work

Tesys [2] is a web e-learning platform running for the students of the University of Craiova enrolled in distance education. It provides the resources need by students, professors and secretaries to perform their activity.

D3.js (or just D3 for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. In contrast to many other libraries, D3.js allows great control over the final visual result. [15] Its development was noted in 2011, [8] as version 2.0.0 was released in August 2011 [17]. The data can be in various formats, most commonly JSON, comma-separated values (CSV) or geoJSON, but, if required, JavaScript functions can be written to read other data formats. [17]

Before we go into concept maps, we must define concepts which are defined as perceived regularities or patterns in events or objects, or records of events or objects, designated by a label and are depicted as shapes in the diagram. [13]

A concept map or conceptual diagram is a diagram that depicts suggested relationships between concepts. It is a graphical tool that instructional designers, engineers, technical writers, and others use to organize and structure knowledge. [9]

Content-based recommendation systems analyze item descriptions to identify items that are of particular interest to the user. [7] In a content-based recommender system, keywords are used to describe the items and a user profile is built to indicate the type of item this user likes.This approach has its roots in information retrieval and information filtering research. [5]

Collaborative Filtering (CF) is a subset of algorithms that exploit other users and items along with their ratings(selection, purchase information could be also used) and target user history to recommend an item that target user does not have ratings for.. CF differs itself from content-based methods in the sense that user or the item itself does not play a role in recommendation but rather how(rating) and which users(user) rated a particular item. (Preference of users is shared through a group of users who selected, purchased or rate items similarly) [3]

An example of use of hybrid filtering is the American global provider of streaming films and television series Netflix, which make recommendations by comparing the watching and searching habits of similar users (i.e. collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (content-based filtering) [11] Basic techniques for recommender systems (collaborative, content-based, knowledge-based, and demographic techniques) have known shortcomings such as the well known cold-start problem for collaborative and content-based systems (what to do with new users with few ratings) and the knowledge engineering bottleneck [10] in knowledge-based approaches, as Wikipedia states in [11].

According to an MIT tutorial for SVD (Singular Value Decomposition) [4], calculating the SVD for a matrix M consists of finding the eigenvalues and eigenvectors of MM at the power if T and M at the power of T multiplied by M. The eigenvectors of Mat the power of T multiplied by M make up the columns of V, the eigenvectors of MM at the power of T make up the columns of U. Also, the singular values in  are square roots of eigenvalues from MM at the power of T or M at the power of T multiplied by M. The singular values are the diagonal entries of the  matrix and are arranged in descending order. The singular values are always real numbers. If the matrix M is a real matrix, then U and V are also real.

Recommender systems for e-Learning platforms are based on many approaches like web mining and information retrieval [6], recommender systems based on the context [14] or even using intelligent agents [16]. One interesting approach of using collaborative filtering in e-Learning systems [1] was to assign greater weights for users with higher knowledge than the users with lower knowledge and to obtain that the authors propose some new equations in the nucleus of the memory-based collaborative filtering. Another interesting paper, presenting clear results regarding recommender systems in smart e-Learning environments presents their approach [12] along with their encouraging results and their aim to extent the system for more faculties.

## 3 Proposed Approach

One of the goals of this project is to offer the users of Tesys a personalized experience by the ability of the Tesys e-learning platform to adapt to the students individual needs in order to enhance the effectiveness of the learning process. For achieving this, a recommender system for choosing questions to be included in a test for a specific subject and chapter has been used. The algorithm used for the recommender system is SVD. The unique path taken by each student in the learning process of a subject is displayed using an evolution concept map based on the concepts covered in questions answered in a self-testing environment.

The following question types were defined: ***Matching question*** - given two columns, A and B, with options on both sides, match the corresponding item from column A with the one on column B given a certain criteria stated in the question, ***Short-answer question*** - given a phrase, fill in the gaps with a

short answer (word or group of words), ***Numerical question*** - given a phrase, fill in the gaps with a numerical value (may be either an integral value or a real number), ***True/False question*** - given a statement, assert its truth value. Therefore, we have designed a SVD based recommender system that performs in the following way. The procedure is presented below:

**Algorithm 1** Recommendation algorithm using SVD

Apply SVD algorithm on matrix M =($m_{(i,j)}$) of size n x m which generates matrices U =($u_{(i,j)}$) of size n x k and V =($v_{(i,j)}$) of size k x m.
**if** the student is new (has taken no tests for the subject and chapter so far) **then**
  Construct user feature row as a row vector L of size 1 x k filled with values of 1.
**else**
  Extract user feature row as a row vector L of size 1 x k from the user-features matrix U.
**end if**
**for each** question feature column vector C **in** the question-features matrix V (index i) **do**
  Perform the dot product p between L and C.
  Save dot product p value to question recommendation vector along with its index i as key.
**end for**
Sort question recommendation vector ascending with respect to values, then keys.
Select the first recommended questions that were not previously answered by the student.
**if** the number of required questions has not been reached **then**
  Select from a list of unanswered questions by any student.
**end if**
**if** the number of required questions has not been reached **then**
  Select the last recommended questions (they were previously answered by student, thus selecting the ones the current student or others answered mostly incorrectly).
**end if**

**Input.** The logical input of our algorithm consists of students results to tests, or more specifically, to questions in tests. We collect this information based on subject and chapter, as was previously stated in the previous section and process it to a form which is understandable to the algorithm we chose for recommendation (singular value decomposition). Therefore, the actual (processed) input of our algorithm consists of: **A list of *n* students** taken into account for the recommendation process (represented, in our implementation, by the list of student identifiers). **A list of *m* questions** taken into account for the recommendation process, that have been answered by students (represented, in our implementation, by the list of question identifiers). **A matrix $\boldsymbol{M} = (m_{(i,j)})$** of size n x m, having values in the range [0, 1] which represent the accuracy of the student $i$ when answering the question $j$ (a measure computed by normalizing the average grade of a student for a question by the maximum grade for that question)

**Output.** The output of the algorithm consists of a list of questions recommended to be included in a test taken by a student for a specific subject. The aim is to provide personalized results that help the student to discover areas which need to be improved in the learning process of a particular subject.

**The recommendation mechanism**. The algorithm relies on the SVD technique which decomposes a matrix into two matrices U and V in a lower dimensional feature space. The decomposition is used to achieve a separation of the user and question models into features that identify individual attributes of the students and questions whose associations give their uniqueness.

## 4 Experimental Results

### 4.1 Usage of SVD method for recommending questions

If M is the initial matrix of m x n dimension, the algorithm finds U of m x m, of m x n and V of n x n dimensions that multiplied give an approximation of the initial M matrix. The U matrix represents the strength of the associations between a user and the features in the hidden feature space while the V matrix represents the strength of the associations between an item and the features in the hidden feature space.  is a diagonal matrix.

In our case, the users are students and items are questions for a specific subject and chapter.

$$U = \begin{pmatrix} 0.44 & -0.22 & -0.90 \\ 0.66 & 0.69 & 0.30 \\ 0 & 0 & 0 \\ 0.61 & -0.73 & 0.31 \end{pmatrix} V = \begin{pmatrix} 0.75 & -0.07 & -0.66 \\ 0.55 & -0.50 & 0.67 \\ 0.38 & 0.86 & 0.33 \end{pmatrix}$$

With these two matrices, we will select a number of q questions for a student at the request of a test on a specific chapter. We identify two cases:

1. student is new and the system knows nothing about him (no row data in the matrix)
2. student has already answered some questions

These two cases will be explained separately in the following sub-chapters.

*Selecting Questions for a New Student*

In the case of a new student, nothing is known about his/her skills (he/she has taken no tests). This is called cold-start. But from the V matrix, the features of each question were extracted from the experience with the other students who already answered some questions.

To deal with this problem, consider a user L line matrix of dimension k filled only with values of 1. This indicates that the features have equal probability for the user and the system will rely solely on how these features impact the questions in the recommendation. By multiplying this line with each questions column from V through the features space (), for each question a value is obtained, thus leading to a one dimensional matrix R.

In our example, the computed L and R matrices are:

$$L = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

$$R = L * \Sigma * V = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} * \Sigma * \begin{pmatrix} 0.75 & -0.07 & -0.66 \\ 0.55 & -0.50 & 0.67 \\ 0.38 & 0.86 & 0.33 \end{pmatrix} = \begin{pmatrix} 0.01 & 0.72 & 1.58 \end{pmatrix}$$

By sorting the values ascending in the R vector and recalling the previous indexes, the first questions are taken for our student as the recommendation.

The result of this approach is the questions that most students answered correctly with a higher probability have priority, so the questions selected to be used to test our new student will test his/her basic skills first and then adjust the difficulty after more tests have been taken.

In the above example, the top 2 recommended questions are the third and second questions.

*Selecting Questions for an Existing Student*

In the case of an existing student, matrix U contains user-features information and the row corresponding to the user can be used to compute the recommendation vector. If the user row in the U matrix is L, then compute the final vector by multiplying it with the V vector containing question-features information through the features space ().

Lets say we want to recommend 2 questions for a test to student. In this case and the conditions of the example above, the computed L and R matrix are:

$$L = \begin{pmatrix} 0.61 & -0.73 & 0.31 \end{pmatrix}$$

$$R = L * \Sigma * V = \begin{pmatrix} 0.61 & -0.73 & 0.31 \end{pmatrix} * \Sigma * \begin{pmatrix} 0.75 & -0.07 & -0.66 \\ 0.55 & -0.50 & 0.67 \\ 0.38 & 0.86 & 0.33 \end{pmatrix} = \begin{pmatrix} 0.30 & 0.91 & -0.29 \end{pmatrix}$$

Like in the new student case, by sorting the values ascending in the R vector and recalling the previous indices, the system can take the first questions for our student as the recommendation. The result will select more similar questions with respect to the features identified which have not been answered by the student or, in case the number of questions needed is higher than the number of questions the student didnt answer, it will take questions from a list of questions unanswered by any student (new questions), and if the number of questions needed for the test still hasnt been reached, it takes the last values in the ascending ordering. The reason to do this is we want to select the questions that the student mostly failed to answer.

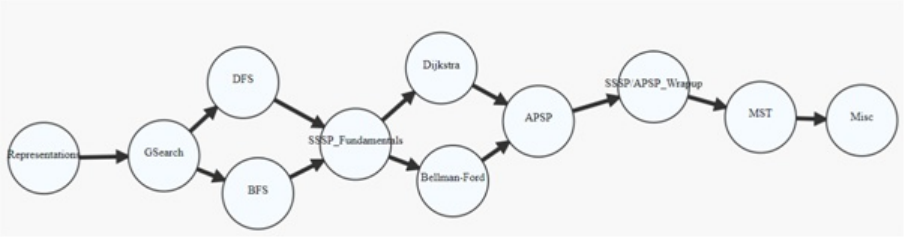
In the above example, the top 2 recommended questions are the third and first questions.

### 4.2 Case Study using Recommender System and Concept Map

A case study has been conducted on bachelor students in 3-rd semester at Computers to assess the recommender systems efficiency. A set of 98 questions were

added for the self-testing of students in the Tesys platform for a chapter module about Graphs at Data Structures course. Further, each question has been associated a concept within a custom built concept map whose structure is presented in figure 1.

**Fig. 1.** The concept map for graphs




Students were divided in two groups: some using the platform with the random approach while others were using the recommender system for picking the questions for the tests. A number of 50 students participated at trial, 21 were randomly selected to receive random questions and 29 received questions from the recommender system. A total number of 140 tests were taken by all

Figure 2 presents the current status of the concepts covered by the student in three tests with random selection of quizzes. For each concept there are provided two values: the number of correctly answered questions and the total number of questions available for that concept. Initially, all nodes are red since there is there are no correct answers, but this situation is not presented in figure 2.

The concept graph from the top of the figure 2 presents the situation after first test is taken. As it may be observed, some nodes are becoming orange or even yellow, as the number of correctly answered question covers a higher percentage from the entire number of questions assigned to a specific concept. For example, concept *GSearch* is covered half, while *SSSP/APSP Wrappup* and *Misc* are less covered. The concept graphs form the second and third test show the randomness of the selection process of quizzes as nodes are becoming orange at the end of the graph while prerequisite concepts are still red, that is not covered.

Figure 3 presents the current status of concepts in five tests with recommended quizzes. Visual analytics of the concept coverage reveals a smooth coloring of concepts from red to orange to yellow and finally to green without major red areas in front of between nodes with progress (i.e., orange, yellow or green).

## 5 Conclusions

The paper presents an SVD based recommender system that offers students the possibility to visualize their knowledge level in a weighted graph of concepts.

**Fig. 2.** Three tests evolution with random quizzes

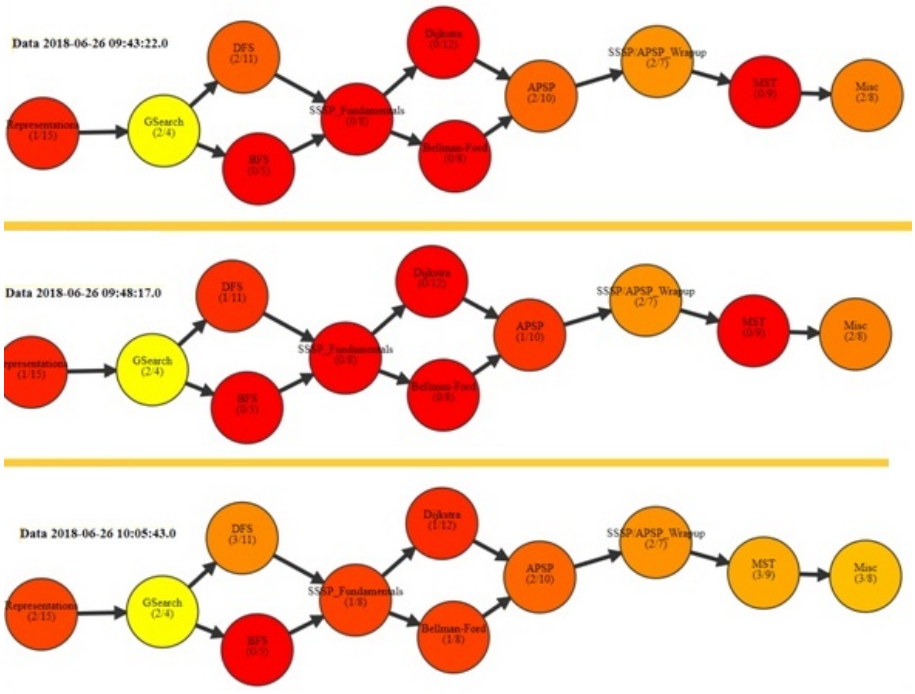
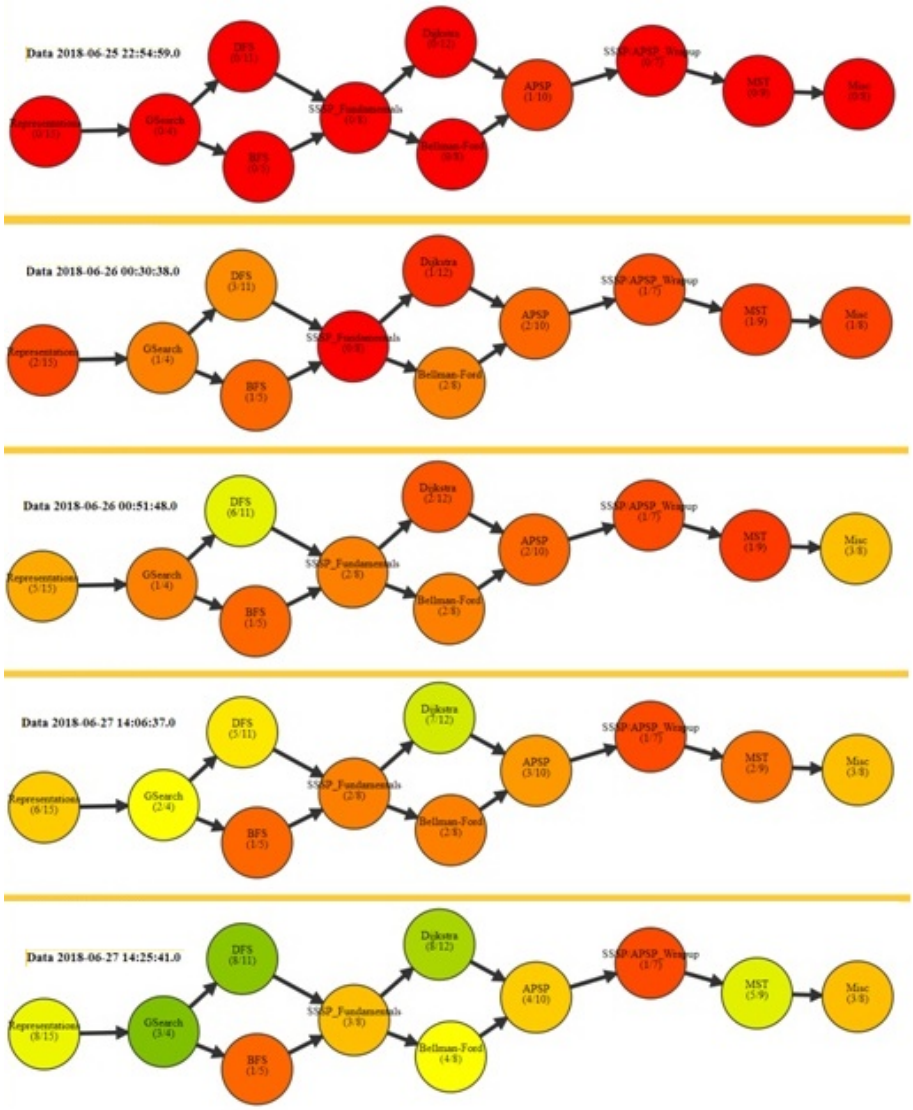


**Fig. 3.** Five tests evolution with recommended quizzes



We defined a concept map as a graph where each node represents a concept which has an associated weight representing the knowledge as a percentage of correctly answered questions from the number of available questions at that concept.The concept map was designed for *Graphs* chapter within the *Data Structures* discipline. The main goal of the recommender system is to provide a personalized set of questions depending on their current status (i.e., correctly and incorrectly questions answered so far) and the status of all other colleagues that have previously taken tests. Visual analytics of the experimental results in terms of impact of recommendation procedure in knowledge coverage of the concept map show pro missing initial results. Further work needs to be performed in terms of defining and integrating appropriate quantitative and qualitative metrics for measuring accumulated knowledge with and without the usage of the recommender system.

## References

1. Bobadilla, J., Serradilla, F., Hernando, A., et al.: Collaborative filtering adapted to recommender systems of e-learning. Knowledge-Based Systems **22**(4), 261–265 (2009)
2. Burdescu, D.D., Mihaescu, M.C.: Tesys: e-learning application built on a web platform. In: ICE-B. pp. 315–318 (2006)
3. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS) **22**(1), 5–53 (2004)
4. Hoekstra, R.: The knowledge reengineering bottleneck. Semantic Web **1**(1, 2), 111–115 (2010)
5. Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems: an introduction. Cambridge University Press (2010)
6. Khribi, M.K., Jemni, M., Nasraoui, O.: Automatic recommendations for e-learning personalization based on web usage mining techniques and information retrieval. In: Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on. pp. 241–245. IEEE (2008)
7. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer (8), 30–37 (2009)
8. Myatt, G.J., Johnson, W.P.: Making sense of data iii: A practical guide to designing interactive data visualizations, vol. 3. John Wiley & Sons (2011)
9. Novak, J.D., Cañas, A.J.: The theory underlying concept maps and how to construct and use them (2008)
10. Ricci, F., Cavada, D., Mirzadeh, N., Venturini, A., et al.: Case-based travel recommendations. Destination recommendation systems: behavioural foundations and applications pp. 67–93 (2006)
11. Shani, G., Heckerman, D., Brafman, R.I.: An mdp-based recommender system. Journal of Machine Learning Research **6**(Sep), 1265–1295 (2005)
12. Soonthornphisaj, N., Rojsattarat, E., Yim-Ngam, S.: Smart e-learning using recommender system. In: International conference on intelligent computing. pp. 518–523. Springer (2006)
13. Sowa, J.F.: Conceptual structures: information processing in mind and machine (1983)

14. 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 Transactions on Learning Technologies **5**(4), 318–335 (2012)
15. Viau, C.: Whats behind our business infographics designer? d3. js of course (2012)
16. Zaíane, O.R.: Building a recommender agent for e-learning systems. In: Computers in education, 2002. proceedings. international conference on. pp. 55–59. IEEE (2002)
17. Zhu, N.Q.: Data visualization with D3.js cookbook. Packt Publishing Ltd (2013)