Continuous Student Knowledge Tracing Using SVD and Concept Maps

Oana Maria TEODORESCU, Paul Ștefan POPESCU,
Lucian Mihai MOCANU, Marian Cristian MIHĂESCU
Faculty of Automation, Computers and Electronics, University of Craiova, 200440, Romania
oteodorescu@software.ucv.ro

***Abstract*—One of the critical aspects of building intelligent tutoring systems regards proper monitoring of student's activity and academic performance. This paper presents a continuous student knowledge tracing method implemented for Tesys e-Learning platform at the Faculty of Automation, Computers and Electronics in the University of Craiova. The student's knowledge level is continuously monitored and, after each recommended test by the SVD-based mechanism, a new set of knowledge weights are computed. We aim to achieve a comprehensive monitoring environment which can provide an accurate insight upon the student's knowledge level at any moment. In our approach, we added weights for both students and tests to improve the student's evolution monitoring and provide more accurate feedback. The setup for validation consisted of ten tests with eight questions per test and we used both current and past year tests data. Results revealed that assigning weights to questions, tests and students and using them in the recommendation process offers a better view of the student's evolution along with more accurate recommendations. Progress in this direction will provide more insight into available teaching materials and SVD-based recommender system such that the e-learning platform that integrates the presented mechanism will provide a better learning experience.**

***Index Terms*—data preprocessing, distance learning, knowledge representation, human computer interaction, recommender systems.**

I. INTRODUCTION

This paper continues the works from [1] which presents a custom validation procedure of the previously developed system [2-3]. The initial works started in [4] present a solution for building personalized quiz sessions which has later been developed into an SVD-based recommender system. The current approach described in this paper offers a better overview of how the recommender system performs as new features are now implemented, along with a more accurate knowledge tracking procedure. Having a recommender system integrated into the e-learning platform represents an essential aspect because it influences the student's engagement into the learning process. If the questions are not adequate to the level of knowledge of the student, taking tests will become dull, and the grades may not reveal their actual knowledge.

The context of the study is Tesys e-Learning platform [2], which is an online educational platform custom developed for the University of Craiova for several distance-education degree programs. Tesys offers a variety of functionalities such as learning resource management, user communication or taking tests and exams. For each course, students can access a set of chapters, homework, and external references, but they can also take many tests (or exams) from questions previously defined by the professor. When a student starts a test, they will receive several questions to answer in a fixed time defined by the professor and the questions can be either chosen randomly or using the recommender system presented in this paper. Questions can be of several types: multiple choice answer, truth-value answer or matching items from two columns.

For our approach, we use SVD (Singular Value Decomposition) [5] as a base algorithm for our recommender system because it offers two important features: it is a fast algorithm, and it recalculates the decomposition matrices before each recommendation, so question knowledge is continuously updated as tests are taken. For providing a logical path in the way the questions are selected for the tests, we use a concept map which can be defined by the professor for each chapter of a course. Furthermore, the professor can map questions to the corresponding concepts from the concept map and the recommender system will ensure concepts from tests are covered in the order they were defined. In a concept map, the nodes define the concepts in a chapter and the arrows define the set of prerequisite concepts, which need to be learnt before the new concept.

The knowledge tracking infrastructure presented in this paper is applied to the Data Structures and Algorithms course taught in the 3-rd semester at the Faculty of Automation, Computers and Electronics in the University of Craiova.

The limitation of previous works was that they lacked an appropriate tracking methodology for assessing student's knowledge level at any point in time. This limitation represents a severe knowledge gap for detailed assessment and further debugging of the proposed recommender system. We hypothesize that a proper knowledge tracking method will bring valuable insight into the SVD based recommender system and allow further improvements.

To test and validate the recommendation algorithm, we considered test data from students in the previous year of study for this subject as prior knowledge for the test recommendation in the current year. For both years of study, the same 11 concepts distributed into 98 questions were considered, with the goal of a student no longer receiving duplicate questions. The test consisted of 8 questions and, for each question, one minute was allocated; therefore, the total time for a test was eight minutes.

Our proposed approach describes a workflow for concepts where students need to answer more than 75% of questions correctly to receive questions from a new (next) concept.

Digital Object Identifier 10.4316/AECE.2021.01008

We also define weights for questions, concepts, and levels for being able to rank them based on their difficulty. To analyze the results and get relevant conclusions, a timeline was developed for each student. Based on the graph of concepts previously assigned to the course/chapter(s), professors can quickly get feedback regarding their subject and analyze how students perform.

We analyzed the test results and compared them to the previous year to evaluate the evolution of the students. Tracking each student's activity is done using a graph of concepts that change their color as they take tests and answer questions from these concepts correctly.

The paper is organized as follows. In Section 2, we perform a literature review with regards to similar approaches. Section 3 describes the proposed approach with a detailed presentation of how the continuous knowledge tracking is performed. Section 4 presents the experimental results on datasets from two academic years. Finally, section 5 contains the conclusions of this work, summarizes the main contributions and discusses potential improvements and applications.

II. Related Works

Adaptive learning and testing represent a pivotal issue in e-Learning recommender systems, and it has been addressed lately in [6] by integrating IRT [7] or Rasch [8] models with various machine learning techniques (i.e., classification and regression trees, random forest) for assessing the knowledge level of a new user.

One particular issue with IRT regards interpretability [9] of parameters (i.e., discrimination, guessing, ability) in the attempts to build adaptive testing systems that integrate machine learning algorithms.

In general, analyzing historical educational records by means of matrix-factorization with temporal course-wise influence (MFTCI) has been addressed in [10] by representing students and courses in a latent knowledge space.

Addressing the problem of ranking items (i.e., questions) in terms of their difficulties has been done in [11] by using KNN with Pearson correlation and a matrix factorization method using SVD to compute latent factors of items and users.

The most well-known method for determining 'learner's knowledge is the usage of q-matrix [12] which is was used in [13] to show that "it is possible to automate the diagnosis of student knowledge states, based solely on student item-response patterns and the relationship between questions and their concepts".

Usage of SVD for implementing recommender systems is a popular approach because it is a fast and straightforward algorithm [14]. Depending on the purpose of the algorithm, users tend to use it as a stand-alone algorithm [15] or in conjunction with others. In [15], authors propose a novel algorithm to choose useful neighbors of users or items for generating the input data, and they claim that their algorithm is useful to all SVD-based recommendation methods. The results are quite relevant, as they used four datasets and their approach outperformed basic SVD methods by more than ten per cent.

In [16], the authors present an ensemble algorithm for getting the top-N items from the SVD results because standard SVD suffers from a computational limitation when delivering the top-N items online. This approach delivers faster and more accurate top-N items than the basic SVD. Still, in the area of top-N recommendations is [17], which is a versatile and efficient latent factor framework for top-N recommendations that include the well-known PureSVD algorithm as a particular case. This paper is strongly related to our approach, as we also choose the top-N recommendations, but in their case, authors use ratings as input, and their result is a prediction, while in ours the input is represented by the student's grades and the task is to recommend questions for the next test.

Another interesting approach for improving the performance of recommender systems is to update the data and, more specifically, to forget obsolete data [18]. The authors introduce several forgetting strategies as methods for selecting this obsolete information, which may be relevant. They need to be taken into consideration even for data in education because students evolve from year to year, and they change their learning techniques and interests.

More educational-related techniques are presented in [19] where authors present a peer assessment method which provides feedback, more consistent grading and aims to students' load in MOOCs (Massive Open Online Courses) [20]. Related to SVD, there is a matrix factorization technique which aims to improve the peer assessment and their overall approach improves the feedback to the student along with a generalization of student's overall knowledge and reducing the students' work burden.

There are also problems regarding scalability and sparsity in the datasets used for recommender systems which are presented in [21]. The authors present a comprehensive overview of the applicability of some advanced techniques, particularly clustering, bi-clustering, matrix factorization, graph-theoretic, and fuzzy techniques in recommender systems. However, the applicability of matrix factorization (in recommender systems for our use case) are such large research areas that truly comprehensive surveys are almost impossible, as the authors state.

Other approaches [22] present a different context of using the SVD algorithm, such as face recognition. This approach reveals the flexibility of the SVD algorithm and how it can be used in many research areas. In this case, the authors build an individual SVD basis set for each image and then learn a standard set of SVs by taking account of the information in the basis sets according to a discriminant criterion across the training images.

Regarding the knowledge tracking proposed methods, one of the most famous ones is BKT (Bayesian Knowledge Tracking) initially proposed in [23] and further used in many approaches such as [24], [25] and many other. The problem reduces to modelling learner's knowledge as a fundamental building block of an intelligent tutoring system. The key ingredients used in various approaches are skills, parameters, the actual learning context and the business logic that is based on a particular algorithm. From this perspective, BKT is based on Bayes Nets implemented in Bayes Net Toolkit-Student Modeling (BNT-SM) [26]. Issues related to these approaches regard the fact that multiple sets of profoundly different parameters may predict

the data equally well, which gives rise to low identifiability, local minima, degenerate parameters, and computational cost during fitting.

Another key ingredient used in modelling students in intelligent tutoring systems regards the usage of concept maps [27]. Measuring various aspects of behaviors has been performed by classical usage of machine learning approach with predictor and predicted variables within a regression context. Usage of the concept maps made possible replacing predictors such as the number of log-in times, or the number of forum posts reads with predictors like the total number of learning objects (i.e., tests), the total number of concepts, total score gained and total time spent online. A correlation has been usually performed by regression analysis and p-value hacking with point-wise conclusions [28].

Lately, knowledge tracking has also been performed by personalized multi-agent systems [29] by analysis of student internal and external interactions as questions and responses. Stack Overflow uses this approach for computing Semantic Correlation and WordNet as a knowledge base to infer semantic relatedness, which opens the way towards Natural Language Processing as a critical ingredient along with machine learning algorithms in the attempt towards efficient knowledge tracking.

In order to build an adaptive learning system, authors in [6] combine Item Response Theory [9] with Machine Learning in order to produce item response prediction for new learners. They also compare their proposed system to alternative approaches by conducting experiments on two different educational datasets. Their chosen algorithm is Random forest which combined with Item Response Theory provides the best prediction results and one conclusion is that this combination of can alleviate the effect of the cold start problem [30] in adaptive learning environments [31].

III. Proposed Approach

A. Data Workflow

We have developed a custom data workflow that is based on an underlying concept map (directed acyclic graph) designed by the professors for the Graph Data Structures chapter within the Data Structures and Algorithms course.

The workflow is presented graphically as a flowchart in Figure 1 and considers a subset of questions from the entire pool of test questions for the recommendation process by:

- constructing levels in the graph representation of the concept map. Each level consists of one or more concepts.
- assessing the target student's completeness of each level by analyzing previous answers to questions in each concept in a level.

Questions are assigned to one concept in the concept map, forming a disjoint q-matrix. Therefore, concepts are always included in a single level in the graphical representation of the concept map. This ensures that questions are selected in a logical order and are only ranked at the student's current level(s) using a recommender system based on SVD. The input matrix for the SVD recommender consists of the average grade of previous answers for all students which have been enrolled in the course and taken tests with questions from the corresponding chapter.

By performing SVD (latent factor model), features and their correlation are extracted from the student-question matrix and used for predicting the next questions for the test of a student.

In the flowchart (Figure 1), the left-side rounded figure (with a triangle at its left side) denotes the starting point, while the right-side one (with a square at its left side) denotes the ending point. Parallelograms denote input/output data processes, while rectangles denote algorithmic processes.

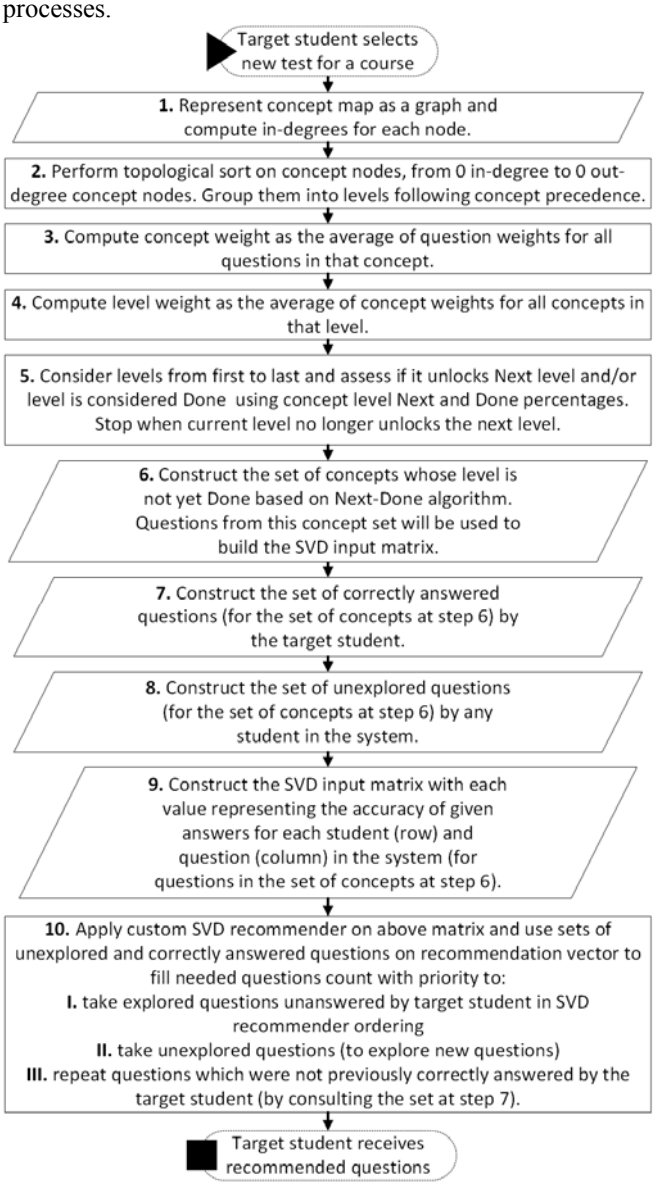

Figure 1. Detailed data workflow for recommending questions in a test

The workflow starts with the target student requesting a test by selecting one or more chapters. The current workflow is designed to work at chapter level, such that each chapter from the course needs its own concept map. This approach gives the opportunity of a finer granularity of concepts, such that we may end up with several manageable concept maps for a course instead of having one large concept map for the whole course.

1. The concept map for the selected chapter is retrieved and stored as a graph and in-degree is computed for each concept node. The in-degree will be later used for ordering and splitting concepts into levels.

2. A topological sort for the concept nodes is

performed, starting from the 0 in-degree nodes to the 0 out-degree ones. Concepts are grouped in levels ensuring no concept depending on a previous one is on the same level as its predecessor and its predecessor is placed in a lower level. Visually, this is equivalent to starting with the root nodes and performing a Breath-first search on the graph to find the levels as stated before.

3. For each concept, we compute the weight. This will result in each concept having a weight between 0 and 1, with 0 being automatically assigned to uncovered concepts. A value closer to 0 means the student answered most questions incorrectly, while a value closer to 1 means the student answered most questions correctly.

4. For each level, we compute the level weight as the average concept weight of concepts in that level. This will also result in each level having a weight between 0 and 1. A value closer to 0 suggests most questions from concepts in that level have been answered incorrectly and a value closer to 1 suggests most questions from concepts in that level have been answered correctly.

5. For each level of concepts at step 2, we assess, based on the level weight, if the level is Done (weight more than 75%) and if it unlocks the next level (weight more than 50%). We stop when the current level does not unlock any other level. This is done for determining the set of concepts from which the recommender may select questions at the next step.

6. Based on a Done/Next mechanism, we determine the set of concepts from levels that are not yet Done by the student. The *Next mechanism* allows the student to receive questions from the next level after it reaches 50% weight of the current level. The *Done mechanism* ensures that the student is levelled-up (i.e., receives questions only from the next level) when at least 75% of the concepts in a level are mastered (i.e., answered correctly). Furthermore, once the level is Done, the student will no longer receive questions from concepts belonging to the Done level.

7. We construct the set of correctly answered questions by the target student. This ensures that, when selecting questions in a level, questions which were correctly answered are no longer repeated. This way, the student only receives incorrectly answered questions after the full set of questions for a level was explored, allowing them to reassess their knowledge and, eventually, level up.

8. We construct the set of unexplored questions by any student in the system. This ensures that newly added questions by the professor are considered.

9. We construct the input matrix $M = (m_{(i,j)})$, of M x N dimension, for the SVD recommender for all students and questions from the set of concepts at step 6. M represents the number of students and N, the number of explored questions. Values $m_{(i,j)}$ are in range [0, 1] and represent the accuracy of student *i* when answering question *j* (a measure computed by normalizing the average grade of all answers given by student *i* for question *j* by the maximum grade for that question). This is used by the SVD algorithm to produce the final recommendation vector which, along with data at steps 7 and 8, is processed as follows in the next (last) step.

10. We apply our custom SVD recommender on matrix $M = (m_{(i,j)})$ at step 9 and use the sets at steps 7 and 8 to select the required number of questions for a test with a rule priority. The rule priority is:

I. Take explored questions (present in the input matrix), but unanswered by the target student, in the order selected by the SVD algorithm.

II. Take unexplored questions for exploring new content.

III. Repeat questions previously answered incorrectly by the target student.

The workflow reaches its final when the student receives the recommended questions.

B. Defined Weights

Question Weight. A question's weight is defined as the average grade of that question when answered by students in the system and is a number between 0 and 1 (see Equation 1).

$$W_Q = \frac{\sum_{i=1}^{S_q} Q_i}{S_q} \quad (1)$$

where, Q_i = average question grade for i^{th} student and S_q = number of students that answered the question.

A value closer to 0 is an indication of a very difficult question (answered correctly by a few students), while a value closer to 1 is an indication of a very easy question.

Concept Weight. A concept's weight is defined as the average weight of all questions in that concept and is a number between 0 and 1 (see Equation 2).

$$W_C = \frac{\sum_{i=1}^{q_c} WQ_i}{q_c} \quad (2)$$

where, WQ_i= weight for i^{th} question and q_c = number of questions in the concept.

This weight shows if the questions from a concept are generally harder, when the weight is closed to 0, or easier, when the weight is closed to 1. A middle value (0.5) indicates a more balanced concept, with either hard and easy questions or questions of a similar (medium) difficulty.

Level Weight. A level's weight is defined as the average weight of all concepts in that level and is a number between 0 and 1 (see Equation 3).

$$W_L = \frac{\sum_{i=1}^{C_L} WC_i}{C_L} \quad (3)$$

where, WC_i = weight for i^{th} concept and C_L = number of concepts in the level.

The level weight can emphasize critical levels, levels of a higher difficulty encountered by students in the learning process of a course. A lower value (closed to 0) shows a difficult level to pass, while a higher value (closer to 1) shows an easier one.

A few more weights defined per test for comparing consecutive tests are described below and will also be shown in the experimental results section.

Average Weight per Test. A test's average weight is defined as the average weight of all questions in that test and is a number between 0 and 1 (see Equation 4).

$$W_T = \frac{\sum_{i=1}^{q_T} WQ_i}{q_T} \quad (4)$$

where, WQ_i = weight for i^{th} question in the test and q_T =

number of questions in the test.

The average weight per test is a measure of the difficulty of the test. Lower values (denoting lower question weights) indicate a harder test while higher values (denoting higher question weights) indicate an easier one.

Answer Correctness per Test. A test's answer correctness is defined as the percentage of correctly answered questions in that test by the student and is a number between 0 and 100 (see Equation 5).

$$\%W_{acT} = \frac{ca_{qT}}{q_T} \cdot 100 \tag{5}$$

where, ca_{qT} = number of correctly answered questions in the test and q_T = number of questions in the test.

The answer correctness weight per test shows the percentage of correctly answered questions from a test. A higher percentage denotes more correctly answered questions and a lower percentage, less correctly answered questions (more questions were incorrectly answered).

Student Weight per Test. A student's weight is defined as the sum of signed weights (positive for correct answers, negative for incorrect) for all questions answered in that test by the student divided by the number of questions in the test and, since it results in a number between -1 and 1, it is normalized to a number between 0 and 1 by adding 1 and dividing by 2 (see Equation 6).

$$SW_T = \frac{\frac{\sum_{i=1}^{q_T} SWQ_i}{q_T} + 1}{2} \tag{6}$$

where, SWQ_i = signed question weight for i^{th} question in the test and q_T = number of questions in the test.

The student weight per test is a measure of how well the student performed in a test. Higher values (closed to 1) mean that the student performed well, while lower values (closer to 0) show that the student struggled in a test.

Student Relative Knowledge Weight. A student's relative knowledge weight is defined as the weight sum of correctly answered questions in taken tests divided by the weight sum of all answered questions in taken tests. It is a number between 0 and 1 (see Equation 7).

$$SKW_r = \frac{\sum_{i=1}^{a_{qT}} WAQ_i}{\sum_{i=1}^{t_{qT}} WQ_i} \tag{7}$$

where, a_{qT} = number of correctly answered questions in all taken tests, t_{qT} = number of questions in all taken tests, WAQ_i = weight for i^{th} correctly answered question in a taken test and WQ_i = weight for i^{th} question in a taken test.

Student Absolute Knowledge Weight. A student's absolute knowledge weight is defined as the weight sum of correctly answered questions in taken tests divided by the weight sum of all questions available for tests. It is a number between 0 and 1 (see Equation 8).

$$SKW_a = \frac{\sum_{i=1}^{a_{qT}} WAQ_i}{\sum_{i=1}^{t_{qT}} WQ_i} \tag{8}$$

where, a_{qT} = number of correctly answered questions in all taken tests, q_T = number of questions available for tests, WAQ_i = weight for i^{th} correctly answered question in a taken test and WQ_i = weight for i^{th} question in the pool of questions for tests.

IV. Experimental Results

The proposed workflow described in the previous section has been tested on second year students of the *Data Structures and Algorithms course* on the *Graphs Data Structures* chapters. Data for two consecutive generations (2018 and 2019) will be presented in this chapter, of which the 2018 students used a simplified workflow (without taking into consideration concept levels), while the 2019 ones benefited from the trained data in terms of question exploration from the previous generation and used the proposed workflow.

A. Students

In our experiments, a number of **140** students have taken **748** tests (with an average of approx. **5.3** tests per student) in **2018**, while **117** students have taken **1068** tests (with an average of approx. **9.1** tests per student) in **2019**.

B. Questions

In both test scenarios, the same pool of **98** questions divided into **11** concepts was available for the Tesys recommender system for selection. These concepts were linked together in a graph called concept map.

Table I depicts the concepts together with the number of questions for each concept of the *Graphs* subject. These can also be seen in each concept map from the timeline of a student presented in Figure 3 as nodes (the concepts) and the second number in the caption of each concept node (number of questions for the concept), respectively. This timeline will be presented in a later sub-section.

TABLE I. Concepts and Number of Questions per Concept

Concept	No. of questions	Concept	No. of questions
Representations (R)	15	Bellman-Ford (BF)	8
GSearch (GS)	5	APSP	10
BFS	5	SSSP/APSP Wrapup (SAW)	7
DFS	11	MST	9
SSSPFundamentals (SP)	8	Misc	8
Dijkstra (DJ)	12		

C. Tests

For students in the year 2018 data set, each test consisted of **10** questions, while students in the year 2019 data set received tests of **8** questions. Each question has an equal weight in a test and student grades are normalized between 0 and 10. Figure 2 displays a bar chart with the test results' distribution in 10-grade ranges (0-10 divided into equal intervals) for both years comparatively. The green bars represent grades for the tests taken by 2018 students, and the blue bars represent grades for the tests taken by 2019 students.

As can be seen from the figure, the blue tests from 2019 form a more bell-shaped curve, showing a proper distribution of grades with only a few students below grade 5 (144 tests out of 1068, approx. 13.5%), while the green tests from 2018 form a more sinusoidal shape, with more values below grade 5 (146 tests out of 748, approx. 19.5%), denoting harder questions, from more advanced concepts, were probably selected from the beginning, in the first tests taken by the students. Almost a double number of students

took grades between 0 and 1 at tests in 2018, comparatively to tests taken in 2019 (26 in 2018 versus 14 in 2019), since the recommender system did not use a concept map and Next/Done mechanism in 2018, so questions from advanced concepts were given in the initial tests.

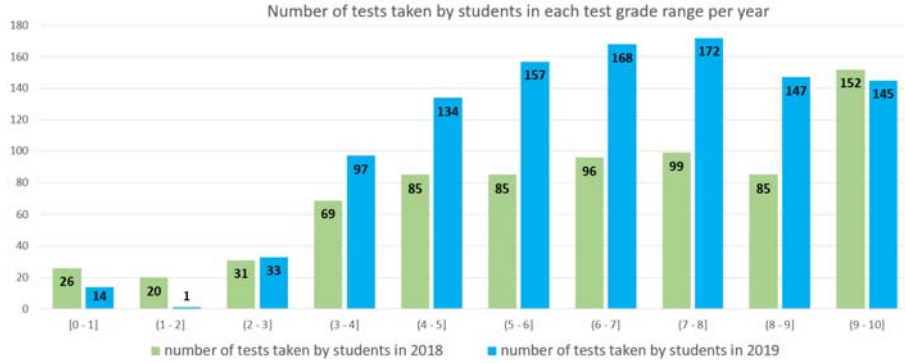

Figure 2. Bar chart representing the number of tests taken by students in each test grade range per year

Another aspect to note is the more significant number of tests with high grades (between 9 and 10) compared to the other grades for students who took tests in 2018. This aspect may be due to the cold-start problem, which caused a random selection of the questions in the initial tests. The students from 2019 benefited from the SVD recommender being more aware of question difficulty after the tests from 2018 explored most questions.

D. Student test activity tracking

Additional tables and a concept map evolution have been implemented to track and assess the student's knowledge level. These will be shown for a specific student.

Our target student is part of the group from 2019, has taken 20 tests and has an average grade of 7.63. The following sub-sections will describe the before-mentioned student testing activity and knowledge tracking features.

E. Student overall testing activity

The student's overall activity can be tracked using a table showing question correctness (along with their concept) given in a test. The correctness is displayed using a green tick for questions answered correctly, and a red cross for questions answered incorrectly. A black dash is used to denote that the question was not selected for that specific test. Figure 4 shows a part of that table for our target student, highlighting questions in tests 7, 8 and 9.

In this table, concepts are topologically sorted from left to right. That is, concepts from the left side of the list are prerequisites and should be mastered before the ones from the right for ensuring a proper learning curve. The table offers a view for monitoring selected questions where, intuitively, the "batch" of green ticks should move from left to right as the student progresses. Red crosses should be gradually replaced by green ticks, therefore questions that were previously answered incorrectly by the student are later recommended in next tests until a correct answer is provided. If the student misses to get over 75\% of a level, we may check that questions for which the student provided continuously wrong answers are selected from the same concepts that were not mastered by the student.

F. Student evolution over time

The student's evolution over time has been captured using a timeline of concept maps after each test, displaying the fill factor of each concept using labels (number of questions answered correctly/number of total questions in the concept) and color shades ranging from red (0-33%) to orange-yellow (34-66%) to green (67-100%). Figure 3 displays concept maps after four tests in the timeline of our target student (tests 5, 10, 15 and 20). Visually, this provides intuitive feedback on how the student progressed, which are the concepts that the student managed to master and which are the ones that still need attention.

Test weights before/after consecutive tests of a student. Figure 5 displays questions included in tests 7, 8 and 9 of the target student (along with their weights and associated concepts). Question weight for questions given in a test is highlighted in blue text, with a green tick for questions answered correctly and a red cross for questions answered incorrectly. In this table, questions in each concept are ordered descending by their weight, from easier to harder questions. This facet presentation allows for the evaluation, in terms of question difficulty, of the selected questions for consecutive tests. The intuition is that, if a student answers correctly to a test, the SVD recommender will select more difficult questions for the next test. On the other hand, if the student answers incorrectly to a test, the SVD recommender will most likely select easier questions for the next test.

Based on data in this table, the two weights from Equations (4) and (6) can be computed for tests 7-9 of our target student; the values are highlighted in Table II.

TABLE II. TEST AND STUDENT WEIGHTS FOR EXPERIMENTAL TESTS 7-9

W_{T7}	W_{T8}	W_{T9}	SW_{T7}	SW_{T8}	SW_{T9}
0.6	0.52	0.61	0.72	0.52	0.68

Equation (7) represents the test weight and is a measure of the difficulty of the test. Lower question weights will result in a harder test with lower test weight. In our example, test 8 is the hardest test of the three, followed by tests 7 and 9 of similar difficulty.

Equation (8) represents the student's weight per test and is a measure of how well the student performed in a test. Higher weights mean that the student performed well, while lower weights show that the student struggled in a test.

In our example, the weights show that our target student performed better in test 7 (with 7/8 questions correctly answered), followed by test 9 (with 6/8 questions correctly answered) and test 8 having the lowest weight (a lower performance, with 4/8 questions correctly answered).

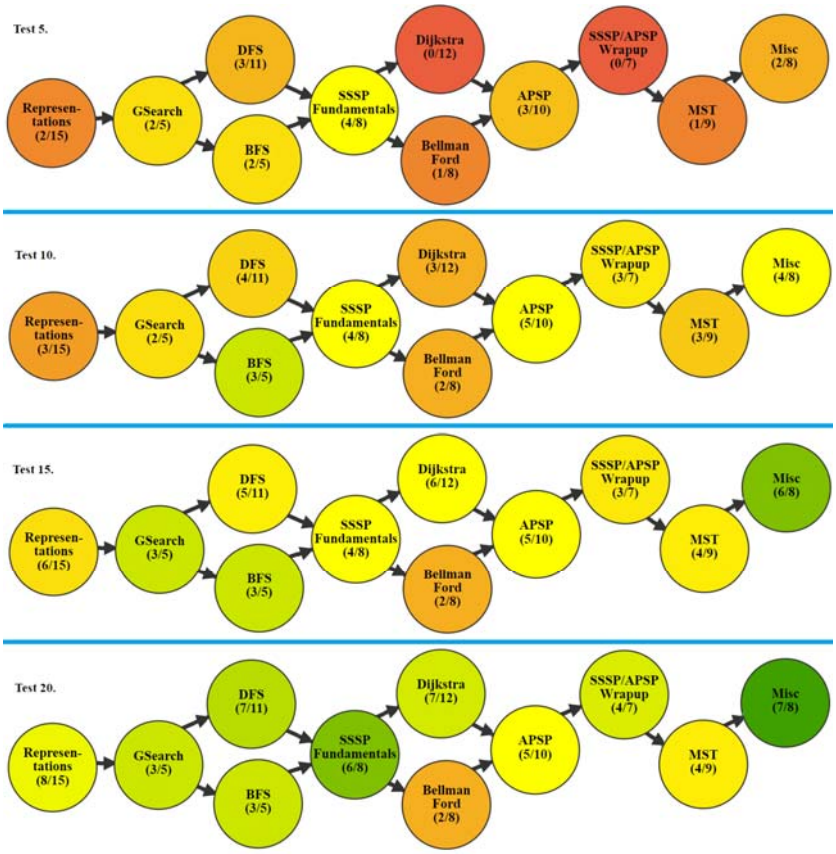

Figure 3. Concept map timeline after 4 tests taken by a student

Test/Question	303	307	314	319	320	334	352	338	339	340	343	346	347	360	362	364	369	372	373	375	386	387
Concept	R	R	BFS	DFS	DFS	SP	BF	DJ	DJ	DJ	DJ	DJ	DJ	APSP	APSP	APSP	SAW	SAW	SAW	MST	Misc	Misc
Test #7	–	✓	–	–	–	✓	✓	–	–	✓	–	✗	✓	–	✓	–	–	–	–	–	✓	–
Test #8	–	–	–	–	–	–	–	✗	✓	–	✗	–	–	✓	–	–	–	✗	–	✓	–	✓
Test #9	✗	–	✓	✗	✓	–	–	–	–	–	✓	–	–	–	–	✓	–	✓	✓	–	–	–

Figure 4. Testing activity for 3 tests (8 questions per test) taken by a student

Test/Question	307	303	314	320	319	334	352	343	339	340	346	338	347	364	362	360	373	372	369	375	386	387
Concept	R	R	BFS	DFS	DFS	SP	BF	DJ	DJ	DJ	DJ	DJ	DJ	APSP	APSP	APSP	SAW	SAW	SAW	MST	Misc	Misc
Test #7	0.67 [62 / 93] ✓	0.51 [42 / 83]	0.49 [47 / 96]	0.72 [60 / 83]	0.53 [49 / 93]	0.48 [41 / 85] ✓	0.67 [64 / 96] ✓	0.7 [63 / 90]	0.74 [75 / 101]	0.71 [76 / 107] ✓	0.66 [63 / 96] ✗	0.61 [51 / 84]	0.51 [47 / 92] ✓	0.82 [77 / 94]	0.68 [67 / 98] ✓	0.36 [33 / 92]	0.8 [90 / 113]	0.34 [30 / 88]	0.27 [30 / 113]	0.81 [62 / 77]	0.43 [42 / 97] ✓	0.32 [30 / 95]
Test #8	0.67 [62 / 93]	0.51 [42 / 83]	0.49 [47 / 96]	0.72 [60 / 83]	0.52 [49 / 94]	0.48 [41 / 85]	0.67 [64 / 96]	0.69 [63 / 91] ✗	0.75 [76 / 102] ✓	0.71 [76 / 107]	0.66 [63 / 96]	0.6 [51 / 85] ✗	0.51 [47 / 92]	0.82 [77 / 94]	0.68 [67 / 98]	0.37 [34 / 93] ✓	0.8 [90 / 113]	0.34 [30 / 89] ✗	0.26 [30 / 114] ✗	0.81 [63 / 78] ✓	0.43 [42 / 97]	0.32 [31 / 96] ✓
Test #9	0.67 [62 / 93]	0.5 [42 / 84] ✗	0.49 [48 / 97] ✓	0.73 [61 / 84] ✓	0.52 [49 / 95] ✗	0.48 [41 / 85]	0.67 [64 / 96]	0.7 [64 / 92] ✓	0.75 [76 / 102]	0.71 [76 / 107]	0.66 [63 / 96]	0.6 [51 / 85]	0.51 [47 / 92]	0.82 [78 / 95] ✓	0.68 [67 / 98]	0.37 [34 / 93]	0.8 [91 / 114] ✓	0.34 [31 / 90] ✓	0.26 [30 / 114]	0.81 [63 / 78]	0.43 [42 / 97]	0.32 [31 / 96]

Figure 5. Questions and their weights after 3 consecutive tests of a student

As a further observation, the middle test 8 also contained harder questions than the previous test 7 in order to challenge the student but was met with a lower performance which ultimately led to the recommendation mechanism returning the student to a more accessible test difficulty (similar to test 7 is test 9). This points out that, by also considering the student's current level of knowledge for a concept, the custom SVD algorithm can adjust the test difficulty based on the student's performance. The relative and absolute knowledge weight of the student, considering the student took only three tests (tests 7-9), can be computed using Equations (7) and (8); the obtained values are listed in Table III. From the student's relative knowledge weight, we can assess whether the student's knowledge is improving or not, as we see that $SKW_{r8} < SKW_{r7}$, so the student is performing worse in the second test (test 8) relative to the first one (test 7). This can clearly be seen by the number of wrong answers given, which increased from 1 to 4.

TABLE III. RELATIVE AND ABSOLUTE WEIGHTS FOR TESTS 7-9

SKW_{r7}	SKW_{r8}	SKW_{a7}	SKW_{a8}
0.86	0.71	0.29	0.46

From the student's absolute knowledge weight, we can assess how much knowledge the student gained from answering the questions and how far this is from the total knowledge gain from answering all questions correctly. Our student gained 29% knowledge from the first test (test 7) and 46% from the second one (test 8).

V. CONCLUSIONS AND FUTURE WORKS

In this paper, we presented a recommender system based on SVD algorithm used in Tesys e-Learning platform to provide better testing experience for students at the Faculty of Automation, Computers and Electronics in the University of Craiova. Providing relevant questions when students take tests is an important task as it influences their engagement.

This paper is a follow-up of three other papers which described the system and a short validation approach, and we focus in this one on the newly developed features.

Based on previous experience, we implemented weights for both students and tests to provide more relevant recommendations. In the experimental results, we present an overview of the system and how it works after the latest features were implemented by applying it on the tests taken during a semester at Data Structures subject. To obtain more relevant results, we also used data from the previous year of study, as this approach will reduce the bias of the algorithm.

As future work we plan to improve the recommender system with deep learning. We need to evaluate the benefits of using Long Short Term Memory networks in adaptive testing environments because deep learning models tend to provide better and better results for recommender tasks.

The trade-off for this approach is that, in our case, we update the model very fast for each test taken by a student and, on the other side, updating the deep learning model takes longer periods of time and these models needs to be reevaluated after each retraining. Based on this assumption, a deep learning model may be feasible for training based on previous results, but it will not update in accordance with the results of the students from current year of study.

REFERENCES

[1] O. Teodorescu, S. P. Popescu, M. Mocanu and M. C. Mihaescu, "Custom Validation Procedure for Tesys Recommender System," International Conference on Software, Telecommunications and Computer Networks (SoftCOM), Split, Croatia, pp. 1-6, 2019. doi:10.23919/SOFTCOM.2019.8903913

[2] C. M. Mihăescu, O. M. Teodorescu, P. Ş. Popescu and M. L. Mocanu, "Learning analytics solution for building personalized quiz sessions," 18th International Carpathian Control Conference (ICCC), Sinaia, 2017, pp. 140-145, 2017. doi:10.1109/CarpathianCC.2017.7970386

[3] O. Teodorescu, P. Popescu, C. Mihaescu, "Taking e-assessment quizzes - A case study with an SVD based recommender system," 19th International Conference, Proceedings, Part I. Madrid, Spain, November 21–23, 2018. doi:10.1007/978-3-030-03493-1_86

[4] D. D. Burdescu, M.C. Mihaescu, "TESYS: e-Learning Application Built on a Web Platform," ICE-B, 315-318, 2006. doi:10.5220/0001424803150318

[5] P. P. M. de Rijk, "A one-sided Jacobi algorithm for computing the singular value decomposition on a vector computer," SIAM Journal on Scientific and Statistical Computing, pp. 359-371, 1989. doi:10.1137/0910023

[6] K. Pliakos, S. H. Joo, Y. Y. Park, F. Cornillie, C. Vens, W. Van den Noortgate, "Integrating machine learning into item response theory for addressing the cold start problem in adaptive learning systems," Computers & Education, pp. 91-103, 2019. doi:10.1016/j.compedu.2019.04.009

[7] W. J. van der Linden, R. K. Hambleton, "Handbook of Modern Item Response Theory," Springer Science & Business Media, pp. 51-65, 2013. doi:10.1007/978-1-4757-2691-6

[8] T. Z. H. T. Petra and J. A. A. Moh, "Investigating reliability and validity of student performance assessment in Higher Education using Rasch Model," In Journal of Physics: Conference Series, vol. 1529, no. 4, IOP Publishing, 2020. doi:10.1088/1742-6596/1529/4/042088

[9] F. Martínez-Plumed, R. B. Prudencio, A. Martínez-Uso, J. Hernandez-Orallo, "Making sense of item response theory in machine learning," In Proceedings of the Twenty-second European Conference on Artificial Intelligence, pp. 1140-1148, 2016. doi:10.3233/978-1-61499-672-9-1140

[10] Z. Ren, X. Ning, H. Rangwala, "Grade prediction with temporal course-wise influence," Proceedings of the 10th International Conference on Educational Data Mining, arXiv: 1709.05433, 2017

[11] A. Segal, K. Gal, G. Shani, B. Shapira, "A difficulty ranking approach to personalization in E-learning," International Journal of Human-Computer Studies, pp. 261-272, 2019. doi:10.1016/j.ijhcs.2019.07.002

[12] T. Barnes, "The Q-matrix method: Mining student response data for knowledge," In American Association for Artificial Intelligence 2005 Educational Data Mining Workshop, Pittsburgh, PA: AAAI Press, pp. 1-8, 2005

[13] K. K. Tatsuoka, "Rule space: An approach for dealing with misconceptions based on item response theory," Journal of educational measurement, vol. 20, no. 4, pp. 345–354, 1983. doi:10.1111/j.1745-3984.1983.tb00212.x

[14] A. K. Menon, E. Charles, "Fast algorithms for approximating the singular value decomposition," ACM Transactions on Knowledge Discovery from Data, 2011, pp. 1-36, doi:10.1145/1921632.1921639

[15] X. Yuan, L. Han, S. Qian, G. Xu, H. Yan, "Singular value decomposition based recommendation using imputed data," Knowledge-Based Systems, Vol. 163, pp. 485-494, 2019. doi:10.1016/j.knosys.2018.09.011

[16] D. Ben-Shimon, L. Rokach, B. Shapira, "An ensemble method for top-N recommendations from the SVD," Expert Systems with Applications, pp. 84-92, 2016. doi:10.1016/j.eswa.2016.07.028

[17] A. N. Nikolakopoulos, V. Kalanzis, E. Gallopoulos, J. D. Garofalakis "EigenRec: generalizing PureSVD for effective and efficient top-N recommendations," Knowledge and Information Systems, Vol. 58, pp. 59-81, 2019. doi:10.1007/s10115-018-1197-7

[18] P. Matuszyk, J. Vinagre, M. Spiliopoulou, A. Jorge, J. Gama, "Forgetting techniques for stream-based matrix factorization in recommender systems," Knowledge and Information Systems, pp. 275-304, 2017. doi:10.1007/s10115-017-1091-8

[19] L. Oscar, J. Diez, A. Bahamonde, "A peer assessment method to provide feedback, consistent grading and reduce students' burden in massive teaching settings," Computers & Education, pp. 283-295, 2018. doi:10.1016/j.compedu.2018.07.016

[20] S. Cooper, S. Mehran, "Reflections on stanford's moocs," Communications of the ACM, pp. 28-30, 2013. doi:10.1145/2408776.2408787

[21] M. Singh, "Scalability and sparsity issues in recommender datasets: a survey," Knowledge and Information Systems, pp. 1-43, 2020. doi:10.1007/s10115-018-1254-2

[22] Y. Tai, J. Yang, L. Luo, F. Zhang, J. Qian, "Learning discriminative singular value decomposition representation for face recognition," Pattern Recognition, pp. 1-16, 2016. doi:10.1016/j.patcog.2015.08.010

[23] A. T. Corbett, J. R. Anderson, "Knowledge tracing: Modeling the acquisition of procedural knowledge," User modeling and user-adapted interaction, pp. 253-278, 1994. doi:10.1007/BF01099821

[24] R. S. J. Baker, A. T. Corbett, V. Aleven, "More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing," International Conference on Intelligent Tutoring Systems. Springer, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-69132-7_44

[25] W. J. Hawkins, N. T. Heffernan, R. S. J. D. Baker, "Learning Bayesian knowledge tracing parameters with a knowledge heuristic and empirical probabilities," International Conference on Intelligent Tutoring Systems. Springer, Cham, 2014. doi:10.1007/978-3-319-07221-0_18

[26] J. Beck, "Difficulties in inferring student knowledge from observations (and why you should care)," Educational Data Mining: Supplementary Proceedings of the 13th International Conference of Artificial Intelligence in Education, 2007.

[27] J. D. Novak, D. B. Gowin, "Learning how to learn," Cambridge University Press, 1984. doi:10.1017/CBO9781139173469

[28] A. Grubisic, et al. "Knowledge tracking variables in intelligent tutoring systems," Proceedings of the 9th International Conference on Computer Supported Education-CSEDU, Vol. 1, 2017. doi:10.5220/0006366905130518

[29] A. Trifa, H. Aroua, L. C. Wided, "Knowledge tracing with an intelligent agent, in an e-learning platform," Education and Information Technologies, 2019. doi:10.1007/s10639-018-9792-5

[30] L. Blerina, K. Kolomvatsos, S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," Expert Systems with Applications, pp. 2065-2073, 2014. doi:10.1016/j.eswa.2013.09.005

[31] P. Jung Yeon, S.-H. Joo, F. Cornillie, H. L. J. van der Maas, W. van den Noortgate "An explanatory item response theory method for alleviating the cold-start problem in adaptive learning environments," Behavior Research Methods, no. 2, pp. 895-909, 2019. doi:10.3758/s13428-018-1166-9