Intelligent Tutoring System: Outer Loop

Harsimran Bath (COMS W3995 sec 028) and Weiman Sun (COMS E6901 sec 028)

February 2, 2018

Contents

1 Abstract

In this proposal, we will be adding intelligent capabilities to the outer loop of SAGE (Socially Addictive Gameful Engineering) to automatically recommend games the students can play based on various features. Furthermore, the recommender will recommend games to the instructor that the students can play to progress in the curriculum. The work will be part of the Gameful Intelligent Tutoring (GIT) Epic in TFS. The feature being proposed is ITS Outer Loop.

2 Introduction

Computation Thinking (CT) is an idea that has been floating around since the 1960s, when Alan Perlis first argued for the need for college students to learn "theory of computations." It has since matured fairly well in the college realm. However, its progression in the K-12 realm has been lacking, until Jeannette Wing published the influential article "Computation Thinking" in 2006. According to Wing, "computational thinking involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science."[8] Since then, a new interest has ignited among researchers to define a curriculum for K-12 students to learn computational concepts.

SAGE is a platform that infuses computational thinking and gamification to bring computer science to students. It is built to ensure students can learn computational concepts and instructors can teach, with the aid of a centralized platform.

In this project, we will be adding intelligent system capabilities to the outer loop. The outer loop is a process that runs when the users complete games and quests. It determines which games the student can pursue next. We will build a sophisticated recommendation engine to make recommendations for the next games the user can pursue. It will be an item- and user- based collaborative filtering system.

Furthermore, the outer loop will also offer recommendations to the instructors. When the instructors are managing their classrooms, the recommender will evaluate various features to recommend what games the instructor can add to the quest that would be beneficial to the students. In addition, the recommender will also recommend games in the quest that do not benefit the students and should be removed.

Finally, we will build a test harness to run and evaluate the recommendation algorithms. The harness will primarily be offline analyses on mock data, but would offer a way to evaluate the performance of the algorithms, as well as, tweaking the different weights.

3 Related Work

3.1 "The Behavior of Tutoring Systems[6]"

This paper explores the function of outer loops in intelligent tutoring systems (ITS). It begins by exploring the various aspects of the outer loop:

1. Task Domain: The skills being taught by the tutor)
2. Task: An activity within the domain.
3. Step: A user interface event within the task.
4. Knowledge Component: A domain-specific concept being taught as part of the task and/or step.

The outer loop essentially runs when the user completes a task and determines what task the user should handle next. There are four main methods to selecting tasks for a student:

1. The outer loop displays all the tasks and lets the user select which the one they want to work on next. This is to some extent similar to Khan Academy, where there is a sequence of assignments, but the student can jump around.
2. The tutor assigns tasks in a predetermined sequence. In this approach, the instructor will have predefined a sequence of tasks and the tutor would automatically assign the next task in the sequence.
3. The tutor assigns tasks from a unit's pool of tasks until the student masters the unit. In this approach, there is likely a mastery level to each unit that the student needs to achieve in order to move on to the next unit. Until the student does so, the tutor will continue assigning tasks from the same unit.
4. The last approach is macroadaptive learning. Essentially, the tutor tracks traits, such as learning styles and mastery of knowledge components, to assign tasks that would be best correlate with user's progress in mastery.

3.2 "A Personalized Courseware Recommendation System based on Fuzzy Item Response Theory"[3]

This paper proposes a personalized courseware recommendation system (PCRS) based on the fuzzy item response theory, which can recommend course with appropriate difficult level to the learner by taking into consideration the learner's understanding percentage for the learned courses. PCRS can dynamically estimate the learner's learning ability by collecting learner feedback information

after studying the recommended course. However, the feedback from students is not real-time; it is collected after students complete courses, and by asking them two simple questions:

1. The difficulty of the course
2. The learner's confidence in the course content

3.3 "Effective e-learning recommendation system based on self-organizing maps and associative mining"[7]

This research constructs a hybrid system with artificial neural network (ANN) and data-mining (DM) techniques. Firstly, ANN is used to classify the e-Learner types. Based on these e-Learner groups, DM will be used to elicit the rules of the best learning path. Then the users can obtain course recommendation from the path. Through this approach, an inclusive curriculum may be suggested to a group of learners with the same interest.

3.4 "User-Based and Item-Based Collaborative Filtering Recommendation Algorithms Design"[9]

This paper builds the recommendation system based on item-based and user-based collaborative filtering. It gives the insight of the details of collaborative filtering recommendation algorithms, such as computing the similarity between users and items and making the prediction based on the similarities. It also compares the results of two models.

3.5 "Combination of machine learning algorithms for recommendation of courses in E-Learning System based on historical data"[1]

In this paper, the authors use machine learning algorithms such as clustering and association rule algorithm in a Course Recommendation System. The approach combines clustering technique – Simple K-means and association rule algorithm – Apriori and finds the result. When it applied the association rule algorithm, it only takes into account the students' interests for courses.

3.6 "Hybrid User-Item Based Collaborative Filtering"[5]

This paper proposes a hybrid user-item based collaborative filtering algorithm to address the data sparsity and scalability challenge in traditional CF. It uses Case Based Reasoning (CBR) combined with average filling to handle the sparsity of data set. It clusters users sharing common attributes using Self-Organizing Map (SOM) network with Genetic algorithms (GA) optimization. At the time of recommendation, for a target user, the closest cluster is first identified for the user, then the traditional item based CF is performed within the respective user cluster.

3.7 "A Multi-Criteria Item-based Collaborative Filtering Framework"[2]

In the traditional Item-based Collaborative Filtering method, each user reveal her admiration about an item based on a single criterion. This paper considers user can rate an item from different aspects, and proposes a multi-criteria collaborative filtering (CF) algorithm. Specifically, the multi-criteria CF computes the similarity of each criteria using the cosine-based similarity in traditional CF, then performs average among these similarities. Another proposed approach to determining the overall rating is to find an aggregation function among criteria.

3.8 "Evaluating Collaborative Filtering Recommender Systems"[4]

This paper breaks down the different approaches in evaluating collaborative filtering recommender system. The paper shows key decisions to consider when building evaluation systems for recommenders. First, we have to decide between Live User Experiment and Offline Analyses. Offline analyses is quick and economical, but the natural sparsity of data sets limit the items that can be evaluated. Live User Experiment, on the other hand, evaluate real user performance, but are expensive and prone to user bias. In addition, we have to decide between natural data sets and synthesized data sets.

The paper also states the approaches to evaluating recommendations. The following approaches are proposed:

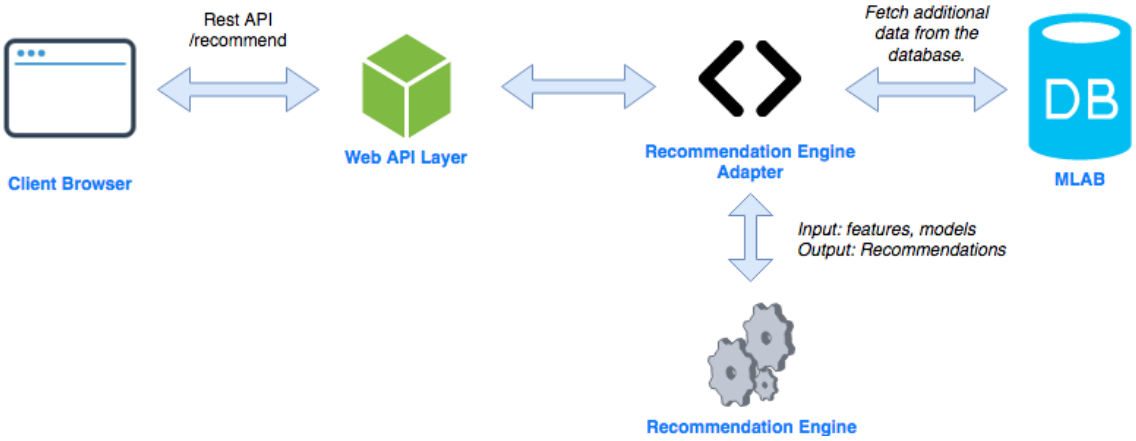
1. Predictive Accuracy Metrics: This approaches measures the recommender system's predicted ratings to the true user settings.

2. Classification Accuracy Metrics: This approach measures the frequency in which the recommender makes correct or incorrect predictions on an item. This approach is good for binary predictions.

3. Rank Accuracy Metrics: This approach measures how well the recommender orders the recommended items similar to the user's ordering.

4 Proposal

4.1 Architecture



We are considering this architecture for the recommendation engine, to ensure the engine is not tightly coupled with the system and can easily respond to the changes to the models in the overall system. Also, this architecture will ensure swift integration of the recommendation engine in the current system.

4.1.1 API Method(s)

The current Node Web Server can expose additional endpoints to get course recommendations for teachers and students. These endpoint methods can then interact with the Recommendation Engine Adapter (below) to retrieve recommendations. Context is key, therefore, the implementors of the recommendation systems can choose to expose multiple new endpoints to specifically request recommendations, or they can add the recommendation abilities to existing endpoints that already return courses on page load. The latter approach will aid in performance, reducing the number of network calls.

4.1.2 Recommendation Engine Adapter

The Recommendation Engine Adapter will be the layer of interaction between the recommendation engine and the rest of the system. The adapter helps the overall system respond the changes without affecting the engine itself, by being the middle man between the system and the recommendation engine. For example, if the models were to change in the system, features were to be added, or preprocessing was required, the implementor of the adapter would take care all of that. Therefore, the recommendation engine would be decoupled from the system, allowing a more dynamic and maintainable design.

4.1.3 Recommendation Engine

The Recommendation Engine is at the core of the design. Its only concern is taking in all the features and models, running the recommendation algorithms in real-time, and returning the output to the adapter. The adapter would be responsible for ensuring the input data structures of the engine are provided in the expected format, and the output format of the engine is handled correctly. The recommendation engine would be built on the principle of single responsibility, and hence, would only perform its core duties, decoupled from the rest of the system.

4.2 Outer Loop: Students

4.2.1 Student model

The student model would be comprised of multiple features (like mastery and duration of games) that would be used by the recommendation engine. For example, the student would have scores of each CT component (from hairball analysis) and duration of completing each CT component in a game. The above two consist of the learning percentage attribute. In addition, each game has difficulty. More student-specific features could potentially be added to enhance recommendations.

4.2.2 Item-based

The item-based recommendation engine will consider the CT components of each game, the difficulty of the game, and the mastery of students, among other features. Next, we will combine the duration for completing each CT component and the respective score to represent the learning percentage of this CT component. Using this learning percentage and the difficulty of the game, we can make a prediction about the mastery probability of other CT components in other games and the mastery probability of this game.

Also, a student cannot learn the same thing all the time; they need to be exposed appropriately to new CT components. Moreover, if they already did well in similar difficulty level games, they will need a slightly more challenging game to progress. Therefore, we will use a threshold to decide whether it is time for a student to play more difficult games, and we will figure out which new CT components that the student needs to learn next. After we figure out the new CT components, we will weight them and incorporate them into the master probability of CT components. In summary, for each game, we will take into consideration the mastery probabilities of its CT components and the appropriate difficulty and recommend them to students.

For example, suppose we have the following data in Table 1. In this case, student 1 has performed adequately on "loop" and "condition" CT Components, but performed poorly on algorithms. Therefore, we need to recommend an easier game for the student to further learn

Table 1: Item-based example

Student ID	Game ID	Game difficulty(1–10)	CT component	Duration(minutes)	Score
1	1	5	loop	5	75%
1	1	5	condition	4	80%
1	2	8	algorithms	10	40%
2	3	4	loop	2	90%
2	3	4	condition	3	95%

fundamental skills before moving to the harder games. On the other hand, Student 2 has done well on "loop" and "condition," both with difficulty 4. Therefore, now we can recommend a more difficult "loop" and "condition" game, or recommend games that contain new CT components.

4.2.3 User-based

The user-based recommendations will take into account the current student's metrics in comparison with other students' metrics on similar completed quests and games, to recommend games and quests the current student can pursue next. We group students based on the above comparison and employ weighted association rule mining technique. Association rule mining is a rule-based machine learning method aims to discover strong rules among variables in large databases. The rules in our scenario are the learning path of students. We think it would be helpful for a student to follow the learning path of those students that have similar behaviors. Thus, by utilizing the association rule mining, we can identify learning paths of groups of students, and then recommend courses to students based on what group they belongs to. For a given student, the similarity between them and another student in the group may vary. In light of this, we should weight the associations obtained from other students in their group differently. That is to say, we compute the similarities between the given student and other students in their group and rank the results. The higher the rank, the more weight we will add to the respective associations.

In order to get a unified solution of recommendation, we will combine the results of item-based and user-based. We will weight these two methods and return the final results.

4.3 Outer Loop: Teachers

4.3.1 Recommend Games

In addition to student game recommendations, the system will also recommend courses to the instructor. When the instructor is building the courses for their classrooms, the recommender will recommend courses that would be a useful addition to the class curriculum.

This part of the recommendation engine will take into account the features of each game in the

current quest (such as difficulty, learning outcomes, etc.) and the average mastery of all students (scores) in the classroom. It will then correlate these with all the games in the system and take into account their respective features. In this way, the recommendation engine will recommend games to the instructor that it believes the students could best work on to improve their skills.

To illustrate with an example, assume a system with 4 games:

1. Conditions: Difficulty of this game is easy
2. Loops 1: Difficulty of this game is medium
3. Easy Algorithms: Difficulty of this game is medium. Prerequisite: Loops, Conditions
4. Loops 2: Difficulty of this game is medium.

The current classroom has "Conditions" and "Loops 1" assigned to the students. If the students have on average achieved a good percentage on both "Loops 1" and "Conditions", the recommendation engine will recommend "Easy Algorithms" to add to the quest. If the students have not done well on "Loops 1" and "Conditions", then the system will not recommend the "Easy Algorithms" course. Instead, it recommend the "Loops 2" course so the students can further practice those essential skills, until they have gained mastery and ready for "Algorithms".

4.3.2 Remove Games not a Good Fit in Current Quest

This feature is slightly different from the recommendations engine. Here, we will help the teacher identify games in the current query that might not be a good fit for the students. Custom algorithm(s) will likely be utilized to verify if a game belongs in a quest. This will take into account features of each game in the quest (such as difficulty, outcomes, etc.) and the average student mastery (scores) of those.

To illustrate in another example, we will build on the example above. Suppose a classroom has "Loops 1," "Conditions 1," and "Easy Algorithms" assigned in the quest. And suppose, the average scores on "Loops 1" and "Conditions 1" are very low. This feature will deduce that the students are not prepared to play the "Easy Algorithms" game, and therefore, recommend the instructor remove it from the quest.

4.4 Outer Loop Evaluation: Test Harness

An important aspect of a recommendation engine is the ability to evaluate its performance. Therefore, we will be building a test harness to evaluate the performance of the recommendation engine algorithms under varying circumstances. The harness will perform offline analyses under a synthesized dataset. For this specific project, real-time analysis would not be possible, as we do not have

real world data available, nor is SAGE in production. The test harness will also have the ability to change weights of different features. This would be a good way to understand the weights of features on the algorithms.

The test harness will perform offline analyses on a mock data-set. We are still researching the best approach to generate a mock data and mock recommendations to compare with the recommender.

4.5 Future Work

4.5.1 Integration of Recommendations in the Affinity Space

Once the recommendation engine is built, it would be essential to integrate the recommendations in the user interface (UI) of SAGE. The UI will automatically fetch and display recommendations in relevant contexts, as well as, display information on how those items were recommended.

5 Timeline

Milestone	Estimated Date
Environment setup	2.3 – 2.7
Build Models (Features for recommendation engine)	2.8 – 2.17
Student recommendation engine: item-based	2.18 – 3.4
Student recommendation engine: user-based	3.5 – 3.18
Teacher recommendation engine	3.19 – 3.30
Teacher: Identify game in course not a good fit	3.31 – 4.8
Test Harness and Evaluation(mock data)	4.9 – 4.15
Integrated with mLab	4.16 – 4.21
Integrated with affinity space	4.22 – 4.28

6 References

References

[1] Sunita B Aher and LMRJ Lobo. Combination of machine learning algorithms for recommendation of courses in e-learning system based on historical data. *Knowledge-Based Systems*, 51:1–14, 2013.

[2] Alper Bilge and Cihan Kaleli. A multi-criteria item-based collaborative filtering framework. In *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on*, pages 18–22. IEEE, 2014.

[3] Chih-Ming Chen, Ling-Jiun Duh, and Chao-Yu Liu. A personalized courseware recommendation system based on fuzzy item response theory. In *e-Technology, e-Commerce and e-Service, 2004. EEE'04. 2004 IEEE International Conference on*, pages 305–308. IEEE, 2004.

[4] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[5] Nitin Pradeep Kumar and Zhenzhen Fan. Hybrid user-item based collaborative filtering. *Procedia Computer Science*, 60:1453–1461, 2015.

[6] Kurt Vanlehn. The behavior of tutoring systems. *International journal of artificial intelligence in education*, 16(3):227–265, 2006.

[7] David Wen-Shung Tai, Hui-Ju Wu, and Pi-Hsiang Li. Effective e-learning recommendation system based on self-organizing maps and association mining. *the electronic library*, 26(3):329–344, 2008.

[8] Jeannette Wing. Computational thinking. *Communications of the ACM*, pages 33–36, 2006.

[9] Guanwen Yao and Lifeng Cai. User-based and item-based collaborative filtering recommendation algorithms design. *University of California, San Diego.*