EduRank: A Collaborative Filtering Approach to Personalization in E-learning

Avi Segal, Ziv Katzir, Ya’akov (Kobi) Gal, Guy Shani
Dept. of Information Systems Engineering
Ben-Gurion University, Israel

Bracha Shapira
Dept. of Information Systems Engineering and
Telekom Innovation Laboratories
Ben-Gurion University, Israel

ABSTRACT
The growing prevalence of e-learning systems and on-line courses has made educational material widely accessible to students of varying abilities, backgrounds and styles. There is thus a growing need to accommodate for individual differences in such e-learning systems. This paper presents a new algorithm for personalizing educational content to students that combines collaborative filtering algorithms with social choice theory. The algorithm constructs a “difficulty” ranking over questions for a target student by aggregating the ranking of similar students, as measured by different aspects of their performance on common past questions, such as grades, number of retries, and time spent solving questions. It infers a difficulty ranking directly over the questions for a target student, rather than ordering them according to predicted performance, which is prone to error. The algorithm was tested on two large real world data sets containing tens of thousands of students and a million records. Its performance was compared to a variety of personalization methods as well as a non-personalized method that relied on a domain expert. It was able to significantly outperform all of these approaches according to standard information retrieval metrics. Our approach can potentially be used to support teachers in tailoring problem sets and exams to individual students and students in informing them about areas they may need to strengthen.

1. INTRODUCTION
Education is increasingly mediated by technology, as attested by the prevalence of educational software in schools and the explosion of on-line course opportunities. As a result, educational content is now accessible to student communities of varied backgrounds, learning styles and needs. There is thus a growing need for personalizing educational content to students in e-learning systems in a way that adapts to students’ individual needs [20, 1]. A popular approach towards personalization in e-learning is to sequence students’ questions in a way that best matches their learning styles or gains [2, 28].

This paper provides a novel algorithm for sequencing content in e-learning systems that directly creates a “difficulty ranking” over new questions. Our approach is based on collaborative filtering [6], which generates a difficulty ranking over a set of questions for a target student by aggregating the known difficulty rankings over questions solved by other, similar students. The similarity of other students to the target student is measured by their grades on common past question, the number of retries for each question, and other features. Unlike other uses of collaborative filtering in education, our approach directly generates a difficulty ranking over the test questions, without predicting students’ performance directly on these questions, which may be prone to error.1

Our algorithm, called EduRank, weighs the contribution of these students using measures from the information retrieval literature. It allows for partial overlap between the difficulty rankings of a neighboring student and the target student, making it especially suitable for e-learning systems where students differ in which questions they solve. The algorithm extends a prior approach for ranking items in recommendation systems [15], which was not evaluated on educational data, in two ways: First, by using social choice theory to combine the difficulty rankings of similar students and produce the best difficulty ranking for the target student. Second, EduRank penalizes disagreements in high positions in the difficulty ranking more strongly than low positions, under the assumption that errors made in ranking more difficult questions are more detrimental to students than errors made in ranking of easier questions.

We evaluated EduRank on two large real world data sets containing tens of thousands of students and about a million records. We compared the performance of EduRank to a variety of personalization methods from the literature, including the prior approach mentioned above as well as other popular collaborative filtering approaches such as matrix factorization and memory-based $K$ nearest neighbours. We also compared EduRank to a (non-personalized) ranking created by a domain expert. EduRank significantly outperformed all other approaches when comparing the outputted difficulty rankings to a gold standard.

The contribution of this paper is two-fold. First, we present a novel algorithm for personalization in e-learning according to the level of difficulty by combining collaborative filtering with social choice. Second, we outperform alternative solutions from the literature on two real-world data sets. Our approach can potentially be used to support both teachers and students, by automatically tailoring problem sets or exams to the abilities of individual students in the classroom, or by informing students about topics which they need to strengthen. Lastly, it can also augment existing ITS systems by integrating a personalized order over questions into the interaction process with the student.

1To illustrate, in the KDD cup 2010, the best preforming grade prediction algorithms exhibited prediction errors of about 25% [25]
2. BACKGROUND

In this section we briefly review relevant approaches and metrics in recommendation systems, social choice, and information retrieval.

2.1 Recommendation Systems and Collaborative Filtering

Recommender systems actively help users in identifying items of interest. For example, the prediction of users’ ratings for items, and the identification of the top-N relevant items to a user, are popular tasks in recommendation systems. A commonly used approach for both tasks is Collaborative Filtering (CF), which uses data over other users, such as their ratings, item preferences, or performance in order to compute a recommendation for the active user.

There are two common collaborative filtering approaches [6]: In the memory-based K nearest neighbor approach, a similarity metric, such as the Pearson correlation, is used to identify a set of neighboring users. The predicted rating for a target user and a given item can then be computed using a weighted average of ratings of other users in the neighborhood. In the model based approach, a statistical model between users and items is created from the input data. For example, the SVD approach [21] computes a latent feature vector for each user and item, such that the inner product of a user and item vectors is higher when the item is more appropriate for the user.

While rating prediction and top-N recommendations are widely researched, not many recommendation system applications require ranking. Thus, there were only a few attempts to use CF approaches to generate rankings. Of these, the most methods order items for target users according to their predicted ratings. In contrast, Liu et al. developed the EigenRank algorithm [15] which is a CF approach that relies on the similarity between item ratings of different users to directly compute the recommended ranking over items. They show this method to outperform existing collaborative filtering methods that are based on predicting users’ ratings.

Using the ratings of similar users, EigenRank computes for each pair of items in the query test set so-called potential scores for the possible orderings of the pair. Afterward, EigenRank converts the pair-wise potentials into a ranked list. EigenRank was applied to movie recommendation tasks, and was shown to order movies by rating better than methods based on converting rating predictions to a ranked list.

2.2 Social Choice

Social Choice Theory originated in economics and political science, and is dealing with the design and formal analysis of methods for aggregating preferences (or votes) of multiple agents [11]. Examples of such methods include voting systems used to aggregate preferences of voters over a set of candidates to determine which candidate(s) should win the election, and systems in which voters rank a complete set of candidates using an ordinal scale. One such approach which we use in this paper is Copeland’s method [8, 17] ordering candidates based on the number of pairwise defeats and victories with other candidates.

The Copland score for an alternative \( q_j \) is determined by taking the number of those alternatives that \( q_j \) defeats and subtracting from this number those alternatives that beat \( q_j \). A partial order over the items can then be inferred from these scores. Two advantages of this method that make it especially amenable to e-learning systems with many users (e.g., students and teachers) and large data sets are that they are quick to compute and easy to explain to users [22]. Pennock et al. [19] highlighted the relevance of social choice to CF and the importance of adapting weighted versions of voting mechanisms to CF algorithms. Our algorithm represents an application of this approach to e-learning systems.

2.3 Metrics for Ranking Scoring

A common task in information retrieval is to order a list of results according to their relevance to a given query [29]. Information retrieval methods are typically evaluated by comparing their proposed ranking to that of a gold standard, known as a “reference ranking”, which is provided by the user or by a domain expert.

Before describing the comparison metrics and stating their relevance for e-learning systems, we define the following notations: Let \( L = \binom{Q}{2} \) denote the set of all non ordered pairs in \( L \). Let \( \succ \) be a partial order of a set of questions \( L \). We define the reverse order of \( \succ \) over \( L \), denoted \( \preceq \) as a partial order over \( L \) such that if \( q_j \succ q_k \) then \( q_k \preceq q_j \). Let \( \succ_1 \) and \( \succ_2 \) be two partial orders over a set of questions \( L \), where \( \succ_1 \) is the reference order and \( \succ_2 \) is the system proposed order. We define an agreement relation between the orders \( \succ_1 \) and \( \succ_2 \) as follows:

- The orders \( \succ_1 \) and \( \succ_2 \) agree on questions \( q_j \) and \( q_k \) if \( q_j \succ_1 q_k \) and \( q_j \succ_2 q_k \).
- The orders \( \succ_1 \) and \( \succ_2 \) disagree on questions \( q_j \) and \( q_k \) if \( q_j \succ_1 q_k \) and \( q_k \succ_2 q_j \).
- The orders \( \succ_1 \) and \( \succ_2 \) are compatible on questions \( q_j \) and \( q_k \) if \( q_j \succ_1 q_k \) and neither \( q_j \succ_2 q_k \) nor \( q_k \succ_2 q_j \).

Given a partial order \( \succ \) over questions \( Q \), the restriction of \( \succ \) over \( L \subset Q \) are all questions \( \langle q_j, q_k \rangle \) such that \( q_j \succ q_k \) and \( q_j, q_k \in L \).

2.3.1 Normalized Distance based Performance

The Normalized Distance based Performance Measure (NDPM) [26, 24] is a commonly used metric for evaluating a proposed system ranking to a reference ranking. It differentiates between correct orders of pairs, incorrect orders and ties. Formally, let \( \delta_{\succ_1, \succ_2}(q_j, q_k) \) be a distance function between a reference ranking \( \succ_1 \) and a proposed ranking \( \succ_2 \) defined as follows:

\[
\delta_{\succ_1, \succ_2}(q_j, q_k) = \begin{cases} 
0 & \text{if } \succ_1 \text{ and } \succ_2 \text{ agree on } q_j \text{ and } q_k; \\
1 & \text{if } \succ_1 \text{ and } \succ_2 \text{ are compatible on } q_j \text{ and } q_k; \\
2 & \text{if } \succ_1 \text{ and } \succ_2 \text{ disagree on } q_j \text{ and } q_k.
\end{cases}
\]  

The total distance over all question pairs in \( L \) is defined as follows:

\[
\beta_{\succ_1, \succ_2}(L) = \sum_{\langle q_j, q_k \rangle \in L} \delta_{\succ_1, \succ_2}(q_j, q_k)  
\]  

Let \( m(\succ_1) = \arg\max_\succ \beta_{\succ_1, \succ}(L) \) be a normalization factor which is the maximal distance that any ranking \( \succ \) can have from a reference ranking \( \succ_1 \). The NDPM score \( s_{ND}(L, \succ_1, \succ_2) \) comparing a proposed ranking of questions \( \succ_2 \) to a reference ranking \( \succ_1 \) is defined as

\[
s_{ND}(L, \succ_1, \succ_2) = \frac{\beta_{\succ_1, \succ_2}(L)}{m(\succ_1)}
\]
Intuitively, the NDPM measure will give a perfect score of 0 to difficulty rankings over the set in $L$ that completely agree with the reference ranking, and a worst score of 1 to a ranking that completely disagrees with the reference ranking. If the proposed ranking does not contain a preference between a pair of questions that are ranked in the reference ranking, it is penalized by half as much as providing a contradicting preference.

The evaluated ranking is not penalized for containing preferences that are not ordered in the reference ranking. This means that for any question pair that were not ordered in the true difficulty ranking, any ordering predicted by the ranking algorithm is acceptable. Not penalizing unordered pairs is especially suitable for e-learning, any ordering predicted by the ranking algorithm is acceptable.

Similarly, we assume that the severity of errors in ranking questions depends on their position in the list. As we are interested in sequencing questions by order of difficulty, properly predicting how easy questions should be ordered is not as important as avoiding the presentation of a difficult question too early, resulting in frustration and other negative effects on the student learning process. Therefore, when evaluating a ranked list of questions, it is often important to consider the position of the questions in the ranked list. We would like to give different weights to errors depending on their position in the list.

To this end, we can use the AP correlation metric [27], which gives more weight to errors over items that appear at higher positions in the reference ranking. Formally, let $\succ_1$ be the reference ranking and $\succ_2$ be a proposed ranking over a set of items. The AP measure compares the order between each item in the proposed ranking $\succ_2$ with all items that precede it with the ranking in the reference ranking $\succ_1$.

For each $q_k, q_j \in L, k \neq j$, let the set $Z^k(L, \succ_2)$ denote all question pairs $(q_k, q_j)$ in $L$ such that $q_j \succ_2 q_k$. These are all the question pairs that are more difficult to the student than question $q_k$.

$$Z^k(L, \succ_2) = \{(q_j, q_k) | \forall q_j \neq q_k \text{ s.t. } q_j \succ_2 q_k \text{ and } q_j, q_k \in L\}$$

We define the indicator function $I^A(q_j, q_k, \succ_1, \succ_2)$ to equal 1 when $\succ_1$ and $\succ_2$ agree on questions $q_j$ and $q_k$.

Let $A^k(L, \succ_1, \succ_2)$ be the normalized agreement score between $\succ_2$ and the reference ranking $\succ_1$ for all questions $q_j$ such that $q_j \succ_1 q_k$.

$$A^k(L, \succ_1, \succ_2) = \frac{1}{k-1} \sum_{(q_j, q_k) \in Z^k(L, \succ_2)} I^A(q_j, q_k, \succ_1, \succ_2)$$

(5)

The AP score of a partial order $\succ_2$ over $L$ given partial order $\succ_1$ is defined as

$$s_{AP}(L, \succ_1, \succ_2) = \frac{1}{|L| - 1} \sum_{k \in L} A^k(L, \succ_1, \succ_2)$$

(6)

The $s_{AP}$ score gives a perfect score of 1 to systems where there is total agreement between the system proposed difficulty ranking and the reference ranking for every question pair above location $i$ for all $i \in L$. The worst score of 0 is given to systems where there is no agreement between the two ranked lists.

3. PROBLEM DEFINITION AND APPROACH

We now formalize our problem and the approach used. The “difficulty ranking problem” includes a target student $s_t$, and a set of questions $L_i$, for which the algorithm must predict a difficulty ranking $\succ_i$ over $L_i$. The predicted difficulty ranking $\succ_i$ is evaluated with respect to a difficulty reference ranking $\succ_1$ over $L_i$ using a scoring function $s(\succ_1, \succ_i, L_i)$.

To solve this problem, we take a collaborative filtering approach, which uses the difficulty rankings on $L_i$ of other students similar to $s_t$ to construct a difficulty ranking over $L_i$ for student $s_t$. Specifically, the input to the problem includes: (1) A set of students $S = \{s_1, s_2, ..., s_m\}$; (2) A set of questions $Q = \{q_1, q_2, ..., q_n\}$; (3) For each student $s_j \in S$, a partial difficulty ranking $\succ_j$ over a set of questions $T_j \subseteq Q$.

For every student $s_j \in S$ there are two disjoint subsets $T_j, L_j \subseteq Q$, where the difficulty ranking of $s_j$ over $T_j$ is known, and is a restriction of $\succ_j$ over all the questions in $Q$. Intuitively, for a a target student $s_t \in S, T_j$ represent the set of questions that the target student $s_t$ has already answered, while $L_j$ is the set of questions for which a difficulty ranking needs to be produced.

The collaborative filtering task is to leverage the known rankings of all students $s_j$ over $T_j$ in order to compute the required difficulty ranking $\succ_1$ over $L_i$ for student $s_t$.

4. THE EDFURANK ALGORITHM

We now present our EduRank algorithm for producing a personalized difficulty ranking over a given set of questions $L_i$ for a target student $s_t$. EduRank estimates how similar other student are to $s_t$, and then combines the ranking of the similar students over $L_i$ to create a ranking for $s_t$. There are two main procedures to the algorithm: computing the student similarity metric, and creating a difficulty ranking based on the ranking of similar users.

For comparing the target student $s_t$ to potential neighbors, we use the $s_{AP}$ metric to encourage greater similarity between students with high agreement in top positions in their respective rankings.

For aggregating the different students’ rankings to create a difficulty ranking for the target student, we use the Copeland method (Section 2.2). We treat each question as a candidate and look at the aggregated voting of neighbors based on their similarity metric. In our aggregated voting calculation, candidate $i$ beats candidate $j$ if the similarity normalized number of wins of $i$ over $j$ computed over all neighbors is higher than the similarity normalized number of loses. The Copeland method then computes for each candidate question the overall number of aggregated victories and aggregated defeats and ranks the candidates accordingly. Before presenting the algorithm we first define $\gamma(q_k, q_2, \succ_i)$ over question pairs $q_k, q_i$.
The algorithm computes a ranking score of the number of retries needed to solve the problem by the student, and we used the following features for each question: question ID, the answering attempts by 575 students, collected during 2005-2006.

We conducted experiments on two real world educational datasets. The first dataset was published in the KDD cup 2010 by the Pittsburg Science of Learning Center (PSLC) \(^2\) [13]. We used the Algebra 1 dataset from the competition, containing about 800,000 answering attempts by 575 students, collected during 2005-2006. We used the following features for each question: question ID, the number of retries needed to solve the problem by the student, and the duration of time required by the student to submit the answer.\(^3\) If the number of retries needed to solve the problem was 0, this means the students solved the problem on a first attempt (we refer to this event as “correct first attempt”).

The second dataset, which we call K12, is an unpublished dataset obtained from an e-learning system installed in 120 schools and used by more than 10,000 students. The records in this dataset were anonymized and approved by the institutional review board of the Ben-Gurion university. This dataset contains about 900,000 answering attempts in various topics including mathematics, English as a second language, and social studies. We used the following features for each question: question ID, the answer provided by the student and the associated grade for each attempt to solve the question. Unfortunately, this dataset does not contain time stamps for each provided response, so we cannot compute the duration of time until a question was answered.

5.2 Feature Selection for Difficulty Ranking

EduRank assumes that each student has a personal difficulty ranking over questions, as described in Section 3. In this section we show how we inferred this ranking from the features in the dataset. An obvious candidate for the difficulty ranking are the grades that the student got on each question. There are several reasons however as to why grades are an insufficient measurement of difficulty. First, in all questions in the PSLC dataset, the “Correct First Attempt” score is either 0 or 1. There were a number of multiple choice questions (between 3 and 4 possible answers) in the datasets, but the dichotomy between low and high grades was also displayed here. To understand this dichotomy, note that students were allowed to repeat the question until they succeeded. It is not surprising that after several retries most students were able to identify the correct answer. A zero grade for a question occurs most often when it was not attempted by the student more than once.

An alternative approach is to consider additional features in addition to grades (or correct first attempts), that are present in the datasets, and which correlate with the difficulty of the question for the individual student. Specifically, we assumed that questions that were answered correctly on a first attempt were easier for the student, while questions that required multiple attempts were harder. We also assumed that questions that required more solution time, as registered in the log, were more difficult to the students.

We realize that these two properties are not perfect indicators of question difficulty for the student. Indeed, it may occur in multiple choice questions that a student guessed the correct answer on the first attempt, even though the question was quite difficult. We also do not account for “gaming the system” strategies that have been modeled in past ITS work [4]. It may also be the case that the length of time reported by the system represents idle time for the student who was not even interacting with the e-learning software, or simply taking a break. However, as we show later in this section, these properties provide a reasonable estimation for the difficulty of the question.

We proceed to describe the following method for identifying the difficulty ranking. We begin by ranking questions by grades. In the PSLC dataset we use “correct first attempt” for this, and in the K12 dataset we find it more informative to use the grade that the student got on each question. In both datasets, and which correlate with the difficulty of the question for the individual student. Specifically, we assumed that questions that were not attempted by the student more than once.

5. EMPIRICAL EVALUATION

We now describe a set of experiments comparing EduRank to other algorithms on the difficulty ranking problem. We describe the datasets that were used and our method for defining a difficulty ranking, then we discuss the performance of the various algorithms.

5.1 Datasets

We conducted experiments on two real world educational datasets. The first dataset was published in the KDD cup 2010 by the Pittsburg Science of Learning Center (PSLC) \(^2\) [13]. We used the Algebra 1 dataset from the competition, containing about 800,000 answering attempts by 575 students, collected during 2005-2006. We used the following features for each question: question ID, the number of retries needed to solve the problem by the student, and the duration of time required by the student to submit the answer.\(^3\) If the number of retries needed to solve the problem was 0, this means the students solved the problem on a first attempt (we refer to this event as “correct first attempt”).

\(^2\)https://pslcdatashop.web.cmu.edu/KDDCup

\(^3\)Note there were other features in this data-set that were not used in the study.
student got on her first attempt. After ranking by grade, we break ties by using the number of attempts that the student took before submitting the correct answer. When the student did not achieve a correct answer we use all the attempts that the student has made. Then, we break ties again on the PSLC dataset using the elapsed time.

To demonstrate that in general, these properties provide a reasonable estimation for the difficulty of the question, Figure 1 shows a distribution over students’ grades (left) and positions in the inferred difficulty ranking which considered grades and retries (right). Note that the different values for grades represent answers to multiple select questions. For example, a grade of 0.9 will be achieved when 9/10 correct answers were selected by the student. As can be clearly seen from the figure, there are substantially more classes in the difficulty ranking when adding additional features.

5.3 Methods
We used two ranking scoring metrics — NDPM and AP (Section 2.3). Many papers in information retrieval also report NDCG, which is a ranking metric for datasets where each item has a score, and thus measures the difference in scores when ranking errors occur. In our case, where we do not have meaningful scores, only pure rankings, NDCG is less appropriate [12].

We compared the performance of a number of different methods to EduRank. First, we used the original EigenRank algorithm, which differs from EduRank in the similarity metric between users as well as the aggregation of the neighbor rankings.

As we explained in section 2.1, a popular alternative in the recommendation systems literature is to predict a score for an item, and then rank by sorting predicted scores. Thus we also used two popular collaborative filtering methods — a memory-based user-user KNN method using the Pearson correlation (denoted UBCF for User Based Collaborative Filtering), and a matrix factorization method using SVD (denoted SVD) to compute latent factors of items and users [6, 30]. In both cases we used the Mahout implementation of the algorithms [23].

The collaborative filtering algorithms require an item score as an input. We used the following scores: We began with the grade (first attempt) that the user got on a question, normalized to the [0 – 1] range. For each retry of the question we reduce this grade by 0.2 points. For the PSLC dataset, we reduce the (normalized) elapsed time solving the question from the score. This scoring method closely captures the difficulty ranking order we describe above. In the K12 dataset we also compared to a non-personalized difficulty ranking from 1-5 for each question, supplied by a domain expert (typically the researcher or teacher authoring the question). We denote this content expert ranking using CER.

Finally, it is common in the educational literature to identify the mastery level of the student on various topics, and then predict the performance of a question from the mastery level of the topic of the question [9]. To implement such an approach we computed the average score (using the scoring mechanism above) that the student got for all questions that belong to the same topic. We then rank the topics by decreasing average score, and rank the questions by the topic they belong to. We denote this method the Topic-Based Ranker (TBR). This measure was used only on the K12 dataset where we have available topic data.

5.4 Results
Based on the problem defined in section 3, we ran the following experiment— for each student $s_i$ we split her answered questions into two sets of equal size: a train set $T_i$, which is given as input to the various algorithms, and a test set $L_i$ that the algorithms must rank. The split is performed according to the time stamp of the answers. Later answers are in the test set. We then compare the result of each algorithm to the difficulty ranking explained above using NDPM and AP. Notice that for NDPM, the lower the score, the better the ranking, while for AP, better rankings result in higher scores. For all approaches, we ordered difficulty ranking in decreasing order of difficulty (harder questions were ranked higher in the list).

As can be seen in Figure 2 EduRank is better than all other approaches on both datasets using both metrics. The results are sta-

\footnote{https://mahout.apache.org/}
Looking at the other collaborative filtering methods we can see that EigenRank and UBCF present comparable performance. This is not very surprising, because these 2 methods do not take as input a ranking, but an item score, as we explain above. As the score is only a proxy to the actual ranking, it is no surprise that these algorithms do not do as well in predicting the true difficulty ranking.

Of the non-CF methods, TBR does relatively well. Our intuition is that identifying the student mastery level in topics is an important factor in establishing the difficulty of a question for that particular student. It is hence interesting to investigate in future research how EduRank can also benefit from the information encapsulated in topics. Nonetheless TBR can be too limiting in practice, because when a teacher wants to create a practice assignment in a particular topic, perhaps one that the student has not yet mastered, then TBR cannot be used to rank questions within that topic.

The method that performed the worst is the content expert ranking (CER). This is especially interesting as this is the only information that is currently available to teachers using the K12 e-learning system for deciding on the difficulty of questions. There can be two sources to this sub-optimal performance; First, it may be that it is too hard, even for experts, to estimate the difficulty of a question for students. Second, this may be an evidence that personalizing the order of questions for a particular student is truly important for this application.

### 5.5 Case Study

To further demonstrate the behaviour of the various algorithms, we took one of the students in the K12 dataset and present the results of the algorithms for that particular student. Table 1 presents a list of 34 test questions for this student and the rankings that were outputted by the different algorithms, in decreasing order of difficulty. The 15 most difficult questions appear in bold. Each question is denoted by (1) its knowledge component (KC) which was determined by a domain expert (this information was not in the database and the algorithms did not use it), and (2) the position of the question in the true difficulty ranking (the gold standard) of the student. This gold standard was used by the NDPM and AP metrics as a reference ranking to judge the performance of all algorithms. As shown in the table, question types involving “multiplication of big numbers” and “order of operations” appear prominently in the 15-most difficulty list, while questions in topics of geometry (“rectangles”, “polygons”) were easier for the student.

The other columns in the table show the suggested rankings by the various algorithms. For each algorithm, we present the ranking location of each question, and the true ranking of this question as obtained from the gold standard. As can be seen from the results, for this particular student, the UBCF algorithm performed poorly, placing many easy questions for the student at high positions in the ranking (e.g., “Multiply Eq 54” which appears at the top of the list but is ranked 12th in the gold standard, and “div mod” appears in 4th position in the list and ranked 11th in the gold standard.) The EigenRank and SVD algorithms demonstrated better results, but still failed to place the most difficult question for the student (e.g., order of operations) at the top of the ranked list. Only the EduRank algorithm was able to place the questions with “multiplication of big numbers” and “order of operation” type problems in the top 15 list, providing the best personalized difficulty ranking for this student.

Table 2 shows the execution time of each algorithm for building the models and computing the recommended rankings. The dataset used is the K12 dataset with 918,792 records. Our experiments were conducted on a Mac Book Air 1.7GHz Intel Core i7 with 8GB RAM.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Run Time (Sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CER</td>
<td>197.6</td>
</tr>
<tr>
<td>UBCF</td>
<td>445.2</td>
</tr>
<tr>
<td>TBR</td>
<td>625.2</td>
</tr>
<tr>
<td>EduRank</td>
<td>631.8</td>
</tr>
<tr>
<td>EigenRank</td>
<td>795.9</td>
</tr>
<tr>
<td>SVD</td>
<td>1490</td>
</tr>
</tbody>
</table>

### 6. RELATED WORK

Our work relates to several areas of research in education and educational data mining. Several approaches within the educational data mining literature have focused on the problem of question difficulty estimation. Some methods focus on estimating the difficulty of an individual question based on student performance on that question. Others focus on estimating the difficulty of a specific topic or skill. Still others build on existing student models to personalize the difficulty of a question for a particular student. This problem of question difficulty estimation is related to the broader problem of student model construction, where the goal is to build a model of a student’s knowledge state that can be used to predict future performance and recommend appropriate learning activities.

One of the key challenges in building student models is collecting and using student data effectively. Education data is often collected in a noisy and fragmented way, and it is often difficult to link this data to the specific student and their performance on specific tasks. Our work proposes a solution to this problem by building on existing student models and using additional information to personalize the difficulty of questions for individual students.

Another challenge in building student models is the need for high-quality student models. As we noted above, students’ knowledge levels can vary widely, and this can make it difficult to accurately estimate the difficulty of a question. Our work proposes a solution to this problem by building on existing student models and using additional information to personalize the difficulty of questions for individual students.

Finally, another challenge in building student models is the need for efficient algorithms to estimate the difficulty of questions. As we noted above, this is a computationally intensive task, and it is important to have algorithms that can efficiently estimate the difficulty of a large number of questions. Our work proposes a solution to this problem by building on existing student models and using additional information to personalize the difficulty of questions for individual students.
data mining community have used computational methods for sequencing students’ learning items. Pardos and Heffernan [18] infer order over questions by predicting students’ skill levels across action pairs using Bayesian Knowledge Tracing. They show the efficacy of this approach on a test-set comprising random sequences of three questions as well as simulated data. This approach explicitly considers each possible order sequence and does not scale to handling large number of sequences, as in the student ranking problem we consider in this paper.

Champaign and Cohen [7] suggest a peer-based model for content sequencing in an intelligent tutor system by computing the similarity between different students and choosing questions that provide the best benefit for similar students. They measure similarity by comparing between students’ average performance on past questions and evaluate their approach on simulated data. Our approach differs in several ways. First, we don’t use an aggregate measure to compute similarity but compare between students’ difficulty rankings over questions. In this way we use the entire ranked list for similarity computation, and do not lose information. Second, we are using social choice to combine similar students’ difficulty rankings over questions. Lastly, we evaluate our approach on simulated data sets.

Table 1: Rankings outputted by the different algorithms for a sample target student

Collaborative filtering (CF) was previously used in the educational domain for predicting students’ performance. Toscher and Jaehrer [25] use an ensemble of CF algorithms to predict performance for items in the KDD 2010 educational challenge. Berger et. al [5] use a model-based approach for predicting accuracy levels of students’ performance and skill levels on real and simulated data sets. They also formalize a relationship between CF and Item Response Theory methods and demonstrate this relationship empirically. Lastly, Loll and Pinkwart [16] use CF as a diagnostic tool for knowledge test questions as well as more exploratory ill-defined tasks.

7. SUMMARY AND FUTURE WORK

This paper presented a novel approach to personalization of educational content. The suggested algorithm, called EduRank, combines a nearest-neighbor based collaborative filtering framework with a social choice method for preference ranking. The algorithm constructs a difficulty ranking over questions for a target student by aggregating the ranking of similar students. It extends existing approaches for ranking of user items in two ways. First, by inferring a difficulty ranking directly over questions for a target student, rather than ordering them according to predicted performance, which is prone to error. Second, by penalizing disagreements between the difficulty rankings of similar students and the target student more highly for harder questions than for easy questions.

The algorithm was tested on two large real world data sets and its performance was compared to a variety of personalization meth-
ods as well as a non-personalized method that relied on a domain expert. The results showed that EduRank outperformed existing state-of-the-art algorithms using two metrics from the information retrieval literature.

In future work we plan to address the cold start problem by applying EduRank in a classroom setting in which we will personalize educational content to both exiting and new students. We also intend to evaluate EduRank’s performance when training on small datasets and in MOOCs settings where the number of data points may dramatically change over time.

8. ACKNOWLEDGEMENTS

This work was supported in part by EU grant no. FP7-ICT-2011-9 #600854 and by ISF grant no. 1276/12.

9. REFERENCES


