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 accomodate for individual differences in such e-learning systems. This paper presents a new algorithm for personliazing 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.

[1]To illustrate, in the KDD cup 2010, the best preforming grade prediction algorithms exhibited prediction errors of about 28% [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, 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 compering 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 $\binom{L}{2}$ 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 $\overline{\succ}$ as a partial order over L such that if $q_j \succ q_k$ then $q_k \overline{\succ} q_j$. 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 $L \subseteq Q$ are all questions (q_k, q_l) such that $q_k \succ q_l$ and $q_k, q_l \in L$.

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:

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

The total distance over all question pairs in L is defined as follows

$$\beta_{\succ_1,\succ_2}(L) = \sum_{(q_j,q_k)\in\binom{L}{2}} \delta_{\succ_1,\succ_2}(q_j, q_k) \tag{2}$$

Let $m(\succ_1) = \text{argmax}_{\succ} \beta_{\succ_1,\succ}(L)$ 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

$$s_{ND}(L, \succ_1, \succ_2) = \frac{\beta_{\succ_1,\succ_2}(L)}{m(\succ_1)} \tag{3}$$

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 systems in which some questions for the target student in L may not have been solved by other students and these questions may remain unordered in the difficulty ranking.

2.3.2 *AP Rank Correlation*

A potential problem with the NDPM metric is that it does not consider the location of disagreements in the reference ranking. In some cases it is more important to appropriately order items that should appear closer to the head of the ranked list, than items that are positioned near the bottom. For example, when ranking movies, it may be more important to properly order the movies that the user would enjoy, than to properly order the movies that the user would not enjoy. Similarly, we assume that the severity of errors in ranking questions depends on their position in the ranked 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 questions that are more difficult to the student than question q_k.

$$Z^k(L, \succ_2) = \{(q_j, q_k) \mid \forall q_j \neq q_k \text{ s.t. } q_j \succ_2 q_k \text{ and } q_j, q_k \in L\} \tag{4}$$

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 $q_j \succ_i q_k$.

$$A^k(L, \succ_1, \succ_2) = \frac{1}{k-1} \sum_{(q_j,q_k) \in Z^k(L,\succ_2)} I^A(q_j, q_k, \succ_1, \succ_2) \tag{5}$$

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

$$s_{AP}(L, \succ_1, \succ_2) = \frac{1}{|L|-1} \sum_{k \in |L|} A^k(L, \succ_1, \succ_2) \tag{6}$$

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 were 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_i, and a set of questions L_i, for which the algorithm must predict a difficulty ranking $\hat{\succ}_i$ over L_i. The predicted difficulty ranking $\hat{\succ}_i$ is evaluated with respect to a difficulty reference ranking $\succ_i$ over L_i using a scoring function $s(\hat{\succ}_i, \succ_i, L_i)$.

To solve this problem, we take a collaborative filtering approach, which uses the difficulty rankings on L_i of other students similar to s_i to construct a difficulty ranking over L_i for student s_i. 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 $T_j, L_j \in Q$, 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 $s_i \in S$, T_i represent the set of questions that the target student s_i has already answered, while L_i 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 $\hat{\succ}_i$ over L_i for student s_i.

4. THE EDURANK 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_i. EduRank estimates how similar other student are to s_i, and then combines the ranking of the similar students over L_i to create a ranking for s_i. 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_i 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 $\gamma(q_k, q_l, \succ)$ over question pairs q_k, q_l

Algorithm 1 EduRank

```
INPUT:
Set of students $S$.
Set of questions $Q$.
For each student $s_j \in S$, a partial ranking $\succ_j$ over $T_j \subseteq Q$.
Target student $s_i \in S$.
Set of questions $L_i$ to rank for $s_i$.
OUTPUT: a partial order $\hat{\succ}_i$ over $L_i$.
1: for each $q \in L_i$ do
2:     $c(q) = \sum_{q_l \in L \setminus q} rv(q, q_l, S)$
3: end for
4: $\hat{\succ}_i \leftarrow \{\forall (q_k, q_l) \in \binom{L_i}{2}, q_k \hat{\succ}_i q_l \text{ iff } c(q_k) > c(q_l)\}$
5: return $\hat{\succ}_i$
```

given a difficulty ranking $\succ$ as follows:

$$\gamma(q_k, q_l, \succ) = \begin{cases} 1 & \text{if } q_k \succ q_l \\ -1 & \text{if } q_l \succ q_k \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

The relative voting $rv(q_k, q_l, S)$ of two questions q_k, q_l given the difficulty rankings of a group of (neighboring) students S is

$$rv(q_k, q_l, S) = sign(\sum_{j \in S \setminus i} s_{AP}(T_i, \succ_i, \succ_j) \cdot \gamma(q_k, q_l, \succ_j)) \quad (8)$$

The Copeland score $c(q, S, L_i)$ of a question q given the difficulty rankings of students S and test questions L_i is

$$c(q, S, L_i) = \sum_{q_l \in L_i \setminus q} rv(q, q_l, S) \quad (9)$$

The EduRank algorithm is shown in Algorithm 1. The input to the EduRank algorithm is a set of students $S = \{s_1, \ldots, s_n\}$, each with a known ranking over a set of questions T_j, such that $Q = T_1 \cup \ldots \cup T_n$. In addition the algorithm is given a target student $s_i \in S$, and a set of questions $L_i \subseteq Q$ that needs to be ranked for s_i. The output of the algorithm is a ranking of the questions in L_i.

The algorithm computes a ranking score $c(q)$ for each question $q \in L_i$, which is the Copeland score for that question, as defined above. The algorithm returns a partial order for student s_i over the test set L_i where questions are ranked by decreasing Copeland score $c(q)$.

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 Pittsburgh 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.

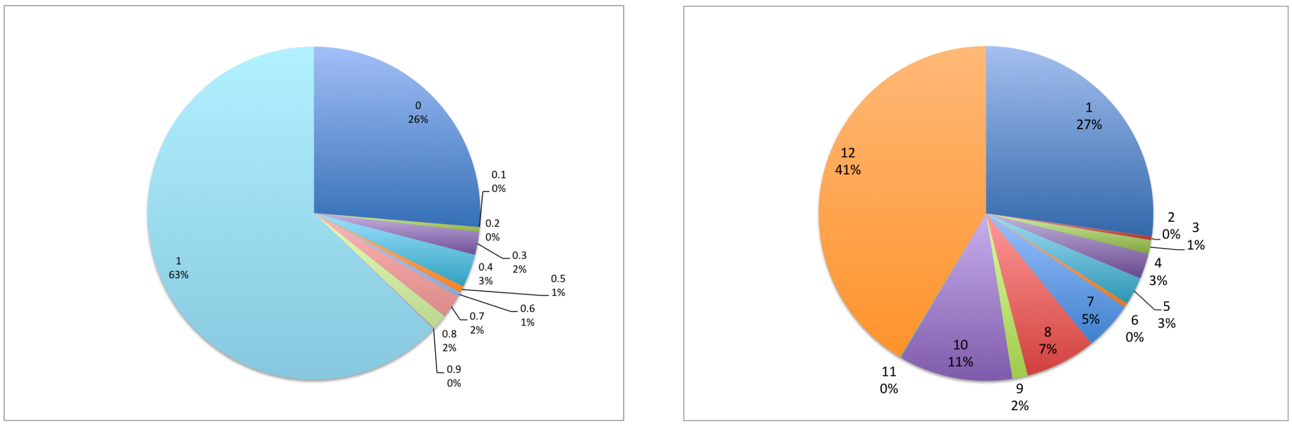
An alternative approach is to consider additional features in addition to grades (or correct first attempts), that are present in the datasets, and which correlate with the difficulty of the question for the individual student. Specifically, we assumed that questions that were answered correctly on a 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

[2] `https://pslcdatashop.web.cmu.edu/KDDCup`

[3] Note there were other features in this data-set that were not used in the study.



(a) Grades (b) Difficulty Ranking

Figure 1: Distribution over grades and difficulty ranking positions for K12 dataset

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[4] 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.

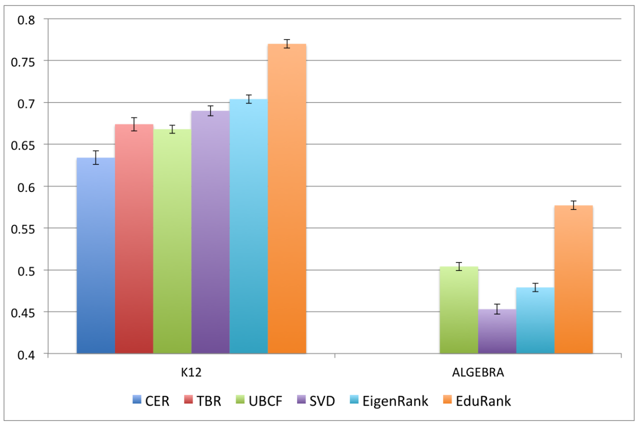
5.4 Results

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.[5] 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).

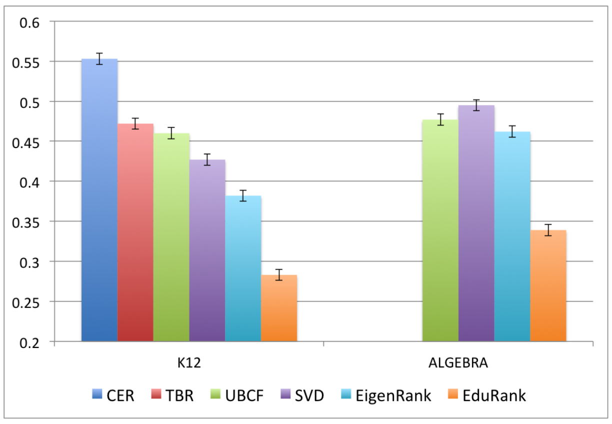
As can be seen in Figure 2 EduRank is better than all other approaches on both datasets using both metrics. The results are sta-

[4] `https://mahout.apache.org/`

[5] Note that the AP metric is also used to measure similarity between neighboring students in EduRank. We note that (1) it is standard practice in ML to use the same metric in the algorithm and the evaluation, and (2) the AP measure was computed over the training set in the algorithm, but over the test set in the evaluation.



(a) AP score (higher is better)



(b) NDPM score (lower is better)

Figure 2: Performance Comparison

tistically significant ($p < 0.05$, paired t-test between EduRank and the leading competing algorithm).

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.

Algorithm	Run Time (Sec)
CER	197.6
UBCF	445.2
TBR	625.2
EduRank	631.8
EigenRank	795.9
SVD	1490

Table 2: Execution Time

6. RELATED WORK

Our work relates to several areas of research in education and educational data mining. Several approaches within the educational

Gold Standard		EduRank Ranking		EigenRank Ranking		UBCF Ranking		SVD Ranking	
KC	True Rank	KC	True Rank	KC	True Rank	KC	True Rank	KC	True Rank
Order of Operations, choose options	**1**	**Order of Operations, choose options**	**1**	**Order of Operations, Brackets**	**7**	**Multuply, Equals 54**	**12**	**Multiply, Big Numbers**	**4**
Letters Order	**1**	**Natural Numbers, Verbal Claims**	**3**	**Natural numbers, In between**	**12**	**Multiply, Choose Between 2**	**12**	**Multiply, Bigger than**	**10**
Multiply, Equals 40	**2**	**Add, Sub, Equals 30**	**10**	**Div, No Mod, Mod 1**	**11**	**Multiply, Bigger than**	**10**	**Order of Operations, Brackets**	**5**
Natural Numbers, Verbal Claims	**3**	**Letters Order**	**1**	**Div, Div and Mod**	**11**	**Div, No Mod, Mod 1**	**11**	**Order of Operations, Equals 5**	**6**
Multiply, Big Numbers	**4**	**Add, Sub, Verbal Claims**	**7**	**Multiply, Big Numbers**	**7**	**Div, No Mod, Mod 2**	**12**	**Natural Numbers, Verbal Claims**	**3**
Order of Operations, Brackets	**5**	**Order of Operations, Equals 5**	**6**	**Div, Exists?**	**8**	**Multiply, Big Numbers**	**4**	**Add, Sub, Equals 30**	**10**
Zero, Equals Zero	**5**	**Order of Operations, Brackets**	**5**	**Multiply, Equals 40**	**2**	**Natural Numbers, Verbal Claims**	**3**	**Order of Operations, Brackets**	**7**
Order of Operations, Equals 5	**6**	**Zero, Equals Zero**	**5**	**Div, Mod 2**	**12**	**Order of Operations, choose options**	**1**	**Div, Mod 2**	**12**
Order of Operations, Brackets	**7**	**Multiply, Big Numbers**	**4**	**Multuply, Choose between 2**	**12**	**Order of Operations, Equals 5**	**6**	**Add, Sub, Verbal Claims**	**7**
Add, Sub, Verbal Claims	**7**	**Div, Mod 2**	**12**	**Order of Operations, Which is bigger**	**11**	**Multiply, Choose between 2**	**12**	**Order of Operations, choose options**	**1**
Multiply, Big Numbers	**7**	**Div, No Mod, Mod 2**	**12**	**Order of Operations, Brackets**	**5**	**Multuply, Choose between 2**	**12**	**Multuply, Equals 54**	**12**
Div, Exists?	**8**	**Order of Operations, Brackets**	**7**	**Div, Mod 1**	**11**	**Order of Operations, Brackets**	**7**	**Div, Exists?**	**12**
Substruction	**9**	**Order of Operations, Which is bigger**	**11**	**Order of Operations, only %, /**	**11**	**Order of Operations, Brackets**	**5**	**Div, No Mod, Mod 2**	**12**
Multiply, Bigger than	**10**	**Order of Operations, only %, /**	**11**	**Polygon, Parallel sides**	**10**	**Letters Order**	**1**	**Multiply, Big Numbers**	**7**
Add, Sub, Equals 30	**10**	**Multiply, Big Numbers**	**7**	**Letters Order**	**1**	**Rectangle, Identify**	**12**	**Natural numbers, In between**	**12**
Polygon, Parallel sides	10	Div, Exists?	12	Order of Operations, Equals 5	6	Multiply, Big Numbers	7	Zero, Equals Zero	5
Order of Operations, only +, -	11	Substruction	9	Substruction	9	Polygon, Identify	12	Order of Operations, Which is bigger	11
Order of Operations, only %, /	11	Polygon, Parallel sides	10	Add, Sub, Verbal Claims	7	Zero, Equals Zero	5	Div, Div and Mod	11
Order of Operations, Which is bigger	11	Order of Operations, only +, -	11	Multiply, Big Numbers	4	Order of Operations, only +, -	11	Letters Order	1
Div, Mod 1	11	Div, No Mod, Mod 1	11	Natural Numbers, Verbal Claims	3	Add, Sub, Equals 30	10	Angles, Find Bigger	12
Div, Div and Mod	11	Multiply, Bigger than	10	Add, Sub, Equals 30	10	Polygon, Parallel sides	10	Multiply, Choose between 2	12
Div, No Mod, Mod 1	11	Div, Exists?	8	Order of Operations, choose options	1	Add, Sub, Verbal Claims	7	Div, Mod 1	11
Natural numbers, In between	12	Div, Mod 1	11	Order of Operations, only +, -	11	Div, Mod 1	11	Multuply, Choose between 2	12
Multuply, Equals 54	12	Multiply, Equals 40	2	Zero, Equals Zero	5	Div, Mod 2	12	Div, No Mod, Mod 1	11
Multiply, Choose between 2	12	Div, Div and Mod	11	Div, No Mod, Mod 2	12	Div, Div and Mod	11	Polygon, Parallel sides	10
Multuply, Choose between 2	12	Multuply, Choose between 2	12	Div, Exists?	12	Order of Operations, only %, /	11	Div, Exists?	8
Div, Mod 2	12	Multiply, Choose Between 2	12	Multiply, Bigger than	10	Order of Operations, Which is bigger	11	Order of Operations, only %, /	11
Div, Exists?	12	Rectangle, Identify	12	Multiply, Choose Between 2	12	Div, Exists?	8	Substruction	9
Div, No Mod, Mod 2	12	Polygon, Identify	12	Rectangle, Identify	12	Div, Exists?	12	Order of Operations, only +, -	11
Angles, Find Bigger	12	Multuply, Equals 54	12	Multuply, Equals 54	12	Natural numbers, In between	12	Multiply, Equals 40	2
Angles, Find Bigger	12	Angles, Find Bigger	12	Angles, Find Bigger	12	Substruction	9	Angles, Find Bigger	12
Rectangle, Identify	12	Angles, Find Bigger	12	Angles, Find Bigger	12	Multiply, Equals 40	2	Multiply, Choose Between 2	12
Polygon, Identify	12	Natural numbers, In between	12	Multiply, Choose between 2	12	Angles, Find Bigger	12	Polygon, Identify	12
Multiply, Choose Between 2	12	Multiply, Choose between 2	12	Polygon, Identify	12	Angles, Find Bigger	12	Rectangle, Identify	12

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

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 over 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.[6] Second, we are using social choice to combine similar students' difficulty ranking over questions. Lastly, we evaluate our approach on two real-world data sets. Li, Cohen and Koedinger [14] compared a blocked order approach, in which all problems of one type are completed before the student is switched to the next problem type to an interleaved approach, where problems from two types are mixed and showed that the interleaved approach yields more effective learning. Our own approach generates an order of the different questions by reasoning about the student performance rather than determining order a-priori.

Lastly, multiple works have used Bayesian Knowledge Tracing as a way to infer students' skill acquisition (i.e., mastery level) over time given their performance levels on different question sequences [9]. These works reason about students' prior knowledge of skills and also account for slips and guessing on test problems. The models are trained on large data sets from multiple students using machine learning algorithms that account for latent variables [3, 10]. We solve a different problem, that of using other students' performance to personalize ranking over test-questions. In addition, these works measure students' performance dichotomously (i.e., success or failure) whereas we reason about additional features such as students' grade and number of attempts to solve the question. We intend to infer students' skill levels to improve the ranking prediction in future work.

Collaborative filtering (CF) was previously used in the educational domain for predicting students' performance. Toscher and Jahrer [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 the 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-

[6]Consider student1 who has accrued grades 60 and 80 on questions (a) and (b) respectively; and student2 who has accrued grades 80 and 60 on questions (a) and (b) respectively. The average grade for both questions will be the same despite that they clearly differ in difficulty level for the students (when ordered solely based on grade).

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

[1] Y. Akbulut and C. S. Cardak. Adaptive educational hypermedia accommodating learning styles: A content analysis of publications from 2000 to 2011. *Computers & Education*, 58(2):835–842, 2012.

[2] H. Ba-Omar, I. Petrounias, and F. Anwar. A framework for using web usage mining to personalise e-learning. In *Advanced Learning Technologies, 2007. ICALT 2007. Seventh IEEE International Conference on*, pages 937–938. IEEE, 2007.

[3] R. S. Baker, A. T. Corbett, and V. Aleven. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In *Intelligent Tutoring Systems*, pages 406–415. Springer, 2008.

[4] R. S. Baker, J. Walonoski, N. Heffernan, I. Roll, A. Corbett, and K. Koedinger. Why students engage in gaming the system behavior in interactive learning environments. *Journal of Interactive Learning Research*, 19(2):185–224, 2008.

[5] Y. Bergner, S. Droschler, G. Kortemeyer, S. Rayyan, D. Seaton, and D. Pritchard. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. In *EDM*, pages 95–102, 2012.

[6] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[7] J. Champaign and R. Cohen. A model for content sequencing in intelligent tutoring systems based on the ecological approach and its validation through simulated students. In *FLAIRS Conference*, 2010.

[8] A. H. Copeland. A reasonable social welfare function. In *University of Michigan Seminar on Applications of Mathematics to the social sciences*, 1951.

[9] A. T. Corbett and J. R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User modeling and user-adapted interaction*, 4(4):253–278, 1994.

[10] M. H. Falakmasir, Z. A. Pardos, G. J. Gordon, and P. Brusilovsky. A spectral learning approach to knowledge tracing. In *EDM*, 2013.

[11] P. C. Fishburn. *The theory of social choice*, volume 264. Princeton University Press Princeton, 1973.

[12] E. Kanoulas and J. A. Aslam. Empirical justification of the gain and discount function for ndcg. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 611–620. ACM, 2009.

[13] K. R. Koedinger, R. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. A data repository for the edm community: The pslc datashop. *Handbook of educational data mining*, pages 43–55, 2010.

[14] N. Li, W. W. Cohen, and K. R. Koedinger. Problem order implications for learning transfer. In *Intelligent Tutoring Systems*, pages 185–194. Springer, 2012.

[15] N. N. Liu and Q. Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 83–90. ACM, 2008.

[16] F. Loll and N. Pinkwart. Using collaborative filtering algorithms as elearning tools. In *42nd Hawaii International Conference on Systems Science*, 2009.

[17] H. Nurmi. Voting procedures: a summary analysis. *British Journal of Political Science*, 13(02):181–208, 1983.

[18] Z. A. Pardos and N. T. Heffernan. Determining the significance of item order in randomized problem sets. In *EDM*, 2009.

[19] D. M. Pennock, E. Horvitz, C. L. Giles, et al. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *AAAI/IAAI*, pages 729–734, 2000.

[20] D. Sampson and C. Karagiannidis. Personalised learning: Educational, technological and standardisation perspective. *Interactive Educational Multimedia*, 4:24–39, 2002.

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[22] F. Schalekamp and A. van Zuylen. Rank aggregation: Together we're strong. In *ALENEX*, pages 38–51, 2009.

[23] S. Schelter and S. Owen. Collaborative filtering with apache mahout. *Proc. of ACM RecSys Challenge*, 2012.

[24] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[25] A. Toscher and M. Jahrer. Collaborative filtering applied to educational data mining. *KDD Cup*, 2010.

[26] Y. Yao. Measuring retrieval effectiveness based on user preference of documents. *JASIS*, 46(2):133–145, 1995.

[27] E. Yilmaz, J. A. Aslam, and S. Robertson. A new rank correlation coefficient for information retrieval. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 587–594. ACM, 2008.

[28] L. Zhang, X. Liu, and X. Liu. Personalized instructing recommendation system based on web mining. In *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, pages 2517–2521. IEEE, 2008.

[29] B. Zhou and Y. Yao. Evaluating information retrieval system performance based on user preference. *Journal of Intelligent Information Systems*, 34(3):227–248, 2010.

[30] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008.