A Difficulty Ranking Approach to Personalization in E-learning

Avi Segal[a,*], Kobi Gal[a], Guy Shani[a], Bracha Shapira[a]

[a]*Ben-Gurion University of the Negev, Israel*

Abstract

The prevalence of e-learning systems and on-line courses has made educational material widely accessible to students of varying abilities and backgrounds. There is thus a growing need to accommodate for individual differences in e-learning systems. This paper presents an algorithm called EduRank for personalizing educational content to students that combines a collaborative filtering algorithm with voting methods. EduRank constructs a difficulty ranking for each student by aggregating the rankings of similar students using different aspects of their performance on common questions. These aspects include grades, number of retries, and time spent solving questions. It infers a difficulty ranking directly over the questions for each student, rather than ordering them according to the student's predicted score. The EduRank algorithm was tested on two data sets containing thousands of students and a million records. It was able to outperform the state-of-the-art ranking approaches as well as a domain expert. EduRank was used by students in a classroom activity, where a prior model was incorporated to predict the difficulty rankings of students with no prior history in the system. It was shown to lead students to solve more difficult questions than an ordering by a domain expert, without reducing their performance.

Keywords: Artificial Intelligence, e-learning, ranking algorithms, human-computer decision-making

*Corresponding author.

Email addresses: avise@post.bgu.ac.il (Avi Segal), kobig@bgu.ac.il (Kobi Gal), shanigu@bgu.ac.il (Guy Shani), bracha.shapira@gmail.com (Bracha Shapira)

1. Introduction

The prevalence of educational software in schools and the explosion of online course opportunities have made educational content a common resource that is accessible to student communities of varied backgrounds and learning abilities. 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 [1, 2, 3, 4, 5, 6].

Many educational applications present a sequence of questions to students, ordered by increasing difficulty. The student is expected to first solve easier questions in a given skill, and only after mastering the skill, move to more difficult and challenging questions. As such, ordering questions by difficulty is an important task in such applications.

This paper provides a novel approach for personalization of educational content that directly creates a *difficulty ranking* over new questions allowing us to order questions differently for different students. Our approach is based on collaborative filtering [7], which is a commonly used technique in recommendation systems for predicting the interests of a user by collecting preferences from their online activities. The explosive growth of e-commerce and online environments mean that users are becoming overloaded by options to consider and they may not have the time or knowledge to personally evaluate these options. Recommender systems have proven to be a valuable way for online users to cope with the information overload and have become one of the most powerful and popular tools in electronic commerce [8].

This paper uses the collaborative filtering approach to generate 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 questions, the number of retries for each question, and other features. Unlike other applications of collaborative filtering in education, our approach directly generates a difficulty ranking over the test questions, avoiding the need to predict the students' performance directly on these questions, which is prone to error. For example, in the KDD cup 2010, the best performing grade prediction algorithms exhibited an error rate of about 28% [9]. We demonstrate the same problem over the datasets that we used in our empirical analysis.

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 on which questions they solve. The algorithm extends a prior approach for ranking items in recommendation systems [10], which was not evaluated on educational data, in two ways: First, by using voting methods from social choice [11] to combine the difficulty rankings of similar students and produce a better 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 when ranking more difficult questions are more detrimental to students than errors made when ranking easier questions. EduRank can 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.

We evaluated EduRank on two large 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, focusing on popular collaborative filtering approaches such as matrix factorization and memory-based nearest neighbours. We also compared EduRank to a (non-personalized) ranking created by a domain expert. EduRank was able to outperform all other approaches when comparing the outputted difficulty rankings to a gold standard.

EduRank was embedded in a real classroom and used to sequence math questions to students by inferred order of difficulty. Its performance was compared to an alternative sequencing approach that selected questions by increasing order of difficulty, as determined by pedagogical experts. We found that students using the EduRank algorithm solved harder questions and spent more time in the system than the expert-based sequencing approach, without impeding their overall performance. This demonstrates the potential of the EduRank approach to contribute to students' learning in the classroom.

The contributions of this paper are three-fold. First, it presents a novel algorithm for personalization in e-learning according to the level of difficulty by combining collaborative filtering with social choice. Second, it is shown to outperform alternative ranking solutions from the literature on two real-world data sets. Finally, it is shown to improve students' performance in the classroom when solving questions of different difficulty levels. This paper extends prior work describing the EduRank algorithm [12] in several ways.

First, we show there is low agreement in the difficulty rankings between students over the same set of questions. This demonstrates that sequencing questions to student in a "one-size-fits-all" approach cannot address these individual differences among students. Second, we have extended the algorithm to handle the "cold-start problem" in e-learning systems in which there is a need to sequence material for new students with little or no history in the system. Third, we deployed the algorithm in a real classroom, where it was compared to an alternative sequencing approach that was designed by pedagogical experts.

2. Background

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

2.1. Online Learning and Intelligent Tutoring Systems

Intelligent Tutoring Systems have been used for computer based instruction since the 1970s. Seeking to apply artificial intelligence techniques for "intelligent" computer-based instruction, their goal is to engage students in sustained reasoning activities and to interact with the student while understanding the student behavior and state. Graesser et. al [13] reviewed the state of ITS and specifically the research on different classes of ITS. They describe the computational mechanism of each type of ITS and the available empirical assessments of the impact of these systems on learning gains.

In recent years we have seen a dramatic change in the education world towards wide-adoption of online learning technologies (e-learning). The huge amount of fine-grained data being collected, coupled with Big Data and artificial-intelligence mechanisms, can be used to develop learning environments that can adapt to the needs of the individual learner. For example, MOOCs have democratized the access to educational resources, making them accessible to anyone with an internet connection [14, 15].

Due to the growing prevalence of online learning settings, how to sequence educational content to students is an important research problem. Existing work in this area focused on using graph search [16], neural networks [17], and heuristic semantics [18] for learning path personalization, among other methods.

2.2. Recommendation Systems and Collaborative Filtering

Recommender systems actively help users in identifying items of interest. 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. For example, a recommendation system for movies, may predict the rating that a user may give to a newly released movie, or recommend a new movie for a user given movies that the user liked in the past. 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 [7]. In the memory-based nearest neighbors 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 popular matrix factorization approach [19, 20] 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, there are 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 [10] 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. EigenRank computes the similarity of users using the Kendall τ metric — a well known metric in information retrieval for comparing two rankings which counts the number of pairwise disagreements between the two lists — rather than by metrics such as Pearson correlation which are popular in rating prediction. Matrix factorization methods have also been suggested for ranking [21].

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 ranking better than methods based on converting rating predictions to a ranked list.

2.3. 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 [22]. Examples of such methods include voting systems used to aggregate preferences of voters over a set of alternatives to determine which alternative(s) wins the election, and systems in which voters rank a complete set of alternatives using an ordinal scale. One such approach which we use in this paper is Copeland's method [23, 24] ordering alternatives based on the number of pairwise defeats and victories with other alternatives.

The Copeland 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 [25]. Pennock et al. [26] highlighted the relevance of social choice to CF, demonstrating that properties and limitations from Social Choice theory apply in the context of CF. Following their work, we consider weighted versions of voting mechanisms to CF algorithms and demonstrate the effect of this approach to e-learning systems.

2.4. 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 [27]. 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, or inferred from the data.

Before describing the comparison metrics and stating their relevance for e-learning systems, we define the following notations: Given a set L of questions of varying difficulties, let $\binom{L}{2}$ denote the set of all non ordered pairs in L. Let $\succ$ be a partial order over the 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 orderings 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 orderings $\succ_1$ and $\succ_2$ as follows:

- The orderings $\succ_1$ and $\succ_2$ *agree* on questions q_j and q_k if $q_j \succ_1 q_k$ and $q_j \succ_2 q_k$. That is, both $\succ_1$ and $\succ_2$ order the two questions in the same order.
- The orderings $\succ_1$ and $\succ_2$ *disagree* on questions q_j and q_k if $q_j \succ_1 q_k$ and $q_k \succ_2 q_j$. That is, while in $\succ_1$ q_j is deemed more difficult than q_k, in $\succ_2$ q_j is considered to be easier than q_k.
- The orderings $\succ_1$ 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 $q_k \succ_2 q_j$. That is, $\succ_2$ does not provide any ordering of questions that are ordered in $\succ_1$. This is not symetric — if the reference order $\succ_1$ does not provide any ordering over a pair, any ordering provided by the system ordering $\succ_2$ is acceptable.

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 $q_k, q_l \in L$. That is, we restrict the order only to a subset of questions that are of interest to us, for example, questions that a specific student has already answered.

2.4.1. Normalized Distance based Performance

The Normalized Distance based Performance Measure (NDPM) [28, 29] is a commonly used metric for evaluating a proposed system ranking to a reference ranking. It differentiates between correct orderings of pairs, incorrect orderings 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:

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

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

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

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

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

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, as well as other collaborative filtering applications, in which many 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.

2.4.2. 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 [30], 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

$$s_{AP}(L, \succ_1, \succ_2) = \frac{1}{|L| - 1} \sum_{k=2}^{|L|} A^k(L, \succ_1, \succ_2) \tag{6}$$

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 $i \in \{1, \ldots, |L|\}$. The worst score of 0 is given to systems were there is no agreement between the two ranked lists.

3. Problem Definition and the EduRank Algorithm

The *difficulty ranking* problem is defined by a target student s_i, and a set of questions L_i, for which the algorithm must predict a difficulty ranking $\hat{\succ}_i$. 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 is:

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 $T_j \subseteq Q$.
4. For each student $s_j \in S$, a subset of questions $L_j \subseteq Q$ that must be ordered.

For every student $s_j \in S$ we require 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 predicted. For example, L_i may be a set of questions in a homework assignment, that needs to be ordered for a particular student, while T_i may be the set of all questions that the student has solved prior to this homework assignment.

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.

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 students 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 over questions in T_i. We prefer s_{AP} to, e.g., NDPM, to encourage greater similarity between students with high agreement in top positions (more difficult questions) in their respective rankings.

For aggregating the different students' rankings to create a difficulty ranking for the target student, we use the Copeland method. We treat each question as an alternative and look at the aggregated voting of neighbors based on their similarity metric. In our aggregated voting calculation, alternative i is preferred over alternative j if the similarity normalized number of wins of i over j computed over all neighbors is higher than the similarity normalized number of losses. The Copeland method then computes for each alternative question the overall number of aggregated victories and aggregated defeats

and ranks the alternatives accordingly. Let the win score $\gamma(q_k, q_l, \succ)$ over question pairs q_k, q_l given a difficulty ranking $\succ$ as follows:

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

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

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

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

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

The EduRank algorithm is shown in Figure 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 $c(q)$.

Complexity: The bottleneck of the computational complexity of the algorithm is computing the relative voting score in line 3 of Figure 1. As shown in Equation 8 the relative voting of a question pair (q_k, q_l) for student s_i combines the similarity score s_{AP} with the ranking score γ. To compute the similarity score between s_i and each student in S, all students in S are traversed once, and for each student all pairs of questions in L_i are traversed. Thus, the similarity computation complexity is $\mathcal{O}(S \cdot |L_i|^2)$. To compute the ranking score all of s_i neighbors are traversed once, and for each neighbor all question pairs are compared for each question location. This leads to a ranking computation complexity bounded by $\mathcal{O}(S \cdot |L_i|^4)$. This complexity is polynomial in the number of questions in the test set of the target student.

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: **function** EDURANK($S, Q, \succ, i, L_i$)
2: **for all** $q \in L_i$ **do**
3: $c(q) = \sum_{q_l \in L_i \setminus q} rv(q, q_l, S)$
4: **end for**
5: $\hat{\succ}_i \leftarrow \{\forall (q_k, q_l) \in \binom{L_i}{2}, q_k \hat{\succ}_i q_l \text{ iff } c(q_k) > c(q_l)\}$
6: **return** $\hat{\succ}_i$
7: **end function**

Figure 1: Psudeocode for the EduRank algorithm

4. 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.

4.1. 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) [1] [31]. We used the Algebra 1 dataset from the competition, containing about 800,000 answering attempts by 575 students, collected during 2005-2006. Data sparsity in this dataset (the amount of empty cells in the student/question matrix) was 99.29%. 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. Other features in this dataset were not used in the study. 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 a *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

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

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. The data sparsity in this dataset was 98.24%. 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.

4.2. Computing the Difficulty Ranking

EduRank assumes that each student has a personal difficulty ranking over questions. 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 most questions in the PSLC dataset, the final grade 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.

Two alternative families of methods for estimating item difficulty are *item response theory* (IRT) [32] and *Bayesian knowledge tracing* (BKT) [33]. The IRT model assumes that many students have completed a test of dichotomous items and assigns each student a proficiency parameter. It models variation of student proficiency across different items. Bayesian Knowledge Tracing captures dynamic changes in student capabilities over time. Both of these models do not take into account the number of retries and response time or reason about similarity between students.

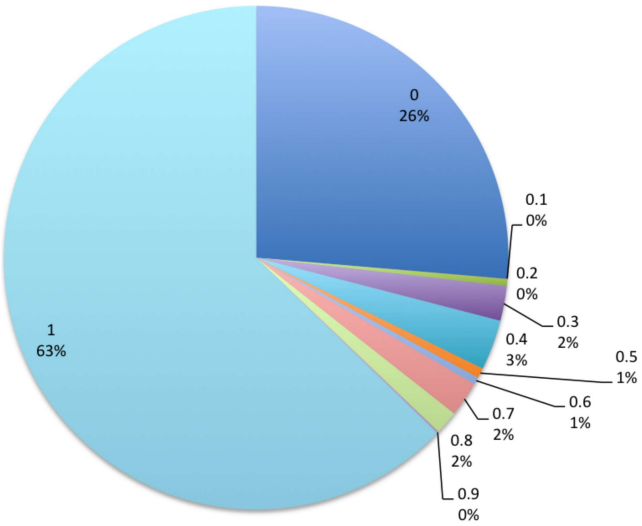
We assumed that questions that were answered correctly on a first attempt were easier for the student, while questions that required multiple attempts were harder. We also assumed that questions that required more solution time, as registered in the log, were more difficult to the students. These two 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 Interactive Learning Environments work [34]. 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 demonstrate later in this section, using grades, number of attempts and response times provide a reasonable approach towards ranking questions.

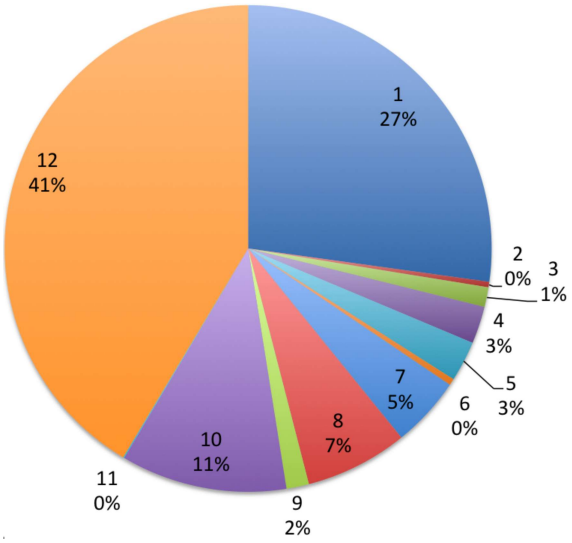
We describe the following method for identifying the difficulty ranking. We begin by ranking questions by grades. In the PSLC dataset we use "correct first attempt" for this, and in the K12 dataset we use the grade that the student got on her first attempt. After ranking by grade, we break ties by using 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 2 shows a distribution over students' grades (top) and positions in the inferred difficulty ranking which considered grades and retries (bottom). 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.

To motivate the EduRank approach, we show that students exhibited a wide degree of variance over difficulty rankings when solving questions. We chose four math topics at random from the dataset, and compared the rankings of all students that solved all questions for each topic using the AP metric defined in Equation 6. In each topic, the AP metric is calculated by choosing one student at random and comparing all other students' rankings to this student ranking. Table 1 presents the topics tested in the K12 datasets. For each such topic we give the topic ID, the number of questions belonging to this topic and the calculated AP metric. As can be seen from the rightmost column in the table, all AP values are significantly lower than 1, which is the value for perfect similarity between difficulty rankings. This means that difficulty rankings with other students are not well correlated. It supports the approach to sequence questions to students in a personal manner as opposed to a "one size fits all" approach.



(a) Grades



(b) Difficulty Ranking

Figure 2: Distribution over grades and difficulty ranking positions for K12 dataset

Topic	Num. questions	AP
fractions	33	0.31
powers	48	0.36
rectangles	17	0.51
circles	61	0.34

Table 1: AP metric measuring students' difficulty rankings for questions in different topics

4.3. Methods

We used the two ranking scoring metrics that we described earlier — NDPM and AP. 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 [35].

We compared the performance of a number of alternative methods to EduRank. First, we compared to the original EigenRank algorithm, which differs from EduRank in that the similarity metric between users and aggregation method is only based on grades. In the K12 dataset we also compared to the default ranking method provided already used in the system. This method (denoted CER) ranks questions according to increasing order of difficulty as determined by the domain experts. Second, we used two popular collaborative filtering methods that rank by decreasing predicted scores — a memory-based user-user KNN method using the Pearson correlation (denoted UBCF for user based collaborative filtering), and a *matrix factorization* (MF) method using SVD (denoted SVD) to compute latent factors of items and users [7, 36, 37]. In both cases we used the Mahout[2] implementation of the algorithms [38]. Finally, we implemented an approach that assigns questions based on a students' average score over topics. This method follows works that consider the student's mastery level of a topic when predicting performance [39]. We computed the average score 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

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

used only on the K12 dataset where we have available topic data.

The collaborative filtering algorithms described above all require an item score as an input. We computed scores as follows: 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 reduced this grade by 0.2 points in accordance with guidelines from the K12 educational expert team. For the PSLC dataset, we reduced the (normalized) elapsed time solving the question (of each attempt) from the score. The elapsed time is normalized to the scale $[0-1]$. In both cases, any negative score is converted to zero.

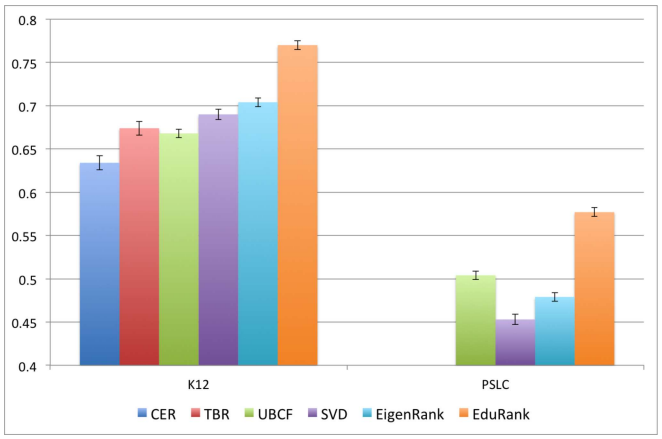
4.4. Results

We ran the following experiment— for each student s_i we split her answered questions into two sets of equal size: a training 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. Earier answers are in the training set, while 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. 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. 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 the questions in L_i by decreasing order of difficulty (harder questions were ranked higher in the list).

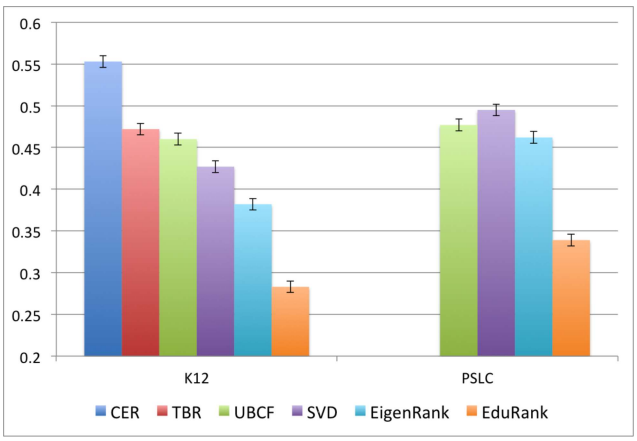
As can be seen in Figure 3, EduRank is better than all other approaches on both datasets using both metrics. The results are statistically significant ($p < 0.05$, paired t-test between EduRank and the leading competing algorithm).

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 es-



(a) AP score (higher is better)



(b) NDPM score (lower is better)

Figure 3: Performance Comparison (error bars represent the 95% confidence interval)

tablishing 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 important for this application.

Producing personalized difficulty rankings for students may be needed many times when used in a classroom, both for multiple students, and for different sessions of the same student. As such, it is critical to choose algorithms that can operate within reasonable time constraints. Table 3 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 answered questions. We report the overall time for computing a single difficulty ranking for all students and per student. Our experiments were conducted on a Mac Book Air 1.7GHz Intel Core i7 with 8GB RAM.

CER is obviously the fastest algorithm, as it requires no computation, only the retrieval of the fixed question difficulty level from the database. The memory-based algorithms are next, with UBCF being the fastest among the remaining algorithms, followed by EduRank and EigenRank. The SVD algorithm, requiring the construction of the matrix factorization model is the slowest here. That being said, as the number of students and questions grow, model-based approaches are expected to work faster than memory-based. In that case, we can move some computations to an initialization phase. For example, we can compute the similarity between students offline, and cache the top k nearest neighbors, reducing the online computation time only to the aggregation of the rankings of similar users.

4.5. Case Study

To demonstrate the behaviour of the various algorithms, we present the results of the algorithms for a particular student from the K12 dataset. Ta-

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**	**Multiply, 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**	**Multiply, 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**	**Multiply, Choose between 2**	**12**	**Multiply, 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	Multiply, Choose between 2	12
Multiply, 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
Multiply, Choose between 2	12	Multiply, 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	Multiply, Equals 54	12	Multiply, 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 2: Rankings outputted by the different algorithms for a sample target student

Algorithm	Run Time (Sec)	Time per Student (millisec)
CER	197.6	19.2
UBCF	445.2	43.2
TBR	625.2	60.6
EduRank	631.8	61.2
EigenRank	795.9	77.2
SVD	1490	144.4

Table 3: Execution Time

ble 2 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 (as computed by the grade of the student and her number of retries when solving the question). 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 vari-

ous 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.

4.6. Addressing the Cold Start Problem

In a real classroom new students join the system on an ongoing basis. At the onset, the system has little to no information on these students. This is known as the *cold start* problem in the recommendation systems literature [40]. To be able to use EduRank in a real classroom, we need to overcome situations in which there are not enough questions completed by a target student to search for similarities in the data set.

To tackle this problem, we incorporated a prior score $pr(q_k)$ for every question q_k in the training set by averaging over the scores of all students that solved this question in the training set. The difference between the prior score for a question pair (q_k, q_l) can be used as a proxy for the pairwise ranking between these two questions. We use a linear combination of the prior score and the student similarity score. To this end we replace Equation 8 used by the EduRank algorithm to compute the relative voting $rv(q_k, q_l, S)$ between the two questions q_k and q_l in the test set S given a training set T_i of questions for the target student i by the following equation:

$$\begin{aligned} rv(q_k, q_l, S) =& \alpha_{k,l} * sign(\sum_{j \in S \setminus i} s_{AP}(T_i, \succ_i, \succ_j) \cdot \gamma(q_k, q_l, \succ_j)) + \\ & (1.0 - \alpha_{k,l}) * sign(pr(q_k) - pr(q_l)) \end{aligned} \tag{10}$$

The value of $\alpha_{k,l}$ is set to the number of available rankings of (q_k, q_l) in the training set divided by the neighboring size used by the algorithm when selecting similar students. That is, as the number of students that have solved

Experiment	Training Data	Testing Data	% New Students
1	Week 1	Week 41	99%
2	Weeks 1-2	Week 41	78%
3	Weeks 1-3	Week 41	71%
4	Weeks 1-4	Week 41	66%
5	Weeks 1-8	Week 41	43%
6	Weeks 1-40	Week 41	6%

Table 4: Training and Testing data for Cold-Start Experiments

both q_k and q_l increases, the weight of the prior is reduced. Hence, as the amount of ranking information that is available in the training set increases, we rely more on this information and less on the computed priors. We note that our solution handles the cold start problem for new students, but not the cold start problem for new questions that are added to the system.

We study the effect of the amount of data collected about students on the prediction performance of EduRank, and the extended algorithm for handling new students, denoted EduRank+Prior. Table 4 demonstrates the training and testing methodology. We varied the training set to consist of the first week, first two weeks, first three weeks, first four weeks, first eight weeks and first 40 weeks of records in the system. The test set consists of the 41st week. The table also denotes the percentage of new students in the testing set for the corresponding training set. This setup was chosen to reflect the the use of the EduRank system in a real classroom, in which data that is supplied to the training algorithm from each student increases gradually.

Figure 4 shows the results of these experiments, comparing between the algorithm without the prior component and the algorithm with the prior component. We vary the amount of weeks worth of data in the training set, and also show the percentage of students in the test set not seen in the training set. The performance of both models are identical with only 1 week of training data. Then, the EduRank+Prior algorithm outperforms the basic EduRank algorithm, showing significant difference as late as the 8th week of training. As more weeks of training data are available, the ratio of new students drop, and EduRank increases the weight of the similarity score over the prior. When both algorithms gain substantial amount of 40 weeks of record data, their performance is identical as it was in week 1 (no statistically significant difference).

To show the benefit of using the EduRank+Prior algorithm when dealing

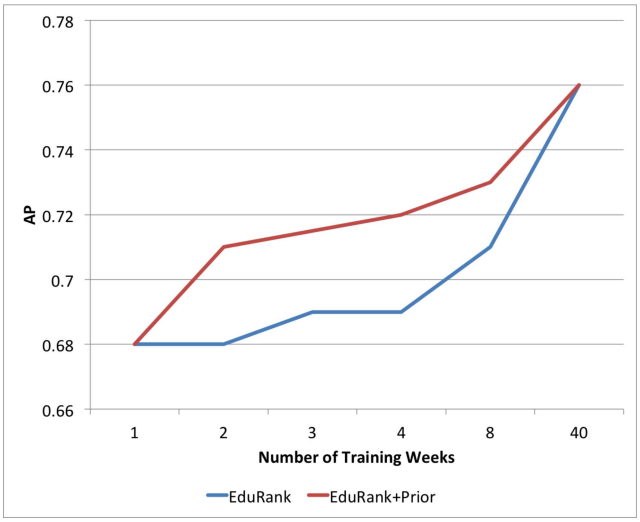


Figure 4: AP for EduRank algorithm with and without prior component

with new students, we evaluate the algorithm performance on specific students with low training data. For this test we randomly chose 50 students and removed most of their training data when predicting their test ranking. Specifically, in each such prediction step, 90% of training data of each target student was removed, while the training data of other students was retained and an overall prediction score for the target student was computed. We then compared the performance of EduRank with and without the prior component on this data set, averaging the results across all 50 random students. Our results indicate that the EduRank+Prior algorithm obtained an average AUC score of 0.72 while the EduRank algorithm (without the prior component) obtained an average AUC score of 0.64 for the target students.

Both of the above results demonstrate that using a prior-based approach is indeed beneficial for EduRank in cold start scenarios which are to be expected in real world situations.

5. Deployment in the Classroom

The previous section evaluated the EduRank approach on offline data. We now report on the deployment of EduRank in pedagogical context. The experiment was conducted during a summer school mathematics session in

Israel, from July 21, 2015 to August 15, 2015.[3]

The summer school is an elective school for strengthening Math and English as a second language capabilities conducted during the summer holiday. The school is run by professional Math and English teachers. Between classes the students were requested to practice using the K12 system, but this was not mandatory. The teachers and students were unaware of the experiment.

5.1. Methods

We compared two methods for sequencing questions to students. The first approach used the EduRank+Prior algorithm to order questions for students by increasing order of difficulty, as inferred by the algorithm. The second sequencing approach, denoted ASC, was based on pedagogical experts that created predetermined difficulty level for each of the questions in the K12 data set ranging from 1 (easiest) to 5 (most difficulty). The sequencing strategy determined by the pedagogical experts included a set of questions randomly sampled from the different level of difficulties as follows: 20% questions of level 1 difficulty; 30% questions of level 2 difficulty; 40% questions of level 3 difficulty; 10% questions of level 4 difficulty. The pedagogical experts decided not to include questions from the most difficult level. The questions were sequenced to students in ascending order of difficulty. Both of the approaches followed the mastery learning principle by which knowledge of simple skills should be demonstrated before moving on to more difficult questions relating to more complex skills [41] but they differed on how they selected questions for students.

5.2. Procedure

We conducted the experiment in two of the summer school classes using K12, one of which was randomly assigned to use the EduRank+Prior algorithm (denoted EduRank), while the other classroom was assigned to the ascending (ASC) algorithm. Each sequencing algorithm was used in real-time to present questions to students as part of their coursework. After submitting an answer, students received feedback from the system about their score for each question. The number of consecutive retries allowed for each question was limited to three. Students could also choose not to answer a question that is given to them by the algorithm. Students were randomly assigned to the

[3]IRB approval was obtained from Ben-Gurion University.

summer school classes and the questions presented by the system matched the topics covered in the classroom. Additionally, the curriculum was identical for the two classes (conditions). The data for the EduRank algorithm was updated each night after class to account for ongoing student learning. We hypothesized that using the EduRank approach to sequence questions to students would lead students to better performance on more difficult problems without reducing their motivation in the e-learning system, as compared to the alternative sequencing approach.

Both classes were administered a pre-test consisting of a single session of 15 questions in mathematics that were sampled from different topics and expert difficulty levels in mathematics. The questions for this preliminary test were chosen by a domain expert. There was no statistically significant difference between the two groups in each class in the average score on this preliminary test (ASC condition mean score was 0.64, STD 16 and EduRank condition mean score was 0.63, STD 18). Hence we asserted that students in each group exhibited similar knowledge baselines of the material.

5.3. Results

In the analysis that follows we restricted our attention to sessions of students who completed the preliminary session. We did not track individual students' progress during the study nor control any of the conditions in the classroom beyond the use of the sequence algorithm. Students could use the system with no supervision and practice as many questions as they wanted. Inspecting the two classes in combination, students solved on average 152 questions (stdev 169) and 94% of students reached and solved questions of levels 4 or 5. During the experiment no student ran out of problems to solve.

Table 5 shows the distribution of the number of students in each group, the number of questions solved, the average grade (first attempt), and the average time spent on each question (in seconds). As shown by the table, the average grade obtained by students using the EduRank approach was four points higher than the grade obtained by students using the sequencing approach. This difference is statistically significance (Repeated Measures ANOVA, $F(1, 36361) = 18.55, \quad p = 1.66 \cdot 10^{-5}$). When comparing the two approaches for each level of difficulty (Table 6), we can see that the students using the EduRank approach achieved higher scores for all questions in each of the levels. Specifically, for easy questions (level 1) the performance of the EduRank and ASC approach was identical. However, for more difficult questions (levels 2 and up) students using the EduRank approach achieved

Approach	Num. Students	Num. Questions	Avg. Grade	Time (Seconds)
EduRank	21	2027	93	27.0
ASC	17	1707	89	17.1

Table 5: Classroom Study Statistics

	Num. Questions		Avg. Grade			Num. Students	
Diff. Level	EduRank	Asc	EduRank	Asc	p-value	EduRank	Asc
1	607	466	93.8	93.8	0.15	21	17
2	423	430	**94.2**	90.8	$2.8 \cdot 10^{-7}$	21	17
3	449	578	**92.9**	86.7	$3.0 \cdot 10^{-14}$	21	17
4	368	233	**88.5**	83.0	$3.7 \cdot 10^{-4}$	21	15
5	180	—	84.0	—	—	18	–

Table 6: Performance comparison between the EduRank and ASC sequencing approaches by level of difficulty (p-value: Repeated Measures ANOVA)

higher performance than students using the ASC approach. Note that the ASC algorithm did not use questions from level 5.

Finally, Table 5 also shows that students using the EduRank approach solved more questions and spent more time in the system than students using the ASC approach. This implies that despite solving more difficult questions, the students using EduRank were as motivated to use the e-learning system as the students using the ASC algorithm.

6. Related Work

Our work relates to several areas of research in student modeling. Several approaches within the educational data mining community have used computational methods for sequencing students' learning items. Pardos and Heffernan [42] 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 a large number of sequences, as in the student ranking problem we consider in this paper.

Champaign and Cohen [43] suggest a peer-based model for content sequencing in an intelligent tutoring system by computing the similarity be-

tween 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. This way, we use the entire ranked list for similarity computation, and do not lose information. Consider, e.g., 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). Second, we are using social choice to combine similar students' difficulty ranking over questions. Third, we evaluate our approach on two real-world data sets. Li, Cohen and Koedinger [44] compared a blocked order approach, in which all problems of one type are completed before the student moves 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.

Multiple researchers 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 [39]. These researchers 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 [45, 46]. We solve a different problem — using other students' performance to personalize ranking over test-questions. In addition, these methods 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.

Approaches based on recommendation systems are increasingly being used in e-learning to predict students' scores and to personalize educational content. We mention a few examples below and refer the reader to the surveys by Drachsler et al. [47] and Erdt et al. [48] for more details. Collaborative filtering (CF) was previously used in the educational domain for predicting students' performance. Toscher and Jahrer [9] use an ensemble of

CF algorithms to predict performance for items in the KDD 2010 educational challenge. Berger et. al [49] 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. Schatten et al. [37] use matrix factorization for task sequencing in a large commercial Intelligent Tutoring System, showing improved adaptivity compared to a baseline sequencer. Finally, Loll and Pinkwart [50] use CF as a diagnostic tool for knowledge test questions as well as more exploratory ill-defined tasks. None of these approaches ranked questions according the personal difficulty level of questions to specific students.

7. Discussion and Conclusion

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 methods 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. We extended EduRank to predict the difficulty rankings of questions for students with little or no prior history in the system, and deployed this extended version in a real classroom. We show that sequencing questions using the EduRank approach in the classroom led students to exhibit better performance on more difficult questions, when compared to a baseline sequencing approach that was designed by domain experts.

Considering implementations in the wild, we note that EduRank was tested on two datasets with high sparsity ($> 98\%$) and showed good results.

Additionally, EduRank's exhibits polynomial computational complexity and was shown to be adequate for implementations in the wild, consuming 61.2 milliseconds to compute a ranked list per student on a Mac Book air computer.

We mention several limitations of the classroom experiment and subsequent suggestions for future work. First, our classroom deployment was conducted during an elective summer course in which the usage of the e-learning system was not mandatory. As shown in Table 5 the response time for students using EduRank was higher than that of the students using the alternative ASC ranking approach. This may mean that students in this group were more motivated to stay in the system. Hence we cannot directly claim our results would necessarily carry over to actual classrooms where students exhibit a variety of motivation levels. We are currently running experiments with using EduRank in a real classroom context in which students are assigned questions in increasing order of inferred difficulty that is outputted by EduRank. We are also combining EduRank with a multi-armed bandit approach to sequence questions to students [51].

Second, EduRank assigned more difficult questions (levels 4 and 5) than the alternative ASC algorithm (see Table 6), and may have adversely affected student motivation. In future work we mean to combine EduRank with cognitive models that directly account for skill acquisition and student engagement for sequencing educational content in the classroom.

Third, EduRank requires to rerun the algorithm to account for student learning over time. As students solves more questions, the change to their "learning state" is reflected in the change to the set of students deemed similar to them. Consequently, running EduRank again will output a new difficulty ranking that is adapted to their learning state. EduRank can be run every time it is necessary to account for effects of student learning on the inferred difficulty ranking. The complexity of running EduRank is low enough to enable EduRank to be run multiple times, depending on the needs of the teacher or education researcher. Indeed, in our study we ran EduRank the night after each class. The outputted set of questions for next class accounted for the student learning that occurred in the previous class.

References

[1] D. Sampson, C. Karagiannidis, Personalised learning: Educational, technological and standardisation perspective, Interactive Educational

Multimedia 4 (2002) 24–39.

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

[3] H. Ba-Omar, I. Petrounias, 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, IEEE, 2007, pp. 937–938.

[4] L. Zhang, X. Liu, X. Liu, Personalized instructing recommendation system based on web mining, in: Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for, IEEE, 2008, pp. 2517–2521.

[5] A. S. Najar, A. Mitrovic, B. M. McLaren, Adaptive support versus alternating worked examples and tutored problems: Which leads to better learning?, in: International Conference on User Modeling, Adaptation, and Personalization, Springer, 2014, pp. 171–182.

[6] C. Mazziotti, W. Holmes, M. Wiedmann, K. Loibl, N. Rummel, M. Mavrikis, A. Hansen, B. Grawemeyer, Robust student knowledge: Adapting to individual student needs as they explore the concepts and practice the procedures of fractions, in: Workshop on Intelligent Support in Exploratory and Open-Ended Learning Environments Learning Analytics for Project Based and Experiential Learning Scenarios at the 17th International Conference on Artificial Intelligence in Education (AIED 2015), 2015, pp. 32–40.

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

[8] G. Shani, A. Gunawardana, Evaluating recommendation systems, in: Recommender systems handbook, Springer, 2011, pp. 257–297.

[9] A. Toscher, M. Jahrer, Collaborative filtering applied to educational data mining, KDD Cup.

[10] N. N. Liu, 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, ACM, 2008, pp. 83–90.

[11] F. Brandt, V. Conitzer, U. Endriss, Computational social choice, Multiagent systems (2012) 213–283.

[12] A. Segal, Z. Katzir, K. Gal, G. Shani, B. Shapira, Edurank: A collaborative filtering approach to personalization in e-learning, in: Educational Data Mining 2014, 2014.

[13] A. C. Graesser, M. W. Conley, A. Olney, Intelligent tutoring systems, APA handbook of educational psychology. Washington, DC: American Psychological Association.

[14] R. F. Kizilcec, G. M. Davis, G. L. Cohen, Towards equal opportunities in moocs: affirmation reduces gender & social-class achievement gaps in china, in: Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale, ACM, 2017, pp. 121–130.

[15] D. Clow, Moocs and the funnel of participation, in: Proceedings of the Third International Conference on Learning Analytics and Knowledge, ACM, 2013, pp. 185–189.

[16] C. Zhao, L. Wan, A shortest learning path selection algorithm in e-learning, in: Advanced Learning Technologies, 2006. Sixth International Conference on, IEEE, 2006, pp. 94–95.

[17] N. Idris, N. Yusof, P. Saad, Adaptive course sequencing for personalization of learning path using neural network, International Journal of Advances in Soft Computing and its Applications 1 (1) (2009) 49–61.

[18] S. A. Al-Radaei, R. Mishra, A heuristic method for learning path sequencing for intelligent tutoring system (its) in e-learning, International Journal of Intelligent Information Technologies (IJIIT) 7 (4) (2011) 65–80.

[19] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Application of dimensionality reduction in recommender system-a case study, Tech. rep., DTIC Document (2000).

[20] Y. Koren, R. M. Bell, Advances in collaborative filtering, in: Recommender Systems Handbook, 2015, pp. 77–118.

[21] M. Weimer, A. Karatzoglou, Q. V. Le, A. J. Smola, Cofi rank-maximum margin matrix factorization for collaborative ranking, in: Advances in neural information processing systems, 2008, pp. 1593–1600.

[22] P. C. Fishburn, The theory of social choice, Vol. 264, Princeton University Press Princeton, 1973.

[23] A. H. Copeland, A reasonable social welfare function, in: University of Michigan Seminar on Applications of Mathematics to the social sciences, 1951.

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

[25] F. Schalekamp, A. van Zuylen, Rank aggregation: Together we're strong., in: ALENEX, 2009, pp. 38–51.

[26] 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, 2000, pp. 729–734.

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

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

[29] G. Shani, A. Gunawardana, Evaluating recommendation systems, in: F. Ricci, L. Rokach, B. Shapira, P. B. Kantor (Eds.), Recommender Systems Handbook, Springer US, 2011, pp. 257–297.

[30] E. Yilmaz, J. A. Aslam, 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, ACM, 2008, pp. 587–594.

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

[32] K. Wauters, P. Desmet, W. Van Den Noortgate, Item difficulty estimation: An auspicious collaboration between data and judgment, Computers & Education 58 (4) (2012) 1183–1193.

[33] Z. A. Pardos, N. T. Heffernan, Kt-idem: Introducing item difficulty to the knowledge tracing model, in: User Modeling, Adaption and Personalization, Springer, 2011, pp. 243–254.

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

[35] E. Kanoulas, 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, ACM, 2009, pp. 611–620.

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

[37] C. Schatten, R. Janning, L. Schmidt-Thieme, Integration and evaluation of a matrix factorization sequencer in large commercial its., in: AAAI, 2015, pp. 1380–1386.

[38] S. Schelter, S. Owen, Collaborative filtering with apache mahout, Proc. of ACM RecSys Challenge.

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

[40] A. I. Schein, A. Popescul, L. H. Ungar, D. M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02, ACM, New York, NY, USA, 2002, pp. 253–260. doi:10.1145/564376.564421.

[41] J. H. Block, P. W. Airasian, B. S. Bloom, J. B. Carroll, Mastery learning: Theory and practice, Holt, Rinehart and Winston New York, 1971.

[42] Z. A. Pardos, N. T. Heffernan, Determining the significance of item order in randomized problem sets., in: EDM, 2009.

[43] J. Champaign, 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.

[44] N. Li, W. W. Cohen, K. R. Koedinger, Problem order implications for learning transfer, in: Intelligent Tutoring Systems, Springer, 2012, pp. 185–194.

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

[46] M. H. Falakmasir, Z. A. Pardos, G. J. Gordon, P. Brusilovsky, A spectral learning approach to knowledge tracing, in: EDM, 2013.

[47] H. Drachsler, K. Verbert, O. Santos, N. Manouselis, Panorama of recommender systems to support learning, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer US, 2015, pp. 421–451.

[48] M. Erdt, A. Fernandez, C. Rensing, Evaluating recommender systems for technology enhanced learning: A quantitative survey, Learning Technologies, IEEE Transactions on 8 (4) (2015) 326–344.

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

[50] F. Loll, N. Pinkwart, Using collaborative filtering algorithms as elearning tools, in: 42nd Hawaii International Conference on Systems Science, 2009.

[51] A. Segal, Y. Ben David, J. J. Williams, K. Gal, Y. Shalom, Combining difficulty ranking with multi-armed bandits to sequence educational content, in: C. Penstein Rosé, R. Martínez-Maldonado, H. U. Hoppe, R. Luckin, M. Mavrikis, K. Porayska-Pomsta, B. McLaren, B. du Boulay (Eds.), Artificial Intelligence in Education, Springer International Publishing, Cham, 2018, pp. 317–321.