MOCHI: an Offline Evaluation Framework for Educational Recommendations

Chunpai Wang[1], Shaghayegh Sahebi[1] and Peter Brusilovsky[2]

[1]*University at Albany, State University of New York, Albany, New York, USA*
[2]*University of Pittsburgh, Pittsburgh, Pennsylvania, USA*

Abstract

Evaluating recommendation algorithms with long-term independently-measured rewards, such as educational recommender systems, has proven to be a difficult task, especially using offline data. While many use model-based evaluation strategies to evaluate such recommender systems, we argue that these strategies are unreliable, particularly due to biases introduced via simulation and reward estimation models. In this paper, we showcase this argument by experimenting with a state-of-the-art model-based evaluation model and presenting its flaws. Next, we propose MOCHI, an offline model-free evaluation framework that can be used on sparser collected data with longer trajectories. We experiment with MOCHI and show how it can be used to effectively evaluate educational recommender policies with long-term goals.

Keywords

educational recommender systems, offline evaluation, instructional sequencing

1. Introduction and Related Work

With the rise of online education, the size of classes and, as a result, the need to provide automatic guidance for students grows. Educational recommender systems and instructional sequencing policies aim to provide the best learning materials to students during their studies in online learning platforms [1, 2]. Despite the considerable application of these algorithms in online education platforms, effectively evaluating them has been proven challenging [3]. Particularly, because of scalability problems in case-based user studies and equity considerations in online A/B testing in this domain, offline evaluation strategies are essential for educational recommender systems.

We note that having delayed and independently-measured utility or reward is one of the main reasons for this evaluation to be challenging. Unlike consumer-based and commercial systems that aim to serve users' interests, the main purpose of educational systems is for students to learn. Research has shown that students' interest-based behaviors may not be aligned with their learning goals and can potentially be against them [4, 5]. As a result, the widely used implicit and explicit observable feedbacks, such as user ratings and clicks, are not adequate success indicators in evaluating an educational recommender system. Instead, long-term learning

measures, such as post-test score and learning gain, are more reliable to evaluate educational recommender systems' effectiveness in student learning.

These more reliable scores are, however, evaluated independently of the student trajectory items and as a result not directly observable from the trace data. For example, post-test scores are student grades in a test that is administered at the end of the course and may not include any problems that the student had practiced before. Additionally, these measures are delayed as they are collected at the end of student trajectory and can change as the students interact with the recommended learning materials. Accordingly, for a successful offline evaluation of educational recommenders using these measures, not only full user trajectories are needed, but also the training traces should have vast coverage of all possibly observable trajectories to facilitate a generalizable offline evaluation. These problems complicate the offline evaluation in educational recommender systems. A similar challenge exists in other recommender systems with delayed independently-measured rewards, such as health or weight-loss applications.

To avoid these problems, recent educational recommendation literature has sought model-based evaluation that simulates student trajectories and estimates the potential final reward for offline evaluation [6]. Model-based evaluation methods train a student model using the users' historical performance data from the logged system and then use it as a simulator coupled with a reward model to estimate the performance of a target policy [7, 8, 9, 6]. Yet, these evaluation strategies suffer from many problems. Most importantly, they rely on two estimators that can induce major errors. To simulate student trajectories, a student knowledge model should be used to estimate student knowledge and performance in the recommended items. Naturally, these models have an estimation bias that would exacerbate by the length of the simulated student trajectory. The reward estimator is usually trained using the learned student knowledge model parameters to predict student post-test score or knowledge gain. Again, not only the reward estimator model itself can be biased, but also relying on error-prone estimated parameters can intensify such a bias. Recently, efforts such as Robust Evaluation Matrix (REM) [10] have aimed to address some issues by testing the proposed policies against multiple student simulation models. REM has an innovative approach to explore experiment-worthy policies. It evaluates the policies in a conservative way: policy $\mathcal{A}$ is considered to be better than policy $\mathcal{B}$ only if policy $\mathcal{A}$ outperforms policy $\mathcal{B}$ under all student simulation models. So, if a policy shows to be better than other according to REM results, this policy would be a policy worth exploring in practice. Yet, REM suffers from other uncertainties that we present in this paper.

A more elegant model-free solution is *importance sampling*, which is popularly used in the field of off-policy evaluation in reinforcement learning [11, 12, 13, 14, 15]. The main idea is to re-weight the pre-collected reward from the logged policy to compute an unbiased estimate of the expected reward on a new compatible policy. However, this method requires the trajectory generated by the new policy to preexist in the old data generated by the logged policy. Otherwise, the importance sampling could yield an estimate with large variance, especially when we have very sparse observed rewards from the logged system. In other words, the existing model-free evaluations are not applicable to the offline evaluation of educational recommendation systems with a long trajectory and sparse reward. Given these challenges, having an offline evaluation framework that can handle delayed independently-measured rewards, is independent of recommendation and student models, allows for multiple item recommendations and user choice, and does not only rely on the superficial observed interest-based measures is essential

for the educational recommender systems.

In this paper, we first examine the REM framework and demonstrate the need for a model-free evaluation framework by showing the problems that arise in such model-based methods. Next, we propose Model-free Offline Correlational HIt (MOCHI), a model-free evaluation framework that can work with delayed independently-measured rewards with long trajectories. MOCHI evaluates if higher degrees of following a recommender system's non-trivial suggestions would be associated with higher independently-measured rewards. Experimenting with the offline data from a real-world online education platform, we show that the results generated by our proposed framework are in accordance with our expectations. Additionally, we present ways to interpret MOCHI's results. Our proposed evaluation framework is algorithm-agnostic and can be used to evaluate any adaptive or non-adaptive educational recommendation or instructional sequencing algorithm that either suggests or mandates the next item for the students to work on. It does not limit the number of recommended items to students, neither their trajectory length. Most importantly, it can be used for any application domain, such as health and fitness, in which a delayed long-term independently-measured reward is required rather than immediate superficial observations.

2. Model-Based Evaluation Challenges: A Case Study

In this section, we investigate Robust Evaluation Matrix (REM) [10], to argue that the existing evaluation methods are not sufficient to validate the effectiveness of educational recommendation policies.

2.1. Dataset

We use the data from the MasteryGrids platform [1], collected during Spring 2012, Fall 2012, and Spring 2013 semesters in the Java introductory course with the same curriculum at the University of Pittsburgh. MasteryGrids is an open-learner interface for an intelligent tutoring system, in which students can practice with various kinds of problems and annotated examples. In this paper, we use student trajectories in solving problems that ask the students to read a code snippet and answer simple questions, such as the final output or a variable's value. The items to be recommended in this system are these problems. In MasteryGrids, the programming problems are ordered from left to right by 21 curriculum topics. The topics that cover a wide range of concepts including simple `Variables` and more complex `Wrapper Classes` topics ordered by a domain expert. Each topic includes multiple problems. Although students can freely select any problem to work on, they typically follow the interface's topic order. The students take a pretest before starting their class and a post-test at the end of their course. We normalize the pre-test and post-test scores to be between zero and one. Also, we calculate students' knowledge gain by deducing their pre-test score from their post-test score. Score distributions are presented in Figure 1. In total, trajectories of 86 students with their pre-test and post-tests are available in the dataset. Descriptive statistics of the dataset are shown in Table 1.

---

[1]http://adapt2.sis.pitt.edu/wiki/Mastery_Grids_Interface

Table 1
Descriptive Statistics of MasteryGrids Dataset.

Dataset	#Users	#Questions	#Topics	#Records	Trajectory Length		
					Min	Median	Max
MasteryGrids	86	103	22	17741	11	168	988
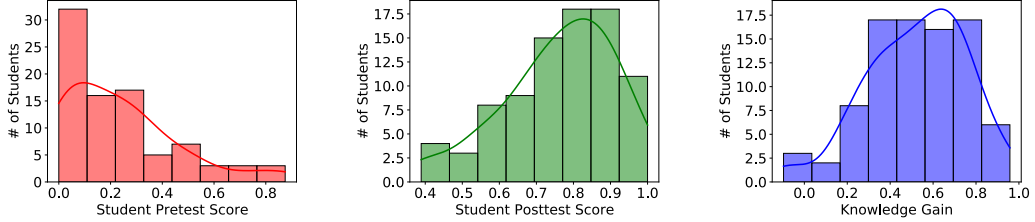
Figure 1: Distribution of Pre-test Score (Left), Post-test Score (Mid.), and Knowledge Gain (Right).

2.2. Setup and Expectations

Model-based evaluations in the education domain use a student model as a *simulator* to estimate the students' performance under the policies that are being evaluated. Since they are simulation-based, they have to also estimate the delayed reward or utility for each student using a *reward model*. Accordingly, in our experiment setup, we use three different student models to evaluate five recommendation policies. Additionally, we use multiple reward models to evaluate how a reward model would affect the evaluation results. We simulate $1,000$ student trajectories and sample the trajectory lengths and pretest scores from the pre-collected training data.
Simulator Student Models. We use the following models as student simulators:

- `Bayesian Knowledge Tracing (BKT)` is a pioneer model based on hidden Markov models that estimates student probability of success based on the probability that the student has learned a topic (mastery) [16].
- `Deep Knowledge Tracing (DKT)` is an LSTM [17] that predicts student's correctness probability according to their knowledge estimate [18].
- `Dynamic Key-Value Memory Network (DKVMN)` is a sequential key-value memory based deep model that represents the student's estimated knowledge in each latent concept [19]. This model has not been used in previous model-based evaluations.

Reward Models. The rewards models aim to estimate the final delayed reward for simulated students. They are trained on the simulator parameters that are learned according to training students' trajectories as their independent variables and the training students' final utilities (e.g., post-test scores) as the dependent variables. Since different simulator student models have different parameters, we have different parametrizations of the dependent variables for their reward models. Additionally, the model to estimate the final reward according to the independent parameters can be different. We use two different reward models for each student

simulator: a linear regression and a ridge regression model. Linear regression is selected according to REM. Ridge regression is chosen to increase linear regression's generalizability.

We also define different independent variables for each student model. For BKT, following [10], we use the trained parameters to infer the mastery probabilities of the 21 topics in MasteryGrids. So, BKT model's knowledge representation is a 21-dimensional vector. To train the reward model for BKT simulator, we fit the regression model with the concatenation of pre-test score and knowledge vector as independent variables to predict the post-test score. Knowledge gain can be calculated based on the difference between the estimated post-test and pre-test scores. For DKT, we use the last estimated student knowledge state vector at the final attempt concatenated with the pretest score as the input for the reward model. For DKVMN, we compute the knowledge state at each time step represented by a 10-dimensional vector [19, 20], 10 being the discovered latent concept size and concatenate it with student pre-test score.

Recommendation Policies. We consider the following standard educational policies that were evaluated in [10]:

- `Random`: is a non-adaptive policy that randomly recommends k learning resources to each student at each attempt.
- `InstructSeq`: is based on the designed instructional order and student's performance. If the student answers the current question correctly, the next k following questions in the instruction sequence will be suggested. Otherwise, the current question and the next $k - 1$ following questions are recommended.
- `Mastery`: is an adaptive policy that leverages BKT [2] student model to estimate student mastery probability at each attempt. It selects the top-k questions that are the farthest from the predefined mastery threshold level, which is usually set to 95%.
- `HighProbCorr`: is also an adaptive BKT-based policy that selects the next top-k questions that have the highest probabilities to be answered correctly by the student based on their current mastery probability estimates.
- `Myopic`: is another adaptive BKT-based policy that selects the next top-k questions that could lead to the largest estimated reward for the student.

Expectations. Among the above policies, we expect the `InstructSeq` policy to be a reasonably good policy, since the course topic sequence has been designed by the domain experts. Particularly, we expect `InstructSeq` to lead to better learning in students, compared to the `Random` policy. In addition, since the `Mastery` policy suggests the items which the student is least likely to have mastered, we anticipate it to recommend very difficult problems. Since the course covers a vast variety of topics with some being the most difficult and presented at the end of the semester, we expect this policy to always recommend the most difficult course problem to all students. Similarly, we anticipate the `HighProbCorr` policy to always suggest the easiest course problems to all students, since they are the most likely to be solved by them. As a result, we expect `InstructSeq` to also be better than both `Mastery` and `HighProbCorr` policies.

---

[2] https://github.com/myudelson/hmm-scalable

2.3. Experimental Results of Robust Evaluation Matrix

Here, we first present the predictive accuracies of different student and reward models. Then, we present REM evaluation results with different setups to show how student and reward model variations can affect the model-based evaluations. Since in REM students are assumed to always follow the recommended item, we recommend the top-1 problem.

Predictive Accuracy of Student Models. We train the three student models with 5-folds user-stratified cross-validation, and report the predictive accuracy of each model in Table 2. ROC-AUC stands for the area under the receiver operating characteristic curve, and PR-AUC denotes the area under the precision-recall curve. As we can see, they all have reasonably good predictive accuracy with the performance order DKT > DKVMN > BKT.

Table 2
Predictive Accuracy of the Student Models. The average performance and 95% confidence intervals are reported.

Dataset	BKT		DKT		DKVMN	
	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
MasteryGrids	0.7002 ± 0.0247	0.8251 ± 0.0259	0.7839 ± 0.0075	0.8759 ± 0.0135	0.7691 ± 0.0057	0.8666 ± 0.0081

Predictive Accuracy of Reward Models. We estimate each reward model's prediction error via 5-fold user-stratified cross-validation. The results are shown in Table 3. We can see that ridge regression's results are only significantly better than regression results in the DKT student model. This overfitting of linear regression can be explained by the large knowledge state representation size in DKT, that results in a high number of reward model parameters.

Table 3
Prediction Errors of 6 Reward Models with 95% confidence interval.

| Student Model	Reward Model with Linear Regression		Reward Model with Ridge Regression	
	RMSE	MAE	RMSE	MAE
BKT	0.1558 ± 0.0207	0.1300 ± 0.0158	0.1428 ± 0.0355	0.1175 ± 0.0282
DKT	0.3743 ± 0.2216	0.2812 ± 0.1700	0.1366 ± 0.0309	0.1101 ± 0.0259
DKVMN	0.1508 ± 0.0270	0.1209 ± 0.0237	0.1456 ± 0.0284	0.1209 ± 0.0204

REM Results with Linear Regression Reward Model. Table 4 presents the expected reward and its standard deviation over 1000 simulations under each combination of student simulation model and recommendation policy. We also show the heatmap of pair-wise Cohen's d effect size in Fig 2. Conventionally, $d = 0.2, 0.5$, and 0.8 respectively represent a 'small', 'medium', and 'large' effect size [21]. As we can see, for BKT student simulation model, the Random policy has a similar effect to InstructSeq, Mastery, and HighProfCorr; the Mastery policy is equivalent to HighProbCorr; and only the Myopic policy is the most different from all with the highest reward. On the other hand, for the DKT student model, we can conclude that InstructSeq equally contributes as HighProbCorr, and the Mastery policy results in the highest reward. For DKVMN, we have Myopic=InstructSeq=HighPropCorr. We can simply see that different simulator student models result in significantly different evaluations. REM selects one policy as the best (worst) policy only if it is the best (worst) in all simulation model experiments. Overall,

since the policy performance is not consistent across different student models, REM will not conclude that any of the policies are better or worse than others.

Table 4
Estimated Regression Rewards of Recommendation Policies under Different Student Simulation Models.

	Random	InstructSeq	Mastery	HighProbCorr	Myopic
BKT	0.7251 ± 0.2943	0.7596 ± 0.1656	0.6896 ± 0.2825	0.6992 ± 0.1175	0.8468 ± 0.3589
DKT	0.6858 ± 0.3636	0.6290 ± 0.3659	0.8120 ± 0.1077	0.5964 ± 0.1285	0.7537 ± 0.2228
DKVMN	0.7124 ± 0.0848	0.7160 ± 0.0976	0.7596 ± 0.0836	0.7368 ± 0.0874	0.7228 ± 0.0846
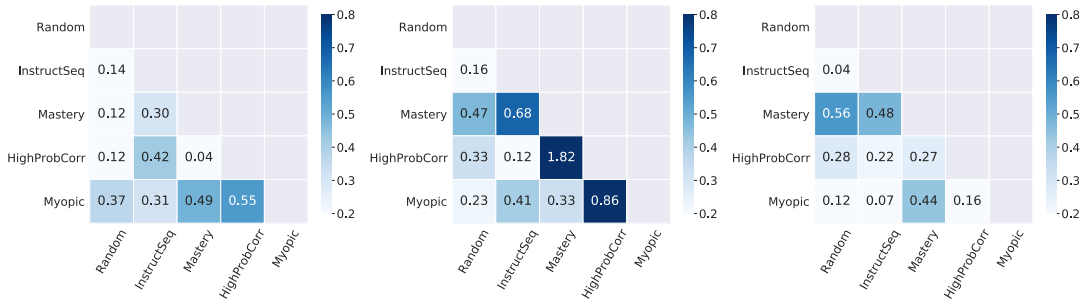
Figure 2: Cohen's d Effect Size for Policy Pairs under BKT (Left), DKT (Mid.), and DKVMN (Right) and Regression Reward Models.

REM Results with Ridge Regression Reward Model. Similarly, we show the estimated rewards for ridge-regression-based reward models in Table 5 and pairwise Cohen's effect size in Fig. 3. Again, with the BKT simulation model, the Myopic policy achieves the best rewards. With DKT, we can see that Mastery is better than HighProbCorr. But, with DKVMN, we see the reverse effect. The difference between policies with DKT and DKVMN policies are smaller, with the Cohen's d effect usually being smaller for DKVMN. As a result, similar to the previous analysis, REM will not conclude that any of the policies are better or worse than others.

Comparing the results of Tables 4 and 5 for each simulation model, the linear regression and ridge regression reward models can also lead to very different and even contradictory conclusions. For example, for DKVMN with linear regression model Mastery policy is better than Random and InstructSeq. But, with ridge regression, Mastery is not different from Random and InstructSeq policies.

Table 5
Estimated Ridge-Regression Rewards of Recommendation Policies under Different Student Simulation Models.

	Random	InstructSeq	Mastery	HighProbCorr	Myopic
BKT	0.7894 ± 0.0779	0.7732 ± 0.0744	0.8448 ± 0.0844	0.6789 ± 0.0374	0.9342 ± 0.0717
DKT	0.7525 ± 0.0148	0.7556 ± 0.0175	0.7665 ± 0.0124	0.7429 ± 0.0132	0.7570 ± 0.0148
DKVMN	0.7333 ± 0.0424	0.7457 ± 0.0297	0.7393 ± 0.0485	0.7541 ± 0.0291	0.7524 ± 0.0451
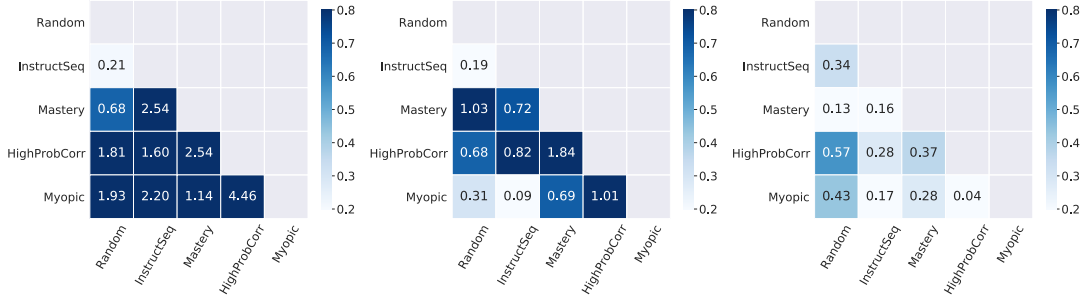
Figure 3: Cohen's d Effect Size for Policy Pairs under BKT (Left), DKT (Mid.), and DKVMN (Right) and Ridge Regression Reward Models.

2.4. Lessons Learned from Robust Evaluation Matrix

As we have seen in our experiment results, REM evaluation can be inconclusive and highly variant depending on the simulation and reward models. It also contradicts our expectations in Section 2.2. Here we discuss the potential problems that may lead to misleading results in REM.

As we have seen, using different simulation models can lead to different results in REM. The problem is that, in practice, it is not clear how many simulators and which classes of student models should be employed to be confident about REM results or any other model-based evaluations. Even if one policy is consistently better than others in all the applied simulation models, it may be contradicted in a new simulation model. Predictive accuracy could be one criterion for student model selection as suggested in [10]. However, since we have no standard of poor simulators, there is no guarantee that results under a particular student model with high predictive accuracy will always generate trustworthy results. Additionally, as we have seen in our experiments, the results of the DKT simulator, which had the best predictive performance, still contradicted our expectations from the policies.

Another similar issue comes from the reward modeling. Reward modeling is needed here to estimate the long-term independently-measured reward. But, simply relying on predictive accuracy of the reward model is inadequate, as we do not know how good of a reward model should be used. As we can see in Table 3, the reported expected prediction error is much higher than the standard error in REM shown in Table 4 and Table 5, which could be an indicator of a poor reward model. In addition, in the MasteryGrids dataset, we only observe a single reward (post-test score) for each trajectory, which is extremely sparse. As we can see, too many factors and variables affect the results of model-based evaluations. As a result, a reliable model-free offline evaluation strategy with fewer variations is needed to evaluate long-term independently-measured reward policies, such as educational recommender systems.

3. Model-free Offline Correlational HIt (MOCHI)

In this section, we present MOCHI, our model-free offline evaluation framework to validate the effectiveness of algorithms with long-term independently measured rewards, such as general educational recommender systems. We assume that a user who often follows a more effective

recommendation policy will have a higher final reward, such as a higher final course grade, compared to a student who adopts a less effective one. Given an offline previously collected dataset, our first counterfactual question is how to quantify if a student would follow the target recommendation policy if it has never been applied to the student. The second question is how significant the student's following of the target policy is, compared to having no, or a different, baseline policy. Finally, the third question is how to determine the effectiveness of a policy, given the significance of student trajectories that would have followed it if it was applied. In the following sections, we introduce our solutions to these questions as building blocks of the MOCHI framework.

3.1. Average Discounted Cumulative Hit

We consider an educational recommender system that ranks the top-k most useful items and learning resources for the target student at every student attempt. For instructional sequencing algorithms that only suggest one item to students, $k = 1$. According to our assumption, we would expect the students who studied the higher-ranked recommended learning resources to have better academic performance than those who chose the lower-ranked ones, or did not follow the recommendations. Hence, we need an "agreement" measure to determine how well a student follows the high-ranked recommended items by an algorithm. Having a ranked-based measure is particularly important in applications with smaller offline trajectory datasets in which users choose to interact with one or a few items at a time and the datasets may not cover all the possible combinations of trajectories. In that case, it is important to know how close the recommender system was to suggest the user's top choices, even if they were not ranked in the highest possible positions. Inspired by the Discounted Cumulative Gain (*DCG*), we design a new metric called *Discounted Cumulative Hit (DCH)* in Equation 1, that provides such a measure for one set of k recommended items to one target student at one attempt.

$$\text{DCH} = \sum_{j=1}^{k} \frac{HIT(j)}{\log_2(j+1)} \tag{1}$$

Here, $HIT(j) = 1$ if the target student had worked on the jth learning resource from the top-k recommendations, and 0 otherwise. According to Equation 1 the lower the selected learning material is in the recommended item list, the lower the *DCH* will be (with a logarithmic scale). Note that the *DCG* measure cannot be used in our problem directly, since its main assumption is that a gold-standard item-ranking is available from the user to be compared to the list of recommended items. In other words, to use *DCG*, users are assumed to be the best judges of their own interests and provide their interests in an ordinal format. However, in the education domain, such a ranked-list of learning resources by students in every step of their trajectories is not available.

3.2. Inverse Probability Weighted Discounted Cumulative Hit

So far, *DCH* only measures how in agreement an algorithm's suggestion is, with one attempt of a student. *DCH* can be used in a controlled online experiment to compare how much students'

choices agree with a recommender algorithm versus another. However, it is not adequate for offline evaluation, as the data is collected without the target algorithms' recommendations being presented to the user. In other words, if the data was collected in a system with no recommender algorithms, *DCH* cannot distinguish if this agreement is because the recommended item is something that the students would have selected even if it has not been recommended to them. We call recommending such an item a *trivial* recommendation. For example, a mandatory reading that is completed by everyone at the beginning of the course can be a trivial recommendation. Ideally, recommending a non-trivial item with a high utility should be more valuable than recommending a trivial item. Additionally, in educational systems with a predetermined order of topics, some students select the items within that fixed topic order. This can create a bias in the collected data, even if no recommender algorithm is used in the system.

In order to reduce this bias from the logged data, we borrow the inverse propensity scoring idea from importance sampling-based off-policy evaluation [22, 11] and normalize the *DCH* score by a propensity score ρ. This propensity score discounts the calculated agreement between the recommended item and the user-selected item by how trivial the item is. We call it *inverse probability weighted DCH (IPW-DCH)*, and formally define it as in Equation 2.

$$IPW\text{-}DCH = \sum_{j=1}^{k} \frac{1}{\rho_j} \frac{HIT(j)}{\log_2(j+1)} \tag{2}$$

In our experiments with no recommendation algorithm at the time of data collection, we simply use the bi-gram item probability as the propensity score. Given the current working item i in the training data, we use the conditional probability of next item j as in Equation 3. It can be interpreted as the sequential item popularity in the dataset.

$$\rho_j = \frac{\# \text{ question } i \text{ followed by question } j}{\# \text{ question } i}. \tag{3}$$

Note that ρ_j optimistically attributes all encounters of the target student following item j after item i to the system bias. Consequently, *IPW-DCH* is a pessimistic indicator of how frequently or favorably a student would "follow" the recommendation generated by the target policy. The propensity score ρ_j can be defined according to the application domain and data collection setup. For example, if a baseline recommender algorithm is active during the data collection, ρ_j should be updated to include the bias introduced by this baseline algorithm in addition to the system bias.

IPW-DCH focuses on one student's interaction in one attempt. But a student has a sequence of attempts and *IPW-DCH* should be extended to represent the whole student trajectory. Since different students have different trajectory lengths, we average all *IPW-DCH* scores of a student trajectory to represent their *Average IPW-DCH* score.

3.3. Correlation between Following Policy and Reward

Finally, with an effective educational recommender system, we expect the students who usually follow the recommendations to have a higher long-term utility. In the education domain, such utility would be a better academic performance or a higher knowledge gain. Therefore, in the

end, we evaluate our proposed model based on the correlation between *Average IPW-DCH* and students' academic performance. A stronger positive correlation indicates better performance on the task of sequential educational learning material recommendation. Particularly, we use Spearman's rank correlation coefficient in our experiments, which is defined as below, where $\mathcal{P}$ is the number of test students, and d_i is the difference in the ranks of the ith student in *Average IPW-DCH* and real rewards.

$$r_s = 1 - \frac{6\Sigma_i d_i^2}{\mathcal{P}\left(\mathcal{P}^2 - 1\right)} \tag{4}$$

4. MOCHI Experiment Results

To demonstrate our proposed evaluation framework, we run it on the real trajectories of the MasteryGrids data for recommending $k = 1$ and $k = 3$ items at each step. The results are presented in Table 6.

Table 6
Experimental Results of MOCHI.

Results of MOCHI for $k = 1$										
	Random		InstructSeq		Mastery		HighProbCorr		Myopic	
Values	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Avg. DCH	0.0097	0.0009	0.3311	0.1346	0.0070	0.0135	0.0143	0.0143	0.0080	0.0095
Avg. IPW-DCH	0.1362	0.1732	1.0493	0.6419	0.0415	0.1491	0.0800	0.1932	0.0521	0.1384
Spearman Correlations	Corr.	P-value	Corr.	P-value	Corr.	P-value	Corr.	P-value	Corr.	P-value
Avg. DCH vs Post-test	−0.0014	0.5111	0.2704	0.0118	−0.0047	0.9658	0.0589	0.5901	−0.0797	0.4657
Avg. DCH vs Knowledge Gain	0.0329	0.5026	0.0788	0.4707	0.0109	0.9210	0.0090	0.9345	0.0720	0.5099
Avg. IPW-DCH vs Post-test	0.0296	0.5225	0.2539	0.0183	0.0323	0.7680	0.0497	0.6494	−0.0580	0.5956
Avg. IPW-DCH vs Knowledge Gain	0.0502	0.4855	0.1442	0.1854	0.0039	0.9717	−0.0017	0.9874	0.0771	0.4805

Results of MOCHI for $k = 3$										
	Random		InstructSeq		Mastery		HighProbCorr		Myopic	
Values	Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
Avg. DCH	0.0208	0.0013	0.3611	0.1427	0.0126	0.0195	0.0345	0.0309	0.0201	0.0180
Avg. IPW-DCH	0.2710	0.2476	1.7750	1.3710	0.1254	0.2979	0.2171	0.3184	0.2316	0.4759
Spearman Correlations	Corr.	P-value	Corr.	P-value	Corr.	P-value	Corr.	P-value	Corr.	P-value
Avg. DCH vs Post-test	0.0054	0.4932	0.2813	0.0087	0.0106	0.9228	0.0472	0.6662	0.1454	0.1818
Avg. DCH vs Knowledge Gain	0.0153	0.4907	0.0931	0.3937	0.0294	0.7880	−0.0309	0.7778	0.0879	0.4211
Avg. IPW-DCH vs Post-test	0.0628	0.4725	0.3116	0.0035	0.0602	0.5821	0.1094	0.3158	0.1844	0.0892
Avg. IPW-DCH vs Knowledge Gain	0.0679	0.4560	0.1068	0.3276	0.0859	0.4316	−0.0449	0.6815	0.1355	0.2134

First, we check the Average DCH and IPW-DCH values. As we can see, the Average DCH values for InstructSeq is higher than all other policies. However, the standard deviation of Average DCH is the largest for InstructSeq, meaning that not all students follow this topic sequence and may need more guidance than the predefined topic order. The next, is High-ProbCorr Average DCH showing that a few tend to solve easier problems. Looking at Average IPW-DCH, values are the highest for InstructSeq. Meaning that, although InstructSeq suggests from the topic sequence, this suggestion is not trivial for all the students. Comparing the Random and HighProbCorr policies, although HighProbCorr has a higher Average DCH, its Average IPW-DCH is lower than Random. This shows the non-triviality of random suggestions, compared to the HighProbCorr ones. Interestingly, unlike when $k = 1$, the Myopic policy has a higher Average IPW-DCH compared to HighProbCorr, when $k = 3$. This can show that the Myopic policy has more non-trivial interesting suggestions in the second or third-ranked recommendations.

Looking at the correlation values, we can see that InstructSeq has the highest correlation values of Average DCH and IPW-DCH with both post-test and knowledge gain scores. Especially, its Average DCH and IPW-DCH values are significantly (p-value < 0.1) correlated with post-test scores. This means that students who followed the InstructSeq policy had higher post-test scores. Next, the Myopic policy's Average IPW-DCH has a significant (p-value < 0.1) positive correlation with students' post-test score when $k = 3$. Meaning that for Myopic policy to help students, it needs to suggest more items to students. The most reliable correlation with knowledge gain score is the positive relationship in Average IPW-DCH with $k = 1$. The rest of the correlations are insignificant and non-conclusive with large p-values. It can be because of the low number of data points, also reflected by low Average DCH values. But, it may also represent the ineffectiveness of the studied policies on student performance. Overall, our results show that InstructSeq is better than other policies when considering post-test score rewards. This is in agreement with our expectations in Section 2.2.

Additionally, in InstructSeq, we can see that when $k = 1$ the correlation with posttest score is lower than when $k = 3$. However, when $k = 1$ the correlation with knowledge gain is higher than the correlation with $k = 3$. This indicates that students with high post-test scores, and high knowledge gain benefit more when $k = 1$. But, students with high post-test scores, but low knowledge gain benefit more when $k = 3$. Meaning that a more strict recommendation ($k = 1$) is needed for the success of students with a lower prior knowledge. But, for students with an already high prior knowledge more freedom ($k = 3$) can be more beneficial.

5. Conclusions

In this paper, we investigated the state-of-the-art offline evaluation method, Robust Evaluation Matrix, on a real-world educational dataset. We found that model-based evaluations are not reliable, and their results can be contradictory and highly dependent on the student simulation and reward models. We concluded that a model-free evaluation method is necessary, especially for domains with delayed independently-measured rewards. We also proposed MOCHI, a model-free offline evaluation framework, as an additional tool for validating the recommendation policies, that does not rely on estimation models, can evaluate list recommendations, and only uses the collected offline data. In our experiments, we showed how MOCHI's results can be interpreted and that our proposed metric meets the expected results and can be an auxiliary tool of offline evaluation of educational recommender systems. MOCHI's limitations include the difficulty to work with policies that have very few instances of agreements with student trajectories in offline data, and hence, resulting in insignificant correlations. In future work, we would like to investigate more on our proposed method with online experiments.

Acknowledgments

References

[1] S. Doroudi, V. Aleven, E. Brunskill, Where's the Reward? a review of reinforcement learning for instructional sequencing, International Journal of Artificial Intelligence in Education 29 (2019) 568–620.

[2] M.-I. Dascalu, C.-N. Bodea, M. N. Mihailescu, E. A. Tanase, P. Ordoñez de Pablos, Educational recommender systems and their application in lifelong learning, Behaviour & information technology 35 (2016) 290–297.

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

[4] S.-Y. Teng, J. Li, L. P.-Y. Ting, K.-T. Chuang, H. Liu, Interactive unknowns recommendation in e-learning systems, in: 2018 IEEE International Conference on Data Mining (ICDM), IEEE, 2018, pp. 497–506.

[5] M. Mirzaei, S. Sahebi, P. Brusilovsky, Detecting trait vs. performance student behavioral patterns using discriminative non-negative matrix factorization, in: The 33rd International FLAIRS Conference, 2020.

[6] C. Mitchell, K. Boyer, J. Lester, Evaluating state representations for reinforcement learning of turn-taking policies in tutorial dialogue, in: Proceedings of the SIGDIAL 2013 Conference, 2013, pp. 339–343.

[7] M. Chi, K. VanLehn, D. Litman, P. Jordan, Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies, User Modeling and User-Adapted Interaction 21 (2011) 137–180.

[8] A. N. Rafferty, E. Brunskill, T. L. Griffiths, P. Shafto, Faster teaching via pomdp planning, Cognitive science 40 (2016) 1290–1332.

[9] J. Rowe, B. Mott, J. Lester, Optimizing player experience in interactive narrative planning: A modular reinforcement learning approach, in: Tenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2014.

[10] S. Doroudi, V. Aleven, E. Brunskill, Robust evaluation matrix: Towards a more principled offline exploration of instructional policies, in: Proceedings of the fourth (2017) ACM conference on learning@ scale, 2017, pp. 3–12.

[11] C. Voloshin, H. M. Le, N. Jiang, Y. Yue, Empirical study of off-policy policy evaluation for reinforcement learning, arXiv preprint arXiv:1911.06854 (2019).

[12] L. Li, W. Chu, J. Langford, R. E. Schapire, A contextual-bandit approach to personalized news article recommendation, in: Proceedings of the 19th international conference on World wide web, 2010, pp. 661–670.

[13] L. Li, W. Chu, J. Langford, X. Wang, Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms, in: Proceedings of the fourth ACM international conference on Web search and data mining, 2011, pp. 297–306.

[14] A. S. Lan, R. G. Baraniuk, A contextual bandits framework for personalized learning action selection., in: Proceedings of the 9th International Conference on Educational Data Mining (EDM-2016), 2016.

[15] X. Zhao, W. Zhang, J. Wang, Interactive collaborative filtering, in: Proceedings of the 22nd ACM international conference on Information & Knowledge Management, 2013, pp.

1411–1420.

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

[17] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.

[18] C. Piech, J. Bassen, J. Huang, S. Ganguli, M. Sahami, L. Guibas, J. Sohl-Dickstein, Deep knowledge tracing, in: Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1, 2015, pp. 505–513.

[19] J. Zhang, X. Shi, I. King, D.-Y. Yeung, Dynamic key-value memory networks for knowledge tracing, in: Proceedings of the 26th international conference on World Wide Web, 2017, pp. 765–774.

[20] C. Wang, S. Zhao, S. Sahebi, Learning from non-assessed resources: Deep multi-type knowledge tracing, in: Proceedings of the 14th International Conference on Educational Data Mining (EDM-2021), 2021.

[21] D. Lakens, Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and anovas, Frontiers in psychology 4 (2013) 863.

[22] J. P. Hanna, S. Niekum, P. Stone, Importance sampling policy evaluation with an estimated behavior policy, arXiv preprint arXiv:1806.01347 (2018).