Predicting Player Moves in an Educational Game: A Hybrid Approach

Yun-En Liu[1], Travis Mandel[1], Eric Butler[1], Erik Andersen[1],
Eleanor O'Rourke[1], Emma Brunskill[2], and Zoran Popović[1]
[1]Center for Game Science, Computer Science & Engineering, University of Washington
[2]School of Computer Science, Carnegie Mellon University
{yunliu, tmandel, edbutler, eland, eorourke, zoran}@cs.washington.edu, ebrun@cs.cmu.edu

ABSTRACT

Open-ended educational tools can encourage creativity and active engagement, and may be used beyond the classroom. Being able to model and predict learner performance in such tools is a critical component to assist the student, and enable tool refinement. However, open-ended educational domains typically allow an extremely broad range of learner input. As such, building the same kind of cognitive models often used to track and predict student behavior in existing systems is challenging. In addition, the resulting large spaces of user input coupled with comparatively sparse observed data, limits the applicability of straightforward classification methods. We address these difficulties with a new algorithm that combines Markov models, state aggregation, and player heuristic search, dynamically selecting between these methods based on the amount of available data. Applied to a popular educational game, our hybrid model achieved greater predictive accuracy than any of the methods alone, and performed significantly better than a random baseline. We demonstrate how our model can learn player heuristics on data from one task that accurately predict performance on future tasks, and explain how our model retains parameters that are interpretable to non-expert users.

Categories and Subject Descriptors

K.8.0 [**Personal Computing**]: General – Games; H.5.0 [**Information interfaces and presentation**]: General

Keywords

Educational games, user modeling

1. INTRODUCTION

Open-ended learning environments offer promises of increased engagement, deep learning, transfer of skills to new tasks, and opportunities for instructors to observe the learning process. One example of such environments is educational games, where players have an opportunity to explore and experiment with a particular educational domain [12]. However, many of these exciting potential applications require low-level behavioral models of how players behave. For example, if we can predict that a player will struggle with a particular concept, we could try to preempt this confusion with tutorials or choose specific levels designed to address those problems. Additionally, as forcing players to complete an explicit knowledge test often breaks the game flow and causes many players to quit, we could estimate a player's knowledge of target concepts by predicting performance on test levels that are carefully designed to measure understanding of those concepts. Finally, we might even be able to compare user populations by examining models learned from their data and hypothesize optimal learning pathways for each population.

Accurate predictions of user behavior have been achieved in existing educational software such as intelligent tutors [10, 9, 11]. However, we cannot directly apply such methods to educational games for two reasons. First, educational games often have very large state and action spaces. For instance, a game involving building one of 10 different structures on 100 locations has a state space of size 10^{100}. Second, games often increase engagement through the addition of game mechanics that are not directly linked to the main educational objectives. One option is to use expert insight to define skills and behavior associated with these skills for the educational game. However, doing so can be extremely labor intensive: for intelligent tutors for structured domains that often include activities labeled with skills, it has been estimated that 200-300 hours of expert development are necessary to produce one hour of content for intelligent tutors [4]. As educational games are more open-ended, allowing students to input a much wider variety of input compared to many popular intelligent tutoring systems, we expect that tagging and building structure models for them would be even more time consuming than for structured topics such as Algebra.

Given these limitations, we would like a method requiring minimal expert authoring, capable of inferring likely user behavior based on collected data. One popular approach with these properties from the field of recommendation systems is collaborative filtering [18, 21]. Collaborative filtering can be effective with no expert authoring at all if there is enough data; however, the large state space of many educational games often results in high degrees of data sparsity. To maintain accuracy in spite of such sparsity, there has been an emergence of hybrid models that supplement collaborative filtering with limited context-specific information when

there is not enough data [16, 24]. Though we are inspired by this work, such methods are not applicable to educational games: we cannot ask users for ranked preferences and are restricted to using behavioral models only, making our task significantly more difficult.

To address these challenges, we create a new ensemble algorithm that leverages the various strengths of multiple disparate models for predicting player behavior. We propose a tripartite methodology that combines elements of collaborative filtering with state-space clustering and modeling players as parameterized heuristic searchers. Using all three methods, we are able to achieve better performance than using any one of these approaches individually. The model reduces the log-likelihood to 68% of a random baseline, outperforming any of its components, which achieve between 73% and 80% log-likelihood of random. Because it uses both a mix of data-driven and model-based approaches, we are able to predict how people will react to any situation in the game, a capability that continues to improve as we observe more players. The model also retains interpretable parameters which we demonstrate by discovering differences in behavior between populations from different websites. Finally, we show that unlike pure collaborative filtering approaches, we can train our model on data from one level and use it to accurately predict behavior on future levels. This allows us to predict how players will respond in situations where we have no data at all, opening up a host of new applications such as adaptive level ordering or invisible assessment based on prediction of player performance on test levels.

2. RELATED WORK

2.1 Educational Technology

There has been substantial research on predicting student outcomes on tests. Some of these methods are based on dynamic assessment, an alternative testing paradigm in which the student receives assistance while working on problems [8, 13]. Intelligent Tutoring Systems (ITSs) include built-in scaffolding and hinting systems, and are therefore an ideal platform for studying dynamic assessment [10]. Studies have shown that this data has strong predictive power. Feng et al. show that 40 minutes of dynamic assessment in the ASSISTment system is more predictive of grades on an end-of-year standaradized test than the same amount of static assessment [9]. Feng et al. also showed that longitudinal dynamic assessment data is more effective at predicting strandardized test scores for middle school students than short-term dynamic assessment data [10]. Fuchs et al. showed that dynamic assessment data from third-grade students was useful for predicting scores on far-transfer problem-solving questions [11]. These methods are useful for predicting student outcomes on tests. However, we require much finer granularity for applications such as predicting how students will respond to new levels without any training data or offering just-in-time hints only when we predict the player is about to make a particular type of move.

2.2 Collaborative Filtering

Machine learning classification is often used to predict user behavior. However, many standard classification techniques are ill-suited for the educational game domain, due to the enormous set of possible inputs (classes) from which a player can choose. We also require a way to predict a player may make a new move that is possible, but has not been done by any previous player.

Another promising approach for predicting user behavior is collaborative filtering. It relies on the assumption that if two users have a similar state, and one user behaves in a particular way in response to a new situation, the other user will likely show the same behavior. A good survey of collaborative filtering approaches can be found in [21]. Several researchers have used these methods in the educational data mining domain, including using matrix or tensor factorization models to predict student item responses [7], or student performance on problems [22]. Unfortunately, their methods do not easily transfer to our problem, which involves predicting choices of transitions between game states instead of performance on problems given out one at a time; our data is simply much more sparse.

Of course, data sparsity is known to offer a key challenge to collaborative filtering, making it unable to issue accurate positions given very limited data [21]. Data sparsity is particularly problematic in our domain, an educational puzzle game with a large search space, because users diverge very quickly and most states are rarely visited. One way of alleviating data sparsity is to combine collaborative filtering with external information. Content-based approaches, which consider users or items to be similar if they share similarity in respect to features selected by a designer [19], can be used to augment collaborative filtering in situations where there is not enough data. For example, Melville et al. [16] use collaborative filtering in situations where there is enough data and look for similarities in content in cases where data is lacking. Ziegler et al. [24] propose a different model in which items are categorized into a taxonomy, and recommendations are made not only through user ratings but also categories of demonstrated interest. These methods do not directly apply in our domain, where we have only behavioral data and must predict transitions between states instead of rankings of items. However, we are inspired by their general ideas; more specifically, our method allows the designer to specify similarity functions to combat data sparsity and takes advantage of our domain structure by modeling players as heuristically guided probabilistic searchers.

2.3 Modeling Players in Interactive Games

Game reseachers have tried to estimate player preferences, skills, and behaviors based on in-game activities [6, 20]. Many of these approaches rely on expert-authored player models, although some have used data-driven techniques. For example, Pedersen et al. tried to predict the player's emotional state in Super Mario Bros by training a neural network on features such as number of deaths [17]. Weber et al. modeled player retention in a sports game with regressions to rank expert-chosen features such as playing style [23]. Our method differs by modeling low-level actions directly, a significantly more complicated task on which standard classification techniques are difficult to apply.

Some work has tried to predict low-level actions. Albrecht et al. used Dynamic Belief Networks to predict players' next actions, next locations, and current goals in an adventure game [3]. This work only applies to games with a very spe-

cific structure involving locations and quests, which we lack. Our work is probably closest to that of Jansen et al [15], who predicted moves of chess grandmasters by modeling them as low-depth heuristic searchers. Unfortunately, this method alone is not very accurate, and as we show later does not tend to improve as we collect more data. Our ensemble method relies on collected data wherever possible and models players as heuristic searchers only as a last resort, giving us significantly better predictive power.

3. GAME DESCRIPTION

We will first describe the game used in our analysis. Refraction is a educational fractions game that involves splitting lasers into fractional amounts. The player interacts with a grid that contains laser sources, target spaceships, and asteroids, as shown in Figure 1. The goal is to satisfy the target spaceships and avoid asteroids by placing pieces on the grid. Some pieces change the laser direction and others split the laser into two or three equal parts. To win, the player must correctly satisfy all targets at the same time, a task that requires both spatial and mathematical problem solving skills. Some levels contain coins, optional rewards that can be collected by satisfying all target spaceships while a laser of the correct value passes through the coin.

At any point in a level, the player may pick up a piece on the board or drop a piece currently not in play onto the board on any location. Let b be the number of open board locations (about 100), and p the number of available pieces (usually at least 6). Then the size of the state space is approximately b permute p, the number of permutations of open board locations for the pieces, and has a branching factor of about bp. Thus the overwhelming majority of game states and transitions have never been observed, a situation common in open-ended educational environments.

In the analysis that follows, we primarily use player data gathered from level 8 of Refraction, the first non-tutorial level. This level was chosen because it was the non-tutorial level for which we had the most data. The layout of the level can be seen in Figure 1.

4. PREDICTIVE TASK

Our objective is to predict player behavior in the educational game Refraction, similar to how student models in intelligent tutoring systems can be used to predict student input. We now define some of the notation we use in the rest of the paper. For a given level, our task is the following. Let $\mathbf{S}$ be the set of all possible *game states* on the level. A game state is a particular configuration of pieces on the board, independent of time. Each player i in a set of players $\mathbf{P}$ of a level goes through a series of game states. We are concerned with predicting the next substantive class of move the player will try, so we preprocess the data to eliminate consecutive duplicate states, leaving us with the list of player's states, $S_{i,1}, \ldots, S_{i,m_i}$. We define a set of collapsed states, $\mathbf{C}$, and a *collapse* function mapping $\mathbf{S} \rightarrow \mathbf{C}$. These are selected by the designer to reduce states to features of interest, as in Table 1. For $s \in \mathbf{S}$, define $succ(s)$ to be the set of collapsed states reachable in one action from s, i.e., $succ(s) = \{collapse(s') \mid s'$ is reachable in one move from $s\}$. The predictive model M assigns a probability that the player will enter a collapsed state depending on his history. Given player i's sequence of states up to time $j \leq m_i$, $S_{i,1}, \ldots, S_{j-1}$, we want to predict the probability of them entering a collapsed state at time j, $Pr(collapse(S_{i,j}) \mid S_{i,1}, \ldots, S_{i,j-1})$, where $\sum_{c \in succ(S_{i,j-1})} Pr(c \mid S_{i,1}, ..., S_{i,j-1}) = 1$. The total probability of the player's sequence of states, P_i, under the model is then $Pr(P_i \mid M) = \prod_{j=1}^{m_i} Pr(collapse(S_{i,j}) \mid S_{i,1}, \ldots, S_{i,j-1})$. The total probability of the set of players' traces P is $Pr(P \mid M) = \prod_{i \in \mathbf{P}} Pr(P_i \mid M)$.

The choice of *collapse* is left up to the designer and depends on the intended application. Prediction of player search behavior in a game with maze-like elements, for example, may only require the model to predict where the player will move to next. Or, a system designed to give mathematical hints might only require a model capable of predicting the value of the next fraction the player will produce. In Refraction, we are primarily concerned with how the player will use pieces and manipulate the laser. This gives us good, though incomplete, overall information on their playing ability, and is described in Table 1.

Figure 1: A game state from level 8 of Refraction, on which we will perform most of our analysis. The pieces are used to split lasers into fractional amounts and redirect them to satisfy the target spaceships. All ships must be satisfied at the same time to win.

Feature	Value
Fringe lasers	2 1/2,East
Pieces used	1 W-NS, 2 S-E, 1 N-E, 1 W-N
Ship values satisfied	(none)
Pieces blocking lasers	Benders: 1 S-E
% coins satisfied	0.0

Table 1: The *collapse* function we use in Refraction, applied to the state in Figure 1. States that share all feature values are considered the same. Pieces are listed as (input)-(outputs) in cardinal directions, such that W-NS is a splitter with a westward input and north-south outputs. Fringe lasers are those at the edge of the laser graph either entering the wrong kind of ship or not entering a piece at all.

5. METRICS

In this section we explain how we will evaluate the performance of our predictive model. Our aim is to build models

that accurately reflect how populations behave in Refraction levels. We use a similar evaluation metric as [15] and [5] by measuring the information content, in bits, of the population under our predictive model. This number is easily calculated as: $I(P \mid M) = \log_2 Pr(P \mid M)$. We then compare the information content of the data under our model as compared to the information content of the data under a random model, $Compression(P \mid M) = \frac{I(P|M)}{I(P|Random)}$. In general, the goal is to find M maximizing $Pr(P \mid M)$, which corresponds to minimizing $Compression(P \mid M)$. We choose this metric as it offers some interpretability, with 0 indicating perfect ability to predict every move and 1 indicating no compression. This metric also retains the same meaning and scale regardless of the number of players. Unless otherwise stated, all evaluations of goodness of fit are done with 5-fold cross-validation on 1000 players, drawn at random from players from the website Kongregate from a period of August 2011 to September 2012.

6. HYBRID BEHAVIOR PREDICTOR

Here, we describe the three portions of our hybrid predictive model and describe the conditions under which each is used. Each individual method has different benefits and drawbacks and is suitable at a different level of data. We use a combination of them to keep all their benefits, giving us good predictive power, interpretability, and generalizability. At the end, we describe the full model in detail.

6.1 Markov

Collaborative filtering models, which search for similar players and use their data to predict the behavior of new players, are an attractive approach for our problem space because they are data-driven and model-free. There are a number of methods for determining the similarity of two players. We describe and compare two methods: a simple Markov model with no knowledge of player history and a model with full awareness of player history.

In the simple Markov model, we compute the probability of a state transition based only on the player's current state. To estimate these state transitions, we use our prior data, aggregating together any examples which start in the same initial state. To prevent the probability of a player from going to 0 when they make a transition that we have not seen before, we add a smoothing parameter r. With r probability, the player will choose between the possible successor states $succ(S_{i,j-1})$ randomly, and with the remaining $1 - r$ probability, the player will move according to the Markov model as outlined above. We empirically determine that the best performance is achieved with $r = 0.3$.

One weakness of this similarity metric is that it ignores player history. We also attempted other collaborative filtering models. For example, we could consider using only the transitions from other players with the same full history of moves on that level when issuing predictions. In the limit of infinite data, we would expect this model to outperform all others based only on order of visits to game states.

We found, however, that the performance of the second history-aware model is worse than the performance of the simple Markov model. The comparison is shown in Figure 2. The underlying issue is that for most observed paths, no previously observed player has followed the exact same path of results. The history-aware model can perform no better than random in these cases, explaining its poor performance. After experimenting with several different collaborative filtering-based models, we settled on the pure Markov model described first as the most straightforward and accurate approach, achieving a base compression of 0.756.

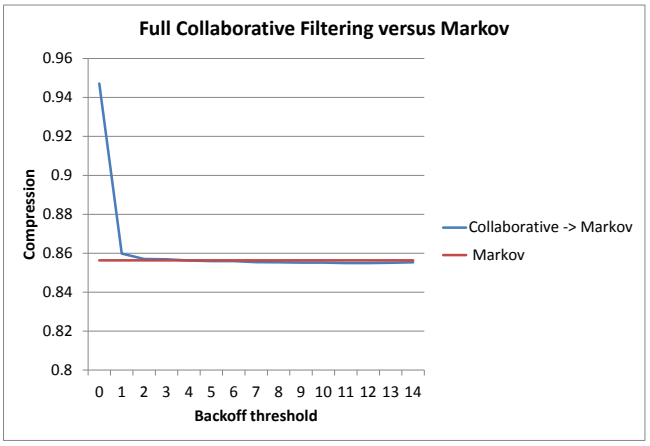


Figure 2: Effect of collaborative filtering models where players are non-Markovian. We create a hybrid model where we first attempt to find similar players under exact path matching, and if there are fewer than n of them, we consult a simple Markov model instead. The Markov model is the same or better at predicting player moves.

6.2 State Aggregation

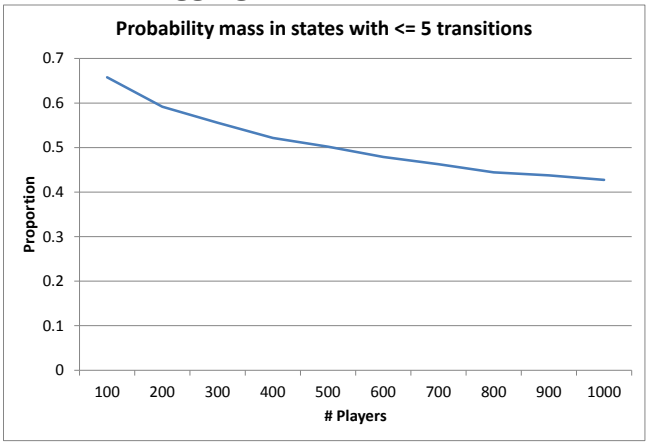


Figure 3: Many moves occur in locations where we have little data, even with 1000 players.

In the limit of infinite data, we would expect the Markov model to outperform all other methods that make the same assumption that players are history-free. However, the amount of data required for good performance can be quite high. In our case, this problem is compounded by the fact that puzzle game state spaces are difficult to search by design. As a result, most states are visited infrequently, as is shown in Figure 3. It is challenging to predict how new play-

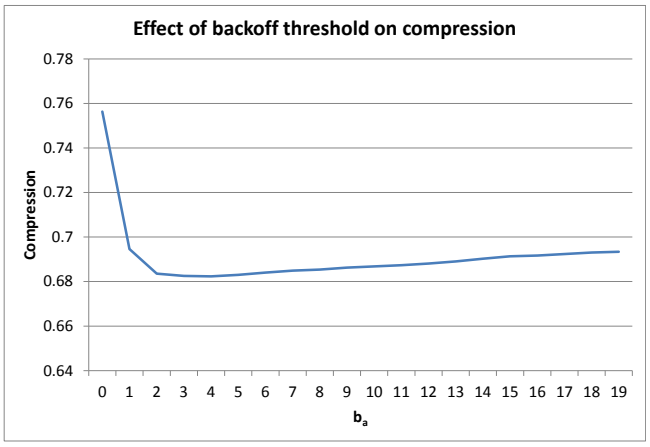


Figure 4: Selection of b_a, the backoff parameter controlling when to consult transitions from all similar states instead of exact states.

ers will behave when they reach these uncommonly-visited states.

One way to address this data sparsity problem is to aggregate data from states that are similar, and use this aggregated data to make predictions. This requires only a minor modification to the original Markov model: instead of looking at the distribution of transitions from the player's current state, we look at the distribution of transitions from all states similar to the player's current state. Here, we use *collapse* to find similar states, though the designer could substitute other functions if desired. To determine when to use this state aggregation approach, we introduce a backoff parameter b_a. When a state has been visited by fewer than b_a players, we switch from the Markov model to the aggregation model.

The aggregated states are not exactly the same as the current state because the *collapse* function throws away some amount of useful data. Thus, we would expect this approach to be most beneficial when data is extremely sparse, and become progressively less beneficial as we gather more data. This is exactly the effect we observe, as seen in Figure 4. In general, the best value of b_a depends on the similarity of player's traces through the state space. For Refraction, the optimal value is $b_a = 4$. The overall compression drops from 0.756 to 0.682 when using an approach that combines state aggregation and the Markov model, compared to using the Markov model alone. Applying state aggregation blindly in all cases erases most of the improvement and results in a compression of 0.732, so it is important to tune this parameter correctly.

6.3 Player Heuristic Search

Both of the previously described models have certain advantages. They both require minimal designer input: state space and transition functions for the Markov model, and a similarity function for the aggregation model. Both models also only improve as more data is gathered. Unfortunately, these methods also have two significant drawbacks: they perform poorly when there is very little data available, and they have parameters that are difficult to interpret. An educator trying to determine whether a game level teaches a particular math strategy, for example, would have difficulty learning this from the transition probabilities of a graph with tens of thousands of states.

In order to address these shortcomings, we use a method that models how players explore the game space in cases where data is particularly sparse. We assume that players are heuristically-guided probabilistic searchers. This assumption is reasonable given that players are attempting to solve fraction puzzles which are fun precisely because the solution is not obvious. This allows us to utilize information from every game state and generalize that information to new states. In comparison, the Markov with state aggregation approach can only utilize data from similar states. We expect this heuristic search approach to be most effective when data is scarce. Since the search model is only an approximation of player behavior, this method will become worse relative to the Markov with state aggregation approach as data become more plentiful, since the Markov approach has the power to precisely fit player behavior with enough data.

We provide a brief summary of the player heuristic search algorithm here, but for a full formalization of the algorithm please refer to Jansen et al. [15]. Note that we make a few modifications to the Jansen algorithm, described below. Our search algorithm assumes that users select moves by following a heuristic function v, which determines the likelihood that a player will visit a particular collapsed state. The function v is a weighted linear sum of simple designer-specified functions $a_1, \ldots, a_n$ that operate on collapsed states $c \in \mathbf{C}$: $v(c) = \sum_{k=1}^{n} \lambda_k a_k(c)$. Players, when they make a move, apply the heuristic to each possible collapsed successor in $c \in succ(S_{i,j-1})$ and assign it probability mass $e^{v(c)}$ to prevent negative probabilities, given by (1).

$$Pr(collapse(S_{i,j}) \mid S_{i,j-1}, \lambda_1, \ldots, \lambda_n) = \frac{e^{v(collapse(S_{i,j}))}}{\sum_i e^{v(C_i)}} \quad (1)$$

We optimize the weights λ_k to maximize the log-likelihood of the data using Covariance Matrix Adaptation: Evolutionary Strategy, an optimizer designed to run on noisy functions with difficult-to-compute gradients [14]. Our algorithm differs from the original in that both the possible successor states and the state that the heuristic operates on are collapsed states, since we want to predict the general type of move players will make rather than their exact move. As before, we also introduce a backoff parameter b_h. When searching for transitions from aggregated players, if there are fewer than b_h datapoints, we switch from the Markov with aggregation model to the heuristic model. Empirically we discover that the optimal value is achieved at $b_h = 4$.

The base heuristics $a_1, \ldots, a_n$ are designer-specified, and should reflect the components of the game that players pay attention to while choosing between moves. The heuristics we use for Refraction are listed in Table 2. In practice, the selection of heuristics is closely related to the definition of the *collapse* function used to project raw game states in the prediction task, since both are chosen according to the game features that the designer views as important.

Table 2: Basic heuristics and values in Figure 1

Heuristic	Value in above state
% ships satisfied	0
% ships matching fringe laser values	1
% pieces used	5/6
% pieces blocking lasers	1/6
% coins satisfied	0

The model of player heuristic search allows us to predict subsequent moves even when a player is visiting a state that has never been seen before. Furthermore, the weights λ_k that are optimized in this model are interpretable; they tell us how the population of players values each of the game features defined in the heuristics a. This information can help designers learn about their games, as described in Section 8.

6.4 Full Hybrid Model

Tying all the components together, we now provide a description of the full hybrid prediction model for Refraction, which combines the simple Markov model, state aggregation, and player heuristic search.

1. Assume a player is currently in state s_a.
2. Consult the Markov model for all other players' transitions to collapsed states from state s_a. If there are b_a or more transitions, predict the observed distribution with the random action smoothing parameter of r.
3. Otherwise, apply the state aggregation function to all the nodes in the graph, and count all transitions from all states with collapsed value $collapse(s_a)$. If there are b_h or more transitions, take the observed distribution, remove any states impossible to reach from s_a, and predict the resulting distribution smoothed by r.
4. Otherwise, apply the heuristic with parameters learned from the training set to each of the successors using Equation (1) to get the probability of each transition.

7. EXPERIMENTS

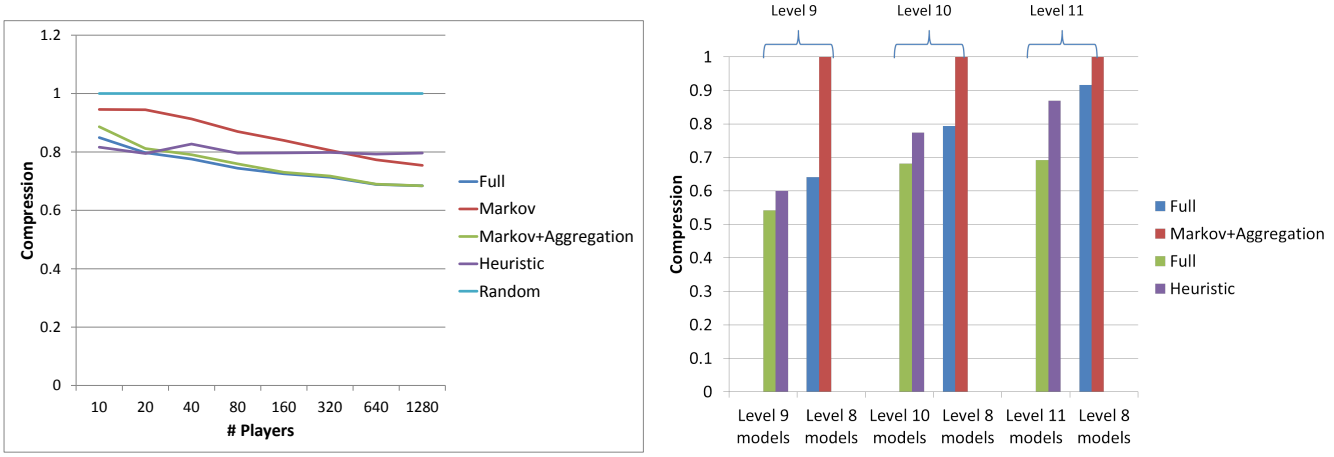
We now evaluate the performance of our predictive model. The performance of the full backoff model, from Markov to state aggregation to player heuristic search depending on the available data, is shown in Figure 5(a). Some features of the graph are worth noting.

- The full model is superior to any of the individual models at nearly all tested levels of data, with the Markov with state aggregation a close second.
- The data-driven approaches continuously improve as more data as added.
- The heuristic function is superior at the start, but its performance does not improve very much as more players are added. This is almost certainly because the model makes very strong assumptions about how players behave that allow it to take advantage of extremely sparse data; however, because the model is not completely accurate, it contains inherent bias that no amount of data will remove.
- The gap between Markov, Markov with state aggregation, and the full backoff model narrow as the amount of data increases. As we gather more players, the amount of probability mass on players in uncommonly visited states shrinks, so the Markov model is used to predict player behavior in more and more situations.

While the heuristic portion of the model seems to offer only incremental improvements, its true power can be seen when we attempt to generalize our models to future levels, as shown in Figure 5(b). Using 1000 players, we first learn heuristic parameters from level 8. We then use the learned heuristic to predict player behavior on levels 9, 10, and 11, comparing these to a Markov with state aggregation model trained on level 8. To get a sense of what "good" performance might look like, we also train player search heuristics and full models learned on the transfer levels and evaluate their compression values with 5-fold cross-validation as usual. We note some features of the graph here.

- We see immediately that the Markov with aggregation portion of the model has no generalization power at all. The state space and possible transitions via *succ* are completely different on future levels, so it's impossible to find similar players and use their transitions to predict moves later on.
- The heuristic portion of the model, on the other hand, allows it to predict what players will do in future levels. When compared to full models fit directly to player data from those levels, it is very good at predicting behavior on level 9, somewhat good at predicting behavior on level 10, and not very good at predicting behavior on level 11. Educational games are explicitly designed to teach players new and better strategies as they play, so we would expect performance to decrease over time.
- In addition, we can see that by level 11 even a heuristic trained on player data from that level is losing power. This means that the features of the state space players pay attention to is no longer being captured by the component heuristic functions $a_1, \ldots, a_n$. As the game introduces new concepts such as compound fractions, equivalent fractions, and fraction addition, players will need to pay attention to more features of the state space than are represented in our choice of $a_1, \ldots, a_n$. This speaks to the importance of choosing these component heuristics for the method's performance.

We caution that the generalization power of our model in these open-ended learning domains can only reasonably be expected to be high for the next few tasks and will be poor if those tasks have radically different state spaces from the training tasks. These caveats notwithstanding, these are promising results that suggest the learned heuristic captures something fundamental about how people navigate the search space of a Refraction level. This could potentially allow designers to guess how players at a certain point in time will behave on levels without ever needing to release updated versions of the game, or allow educators to simulate and evaluate user performance on assessment levels without



(a) Performance

(b) Generalization

Figure 5: Performance and generalization power of our model. The full model is superior to the other models at most amounts of data. In addition, the full model learned from level 8 data is able to generalize to predict behavior in future levels due to the heuristic component.

Table 3: Heuristic parameters learned on different player populations

Population	% ships	% ships matching	% correct pieces	% incorrect pieces	% coins	Compression
Kongregate	2.529	0.116	0.317	-13.391	0.937	0.782
BrainPOP	1.868	-0.149	1.584	-8.527	0.665	0.862

needing to interrupt player engagement by actually giving these tasks.

8. INTERPRETABILITY

One of the key drawbacks of model-free methods are that their results are extremely difficult to interpret, even for experts. The Markov model with state aggregation suffers from this problem, as it is essentially a black box that predicts distributions of transitions. The learned heuristic parameters, on the other hand, offer some glimpses into player behavior. We demonstrate this capability by analyzing some ways in which players differ between two populations. The first is Kongregate, a website whose main demographic is age 18-24 males [2], whose data we have been using up until this point. The second is BrainPOP, a website aimed at schools whose main demographics are children and middle-aged women (who are likely teachers) [1]. We learned heuristic weights on 1000 randomly selected players from each population, shown in Table 3. The goal is not to study how different populations interact with educational puzzle games, so we will not dwell on these results; we simply want to show how these parameters can lead to interesting hypotheses about player behavior.

Two interesting differences are immediately apparent based on these parameters. First, Kongregate players have a stronger negative weight on states with incorrectly used pieces as compared to BrainPOP players, suggesting they are less prone to placing pieces incorrectly. Second, BrainPOP players seem more interested in merely placing pieces down given the relatively high weight on used pieces parameter. Given that they also compress more poorly, one possible explanation is that they have less coherent plans and so place pieces more randomly. These hypotheses cannot be verified merely by looking at the heuristic values, but are sufficiently concrete that we can now run statistical tests to check their validity. For the following analyses, we use the non-parametric Wilcoxon rank-sums test due to the non-normality of our data. As we perform two tests on a dataset after learning parameters from that same dataset, we use the Bonferroni correction to avoid false positives; thus the threshold significance value is set at $\alpha = 0.025$. We report effect sizes as r values, with 0.1 considered small, 0.3 medium, and 0.6 large.

To see if Kongregate players understand piece directionality better than BrainPOP players, we assign to each player the proportion of piece placements such that a laser hits the dropped piece from an incorrect side. We discard players who place no pieces. We find a statistically significant effect of Population on Proportion Drops Incorrect (W=728148, Z=-18.06, r=0.4, p <0.0001), with Kongregate players having a median 0% incorrect drops (N=973) and BrainPOP players having a median of 12% incorrect drops (N=969).

Next, to see if BrainPOP players act with less foresight, we ask how often players make a move, only to take it back immediately. More precisely, for a player who traverses states $s_a, s_b, s_c, s_b, s_c, s_a$, we look at all the triples of moves: (s_a, s_b, s_c), (s_b, s_c, s_b), (s_c, s_b, s_c), and (s_b, s_c, s_a). We then assign to this player the proportion of triples in which the first and third state are the same, discarding players who visit fewer than three states. We find a statistically significant effect of Population on Proportion Takebacks (W=651589, Z=-23.35, r=0.53, p $<$ 0.0001), with Kongregate players having a median of 13% takeback moves

(N=968) and BrainPOP players having a median of 32% takeback moves (N=971).

These analyses show that the learned parameters in our hybrid model can be valuable tools for game designers, educators, and researchers for analyzing how populations use their systems. For instance, because Kongregate players are primarily adults and BrainPOP players are primarily children, we might wonder if children have more difficulty understanding piece directionality and spatial reasoning and plan their moves less carefully than adults do. A researcher might attempt to generalize these results to other strategic tasks, while a game designer might create a different version of Refraction with easier levels, fewer pieces, and clearer graphics for children. Either way, the learned parameters are a useful tool to help understand how players behave.

9. CONCLUSION

Predicting player behavior in open-ended learning environments is an interesting and complex problem. This ability could be used for a host of automatic applications to bolster engagement, learning, or transfer. In this paper, by using a combination of data-driven and model-based approaches, we presented a "best-of-all-worlds" model able to predict player behavior in an educational game. First, our hybrid model's performance is better than any individual component's. Second, the learned weights of the sub-heuristics are human-readable and can give insights into how players behave. We used these parameters to formulate hypotheses about how two populations behave differently and confirmed them with strong statistical results. Finally, we demonstrated how the heuristic portion of the model allows us to generalize and predict how players will behave on levels in which we have no data at all, opening the door to many adaptive applications involving problem ordering and choice.

There are many possible avenues for future work. On a lower level, we could use more powerful collaborative filtering models taking advantage of timestamps in order to find similar players. Automatic generation of state aggregation functions and autotuning the b_a and b_h parameters would remove the need for some expert authoring. On a higher level, trying the same method on other open-ended educational environments, not necessarily games, could tell us how well the method generalizes. Using the model for applications such as dynamic hinting systems just when we predict players will quit or make egregious errors could increase player engagement and learning. Finally, the ability to estimate behavior on future, unseen problems could be used to increase transfer by selecting tasks which specifically target incorrect strategies or concepts we believe the player has, reflected in the heuristics they use.

10. ACKNOWLEDGMENTS

This work was supported by the University of Washington Center for Game Science, DARPA grant FA8750-11-2-0102, the Bill and Melinda Gates Foundation, NSF grants IIS-0811902 and IIS-1048385, and an NSF Graduate Research Fellowship under Grant No. DGE-0718124.

11. REFERENCES

[1] Alexa Ranking, BrainPOP.
[2] Alexa Ranking, Kongregate.
[3] D. W. Albrecht, I. Zukerman, and A. E. Nicholson. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction*, 8(1-2):5–47, Jan. 1998.
[4] V. Aleven, B. M. McLaren, J. Sewall, and K. R. Koedinger. The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. ITS'06, 2006.
[5] I. Althofer. Data compression using an intelligent generator: The storage of chess games as an example. *Artificial Intelligence*, 52(1):109 – 113, 1991.
[6] S. C. Bakkes, P. H. Spronck, and G. van Lankveld. Player behavioural modelling for video games. *Entertainment Computing*, 3(3):71 – 79, 2012.
[7] Y. Bergner, S. Droschler, G. Kortemeyer, S. Rayyan, D. Seaton, and D. Pritchard. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. In *EDM*, 2012.
[8] J. C. Campione. Assisted assessment: A taxonomy of approaches and an outline of strengths and weaknesses. *Journal of Learning Disabilities*, 22(3), 1989.
[9] M. Feng and N. T. Heffernan. Can we get better assessment from a tutoring system compared to traditional paper testing? can we have our cake (better assessment) and eat it too (student learning during the test)? In *EDM*, 2010.
[10] M. Feng, N. T. Heffernan, and K. R. Koedinger. Predicting state test scores better with intelligent tutoring systems: developing metrics to measure assistance required. ITS'06, pages 31–40, 2006.
[11] L. S. Fuchs, D. L. Compton, D. Fuchs, K. N. Hollenbeck, C. F. Craddock, and C. L. Hamlett. Dynamic assessment of algebraic learning in predicting third graders' development of mathematical problem solving. *Journal of Educational Psychology*, 100(4):829 – 850, 2008.
[12] J. P. Gee. Learning by design: Games as learning machines. *Interactive Educational Multimedia*, 8:15–23, 2004.
[13] E. L. Grigorenko and R. J. Sternberg. Dynamic testing. *Psychological Bulletin*, 124(1):75 – 111, 1998.
[14] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, June 2001.
[15] A. R. Jansen, D. L. Dowe, and G. E. Farr. Inductive inference of chess player strategy. PRICAI'00, pages 61–71, Berlin, Heidelberg, 2000. Springer-Verlag.
[16] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. AAAI-02, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.
[17] C. Pedersen, J. Togelius, and G. N. Yannakakis. Modeling player experience in super mario bros. CIG'09, pages 132–139, Piscataway, NJ, USA, 2009. IEEE Press.
[18] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: an open architecture for collaborative filtering of netnews. CSCW '94, 1994.
[19] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems, 2007.
[20] A. M. Smith, C. Lewis, K. Hullett, G. Smith, and A. Sullivan. An inclusive taxonomy of player modeling. *University of California, Santa Cruz, Tech. Rep. UCSC-SOE-11-13*, 2011.
[21] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009, Jan. 2009.
[22] N. Thai-Nghe, T. Horváth, and L. Schmidt-Thieme. Factorization models for forecasting student performance. In *EDM*, pages 11–20, 2011.
[23] B. G. Weber, M. Mateas, and A. Jhala. Using data mining to model player experience. In *FDG Workshop on Evaluating Player Experience in Games*, Bordeaux, France, 06/2011 2011. ACM, ACM.
[24] C.-N. Ziegler, G. Lausen, and L. Schmidt-Thieme. Taxonomy-driven computation of product recommendations. CIKM '04, 2004.