Discovering Student Models with a Clustering Algorithm Using Problem Content

Nan Li
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15232
nli1@cs.cmu.edu

William W. Cohen
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15232
wcohen@cs.cmu.edu

Kenneth R. Koedinger
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15232
koedinger@cs.cmu.edu

ABSTRACT

One of the key factors that affects automated tutoring systems in making instructional decisions is the quality of the student model built in the system. A student model is a model that can solve problems in various ways as human students. A good student model that matches with student behavior patterns often provides useful information on learning task difficulty and transfer of learning between related problems, and thus often yields better instruction on intelligent tutoring systems. However, traditional ways of constructing such models are often time consuming, and may still miss distinctions in content and learning that have important instructional implications. Automated methods can be used to find better student models, but usually require some engineering effort, and can be hard to interpret. In this paper, we propose an automated approach that finds student models using a clustering algorithm based on automatically-generated problem content features. We demonstrate the proposed approach using an algebra dataset. Experimental results show that the discovered model is as good as one of the best existing models, which is a model found by a previous automated approach, but without the knowledge engineering effort.

Keywords

student model, machine learning, learner modeling

1. INTRODUCTION

A student model is an essential component in intelligent tutoring systems. It encodes how to solve problems in various ways as human students do. One common way of representing such student models is a set of *knowledge components (KC)* encoded in intelligent tutors to model how students solve problems. As defined in [9], a knowledge component is an *acquired* unit of cognitive function or structure that can be *inferred* from performance on *a set of related tasks*. The set of KCs includes the component skills, concepts, or percepts that a student must acquire to be successful on the target tasks. For example, a KC in algebra can be how students should proceed given problems of the form $Nv=N$ (e.g. $3x = 6$). A student model provides automated tutoring systems with important information on how to make instructional decisions. Better student models are capable of predicting task difficulty and transfer of learning between related problems. Thus, intelligent tutoring systems with better student models often provide more effective learning experience.

Traditional ways to construct student models include structured interviews, think-aloud protocols, rational analysis, and so on. However, these methods are often time-consuming, and require domain expertise. More importantly, they are highly subjective. Previous studies [11] have shown that human engineering of these models often miss components of knowledge acquisition (e.g., that learning to read algebraic sentences is difficult) that have important instructional implications. Other methods that apply machine learning techniques to generate student models [16, 27] can find models that are better than human-generated ones, but may suffer from challenges in interpreting the results. For example, Learning Factor Analysis (LFA) [6] apply an automated search technique to discover student models. Nevertheless, one key limitation of LFA is that it carries out the search process only within the space of human-provided factors. If a better model exists but requires unknown factors, LFA will not find it. Another approach is to use a learning agent, *SimStudent*, to automatically discover student models [15]. Although this method is less dependent on human-provided factors, it still needs some knowledge engineering effort in constructing the learning agent.

To address this issue, we formulate the student model discovery approach as a clustering problem, and propose another automated method that discovers student models using a machine learning algorithm, *k-means*, based on automatically-generated features. To accommodate for both the performance prediction accuracy and the interpretability of the discovered model, the features include both problem content features and performance features, so that problem steps in the same cluster are of similar forms and are associated with similar performance on human students. Each cluster corresponds to a KC that students need to learn. We evaluated the approach in algebra using real student data. Experiment results show that the discovered model fits with real student data as good as the model found by SimStudent.

Table 1: An Example List of Steps with Their Content Features.

Step	Tokenized Step	-N	Nv	v=	=N	-Nv	Nv=	v=N	v+	+N	N=	Nv+	v+N	+N=	N=N
-3x = 6	-Nv=N	1	1	1	1	1	1	1	0	0	0	0	0	0	0
2y+5=7	Nv+N=N	0	1	0	1	0	0	0	1	1	1	1	1	1	1

In the following sections, we start by describing how to statistically evaluate the quality of a student model. Then, we explain how to generate features and to apply a clustering algorithm to find student models that meet such criteria. Next, we report experimental results on the comparison between the clustering-based model and the SimStudent model, along with an in-depth study using a recently developed analysis technique, Focused Benefits Investigation (FBI) [10]. After this, we discuss the generality of the proposed approach, and possible improvements that can be made using SimStudent. In closing, we describe some related work as well as conclusions drawn from this work.

2. STATISTICAL EVALUATION OF STUDENT MODEL QUALITY

As we have mentioned before, a student model can be represented by a set of knowledge components, where each problem step is associated with one KC that encodes how to proceed given the current step. Therefore, the problems we have is that given a dataset recording how human students solve problems in one domain, how to find a set of KCs that matches with student behavior well.

There are various ways of matching a student model with student data. Although other models are also possible (e.g. [8], in our case, we use the Additive Factor Model (AFM) [6] to measure the quality of a student model. AFM is an instance of logistic regression that models student success using each student, each KC, and the KC by opportunity interaction as independent variables,

$$\ln \frac{p_{ij}}{1-p_{ij}} = \theta_i + \sum_k \beta_k Q_{kj} + \sum_k \beta_k Q_{kj}(\gamma_k N_{ik})$$

Where:

i represents a student i.

j represents a step j.

k represents a skill or KC k.

p_{ij} is the probability that student i would be correct on step j.

θ_i is the coefficient for proficiency of student i.

β_k is coefficient for difficulty of the skill or KC k.

Q_{kj} is the Q-matrix cell for step j using skill k.

γ_k is the coefficient for the learning rate of skill k.

N_{ik} is the number of practice opportunities student i has had on the skill k.

Hence, the better the student model is; the more accurate the predictions are. To train the parameters, we use maximum-likelihood estimation (MLE). In order to avoid overfitting, we use cross-validation (CV) to validate the quality of the student model.

3. STUDENT MODEL DISCOVERY USING A MACHINE LEARNING ALGORITHM

Given the above evaluation method, we would like to note that our task here is not only to find a model that predicts student behavior well, we also want to find a model that is conceptually meaningful. In other words, steps within the same KC should be both conceptually similar and performance-wise similar. In fact, finding a student model over a set of problem steps is a *clustering task*, where the algorithm groups a set of problem steps in a way that steps in the same group (called cluster) are more similar in some sense to each other than to those in other groups (clusters). In our case, each cluster corresponds to a KC in the student model. From this clustering point of view, if the metric of similarity measures both content similarity and performance similarity, the KCs that the algorithm finds would have the desired properties we discussed above. Therefore, we use two types of features for clustering, *content features* and *performance features*.

3.1 Preprocessing

Before generalizing the features, we first tokenize the problem steps, so that all numbers are replaced by N, and all variables are represented as v. For example, the tokenized representation of $-3x = 6$ is $-Nv = N$. In fact, the level of tokenization affects the result of the discovered model, since this preprocessing step removes the difference among steps that are of the same form but with different numbers. This may cause problems in some cases. For instance, solving $-3x = 6$ can be potentially much easier than solving $-452x = 904$, but the preprocessing step gives both steps the same tokenized representation $-Nv = N$. As we will discuss later, by making use of SimStudent, we could automatically get different levels of tokenization.

3.2 Feature Generation

After preprocessing, we now generate features for these tokenized steps. There are two types of features, content features and performance features.

3.2.1 Content Features

Content features are defined based on the problem content information of the tokenized steps. More specifically, we generate all of the bigrams and trigrams in each of the tokenized steps. For each bigram or trigram, we set the value of that feature to be 1 if the bigram or trigram appears in the current step, and 0 otherwise.

Table 2: The List of Performance Features Used for Clustering.

Feature	Meaning
Avg. Incorrects	Average number of incorrect attempts for the current step
Avg. Hints	Average number of the student asking for a hint for the current step
Avg. Corrects	Average number of correct attempts for the current step
% First Attempt Incorrects	The percentage of times that the first attempt is incorrect
% First Attempt Hints	The percentage of times that the first attempt is asking hint
% First Attempt Corrects	The percentage of times that the first attempt is correct
Avg. Step Duration (sec)	Average number of seconds the student spending on this step
Avg. Correct Step Duration (sec)	Average number of seconds the student spending on this step when the student gets this step correct
Avg. Error Step Duration (sec)	Average number of seconds the student spending on this step when the student gets this step incorrect
Total Students	Average number of total students working on this step
Total Opportunities	Average number of total opportunities that the student has in solving the current step

For example, for step $-Nv = N$, all of the bigram features it generates are $-N$, Nv, $v =$, and $= N$, and all of the trigram features it generates are $-Nv$, $Nv =$, and $v = N$. Considering a bigram feature Nv, and a trigram feature $+N =$, for step $-Nv = N$, the value of the feature Nv is 1, whereas the value of the feature $+N =$ is 0, since $+N =$ does not appear in $-Nv = N$. But for another step $Nv + N = N$, both the value of Nv and the value of $+N =$ are 1, since both of them appear in $Nv + N = N$. The trigram feature $+N =$ here can be used to identify the steps for subtracting both sides with N. Table 1 shows an example list of steps with their content features.

By using these content features, we make sure that the steps in the same cluster share some common content features, and thus look similar to each other. This satisfies the property of having conceptually-similar steps in the same cluster. Moreover, by having steps that share content features clustered in one KC, it is easier for human to interpret the results.

3.2.2 *Performance Features*

The second set of features we used in the algorithm are performance features. These features measure the average performance of human students on each format of the tokenized steps. Examples of such measurements are the time to response, and whether the student's first attempt was correct. Table 2 shows the full list of performance features used for clustering.

Note that performance features are only used to create the clusters of the training data. Since we are predicting the performance of human students, performance data should not be used in testing. For testing data, we only use the content features to assign the cluster of the current step. In other words, for each testing data point, we calculate the distance of the data point to all of the training data points based on perceptual features, and assign the testing data point to the cluster associated with the closest training data point.

3.3 Principal Component Analysis

Before clustering, we normalize all the features to range from 0 to 1. Then, we perform a principal component analysis over the features we generated. Principal component analysis is a mathematical procedure that projects a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables. These linearly uncorrelated variables are called principal components. The first principal component points to the direction that accounts for the largest possible variance. The succeeding components are orthogonal to the previous components, and account for smaller variance.

After this transformation process, all of the features in the projected space are orthogonal to each other. Moreover, in order to remove less informative features, we only select the first 40 principal components in the projected space. It covers approximately 95% of the variance in the data.

Algorithm 1: K-Means

Input: Points to be clustered P, Number of clusters k
Output: Cluster centroids C, cluster membership M.

initialize C with k randomly selected data points in P
forall the $p_i \in P$ **do**
  $m_i := argmin_{j \in 1..k} distance(p_i, c_j)$
end
while m *changed* **do**
  foreach $i \in \{1..n\}$ **do**
    Recompute c_i as the centroid of $\{p_j | m_j = i\}$
  end
  sum_ratios := 0
  forall the $p \in c \cap d$ **do**
    sum_ratios += $w_c(p)/w_d(p)$
  end
  forall the $p_i \in P$ **do**
    $m_i := argmin_{j \in 1..k} distance(p_i, c_j)$
  end
end
return C, M

3.4 Student Model Discovery with a Clustering Algorithm

To discover student models, we use k-means to cluster the data over the automatically-generated features. The dis-

Table 3: Cross Validation Results on the Clustering-Based Model and the SimStudent Model.

	SimStudent RMSE	Clustering RMSE
Run 1	0.4105	**0.4102**
Run 2	0.4109	**0.4106**
Run 3	0.4113	**0.4105**
Run 4	**0.4107**	0.4111
Run 5	0.4106	**0.4095**
Run 6	0.4109	**0.4102**
Average	0.4108	**0.4104**

tance between data points is measured by the Euclidean distance in the feature space.

The algorithm uses an expectation-maximization style approach. Algorithm 1 shows the psuedocode of the clustering procedure. Fristly, the algorithm randomly selects k points as the initial centers of each cluster. Then, in the assignment step, the rest of the points are assigned to the cluster whose mean is closest to it among all of the existing clusters. Next, in the update step, the algorithm calculates the new means of the new clusters as the centroids of the data points. This process continues until converge.

K-means needs the number of clusters k to be given as input. Since we do not know how many clusters are there, we set the number of clusters to be 20, 25, and 30. The algorithm then picks the one with the best cross validation result[1]. Note that even with the same number of clusters, different initialization of the clusters can lead to different clustering results. In this study, we just run k-means once for each value k. In future study, we could run the clustering algorithm multiple times, and select the clusters that have the smallest intra-cluster difference and the largest inter-cluster difference.

4. EXPERIMENT STUDY

In order to evaluate the effectiveness of the proposed approach, we carried out a study using an algebra dataset. We compared the clustering-based model with a SimStudent model. The SimStudent model is discovered by a learning agent, which is also one of the best student models we have in the database.

4.1 Method

To generate the SimStudent model, SimStudent was tutored on how to solve linear equations by interacting with a Carnegie Learning Algebra I Tutor like a human student. We selected 40 problems that were used to teach real students as the training set for SimStudent. Given all of the acquired production rules, for each step a real student performed, we assigned the applicable production rule as the KC associated with that step. In cases where there was no applicable production rule, we coded the step using a human-generated KC model (Balanced-Action-Typein). The human-generated model is the best model constructed by domain experts. It has been shown that the SimStudent model is better than the human-generated model, and provides useful instructional implications.

[1]We tried smaller numbers, but it turns out that when k is between 20 and 30, the cross validation result is often better.

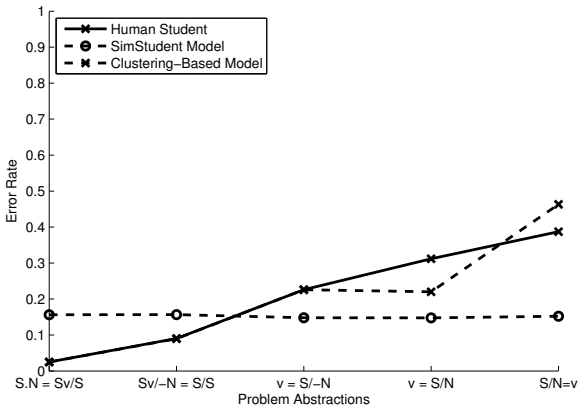

Figure 1: Error rates of human students and predicted error rates of two student models. S stands for a signed number, N represents an integer, and v is a variable.

The clustering-based model was discovered using the approach described above. Each time a student encounters a step using some KC is considered as an "opportunity" for that student to show mastery of that KC. In both models, a total of 6507 steps are coded.

In order to get a better understanding on how the clustering-based model differs from other student models, we further utilized DataShop, a large repository that contains datasets from various educational domains as well as a set of associated visualization and analysis tools, to facilitate the process of evaluation, which includes generating learning curve visualization, AFM parameter estimation, and evaluation statistics including AIC (Akaike Information Criterion) and cross validation.

4.2 Dataset

We analyzed data from 71 students who used an Carnegie Learning Algebra I Tutor unit on equation solving. The students were typical students at a vocational-technical school in a rural/suburban area outside of Pittsburgh, PA. The problems varied in complexity, for example, from simpler problems like *3x=6* to harder problems like *x/-5+7=2*. A total of 19,683 transactions between the students and the Algebra Tutor were recorded, where each transaction represents an attempt or inquiry made by the student, and the feedback given by the tutor.

4.3 Measurements

To test whether the generated model fits with real student data, we used 10-fold cross validation. The cross validation was performed over ten folds with the constraint that each of the three training sets must have data points for each student and KC. For the clustering-based model, performance features of the testing steps were not used in constructing the KCs. We calculated the root mean-squared error (RMSE) averaged over ten test sets. Due to the random nature of the fold generation process in cross validation, we repeated this process six times.

Table 4: FBI Results on Selected KCs That Are Improved in the Clustering-Based Model.

SimStudent KCs	SimStudent Model RMSE	Clustering-Based Model RMSE	% change of RMSE
ctat-divide	0.5289	0.3984	-24.67
ctat-distribute	0.4292	0.3553	-17.21
ctat-multiply	0.4634	0.3962	-14.50
ctat-clt	0.3757	0.3445	-8.325
ctat-divide-typein	0.3674	0.3368	-8.321

In order to better understand this machine learning approach, we carried out an in-depth study using FBI [10] on the clustering-based model and the SimStudent. FBI is a recently developed technique. It is designed to analyze which of the differences between the models improves the prediction the most, and by how much.

4.4 Experimental Results

As shown in Table 3, in five out of the six runs, the clustering-based models get lower RMSEs than the SimStudent model, which indicates that the clustering-based model is at least as good as the SimStudent model. Averaged over the six runs, the clustering-based models get an average RMSE of 0.4104, while the SimStudent model gets a slightly higher RMSE (i.e., 0.4108).

As you may have noticed, the difference between the RMSEs of the two models is small, but this does not mean that the difference between the two models is small. Instead of using cross validation to measure the quality of the model as a whole, we applied FBI to evaluate the difference at the knowledge component level. Table 4 shows the top five KCs in the SimStudent model that are improved in the clustering-based model. As we can see that all of these KCs' names start with "ctat", which means these KCs are from the human-generated model. Recall that in the SimStudent model, if SimStudent could not find any applicable production rule to a step, the step would be coded by the human-generated model. This suggests that the clustering-based approach is more general than the SimStudent approach in the sense that it is able to code steps that are not supported by SimStudent. Among the nine KCs generated by SimStudent, three of them were improved in the clustering-based student model.

In these five KCs, the clustering-based model successfully reduced the RMSE by at least 8%. In the KC "ctat-divide", the RMSE was reduced by around 25%. This indicates that the clustering-based approach is able to find KCs that are better than the existing ones. We can inspect the data more closely to get a better qualitative understanding of how the two models are different and what implications there might be for improved instruction.

We took a closer look at the KC "ctat-divide-typein". In the SimStudent model, all steps that require division are assigned to the "ctat-divide-typein" skill. However, there are differences among these steps. We checked the KCs in the clustering-based model associated with these steps, and found out that these steps were split into different KCs in the clustering-based model. Table 5 shows the five biggest KCs associated with the "ctat-divide-typein" steps. Since we used problem content based features, the KCs in the model were relatively easy to interpret. Each KC name (e.g., 25) in the table is followed by the most common form of the division steps in the KC (e.g., $S.N = Sv/S$), where N represents an integer, S means a signed number, and v stands for a variable. We calculated the average error rate of human students solving these steps, as well as the predicted error rates of the steps based on the two student models. As presented in Figure 1, the predicted error rates of the clustering-based model are closer to human students' actual error rates than the predicated error rates of the SimStudent model. Since the SimStudent model considers all these steps correspond to one KC, it predicts that they should have similar error rates, which is reflected by the flat line in Figure 1. The clustering-based model, on the other hand, predicts different error rates for problem steps of different forms.

More specifically, according to human student performance, steps associated with KC 25 are easier than problem steps from other KCs. A careful inspection at the data shows that KC 25 is associated with problem steps of the form $S.N = Sv/S$, which means that the left side of the equation is a decimal number. On the other hand, the problem steps in other KCs are associated with fractions. For steps with fractions, human students may have to simplify the fractions in order to get the final solution, whereas for steps with decimal numbers, students only need to copy the decimal numbers as the solution. Therefore, steps associated with KC 25 have a lower error rate than the other steps, which are correctly modeled by the clustering-based model. Moreover, among KC 6, KC 22, and KC 29, human students have a higher error rate when the variable is on the right side of the equation (i.e., steps associated with KC 6). This is also correctly captured by the clustering-based model, while the SimStudent model again incorrectly predicts similar error rates.

These results are confirmed in the FBI study as well. As shown in Table 5, the largest improvement comes from KC 25 reaches 40%, partially because it separates divide-typein problems with decimal numbers from problems with fractions. KC 15 further models problem steps that have the variable with coefficients from the other steps that have the single variable in one side of the equation. This contributes to an improvement around 8%. The other three KCs differentiate problem steps that have the variable in the left side of the equation from the ones that have the variable in the right side of the equation. Two out of these three KCs get better RMSE. The third KC's increase in RMSE is mainly caused by other none divide-typein steps.

Table 5: Selected KCs in the Clustering-Based Model That Correspond to KC "ctat-divide-typein".

Clustering KCs	Clustering-Based Model RMSE	SimStudent Model RMSE	% change of RMSE
25 (S.N = Sv/S)	0.1547	0.2170	40.34
6 (S/N=v)	0.4654	0.5516	18.54
15 (Sv/-N = S/S or S/S = Sv/-N)	0.2969	0.3205	7.939
22 (v = S/-N)	0.4194	0.4279	2.016
29 (v = S/N)	0.4238	0.4073	-3.898

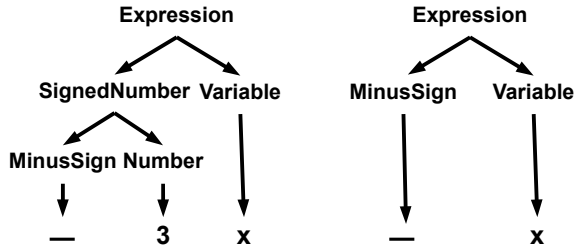

Figure 2: Different parse trees for -3x and -x.

The clustering algorithm's split of the original divide-typein KC into five KCs suggests that human students should be taught separately on each type of problems. More specifically, intelligent tutors cannot make the assumption that if students have learned the divide-typein KC for decimal problems, they will also know how to solve the fraction problems. Furthermore, the tutoring system should teach human students not only with problems that have the variables on the left side, but also with problems that have the variables at the right side of the equation, so that students get familiar with the concept that variables can be at the either side of the equation.

5. DISCUSSION

Given the promising results, we would like to further discuss some interesting future steps for this algorithm.

5.1 Automated vs. Manual Model Discovery

One question we should ask is that why we should use automated student model discovery approach rather than manual construction. This is mainly due to the fact that much of human expertise is only tacitly known. In many of the cases, we know how to solve the problems, while it can be hard to explain how we solved the problem. For instance, in language learning, native speakers can accurately select the correct article in a sentence, but do not know why they pick that article. Similarly, most algebra experts have no explicit awareness of subtle transformations they have acquired. Even though such instructional designers may be experts in a domain, they may still have some blind spots regarding subtle perceptual differences like this one, which may make a real difference for novice learners. A machine learning approach can help get past such blind spots by revealing challenges in the learning process that experts may not be aware of. In addition, these discovered KCs can serve as a basis for traditional ways of student model discovery.

5.2 Feature Generation Using SimStudent

Furthermore, in this paper, we simply use bigrams and trigrams of the tokenized steps as the content features. Some of these features may not be very helpful in differentiating KCs needed for the steps. Moreover, it is possible that different tokenization procedures and longer n-grams would lead to better results in other domains. We can, of course, keep adding new features in the feature space, and let the learning algorithm search through the larger space. However, this is not ideal due to the curse of dimensionality. In previous work, we have shown that hierarchical representations of the problem steps as shown in Figure 2 can be acquired by grammar induction techniques [14]. These hierarchical representations capture "deep features" in solving problems at different levels of abstractions. In the future, it would be interesting to see that whether we can make use of such representations to automatically generate high-quality content features, and lead to the discovery of better student models.

Moreover, the problem content features used in this paper are perceptual features. This is sufficient for domains like algebra, since the structure of the problem steps is enough to decide which skill to apply. But in other domains such as fraction addition, deciding whether two numbers are coprimed or not is impossible if using only perceptual features. In these cases, being able to generate operational features is required.

In response to this, we propose to use SimStudent to generate these features. SimStudent is an intelligent learning agent that uses machine learning techniques to acquire skills. It has three sets of prior knowledge, *a perceptual hierarchy*, *a set of operator functions*, and *a set of feature predicates*. Previous work has shown that by integrating representation learning with skill learning, instead of manually encoding this prior knowledge, the learning agent can learn, automatically generate, or partially reduce the need of such prior knowledge. The extended learning agent becomes a better model of student learning. To get a more general approach, we plan to make use of the acquired prior knowledge as well as the learned skills to generate both perceptual features and operational features.

5.3 Objective Function Guided Clustering

In this paper, the student model is discovered purely based on the clustering procedure. The discovered model is then used to fit with student data in the AFM model. In other words, the student model does not change once the clustering process is completed. Another interesting approach is to guide the student model discovery / clustering process by the fit to student performance data. Since the second approach is fully guided by the objective function, presumably, we could get a model with better predictions than the approach proposed in this paper.

However, there are two major issues with this objective func-

tion guided approach. As mentioned before, each knowledge component in a student model is an *acquired* unit of cognitive function or structure that can be *inferred* from performance on *a set of related tasks* [9]. If the student model is discovered purely based on the fit to student performance data, the KCs discovered may not be able to provide meaningful instructional insights. For example, if human students found both problems of the form $-v = N$ and $-N/v = N$ hard, does that mean that the tutor should teach these problem steps together? One possible way to address this issue is to also include the problem content similarity measurement in the objective function, so that the search is guided not only by performance, but also by task similarity.

Another issue is that this objective function guided approach often takes longer, as it has to fit the model with the data on each node expansion during the search. Therefore, in this paper, we take the clustering approach since it is more efficient, and can find KCs that are easier to interpret. In the next study, it would be interesting to compare the proposed approach with the objective function guided approach.

5.4 Other Clustering Techniques

One additional possible study is to try other clustering techniques. In this work, we only applied k-means to discover student models. There are other clustering algorithms such as hierarchical agglomerative clustering and spectral clustering [19]. These clustering algorithms have different properties, and may be better fit with the student model discovery task. In the future, we would like to further explore in this direction with other clustering techniques.

5.5 Generality

The last study we are interested in carrying out is to test the generality of the proposed approach. The Pittsburgh of Science of Learning Center's DataShop contains over 200 datasets in algebra and other domains that could be used for such cross-dataset or cross-domain validation. The current study used a single dataset in a single domain. The generality and validity of the proposed student-modeling technique could be extended by clustering problem steps in one dataset and applying the discovered KC model to other datasets. For example, the dataset we used is associated with students in one high school. It would be interesting to see whether the generated student model applies to other high schools at the same level.

In addition, we plan to apply this approach in other domains such as stoichiometry, fraction addition and so on. As we have mentioned above, it is possible that operational features are also needed in these domains. In this case, extending the current approach with other learning techniques such as SimStudent would be a promising future step. On the other hand, the language learning domain does not require complex problem solving, but needs complex perceptual knowledge and large amounts of background knowledge. An interesting future work is to apply existing linguistic tools to English sentences, and then automatically generate problem content features based on the parsed sentences.

6. RELATED WORK

The objective of this paper is using a clustering algorithm to automatically construct student models. A lot of efforts have also been put toward comparing the quality of alternative student models. LFA automatically discovers student models, but is limited to the space of the human-provided factors. SimStudent is less dependent on human-provided factors, but still needs some knowledge engineering effort in constructing the agent. Moreover, as we have shown in the experiments, the clustering based algorithm is able to find KCs that are better than those found by SimStudent. Other works such as [16, 27] are less dependent on human labeling, but may suffer from challenges in interpreting the results. In contrast, the clustering-based approach has the benefit that the acquired KCs usually have a straightforward interpretation. Baffes and Mooney [2] apply theory refinement to the problem of modeling incorrect student behavior. Other systems [23, 3] use Q-matrix to find knowledge structure from student response data. Our approach also uses machine learning algorithms to discover student models. In addition to model student performance, we emphasize on the interpretability of the models by adding content features to the clustering approach.

There has also been considerable amount of research on using artificial intelligence and machine learning techniques to model human students. Langley and Ohlsson's [13] ACM applies symbolic machine learning techniques to automatically construct student models. Brown and Burton's [5] DEBUGGY, and Sleeman and Smith's [20] LMS also make use of artificial intelligent tools to construct models that explain student's behavior in math domains. VanLehn's [25] Sierra models the impasse-driven acquisition of hierarchical procedures for multi-column subtraction from sample solutions. Research on models of high-level learning [12, 1, 22, 21, 24, 18] is also closely related to our work, but to the best of our knowledge, has not been evaluated by the fit to student learning curve data as we do in this work. In addition, most of these work took a more symbolic approach, while our algorithm is more statistical based.

Other research on creating simulated students [26, 7, 17] also share some resemblance to our work. VanLehn [25] created a learning system and evaluated whether it was able to learn procedural "bugs" like real students. Biswas et al.'s [4] system learns causal relations from a conceptual map created by students. None of the above approaches except for the SimStudent model discovery approach compared the system with learning curve data. To the best of our knowledge, our work is the very few who combines the two whereby we use cognitive model evaluation techniques to assess the quality of a simulated learner.

7. CONCLUSION

In this paper, we introduced an innovative application of a machine learning algorithm for an automatic discovery of student models. In order to discover KCs that are effective in predicting human student performance, while being easy to interpret, we added problem content features in the feature space, and applied a clustering algorithm to find student models. Our evaluation demonstrated that discovering student models based on problem content features was able to produce models of good prediction accuracies, and showed how the discovered model could provide important instructional implications. We further discussed possible extensions to the existing approach, and described how a learning agent

such as SimStudent can be used to automatically generate content features as well as operational features to improve the generality of the proposed approach. This work is one step forward in applying machine learning techniques to construct student model. We believe that there are a lot of of fruitful future steps in this direction. They are natural extensions under the current framework.

8. ACKNOWLEDGEMENTS

The research reported here was supported by the Pittsburgh Science of Learning Center, which is funded by the National Science Foundation Award No. SBE-0836012. The authors would also like to thank Mike Wixon and Dandiel Seaton for helpful discussions, and Hui Cheng for running the experiments.

9. REFERENCES

[1] J. R. Anderson. *Rules of the Mind*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1993.

[2] P. T. Baffes and R. J. Mooney. A novel application of theory refinement to student modeling. In *Proceedings of the thirteenth national conference on Artificial intelligence*, pages 403–408. AAAI Press, 1996.

[3] T. Barnes. The Q-matrix method: Mining student response data for knowledge. In *Proceedings AAAI Workshop Educational Data Mining*, pages 1–8, Pittsburgh, PA, 2005.

[4] G. Biswas, D. Schwartz, K. Leelawong, and N. Vye. Learning by teaching: A new agent paradigm for educational software. *Applied Artificial Intelligence*, 19:363–392, March 2005.

[5] R. R. Burton. Diagnosing bugs in a simple procedural skill. In *Intelligent Tutoring Systems*, pages 157–184. Academic Press, 1982.

[6] H. Cen, K. Koedinger, and B. Junker. Learning factors analysis - a general method for cognitive model evaluation and improvement. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, pages 164–175, 2006.

[7] T.-W. Chan and C.-Y. Chou. Exploring the design of computer supports for reciprocal tutoring. *International Journal of Artificial Intelligence in Education*, 8:1–29, 1997.

[8] Y. Gong, J. E. Beck, and N. T. Heffernan. Comparing knowledge tracing and performance factor analysis by using multiple model fitting procedures. In *Proceedings of the 10th international conference on Intelligent Tutoring Systems - Volume Part I*, pages 35–44, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The Knowledge-Learning-Instruction (KLI) Framework : Toward Bridging the Science-Practice Chasm to Enhance Robust Student Learning. *Cognitive Science*, 36(5):757–798, 2012.

[10] K. R. Koedinger, E. A. McLaughlin, and J. C. Stamper. Automated student model improvement. In *Proceedings of the 5th International Conference on Educational Data Mining*, pages 17–24, Chania, Greece, 2012.

[11] K. R. Koedinger and M. J. Nathan. The real story behind story problems: Effects of representations on quantitative reasoning. *The Journal of Learning Sciences*, 13(2):129–164, 2004.

[12] J. E. Laird, P. S. Rosenbloom, and A. Newell. Chunking in soar: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

[13] P. Langley and S. Ohlsson. Automated cognitive modeling. In *Proceedings of the Fourth National Conference on Artificial Intelligence*, pages 193–197, Austin, TX, 1984. Morgan Kaufmann.

[14] N. Li, W. W. Cohen, and K. R. Koedinger. Efficient cross-domain learning of complex skills. In *Proceedings of the Eleventh International Conference on Intelligent Tutoring Systems*, pages 493–498, Berlin, 2012. Springer-Verlag.

[15] N. Li, N. Matsuda, W. W. Cohen, and K. R. Koedinger. A machine learning approach for automatic student model discovery. In *EDM*, pages 31–40, 2011.

[16] P. I. Pavlik, H. Cen, and K. R. Koedinger. Learning Factors Transfer Analysis: Using Learning Curve Analysis to Automatically Generate Domain Models. In *Proceedings of 2nd International Conference on Educational Data Mining*, pages 121–130, 2009.

[17] T. N. Pentti Hietala. The competence of learning companion agents. *International Journal of Artificial Intelligence in Education*, 9:178–âĂŞ192, 1998.

[18] U. Schmid and E. Kitzelmann. Inductive rule learning on the knowledge level. *Cognitive System Research*, 12(3-4):237–248, Sept. 2011.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[20] D. H. Sleeman and M. J. Smith. Modeling students' problem solving. *Artificial Intelligence*, 16:171–187, 1981.

[21] R. Sun. Cognitive social simulation incorporating cognitive architectures. *IEEE Intelligent Systems*, 22(5):33–39, Sept. 2007.

[22] N. A. Taatgen and F. J. Lee. Production compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, 45(1):61–75, 2003.

[23] K. K. Tatsuoka. Rule space: An approach for dealing with misconceptions based on item response theory. *Journal of Educational Measurement*, pages 345–354, 1983.

[24] J. B. Tenenbaum and T. L. Griffiths. Generalization, similarity, and bayesian inference. *Behavioral and Brain Sciences*, 24:629–640, 2001.

[25] K. VanLehn. *Mind Bugs: The Origins of Procedural Misconceptions*. MIT Press, Cambridge, MA, USA, 1990.

[26] K. Vanlehn, S. Ohlsson, and R. Nason. Applications of simulated students: an exploration. *Journal of Artificial Intelligence in Education*, 5:135–175, February 1994.

[27] M. Villano. Probabilistic student models: Bayesian belief networks and knowledge space theory. In *Proceedings of the 2nd International Conference on Intelligent Tutoring Systems*, pages 491–498, Heidelberg, 1992.