Using the Hint Factory to Compare Model-Based Tutoring Systems

Collin Lynch
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
cflynch@ncsu.edu

Thomas W. Price
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
twprice@ncsu.edu

Min Chi
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
mchi@ncsu.edu

Tiffany Barnes
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
tmbarnes@ncsu.edu

ABSTRACT

Model-based tutoring systems are driven by an abstract domain model and solver that is used for solution validation and student guidance. Such models are robust but costly to produce and are not always adaptive to specific students' needs. Data-driven methods such as the Hint Factory are comparatively cheaper and can be used to generate individualized hints without a complete domain model. In this paper we explore the application of data-driven hint analysis of the type used in the Hint Factory to existing model-based systems. We present an analysis of two probability tutors Andes and Pyrenees. The former allows for flexible problem-solving while the latter scaffolds students' solution path. We argue that the state-space analysis can be used to better understand students' problem-solving strategies and can be used to highlight the impact of different design decisions. We also demonstrate the potential for data-driven hint generation across systems.

1. INTRODUCTION

Developers of model-based tutoring systems draw on domain experts to develop ideal models for student guidance. Studies of such systems have traditionally been focused on their overall impact on students' performance and not on the students' user-system interaction. The Hint Factory, by contrast, takes a data-driven approach to extract advice based upon students' problem solving paths. In this paper we will apply the Hint Factory analytically to evaluate the impact of user interface changes and solution constraints between two closely-related tutoring systems for probability.

Model-based tutoring systems are based upon classical expert systems, which represent relevant domain knowledge via static rule bases or sets of constraints [9]. These knowledge bases are generally designed by domain experts or with their active involvement. They are then paired with classical search algorithms or heuristic satisfaction algorithms to automatically solve domain problems, identify errors in student solutions, and to provide pedagogical guidance. The goal of the design process is to produce expert models that give the same procedural advice as a human expert. Classical model-based tutors have been quite successful in field trials, with systems such as the ACT Programming Tutor helping students achieve almost two standard deviations higher than those receiving conventional instruction [5].

Data-driven hint generation methods such as those used in Hint Factory [17] take a different approach. Rather than using a strong domain model to generate *a-priori* advice, data-driven systems examine prior student solution attempts to identify likely paths and common errors. This prior data can then be used to provide guidance by directing students towards successful paths and away from likely pitfalls. In contrast to the expert systems approach, these models are primarily guided not by what experts consider to be *ideal* but by what students *do*.

Model-based systems such as Andes [18] are advantageous as they can provide appropriate procedural guidance to students *at any point* in the process. Such models can also be designed to reinforce key meta-cognitive concepts and explicit solution strategies [4]. They can also scale up rapidly to include new problems or even new domain concepts which can be incorporated into the existing system and will be available to all future users. Rich domain models, however, are comparatively expensive to construct and require the long-term involvement of domain experts to design and evaluate them.

Data-driven methods for generating feedback, by contrast, require much lower initial investment and can readily adapt to individual student behaviors. Systems such as the Hint Factory are designed to extract solutions from prior student data, to evaluate the quality of those solutions, and to compile solution-specific hints [17]. While this avoids the need for a strong domain model, it is limited to the space of solutions explored by prior students. In order to incorporate new problems or concepts it is necessary to collect additional data. Additionally, such methods are not generally designed to incorporate or reinforce higher-level solution strategies.

We believe that both of these approaches have inherent advantages and are not necessarily mutually exclusive. Our goal in this paper is to explore what potential data-driven methods have to inform and augment model-based systems. We argue that data-driven methods can be used to: (1) evaluate the differences between closely-related systems; (2) assess the impact of specific design decisions made in those systems for user behaviors; and (3) evaluate the potential application of data-driven hint generation across systems. To that end we will survey relevant prior work on model-based and data-driven tutoring. We will describe two closely-related tutoring systems and data collected from them. We will then present a series of analyses using state-based methods and discuss the conclusions that we drew from them.

2. BACKGROUND

2.1 Model-Based Tutoring

Model-based tutoring systems take a classical expert-systems approach to tutoring. They are typically based upon a strong domain model composed of declarative rules and facts representing domain principles and problem-solving actions coupled with an automatic problem solver. This knowledge base is used to structure domain knowledge, define individual problems, evaluate candidate solutions, and to provide student guidance. Novices typically interact with the system through problem solving with the system providing solution validation, automatic feedback, pedagogical guidance, and additional problem-solving tasks. The Sherlock 2 system, for example, was designed to teach avionics technicians about appropriate diagnostic procedures [11]. The system relies on a domain model that represents the avionics devices being tested, the behavior of the test equipment, and rules about expert diagnostic methods. Sherlock 2 uses these models to pose dynamic challenges to problem solvers, to simulate responses to their actions, and to provide solution guidance.

Andes [19, 18, 20] and Pyrenees [4] are closely-related model-driven ITSs in the domains of physics and probability. They were originally developed at the University of Pittsburgh under the Direction of Dr. Kurt VanLehn. Like other model-based systems, they rely on a rule-based domain model and automatic problem solvers that treat the domain rules as problem-solving steps. They distinguish between higher-level domain concepts such as Bayes' Rule, and atomic steps such as variable definitions. Principles are defined by a central equation (e.g. $p(A|B) = (p(B|A) * p(A))/p(B)$) and encapsulate a set of atomic problem-solving steps such as writing the equation and defining the variables within it.

The systems are designed to function as homework-helpers, with students logging into the system and being assigned or selecting one of a set of predefined problems. Each problem is associated with a pre-compiled solution graph that defines the set of possible solutions and problem-solving steps. The system uses a principle-driven automated problem solver to compile these graphs and to identify the complete solution paths. The solver is designed to implement the Target Variable Strategy (TVS), a backward-chaining problem solving strategy that proceeds from a goal variable (in this case the answer to the problem) via principle applications to the given information. The TVS was designed with the help of domain experts and guides solvers to define basic solution information (e.g. given variables) and then to proceed from the goal variable and use principles to define it in terms of the given variables.

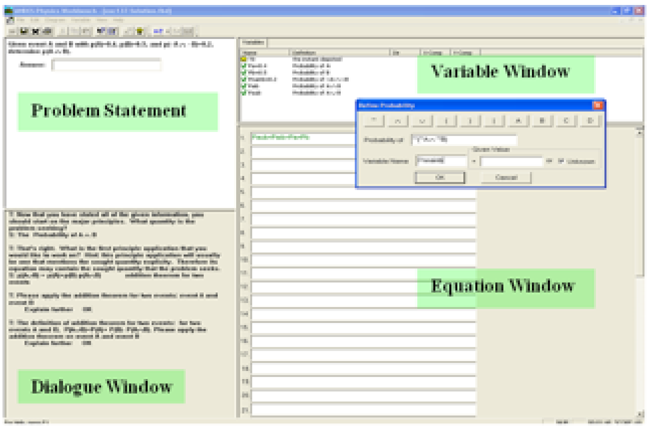
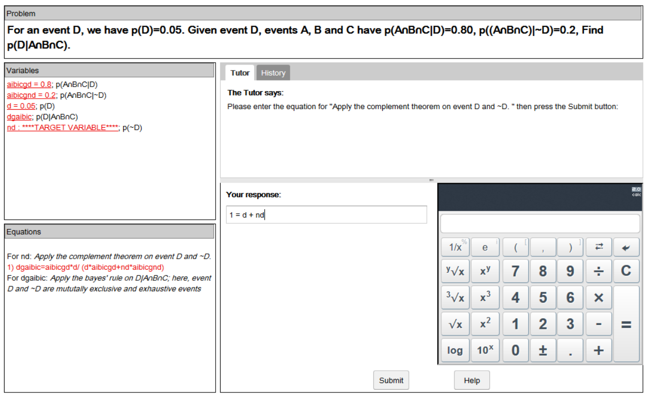


Figure 1: The Andes user interface showing the problem statement window with workspace on the upper left hand side, the variable and equation windows on the right hand side, and the dialogue window on the lower left.

Students working with Andes use a multi-modal user interface to write equations, define variables and engage in other atomic problem-solving steps. A screenshot of the Andes UI can be seen in Figure 1. Andes allows students to solve problems flexibly, completing steps in any order so long as they are valid [20]. A step is considered to be valid if it matches one or more entries in the saved solution paths and all necessary prerequisites have been completed. Invalid steps are marked in red, but no other immediate feedback is given. Andes does not force students to delete or fix incorrect entries as they do not affect the solution process. In addition to validating entries, the Andes system also uses the precompiled solution graphs to provide procedural guidance (*next-step-help*). When students request help, the system will map their work to the saved solution paths. It will then select the most complete solution and prompt them to work on the next available step.

One of the original goals of the Andes system was to develop a tutor that operated as an "intelligent worksheet." The system was designed to give students the freedom to solve problems in any order and to apply their preferred solution strategy. The system extends this freedom by allowing invalid steps in an otherwise valid solution and by allowing students to make additional correct steps that do not advance the solution state or are drawn from multiple


Problem

For an event D, we have p(D)=0.05. Given event D, events A, B and C have p(AnBnC|D)=0.80, p((AnBnC)|~D)=0.2, Find p(D|AnBnC).

Variables

aibicgd = 0.8; p(AnBnC|D)
aibicgnd = 0.2; p(AnBnC|~D)
d = 0.05; p(D)
dgaibic; p(D|AnBnC)
nd : ****TARGET VARIABLE****; p(~D)

Equations

For nd: *Apply the complement theorem on event D and ~D.*
1) dgaibic=aibicgd*d/ (d*aibicgd+nd*aibicgnd)
For dgaibic: *Apply the bayes' rule on D|AnBnC; here, event D and ~D are mututally exclusive and exhaustive events*

The Tutor says:

Please enter the equation for "Apply the complement theorem on event D and ~D. " then press the Submit button:

Your response:

1 = d + nd

Figure 2: The Pyrenees user interface showing the problem statement at the top, the variable and equation lists on the left, and the tutor interaction window with calculator on the lower right.

solution paths. This was motivated in part by a desire to make the system work in many different educational contexts where instructors have their own preferred methods [20]. The designers of Andes also consciously chose only to provide advice upon demand when the students would be most willing to accept it. For the students however, particularly those with poor problem-solving skills, this passive guidance and comparative freedom can be problematic as it does not force them to adhere to a strategy.

This problem motivated the development of Pyrenees. Pyrenees, like Andes acts as a homework helper and supports students with on-demand procedural and remediation help. It uses an isomorphic domain model with the same principles, basic steps, problems, and solution paths. Unlike Andes, however, Pyrenees forces students to applying the target-variable-strategy during problem solving. It also requires them to repair incorrect entries immediately before moving on. Students are guided through the solution process with a menu-driven interface, shown in Figure 2. At each step, the system asks students what they want to work on next and permits them to make any valid step that is consistent with the TVS. Chi and VanLehn [3] conducted a study of the two systems and found that scaffolding the TVS in Pyrenees helped to eliminate the gap between high and low learners. This effect was observed both in the original domain where it was taught (in their case probability) and it transferred to a new domain (physics), where students used Andes alone.

2.2 Data-Extraction and Data-Driven Tutoring.

One of the longstanding goals of educational data-miners is to support the development of data-driven tutoring systems. Such systems use past student data to structure pedagogical and domain knowledge, administer conceptual and pedagogical advice, or evaluate student performance and needs. A number of attempts have been made to address these goals. One of the most successful data-driven systems is the Hint Factory [1, 2, 17]. The Hint Factory takes an MDP-based approach to hint generation. It takes as input a set of prior student logs for a given problem, represented as a network of interactions [6, 7]. Each vertex in this network represents the state of a student's partial solution at some point during the problem solving process, and each edge represents an action that takes the student from one state to another. A complete solution is represented as a path from the initial state to a goal state. Each state in the interaction network is assigned a weight via a value-iteration algorithm. A new student requesting a hint is matched to a previously observed state and given context-sensitive advice. If, for example, the student is working on a problem that requires Bayes' Rule and has already defined $p(A)$, $p(B)$, and $p(B|A)$ then the Hint Factory would first prompt them to consider defining $p(A|B)$, then it would point them to Bayes Rule, before finally showing them the equation $p(A|B) = (p(B|A) * p(A))/p(B)$.

These hints are incorporated into existing tutoring systems in the form of a lookup table that provides state-specific advice. When a user asks for help the tutor will match their current state to an index state in the lookup table and will prompt them to take the action that will lead them to the highest value neighboring state. If their current state is not found then the tutor will look for a known prior state or will give up. The Hint Factory has been applied successfully in a number of domains including logic proofs [17], data structures [8], and programming [15, 10, 13]. Researchers have also explored other related methods for providing data-driven hints. These include alternative state representations [13], path construction algorithms [16, 14], and example-based model-construction [12].

The primary goal of the Hint Factory is to leverage prior data to provide optimal state-specific advice. By calculating advice on a per-state basis, the system is able to adapt to students' specific needs by taking into account both their current state and the paths that they can take to reach the goal. As a consequence the authors of the Hint Factory argue that this advice is more likely to be in the students' Zone of Proximal Development and thus more responsive to their needs than a less-sensitive algorithm.

3. METHODS

In order to investigate the application of data-driven methods to model-based tutoring systems, we collected data from two studies conducted with Andes and Pyrenees in the domain of probability. We then transformed these datasets into interaction networks, consisting of states linked with actions. We used this representation to perform a variety of quantitative and qualitative analyses with the goal of evaluating the differences between the two systems and the impact of the specific design decisions that were made in each.

3.1 The Andes and Pyrenees Datasets

The Andes dataset was drawn from an experiment conducted at the University of Pittsburgh [3]. This study was designed to assess the differential impact of instruction in Andes and Pyrenees on students' meta-cognitive and problem-solving skills. Participants in this study were college undergraduates who were required to have taken high-school level algebra and physics but not to have taken a course in probability or statistics. The participants were volunteers and were paid by time not performance.

Forty-four students completed the entire study. However for the purposes of the present analysis, we drew on all

66 students who completed at least one problem in Andes-Probability. This is consistent with prior uses of the Hint Factory which draw from all students including those who did not complete the problem. The Pyrenees-Probability logs from this study were not used due to problems with the data format that prevented us from completing our analysis. From this dataset we drew 394 problem attempts covering 11 problems. The average number of steps required to solve the problems was 17.6. For each problem we analyzed between 25 and 72 problem attempts, with an average of 35.8 attempts per problem. Some attempts were from the same student, with at most two successful attempts per student. Over all problems, 81.7% of the attempts were successful, with the remainder being incomplete attempts.

The Pyrenees dataset was drawn from a study of 137 students conducted in the 200-level Discrete Mathematics course in the Department of Computer Science at North Carolina State University. This study used the same probability textbook and pre-training materials as those used in the Andes study. The students used Pyrenees as part of a homework assignment, in which they completed 12 problems using the tutoring system. One of these problems was not represented in the Andes dataset. We therefore excluded it from our analysis, leaving 11 shared problems.

Unlike the Andes students, however, the Pyrenees students were not always required to solve every problem. In this study the system was configured to randomly select some problems or problem steps to present as worked examples rather than as steps to be completed. In order to ensure that the results were equivalent we excluded the problem-level worked examples and any attempt with a step-level worked example from our analysis. As a consequence, each problem included a different subset of these students. For each problem we analyzed between 83 and 102 problem attempts, with an average of 90.8 attempts per problem. Some attempts were from the same student, with at most one successful attempt per student. Over all problems, 83.4% of the attempts were successful.

3.2 State and Action Representations

In order to compare the data from both tutors, we represented each problem as an interaction network, a representation used originally in the Hint Factory [7]. In the network a vertex, or state, represents the sum total of a students' current problem solving steps at a given time during a problem-solving attempt. Because Andes permits flexible step ordering while Pyrenees does not, we chose to represent the problem solving state s_t as the set of valid variables and equations defined by the student at time t.

A variable is a probabilistic expression, such as $P(A \cup B)$, that the student has identified as important to solving the problem, for which the probability is known or sought. An equation represents the application of a principle of probability, which relates the values of defined variables, such as the Complement Theorem, $P(A) + P(\neg A) = 1$. Because such equations can be written in many algebraically equivalent ways, we represent each equation as a 2-tuple, consisting of the set of variables included in the equation (e.g. $\{P(A), P(\neg A)\}$) and the principle being applied (e.g. Complement Theorem). Because we only represent valid equations, this representation uniquely identifies any equation for a given problem. Because we used the same state representation for both tutors, we were able to compare states directly across tutors.

Additionally, we opted to ignore incorrect entries. Pyrenees prevents students from applying the principles of probability improperly and forces them to correct any mistakes made immediately therefore any errors in the student logs are immediately removed making the paths uninformative. Andes, by contrast, gives students free reign when writing equations and making other entries. This freedom resulted in syntactic errors and improper rule application errors arising in our dataset. The meaning of these invalid equations is inherently ambiguous and therefore difficult to incorporate into a state definition. However such errors are immediately flagged by the system and may be ignored by the student without consequence as they do not affect the answer validity therefore they may be safely ignored as well.

An edge, or action, in our network represents the correct application of a rule or a correct variable definition and leads to a transition from one state to another. For the present dataset and state representation, the possible actions were the definition or deletion of variables or equations. Each of these actions was possible in both tutors.

4. ANALYSIS

In order to develop a broader understanding of our datasets, we first visualized the interaction network for each problem as a weighted, directed graph. We included attempts from both Andes and Pyrenees in the network, and weighted the edges and verticies by the frequency with which it appeared in the logs. We annotated each state and edge with the weight contributed by each tutor. Two examples of these graphs are given in Figure 3.

Throughout this section, we will use these graphs to address the points we outlined at the end of Section 1. We begin with a case study from one problem and will explore the student problem solving strategies using our graph representation. We will then compare the Andes and Pyrenees Systems with a variety of metrics based on this representation. We will relate our observations back to the design decisions of each system and identify evidence that may support or question these decisions. Finally, we will show how the analysis methods associated with data-driven hint generation can be used to validate some of these findings.

4.1 Case Study: Problem Ex242

The graphical representation of a problem is very helpful for giving a high-level overview of a problem and performing qualitative analysis. Problem *Ex242*, shown in Figure 3, presents an interesting scenario for a number of reasons. The problem was the 10th in a series of 12 practice problems, and asked the following:

> Events A, B and C are mutually exclusive and exhaustive events with $p(A) = 0.2$ and $p(B) = 0.3$. For an event D, we know $p(D|A) = 0.04$, $p(D|B) = 0.03$, and $p(C|D) = 0.3$. Determine $p(B|D)$.

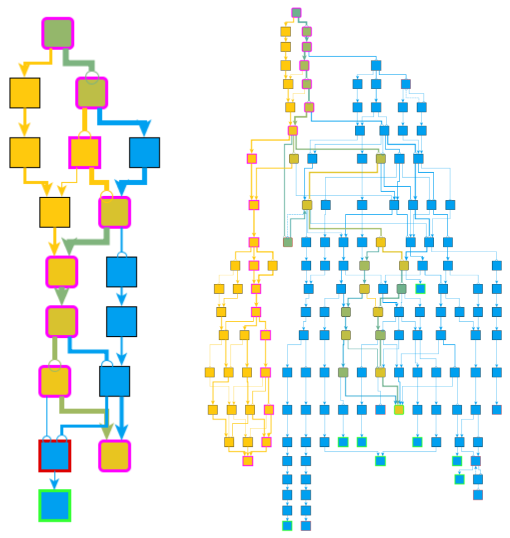


Figure 3: A graph representation of problems Ex252a, left, and Ex242, right. States and edges are colored on a gradient from blue to yellow, indicating the number of students who reached that state in the Andes and Pyrenees tutors respectively. Rounded edges indicate that at least one student from both tutors is present in a state. A green border indicates a solution state, and a pink border indicates that a state is contained in the pedagogically "ideal" solution. Edge thickness corresponds to the natural log of the number of problem attempts which included the given edge.

The problem is notable in the Pyrenees dataset because it was the only one in which the students were split almost evenly among two solution paths. For most of the problems in the dataset the vast majority of students followed the optimal solution path with only a few finding alternatives. The ideal solution path, as suggested by Pyrenees' domain model, employed repeated applications of the Conditional Probability Theorem: $P(A \cap B) = P(A|B)P(B)$, which the problem was designed to teach. The students had been previously exposed to Bayes' Theorem however, and over half of them chose to apply it instead. This allowed them to circumvent one variable definition and two applications of the Conditional Probability Theorem, achieving a slightly shorter solution path. We make no argument here which path the tutor should encourage students to take, but it is worth noting that the Hint Factory, when trained on the Pyrenees data for this problem, recommends the shorter, more popular path.

The Andes dataset gives us a very different set of insights into this problem. Because Andes lacks the strong process scaffolding of Pyrenees, students were able to make a wider variety of choices, leading to a graph with many more, less populous states. While almost every state and edge in the Pyrenees graph represents multiple students, the Andes graph contains a number of paths, including solutions, that were reached by only one student. In some state-based analyses the authors choose to omit these singleton states, for instance when generating hints. We have chosen to include them as they represent a fair proportion of the Andes data. For instance, 62 of the 126 Andes states for Ex242 were singleton states.

Interestingly, while there were small variations among their solutions, all of the Andes students choose to apply Bayes' Rule rather than relying solely on the Conditional Probability Theorem as suggested by Pyrenees. This, coupled with the strong proportion of Pyrenees students who also chose the Bayes' Rule solution, indicates that the solution offered by Pyrenees may be unintuitive for students, especially if they have recently learned Bayes' Rule. Again, this can be interpreted as evidence that Pyrenees' strong guidance did have an impact on students' problem solving strategies, but it also raises concerns about how reasonable this guidance will appear to the students. Regardless of one's interpretation, an awareness of a trend like this can help inform the evolution of model-based tutors like Andes and Pyrenees.

4.2 Comparing datasets

We now turn to qualitatively comparing the datasets. While it is not a common practice to directly compare data from different tutors, we argue that it is appropriate, especially in this context. In longstanding tutoring projects it is common for developers and researchers to make many substantive changes. The Andes system itself has undergone substantial interface changes over the course of its development [20]. These changes can alter student behavior in substantial ways, and it is important for researchers to consider how they affect not just learning outcomes but also problem solving strategies, as was investigated by Chi et al. [4].

In many respects the close relationship between Andes and Pyrenees makes them analogous to different versions of the same tutor and the presence of an isomorphic knowledge base and problem set makes it possible for us to draw meaningful comparisons between students. In this section we will inspect how the scaffolding design decisions made when constructing the tutors affected the problem solving strategies exhibited by the students.

A visual inspection of the state graphs for each problem revealed significant portions of each graph were shared between the two datasets and portions that were represented in only one of the two. Despite the fact that students using Andes were capable of reaching any of the states available to students in Pyrenees, many Pyrenees states were never discovered by Andes students. As noted in Section 4.1, this suggests that guidance from the Pyrenees tutor is successful in leading students down solution paths that they would not otherwise have discovered, possibly applying skills that they would otherwise not have used.

To quantify these findings, we calculated the relative similarity of students in each tutor. For a given problem, we defined the state-similarity between datasets A and B as the probability that a randomly selected state from a student in A will be passed through by a randomly selected student in B. Recall from Section 3.2 that our state representation allows us to directly compare states across tutors. By this definition, the self-similarity of a dataset is a measure of how closely its students overlap each other while the cross-similarity is a measure of how closely its students overlap

States	Andes	Pyrenees	Solution
Andes	0.551 (0.134)	0.494 (0.153)	0.478 (0.186)
Pyrenees	0.460 (0.141)	0.688 (0.106)	0.669 (0.146)
Actions	Andes	Pyrenees	Solution
Andes	0.878 (0.085)	0.874 (0.118)	0.851 (0.140)
Pyrenees	0.828 (0.117)	0.936 (0.021)	0.923 (0.036)

Table 1: Pairwise similarity across tutors and the ideal solution path. Similarities were calculated for each problem, and each cell lists the mean (and standard deviation) over all problems. The top half covers the state similarity metrics while the bottom half of each table covers action similarity.

with the other dataset. Similarity measures for the datasets can be found at the top of Table 1.

Predictably, both datasets have higher self-similarity than cross-similarity, with Pyrenees showing higher self-similarity than Andes. This indicates that Pyrenees students chose more homogeneous paths to the goal. This is reasonable and consistent with the heavy scaffolding that is built into the system. It is important to note that our similarity metrics are not symmetric. The cross-similarity of Pyrenees with Andes is higher than the reverse. This indicates that the path taken by Pyrenees students are more likely to have been observed by Andes students than vice-versa. This has important implications for designers who are interested in collecting data from a system that is undergoing modifications. If a system becomes increasingly scaffolded and restrictive over time, past data will remain more relevant than in a system that is relaxed. In many ways this simply reflects the intuition that allowing students to explore a state space more fully will produce more broadly useful data, and restricting students will produce data that is more narrowly useful. Note that here we are only observing trends, and we make no claims of statistical significance.

In our analysis, we found that, within both datasets, many solution paths or sub-paths differed only in the order that actions were performed. In our domain, many actions do not have ordering constraints. It is possible, for example, to define the variables A and B in either order, and the resulting solution paths would deviate from one another. We thus sought to determine how much of the observed difference between our two datasets was due to these ordering effects. To that end we define the *action-similarity* between datasets A and P as the probability that a randomly selected action performed by a student in A will have been performed by a by a randomly selected student in B. These values are shown in the bottom of Table 1, and each of the trends observed for state-similarity hold, with predictably higher similarity values.

It is notable that the similarity between Pyrenees and Andes is almost as high as Andes' self-similarity, indicating that the actions taken by Pyrenees students are almost as likely to be observed in Andes students as Pyrenees students. This suggests that, for the most part, the Andes students performed a superset of the actions performed by the Pyrenees students. Thus the impact of Pyrenees is most visible in the *order* of execution, not the actions chosen. This is consistent with the design goals of Pyrenees which was set up to guide students along the otherwise unfamiliar path of the TVS.

States	Andes	Pyrenees	Solution
Andes	0.419 (0.150)	0.327 (0.139)	0.253 (0.172)
Pyrenees	0.372 (0.193)	0.709 (0.145)	0.601 (0.214)
Actions	Andes	Pyrenees	Solution
Andes	0.818 (0.151)	0.582 (0.168)	0.636 (0.205)
Pyrenees	0.678 (0.212)	0.899 (0.038)	0.879 (0.055)

Table 2: Pairwise similarity across tutors and the ideal solution path calculated using a variable-free state representation. Rows and columns are the same as in Table 1.

Problem	**Andes**	**Pyrenees**
ex132	26 (47.5%)	2 (98.7%)
ex132a	13 (33.3%)	1 (100.0%)
ex144	2 (96.6%)	1 (100.0%)
ex152	11 (0.0%)	4 (0.0%)
ex152a	8 (59.0%)	3 (97.4%)
ex152b	12 (0.0%)	1 (100.0%)
ex212	8 (71.4%)	1 (100.0%)
ex242	9 (0.0%)	2 (49.38%)
ex252	7 (76.9%)	2 (98.4%)
ex252a	4 (81.8%)	2 (98.6%)
exc137	19 (0.0%)	2 (98.75%)

Table 3: For each problem, the tables gives the number of unique solution states represented in each tutor's dataset. Note that there may exist many solution paths which reach a given solution. The following percent (in parentheses) represents the percent solution paths that ended in the pedagogically ideal solution.

We also opted to examine the impact of the variable definitions on our evaluation. As noted above, variable definitions are an atomic action. They do not depend upon any event assertion and thus have no ordering constraints unlike the principles. We did so with the hypothesis that this would increase the similarity metrics for the datasets by eliminating the least constrained decisions from consideration. Our results are shown in Table 2 below. Contrary to our expectations, this actually reduced the similarity both within and across the datasets, with the exception of Pyrenees' self-similarity. Thus the unconstrained variable definitions did not substantially contribute to the dissimilarity. Rather, most of the variation lay in the order of principle applications.

4.3 Similarity to an "ideal" solution

We now turn to exploring how well the ideal solution was represented in the datasets. For both tutors the ideal solution is the pedagogically-desirable path constructed via the TVS. Our measure of cross-similarity between two datasets can also be applied between a single solution path and a dataset by treating the single solution as a set of one. We can thus measure the average likelihood of an ideal solution state appearing in a student's solution from each dataset. The results of this calculation are shown in tables 1 and 2, using both the state- and action-similarities explained earlier. Predictably, the solution has a high similarity with Pyrenees students, as these students are scaffolded tightly and offered few chances to deviate from the path.

As Table 3 shows, Pyrenees students were funneled almost exclusively to the ideal solution on the majority of problems, even if their paths to the solution were variable. We found only one problem, Ex152, where the Pyrenees students missed the ideal path. That was traced to a programming error that forced students along a similar path. Otherwise, there was only one problem, Ex242 (discussed in Section 4.1), where a meaningful percentage of students chose a different solution. The Andes students, by contrast, were much less likely to finish in the ideal solution state, but this was also problem-dependent.

4.4 Applications of the Hint Factory

Finally, having shown that the datasets differ, and that these differences are consistent with the differing design choices of the two tutors, we sought to determine what effect those differences would have on data-driven hint generation. Our goal was to determine how applicable a hint model of the type produced by the Hint Factory would be for one dataset if it was trained on another. To that end we performed a modified version of the Cold Start Experiment [1], which is designed to measure the number of state-specific hints that Hint Factory can provide given a randomly selected dataset. The Cold Start experiment functions like leave-one-out cross-validation for state-based hint generation. In the original Cold Start experiment, one student was selected at random and removed from the dataset, to represent a "new" student using the tutor. Each remaining student in the dataset was then added, one at a time, in a random order to the Hint Factory's model. On each iteration, the model is updated and the percentage of states on the 'new' student's path for which a hint is available is calculated. This is repeated a desired number of times with new students to account for ordering effects.

For the present study we calculated cold-start curves for both the Pyrenees and Andes datasets. We also calculated curves using the opposing dataset to illustrate the growth rate for cross-tutor hints. For these modified curves we selected the hint-generating students from the opposing dataset. All four curves are shown in Figure 4. Here *AvA* and *PvP* designate the within tutor curves for Andes and Pyrenees respectively while *PvA* and *AvP* designate the cross-tutor curves for hints from Pyrenees provided to Andes users and vice-versa. Figure 4 represents an average over all problems, and therefore the x-axis extends only as far as the minimum number of students to complete a problem. As the curves illustrate, the within-tutor curves reach high rates of coverage relatively quickly with *PvP* reaching a plateau above 95% after 21 students and *AvA* reaching 85%.

The cross-tutor curves, by contrast, reach much lower limits. *AvP* reaches a plateau of over 75% coverage, while *PvA* reaches a plateau of 60% coverage. This reflects the same trends observed in Tables 1 and 2, where Andes better explains the Pyrenees data than vice-versa; however, neither dataset completely covers the other. On the one hand this result is somewhat problematic as it indicates that prior data has a limited threshold for novel tutors or novel versions of a

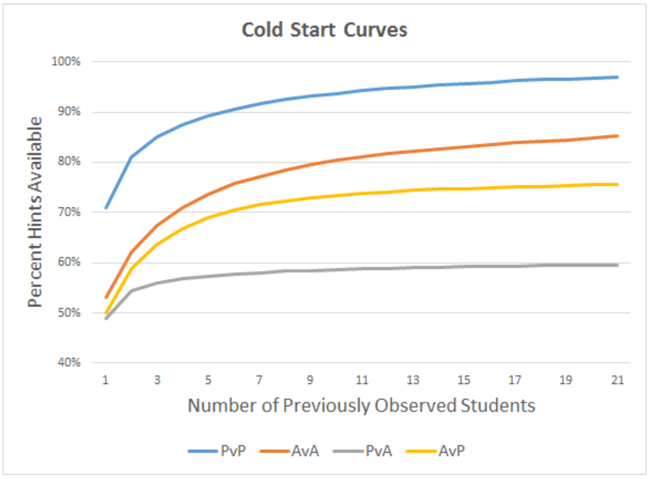


Figure 4: The four Cold Start curves, averaged across all problems. The x-axis shows the number of students used to train the model, and the y-access shows the percentage of a new student's path that has available hints. The curve labeled "XvY" indicates training on the X dataset and selecting a new student from the Y dataset (A = Andes; P = Pyrenees).

system in the same domain. Clearly a substantive interface and scaffolding change of the type made in Pyrenees can change the state space sufficiently that we cannot trivially rely on our prior data. On the other hand, while the cross-application of data does have upper limits, those limits are comparatively high. Clearly data from a prior system can be reused and can serve as a reliable baseline for novel system, with the caveat that additional exploratory data is required.

5. DISCUSSION AND CONCLUSION

Our goal in this paper was to evaluate the application of data-driven methods such as the Hint Factory to model-based tutoring systems. To that end we analyzed and compared datasets collected from two closely-related tutoring systems: Andes and Pyrenees. Through our analysis we sought to: (1) evaluate the differences between closely-related systems; (2) assess the impact of specific design decisions made in those systems for user behaviors; and (3) evaluate the potential application of data-driven hint generation across systems.

We found that, while the systems shared isomorphic domain models, problems, and ideal solutions, the observed user behaviors differed substantially. Students using the Andes system explored the space more widely, were more prone to identify novel solutions, and rarely followed the ideal solution path. Students in Pyrenees, by contrast, were far more homogeneous in their solution process and were limited in the alternative routes they explored. Contrary to our expectations, we found that this variation was not due to simple ordering variations in the simplest of steps but of alternative strategy selection for the higher-level domain principles. This is largely consistent with the design decisions that motivated both systems and with the results of prior studies.

We also found that the state-based hint generation method used in the Hint Factory can be applied to the Andes and Pyrenees data given a suitable state representation. For this analysis we opted for a set-based representation given the absence of strong ordering constraints across the principles. We then completed a cold-start analysis to show that the cross-tutor data could be used to bootstrap the construction of hints for a novel system but does not provide for complete coverage.

Ultimately we believe that the techniques used for data-driven hint generation have direct application to model-based systems. Data-driven analysis can be used to identify the behavioral differences between closely related systems and, we would argue, changes from one version of a system to another. We also found that these changes can be connected to the specific design decisions made during development. Further, we found that data-driven methods can be applied to model-based tutoring data to generate state-based hints. We believe that hint information of this type may be used to supplement the existing domain models to produce more user-adaptive systems. In future work we plan to apply these analyses to other appropriate datasets and to test the incorporation of state-driven hints or hint refinement to existing domain models.

6. ACKNOWLEDGMENTS

Work supported by NSF Grant #1432156 "Educational Data Mining for Individualized Instruction in STEM Learning Environments" Min Chi & Tiffany Barnes Co-PIs.

7. REFERENCES

[1] T. Barnes and J. Stamper. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In *Intelligent Tutoring Systems (ITS)*, pages 373–382, 2008.

[2] T. Barnes and J. C. Stamper. Automatic hint generation for logic proof tutoring using historical data. *Educational Technology & Society*, 13(1):3–12, 2010.

[3] M. Chi and K. VanLehn. Eliminating the gap between the high and low students through meta-cognitive strategy instruction. In *Intelligent Tutoring Systems (ITS)*, volume 5091, pages 603–613, 2008.

[4] M. Chi and K. VanLehn. Meta-Cognitive Strategy Instruction in Intelligent Tutoring Systems: How, When, and Why. *Educational Technology & Society*, 3(1):25–39, 2010.

[5] A. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *User Modeling 2001*, pages 137–147, 2001.

[6] M. Eagle and T. Barnes. Exploring Differences in Problem Solving with Data-Driven Approach Maps. In *Educational Data Mining (EDM)*, 2014.

[7] M. Eagle and T. Barnes. Exploring Networks of Problem-Solving Interactions. In *Learning Analytics (LAK)*, 2015.

[8] D. Fossati, B. D. Eugenio, and S. Ohlsson. I learn from you, you learn from me: How to make iList learn from students. In *Artificial Intelligence in Education (AIED)*, 2009.

[9] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. *Building Expert Systems*. Addison-Wesley Publishing Company Inc., Reading, Massachusetts, U.S.A., 1983.

[10] A. Hicks, B. Peddycord III, and T. Barnes. Building Games to Learn from Their Players: Generating Hints in a Serious Game. In *Intelligent Tutoring Systems (ITS)*, pages 312–317, 2014.

[11] S. Katz, A. Lesgold, E. Hughes, D. Peters, G. Eggan, M. Gordin, and L. Greenberg. Sherlock 2: An intelligent tutoring system built on the lrdc framework. In C. P. Bloom and R. B. Loftin, editors, *Facilitating the Development and Use of Interactive Learning Environments*, Computers, Cognition, and Work, chapter 10, pages 227 – 258. Lawrence Erlbaum Associates, Mawah New Jersey, 1998.

[12] R. Kumar, M. E. Roy, B. Roberts, and J. I. Makhoul. Toward automatically building tutor models using multiple behavior demonstrations. In S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, editors, *Proceedings of the* 12^{th} *International Conference on Intelligent Tutoring Systems*, LNCS 8474, pages 535 – 544. Springer Verlag, 2014.

[13] B. Peddycord III, A. Hicks, and T. Barnes. Generating Hints for Programming Problems Using Intermediate Output. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*, pages 92–98, 2014.

[14] C. Piech, M. Sahami, J. Huang, and L. Guibas. Autonomously Generating Hints by Inferring Problem Solving Policies. In *Learning at Scale (LAS)*, 2015.

[15] K. Rivers and K. Koedinger. Automatic generation of programming feedback: A data-driven approach. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013.

[16] K. Rivers and K. Koedinger. Automating Hint Generation with Solution Space Path Construction. In *Intelligent Tutoring Systems (ITS)*, 2014.

[17] J. C. Stamper, M. Eagle, T. Barnes, and M. J. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *I. J. Artificial Intelligence in Education*, 22(1-2):3–17, 2013.

[18] K. VanLehn, C. Lynch, K. G. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Five years of evaluations. In C. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker, editors, *Artificial Intelligence in Education - Supporting Learning through Intelligent and Socially Informed Technology, Proceedings of the 12th International Conference on Artificial Intelligence in Education, AIED 2005, July 18-22, 2005, Amsterdam, The Netherlands*, volume 125 of *Frontiers in Artificial Intelligence and Applications*, pages 678–685. IOS Press, 2005.

[19] K. VanLehn, C. Lynch, K. G. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Lessons learned. *I. J. Artificial Intelligence in Education*, 15(3):147–204, 2005.

[20] K. VanLehn and B. van de Sande. The Andes physics tutoring system: An experiment in freedom. *Advances in intelligent tutoring systems.*, pages 421–443, 2010.