Do Students Who See More Concepts in an ITS Learn More?

Moffat Mathews and Tanja Mitrović
{moffat, tanja}@cosc.canterbury.ac.nz
Intelligent Computer Tutoring Group, Computer Science & Software Engineering,
University of Canterbury

Abstract. Active engagement in the subject material has been strongly linked to deeper learning. In traditional teaching environments, even though the student might be presented with new concepts, it is possible for the student to remain passive to such an extent that it is detrimental to learning. This research explores whether experiencing new concepts in an ITS necessarily equates to learning. Initial analysis of data mining student models in SQL-Tutor, a CBM tutor, shows a strong positive correlation between the number of constraints seen and the number of constraints learned. This global trend is mitigated at an individual level, possibly due to individual differences in learning style and behavior. The number of constraints not learned remains relatively constant for all students; however, the proportion of constraints not learned is inversely proportional to the constraints seen. The author suggests deeper analysis into the factors that might cause variability amongst individuals from this population trend.

1 Introduction

For many years, researchers have been stating the importance of active participation and engagement in learning [1, 2]. Constructivists would argue that for effective learning to occur, the student has to be actively involved in constructing the appropriate mental model of the domain, ideally guided by the teacher [3]. Learning which requires the student to actively construct their knowledge leads to better performance (both in time and the number of important errors) [4]. Students who work at constructing their own knowledge using methods such as self-explanation, become better problem solvers [5]. In contrast, students that passively sit in a traditional class environment learn poorly compared to those that are actively involved in constructing their knowledge [6].

It is easy to understand the scenario of a student who learns very little in a traditional lecture environment even if many new concepts are introduced, because they are disengaged from their learning (e.g. daydreaming). However, does this same principle apply in ITSs? Can students use an ITS and learn nothing because they are so passive with their learning? Basic observation would suggest that there are differences between the way a student learns in a traditional lecture environment and within an ITS. In the traditional lecture environment, the student can be passive (in spite of being presented with new concepts) to the extent that they learn nothing. For example, an extreme case might mean they could have slept through the lecture. The progress through material, including the rate at which it is delivered, is usually dependent on an external source (e.g. the lecturer) rather than the student. However, in an ITS, the student generally must initiate all progress i.e. to be presented with anything, the student has to do something. This might be as small as selecting a new problem or requesting help. Disengaging totally

from the learning would mean not progressing through the ITS. There usually is no external entity built into the ITS that controls and maintains the steady flow and delivery of knowledge. Progress is student-dependent. Due to this basic difference, one could assume that if a student progresses through problems in an ITS (in any manner), they are at least in some way actively involved in some part of their learning. As active participation is correlated to learning, students in an ITS must learn more as they progress through the system.

The aim of this research is to mine student logs in an ITS to see if students learn more as they see and experience more concepts or domain principles in the ITS. The goal of this paper is to complete the first stage of this research i.e. to use simple statistics to find general trends within populations. Once these trends are established, in-depth analysis can be performed.

In the next section we introduce the ITS used, namely SQL-Tutor. After describing the dataset, we outline the methods used to perform this analysis. We finish with a discussion of the results and future work.

2 SQL-Tutor

The data used for this research was gathered from SQL-Tutor [7], an Intelligent Tutoring System. It provides students with intelligent and adaptive guidance as they practice their skills within a specifically designed problem solving environment for the domain of database querying using Structured Query Language (SQL). It has been used by students in tertiary database courses since 1998. Having undergone several evaluation studies, it has been shown to produce high levels (both rates and depth) of learning [8].

The domain in SQL-Tutor is modeled using constraints. Constraints are domain principles. They contain a relevance condition and a satisfaction condition. The relevance condition is the condition in which the constraint is applicable (or is relevant). The satisfaction condition states what must be true for this constraint to be correct. For example, for the domain "driving", one constraint might state "if you are driving in New Zealand, you must be on the left-hand side of the road." The relevance condition informs us on when this constraint applies: *"if you are driving in New Zealand"*. If this constraint is relevant, then for it to be correct, the satisfaction condition must be true, i.e. *"you must be on the left-hand side of the road"*. See [9] for a detailed explanation of Constraint-Based Modeling.

Each time a student attempts a problem in SQL-Tutor, the constraints that are relevant for the attempt are recorded in the student model. For each relevant constraint, a history of correct usage for each attempt is also recorded.

SQL-Tutor contains approximately 280 problems. Each problem is assigned a difficulty level by the teacher, using their expertise in the domain. Difficulty levels range from 1 (easiest) to 9 (hardest). For each solution to a problem, several constraints could be relevant i.e. each problem could relate to several concepts. The number and complexity of the relevant constraints also determines the difficulty of the problem. For example, a

problem with many relevant constraints, or a problem with complex relevant constraints could be said to contain many principles or complex principles respectively, making the problem more difficult than one with fewer, simpler constraints. This means that when solving more difficult problems, the student could be dealing with either a large number of constraints or more complex constraints.

3 Data Used

The data for this analysis was collated from student models and logs from an online version of SQL-Tutor. Students from all over the world were given free access to this version of SQL-Tutor when they bought certain database textbooks. Students that made less than five attempts were excluded from this analysis. The final dataset consisted of 1,803 students who spent just over 1,959 combined hours while making a total of 104,062 submissions. In total, these students solved just over 70% (19,604) of the problems they attempted (27,676).

Certain constraints (such as basic syntactic constraints) are always relevant for every query i.e. those concepts apply to every problem. As these constraints are principles that are very easily learned and do not provide in-depth knowledge into the domain or its concepts, they were excluded from this study. The number of constraints that students saw at least once while solving problems varied from 6 to 333. While engaged in their problem solving activities, students experienced a combined total of 174,309 constraints.

4 Method

The method utilized was very simple. We first extracted constraint histories from individual logs and student models. Using this data, we counted the number of unique constraints that were relevant for each student. These are the constraints that the student saw or experienced during their practice; we listed these as "constraints seen".

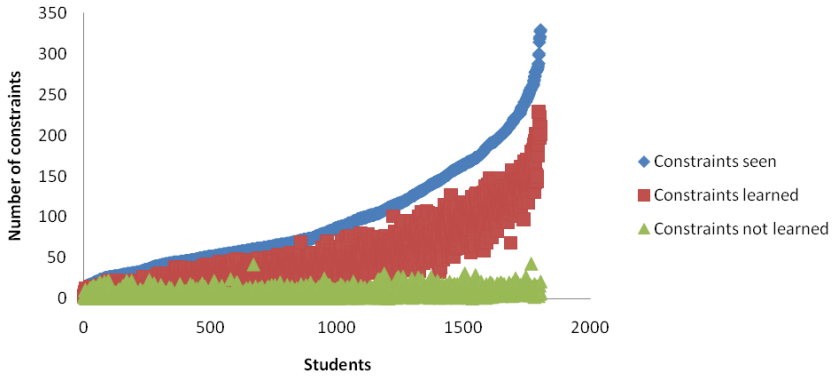

Figure 1: Constraints seen, constraints learned, and constraints not learned for each student, ordered by constraints seen.

To calculate whether a constraint was learned, we used two separate methods. In both methods, we used a window to focus on the recent usage of each constraint. We used the most recent history as we determined that this would better indicate the current state of

learning with regards to that particular constraint. The main difference between the two methods was the size of the window.

In the first method, if the total history of a particular constraint consisted of five or more attempts, we used a window size of five. If the history consisted of less than five attempts, we used a window size of three. In the second method, we always used a window size of five. Each method makes an assumption of how many attempts are required to determine the state of learning for each constraint.

We then calculated the proportion the constraint was learned by dividing the number of correct usages of the constraint in the window by the window size.

$$Proportion\ constraint\ learned = \frac{Correct\ usages\ of\ constraint\ in\ window}{Window\ size}$$

This gave us a number between zero and one, where zero meant that the constraint was not learned at all whereas one meant that the constraint was learned. All other numbers in between zero and one showed the proportion the constraint was learned. We plotted graphs for all users depicting their number of constraints seen, the number of constraints learned, and the number of constraints not learned, using both the methods described above. Both methods above gave very similar graphs. Figure 1 shows the values from method 2. We used these methods as they were previously used somewhat successfully in various versions and evaluation studies.

We also calculated the average difficulty level of problems attempted and problems solved for each student. This is shown in Figure 2.

The total time spent on the system by each student was recorded and graphed against the number of constraints seen (Figure 3). Furthermore, we calculated the *time per constraint seen* and the *time per constraint learned* for each student (Figure 4). Time per constraint seen was calculated by dividing the total time by number of constraints seen. Time per constraint learned was similarly calculated by dividing total time by the number of constraints learned. These calculations were to give us an idea of how much time each student spent in relation to the constraints they had seen (how quickly they were progressing through the system) and to the constraints they had learned (how quickly they were learning constraints).

5 Results and Discussion

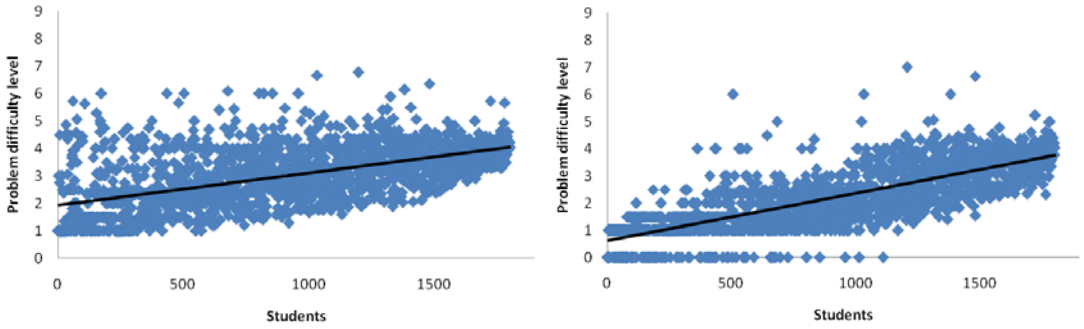

Figure 2: Average difficulty levels of problems attempted (left) and solved (right) for all students ordered by number of constraints seen.

5.1 Constraints

As seen in Figure 1 and from correlation calculations, the number of constraints a student sees is strongly correlated to the constraints learned ($Pearson's\ r = 0.947$). This means that the students who saw more domain principles learned more. This supports our original hypothesis that learning on an ITS requires some form of active engagement which translates to learning. Remember, here we are saying that to have *seen* a constraint, they must have made at least an attempt where that constraint was relevant. Simply moving through and viewing the problem is not counted as progressing through the system as the student would not have experienced any constraints i.e. they have not made any attempts. The variability in the constraints learned could be attributed to individual differences. These differences could be due to the quality and quantity of engagement, the amount of help used, gaming [10], and low skill levels (e.g. low meta-cognitive abilities).

The number of constraints not learned remains relatively constant for all users, regardless of the number of constraints seen. This is interesting, but makes sense as the proportion of constraints that are wrong decreases as the number of constraints seen increases. However, it seems that the types of constraints that are not learned are different between users. Even though all students are getting approximately the same number of constraints wrong, the students who have seen more constraints are dealing with more difficult constraints. This can be seen from the graphs in Figure 2. Students who had low numbers of constraints seen (to the left of each graph) are generally attempting and solving easier problems compared to those who have seen higher numbers of constraints (to the right of each graph). As mentioned earlier, this means that the students on the right of the graph are using more complex constraints or a greater number of constraints in a single problem than the ones on the left.

5.2 *Time*

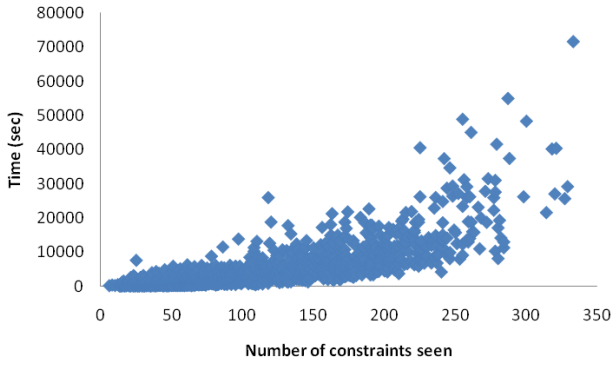

Figure 3: Number of constraints seen against total time taken by students

Figure 3 shows the number of constraints seen plotted against the total time spent in the system. From the graph, we can see that the total time is proportional to the number of constraints seen i.e. the students that saw more constraints spent longer in the system. This is understandable as the greater the number of concepts explored by the student, the longer they will spend in the system. Although this may seem a trivial point, it is included for completeness.

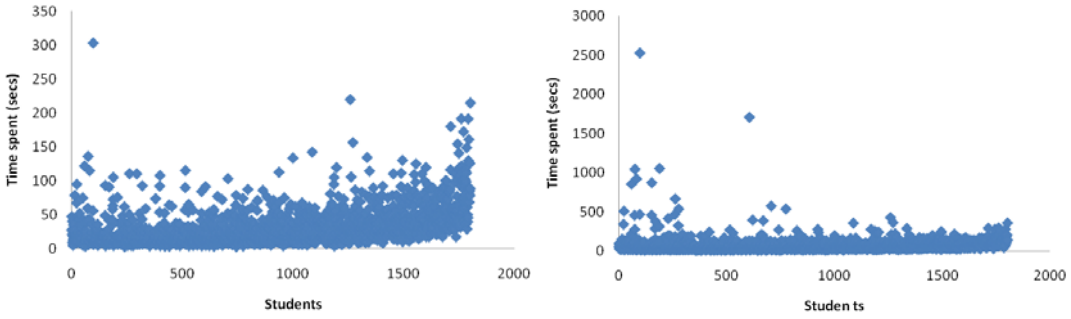

Figure 4: Time spent per constraint seen (left) and constraint learned (right) for each student

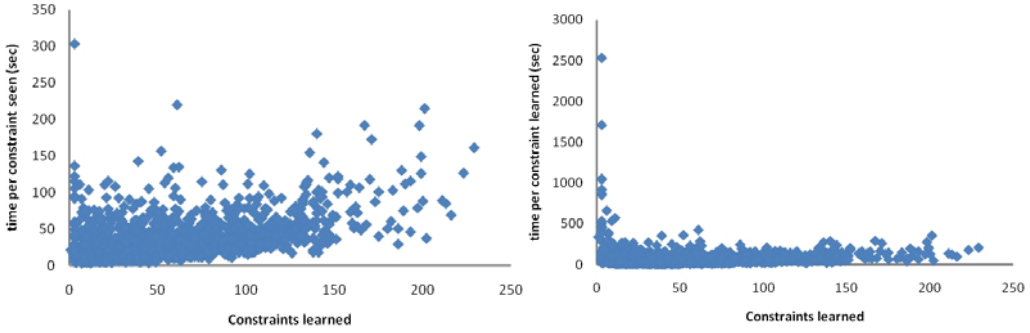

Figure 5: Time spent per constraints seen (left) and constraints learned (right) against constraints learned

The time spent per constraint seen and the time spent per constraint learned (Figure 4) is relatively constant across students, irrespective of the number of constraints seen or learned. In Figure 4, the students who have seen the least number of constraints are towards the left of the graphs. This means that even for the relatively difficult constraints, students on average still spent the same amount of time on each constraint. This could be because students that are seeing more constraints are generally the ones that have progressed to higher levels of expertise. The outliers could be students who are attempting more difficult problems than their expertise level. Similarly, Figure 5 shows that the time spent per constraint seen is relatively constant (or increases slightly with high variability) as more constraints are learned. The time per constraint learned is relatively constant as the number of constraints learned increases. This means that although students spent varying amounts of time on the ITS (depending on the number of constraints they saw), they spent on average the same amount of time per constraint.

6 Conclusion and Future Work

This paper looked at the broad global trends within a population of students using an Intelligent Tutoring System (SQL-Tutor). On a global (population) level, there is a strong positive correlation between the number of constraints a student sees and the number of constraints they learn within an ITS. This seems to support our initial hypothesis that those students who progress through problems in an ITS do learn concepts as they require at least a certain amount of active participation. However, at a more localized level, there are students who have seen many constraints but have learned far fewer constraints than their counterparts who have seen fewer constraints. This could be due to individual differences in the way they learn (e.g. their styles of learning) or in their behavior (e.g. gaming the system). More detailed analysis is required to understand what factors play a part in this process, including factors that affect causality.

We would like to perform deeper analysis on the types of constraints that the students do not learn. What concepts in SQL do they not understand? Are there similarities between these constraints? Are there certain constructs that are inherently more difficult to such a point that no students understand them? Are students guessing their answers and therefore seeing constraints that are not necessarily relevant for the problems they are solving? The answers to these questions would give us better insight into the domain and the way in which students learn concepts.

Another part of this research which requires further exploration is the method by which we calculate constraints learned. In our research, we used two separate methods which gave very similar results. However, *"what does it mean to have learned a constraint or principle?"* is still a very valid question. How much of the constraint history should we use to determine if the constraint has been learned?

In the introduction, we made a comparison between the student in the traditional classroom environment and a student working on an ITS. We said that the extremely passive student in a traditional classroom might not learn anything while the student working on an ITS has a higher chance of learning concepts as they see more concepts. However, it should be noted that the extremely passive student might be one that lacks

motivation to use the ITS. In fact, we did not include anyone who made less than five attempts in our dataset. In this research we do not make any mention about motivating students to learn; merely the trends found in students that do use ITSs.

In spite of these points, this initial study is encouraging, indicating the effectiveness of learning for students that progress through the material in ITSs.

References

[1] R. S. Prawat, "Teachers' Beliefs about Teaching and Learning: A Constructivist Perspective," *American Journal of Education*, vol. 100, pp. 354-395, 1992.

[2] F. N. Akhras and J. A. Self, "System Intelligence in Constructivist Learning," *International Journal of Artificial Intelligence in Education*, vol. 11, pp. 344-376, 2000.

[3] M. Ben-Ari, "Constructivism in Computer Science Education," presented at SIGSCE, Atlanta, GA, USA, 1998.

[4] R. Catrambone and M. Yuasa, "Acquisition of procedures: The effects of example elaborations and active learning exercises," *Learning and Instruction*, vol. 16, pp. 139-153, 2006.

[5] M. T. H. Chi and K. A. VanLehn, "The Content of Physics Self-Explanations," *The Journal of the Learning Sciences*, vol. 1, pp. 69-105, 1991.

[6] L. B. Resnick, "Constructing Knowledge in School," in *Development and Learning: Conflict or Congruence?*, L. S. Liben, Ed. Hillsdale, NJ: Lawrence Erlbaum, 1987, pp. 19-50.

[7] A. Mitrović, "An Intelligent SQL Tutor on the Web," *International Journal of Artificial Intelligence in Education*, vol. 13, pp. 173-197, 2003.

[8] A. Mitrović, B. Martin, and P. Suraweera, "Intelligent Tutors for All: The Constraint-Based Approach," *IEEE Intelligent Educational Systems*, vol. 22, pp. 38-45, 2007.

[9] S. Ohlsson, "Constraint-Based Student Modelling," in *Student Modelling: The Key to Individualized Knowledge-based Instruction*, vol. 125, J. E. Greer and G. I. McCalla, Eds. Berlin: Springer-Verlag GmbH, 1994, pp. 167-189.

[10] R. S. Baker, A. T. Corbett, K. R. Koedinger, and I. Roll, "Detecting When Students Game the System, Across Tutor Subjects and Classroom Cohorts," presented at UM 2005, Berlin, Heidelberg, 2005.