G. Biswas et al. (Eds.) (2012). Proceedings of the 20th International Conference on Computers in Education. Singapore: Asia-Pacific Society for Computers in Education

Data-Driven Misconception Discovery in Constraint-based Intelligent Tutoring Systems

Myse ELMADANI[*], Moffat MATHEWS[b] & Antonija MITROVIC[c]
Intelligent Computer Tutoring Group, University of Canterbury, New Zealand
*myse.elmadani@pg.canterbury.ac.nz

Abstract: Students often have misconceptions in the domain they are studying. Misconception identification is a difficult task but allows teachers to create strategies to appropriately address misconceptions held by students. This project investigates a data-driven technique to discover students' misconceptions in interactions with constraint-based Intelligent Tutoring Systems (ITSs). This analysis has not previously been done. EER-Tutor is one such constraint-based ITS, which teaches conceptual database design using Enhanced Entity-Relationship (EER) data modelling. As with any ITS, a lot of data about each student's interaction within EER-Tutor are available: as individual student models, containing constraint histories, and logs, containing detailed information about each student action. This work can be extended to other ITSs and their relevant domains.

Keywords: Misconceptions, data mining, constraint-based modeling

Introduction

It is common for students to make errors while solving problems. Intelligent Tutoring Systems (ITSs) usually respond very well to such errors by identifying the exact error and providing appropriate instructional feedback at varying levels, from error flags to bottoming out and providing the solution. Some of these ITSs also provide hints on what to do next in a particular step or problem in the student's solution. Both Cognitive Tutors [8] and Constraint-based tutors [10] have domain knowledge represented at an atomic level to give the most precise and helpful feedback on the incorrect step or solution state. Domain knowledge is represented in Cognitive Tutors as production rules or as constraints in Constraint-Based Modelling (CBM) tutors.

Making an error, however, could mean that the student has either got something genuinely incorrect (e.g. in the case of a novice who has faulty or incomplete knowledge) or they have made a slip (i.e. entered an incorrect answer accidentally). Long-term modelling of the student could show whether such cases are slips or genuine errors. In this paper, we are only concerned with genuine errors.

Some students fairly consistently make errors in certain larger portions of the domain (at a higher conceptual level). Glancing at the individual constraints this might not be obvious because of the low level of granularity at which constraints represent domain knowledge. Each constraint covers only one aspect of a domain concept. Such a fine level of granularity is necessary in order to provide very specific feedback on the error, so that the student can correct it. However, students do have misconceptions that cover several constraints. In constraint-based tutors, constraints are independent of each other, but subsets of constraints do cover the same domain concept. For example, student *A* violates constraint 27 every time they violate constraint 349 and 243. Even though our ITSs do a relatively good job in correcting these errors eventually via their atomic feedback, it is still pedagogically interesting to find out why these errors occur frequently together. Finding out these reasons might also help us instantiate different pedagogical strategies in these cases.

With genuine errors, students often have misconceptions at a conceptual level rather than just at a step or constraint level. These misconceptions may be due to the student having incorrect or incomplete domain knowledge involving parts of several concepts. Having misconceptions could result in the student making a number of uniquely different but similarly related errors while solving problems in that part of the domain.

Misconceptions and domain concepts are more abstract than the production rules or constraints. Many constraints, therefore, form part of a concept and a misconception results in many constraints (related to that concept) being violated. Often misconceptions are fudged within buggy rules or constraint feedback as the ITS author assumes that there must be a misconception occurring here if this step/state was done incorrectly.

It would be very useful if we could identify common misconceptions, not just at the rule/constraint level, but at a broader conceptual level, empirically, and have pedagogical strategies to deal with these known misconceptions. Normally, this is difficult, requiring human domain experts to manually observe large numbers of students working on tasks within that domain [5]. During the identification process, experts attempt to deduce misconceptions that students might have about that part of the domain by: using their expertise (e.g. knowing what is difficult in the domain); their account of each student's task-specific behaviour (e.g. errors the student made); and if available, introspective notes from the student. Once these misconceptions are identified, teaching strategies can be better aligned with those knowledge areas that are incorrect or lacking. The automation of misconception discovery would not only be of benefit for both teachers and students but would provide another level of adaptivity in ITSs.

The data-driven technique trialled in [5] provides a semi-automated method for misconception discovery. The study consisted of identifying domain misconceptions through the analysis of students' answers to multiple-choice test questions. Our project modifies the technique to elicit misconceptions in student knowledge in constraint-based tutors with more complex, ill-defined tasks. To show how this technique could be used, we implemented it in the domain of Enhanced Entity-Relationship (EER) data modelling [4] as taught to students through EER-Tutor [16, 17, 20]. This research investigates whether a list of misconceptions can be created; one that will reflect how constraints are actually used by students. The data-driven technique can then be applied to other ITSs. Having this new information about common misconceptions, the ITS will not only be able to identify that a student has a misconception and provide additional misconception-specific guidance (more than what a rule or constraint might offer) but also offers other possible applications. An example application is to create novel tasks for students, like the dynamic generation of erroneous solutions for students to correct. Such erroneous solutions are currently being created manually in other projects in the Intelligent Computer Tutoring Group.

In this paper, we discuss whether eliciting misconceptions in this manner is even possible in a constraint-based tutor within an ill-defined task (EER-Modelling) by data mining student models and logs from EER-Tutor.

1. EER-Tutor

EER-Tutor is a constraint-based ITS which provides an intelligent learning environment for students to practise and learn database design using the EER data model. Currently, EER-Tutor has 57 problems, where each subsequent problem increases in difficultly such that problem 1 is the easiest and problem 57 is the most difficult. Users create EER schemas satisfying a set of requirements which are checked for constraint violations on submission [16, 17]. EER-Tutor records detailed session information, including each student's attempt at each problem and its outcome, as well as the history of each constraint [18, 20].

A constraint is an ordered pair <Cr, Cs>, where Cr is the relevance condition and Cs is the satisfaction condition [11]. EER-Tutor has 225 constraints which evaluate the student's solution for semantic and syntactic correctness. When checking a student's solution, the ITS uses the relevance condition to check whether a constraint is relevant for this particular solution. If it is relevant, then it checks whether the student's solution adheres to the satisfaction condition. The constraint is violated otherwise, indicating an error in the submitted schema. Appropriate corrective steps are then taken by the ITS. To learn from their errors requires that students be able to detect their errors before taking corrective steps [12]. On the basis of violated constraints, the system generates feedback, which allows the student to correct their errors.

We have large datasets of anonymised interaction data from our tutors as they are used by students worldwide. These datasets allow us to conduct data mining based research. Data collected is not from controlled experiments; the students used EER-Tutor over the Web, in a way that suited them.

2. Related Work

Educational data mining is a growing field that focuses on the analysis of large datasets of student-computer interaction logs to answer educational research questions [3]; "an emerging discipline, concerned with developing methods for exploring the unique types of data that come from educational settings, and using those methods to better understand students, and the settings which they learn in" [7].

One such way of exploring data is to use association rules. An *association rule* represents relationships between different attribute-value pairs; stated simply, if there is an association between two attribute-value pairs (X,x) and (Y,y), then if X=x then it must also hold that Y=y. Association rule inference algorithms take a set of uniquely identifiable transactions, with each transaction being a set of attribute-value pairs that occur together [5]. An *itemset* is the set of attribute-value pairs or items. An itemset is *large* if it appears at least as many times in the transaction dataset as required by the predefined *minimum support* value [1]. Non-large itemsets are discarded and do not appear in the output association rules. Each association rule also has a *support*, a number of transactions containing the itemset and a *confidence*, the number of data instances it correctly predicts.

Current work in using a data-driven technique for semi-automatic misconception elicitation in a domain is described in [5]. This research involved the discovery of potential misconceptions by identifying the most frequent associations among incorrect answers in student solutions to multiple-choice questions. The hypothesis was that incorrect answers reflect misconceptions held by the student. The result of mining is a set of incorrect choices selected most frequently by the students. These itemsets were ordered by frequency, with the most frequent answer associations corresponding to potential misconceptions. Domain experts were required at this stage to identify which potential misconceptions could in fact be considered misconceptions. They found that students had the misconception "misunderstanding of the difference between binary and text files" for example.

The elicited misconceptions form a misconception layer (Figure 1) in the evidence model proposed in [5]. The model specifies the relationships between individual tasks, assessment activities such as test questions, and their relevant domain-specific concepts and misconceptions. This knowledge aids in more personalised and specific tutoring, such as problem selection for students based on their known misconceptions about a domain. We

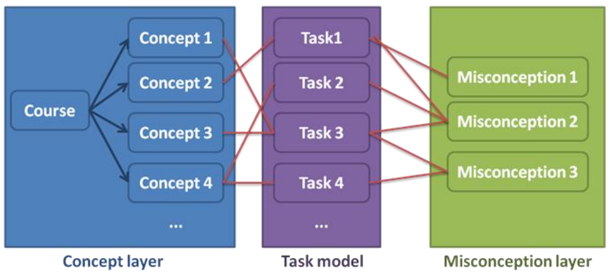

Figure 1. Relationships between misconceptions and domain concepts [5]

therefore provide an adaptation of this technique described in [5] for use with constraint-based ITSs.

INFER* and MALGEN [15] are rule-based algorithms for identifying misconceptions. Using incorrect student actions, INFER* creates faulty rules. MALGEN conversely modifies existing operators to create faulty ones and tests them out. The suitability of rules generated by both algorithms is considered by human experts before inclusion in the bug library. ASSERT [2] was the first student modeling system to automatically create bug libraries. It uses theory refinement, which takes examples of student's behavior as input and modifies and creates rules if the behavior cannot be explained with the existing domain rules. MEDD [14] learns student and reference Prolog programs and uses similarity- and causality-based clustering of discrepancies between them. Each reference program has an associated error hierarchy, which is refined using the discrepancies.

3. Design and Implementation

Starting with all student models and logs collected by EER-Tutor from 2004 to 2006[1], we extracted the data about students who had attempted at least one problem. Here, an attempt is a student's solution submission, which could contain multiple steps. We decided to use all resulting 1135 student models and logs together for this project but different subsets can be processed separately in the future.

The logs were pre-processed (see Figure 2) to simplify information extraction. We extracted the required data from each pre-processed log and stored it in a database. The data is sparse because for each problem there is a number of relevant constraints, but not all constraints are relevant for every attempt at the problem. We output the data to sparse Attribute-Relation File Format (ARFF) files in order to carry out the data-mining process. There were 53,360 attempts at the various EER-Tutor problems in this data set.

RapidMiner (RM) is an open source data mining system [9]. RM provides an implementation of the Frequent Pattern-Growth (FP-Growth) algorithm [6], which generates frequent itemsets that can be used to generate association rules. The choice of algorithm differs from that in [5] because the use of Apriori with ARFF files did not allow us to treat irrelevant constraints as such within RM.

The process involved reading the ARFF files, converting all nominal values to binominal attributes and inputting all the data into the FP-Growth operator. This outputted all the frequent itemsets, which were inputs into the Create Association Rules operator to generate the association rules with a minimum confidence of 0.9. Initially the minimum

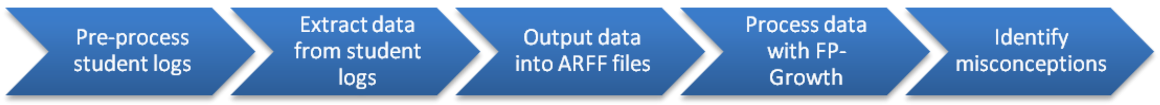
Pre-process student logs
Extract data from student logs
Output data into ARFF files
Process data with FP-Growth
Identify misconceptions

Figure 2. A high-level outline of the data-driven technique described

[1] From Addison-Wesley's DatabasePlace Web portal: http://www.aw-bc.com/databaseplace/

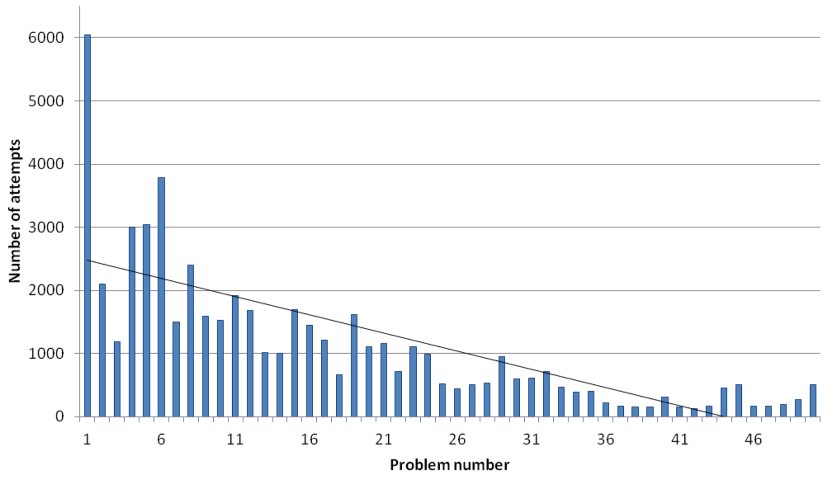

Figure 3. The number of attempts for each problem in EER-Tutor

support of FPGrowth was set to 0.25 as in [5], resulting in an average of 12.14 itemsets for each problem. In order to generate more itemsets, the minimum support was lowered to 0.1, with an average of 159.2 itemsets generated per problem and ranging from 7 itemsets for problems 1, 3 and 12 to 2085 itemsets for problem 23.

As there was a varying number of attempts at each problem, ranging from 124 to 6045 (see Figure 3), the number of attempts required to satisfy the minimum support level also varied. A minimum support of 0.1 requires about 600 attempts in which a particular constraint is violated for problem 1, but only around 13 for problem 24, for example. Instead of controlling minimum support, RM allows us to specify a minimum number of itemsets to be generated. This returns the minimum number of itemsets with the highest support values, regardless of the minimum support value [13].

An alternative method of analysing the data set was to look at all attempts at all problems in the system at the same time, so an ARFF file containing all constraints violated during all attempts at all problems was generated as well. This was processed with the FP-Growth algorithm, with the minimum number of itemsets set to 500, resulting in the generation of 912 itemsets. Due to time constraints, the per-problem ARFF files were not reprocessed using this second mode of FP-Growth. It would be interesting to find out if there are any misconceptions that appear only in specific problems however.

The itemsets output formed a list of potential misconceptions that domain experts had to inspect in order to confirm that they were actual misconceptions. Domain experts inspected itemsets with the largest size first (see Table 1), in order of decreasing support, creating descriptions for each misconception. Both a general and a more specific description were generated for each. An example of such general descriptions is “wrong type” where the specific misconception is that “a regular entity is used instead of a weak entity”.

Table 1. The number of itemsets for all attempts at all problems

Itemset size	Number of itemsets
7	7
6	46
5	131
4	215
3	233
2	211
1	69

4. Results and Discussion

Misconception discovery is difficult, requiring human domain experts to observe large numbers of students in order to identify misconceptions [5]. By identifying which misconceptions are commonly held by students, they can be addressed in order to improve students' knowledge of a particular domain. The data-driven technique described in [5] has been adapted and implemented to identify domain misconceptions through the analysis of student-tutor interaction logs of a constraint-based ITS. Although our implementation used EER-Tutor, this method can be carried out with any constraint-based ITS.

There were a total of 912 itemsets generated by the FP-Growth algorithm, ranging in support from 0.221 to 0.01. This means that there were at least 11,793 attempts in which students violated a particular set of constraints in the first case and 534 in the latter. On average, a constraint appeared in 44 itemsets, with the highest occurrence being 238 for constraint 16. This constraint appeared as a single itemset with a support of 0.126, the equivalent of appearing in over 6700 attempts, and was found to correspond to the misconception related to the use of weak entities (with a label *weak entities are missing*).

Given that there is a total of 912 itemsets generated for all attempts at all problems, the process of labelling misconceptions is currently manual and time consuming. Because the student-ITS interaction logs are already stored and only new itemsets would need to be inspected for labelling in the future, we believe that this process is still manageable. We have started creating a hierarchy of misconceptions in the domain of EER modelling. This is beyond the scope of this paper but will be similar to the hierarchy in [19]. Because of time constraints for this project, domain experts did not inspect all itemsets. Some of the related misconceptions identified are shown in Table 2. Others included “Using simple attributes to represent regular entities” for example. The difficulty in creating the hierarchy is that violated constraints at the bottom of the misconception hierarchy could necessarily mean that the whole branch of the hierarchy is a misconception. The ITS could traverse up that branch until it finds the start of the misconception and take remedial action from that point onwards.

In some cases where two itemsets differed by a single violated constraint both corresponded to the same misconception. The most commonly occurring misconception is “using regular entities instead of weak entities”. Because weak entities are introduced as early as problem 6, the results indicate that this could be a difficult concept for students to master. This misconception was identified for problem 6 in fact, with a support value of 0.306. The understanding of weak entities requires knowledge of the difference between partial and primary keys and between regular and identifying relationships among other concepts. Itemsets corresponding to this misconception could therefore be supersets of other itemsets, which correspond to related misconceptions that can also be addressed separately.

Thousands of association rules with confidence of at least 0.9 were generated. Such a rule is:

constraint21_B → constraint21_AR (confidence 0.976)

This means that when constraint 21_B (used a recursive relationship when one is not needed) is violated, we can be 97.6% confident that the constraint 21_AR (no role names defined for a recursive relationship) is also violated.

Table 2. One related list of identified misconceptions

Misconception	Support
Using regular relationships instead of identifying relationships	0.079
Using identifying relationships instead of regular relationships	0.038
Using regular entities instead of weak entities	0.023
Using weak entities instead of regular entities	0.015

Domain experts need to analyse the generated association rules to determine their usefulness. For example if the student violates constraints corresponding to the misconception that all regular entities require one key amongst the set of keys that is a primary key (i.e. *a regular entity is missing a primary key*), it may be that we could be 90% confident that they will hold the misconception that they are using primary keys to represent entities (i.e. *using a regular entity to represent a primary key*) and will violate the corresponding constraints. In this case, instead of giving atomic feedback relating to each constraint, we could give broader feedback that addresses a potential misconception by explaining both the meaning of entities and primary keys and their association.

Since we were initially analysing the original data set on a per problem basis with a minimum support value of 0.1, we also looked at how many itemsets of various sizes were generated for all attempts at all problems with the same minimum support value. It may be that there are more itemsets for the per problem files (thousands of itemsets in total) as there are duplicates or because some misconceptions are problem specific. The latter case may be due to variance in relevant constraints per problem, an average of 95 across the problems. When eliciting misconceptions for each problem, a problem might be flagged for further investigation if many itemsets are generated extraordinarily; for example, it may have a context that is too unfamiliar to students. As an example, this is what probably occurs for problem 23, where 2085 itemsets or potential misconceptions are generated when minimum support is set to 0.1 despite there being 1,112 attempts at the problem. Problem 23 describes a surveillance system database and includes some specific terminology regarding surveillance systems, which may have confused students; however these would need to be investigated further on a case by case basis.

5. Conclusion and Future Work

Misconception identification is important as it potentially aids the construction of pedagogical strategies to enhance a student's learning. Being able to semi-automate such a difficult and time consuming task allows teachers to focus on enhancements such as customised feedback and problem selection focussing on remedying misconceptions specific to a particular student. Teaching tasks can therefore be better aligned with knowledge areas that are incorrect or lacking. In this paper, we have shown that it is possible to semi-automatically identify domain misconceptions by using a data-driven technique to analyse student-tutor interaction logs of constraint-based ITSs. Students using EER-Tutor have misconceptions about the use of weak entities for example.

Although we have identified some misconceptions in order to determine that this could be done with constraint-based ITSs, the next step is to integrate this knowledge within the ITSs. For example, we can guide problem selection for students according to the specific misconceptions they hold. Further work is also required to make the data-driven technique more streamlined by investigating opportunities for automation in the developed system. In addition, the labeling of misconceptions and the creation of the misconception hierarchy is still a time-consuming and challenging job at this stage, and we can explore improvements such as developing tools to help domain experts. For example, a tool can be developed that displays only a subset of itemsets to the experts for inspection based on factors including similarity between itemsets. This naturally extends to providing some way of presenting the generated association rules to determine whether any are useful, perhaps in guiding feedback provided by the ITS.

Acknowledgements

This research would not have been possible without the support of other members of the Intelligent Computer Tutoring Group at the University of Canterbury, New Zealand.

References

[1] Agrawal, R., & Srikant, R. (1994) Fast algorithms for mining association rules. In Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) *Proceedings of 20th Int. Conf. Very Large Data Bases VLDB*, *1215*, 487-499.

[2] Baffes P., & Mooney, R. (1996) Refinement-based Student Modeling and Automated Bug Library Construction. *J. of Artificial Intelligence in Education*, *7*(1), 75-116.

[3] Baker, R., & Yacef, K. (2009) The state of educational data mining in 2009: A review and future visions. *Educational Data Mining, 1*, 3-17.

[4] Elmasri, R., & Navathe, S. (2007) *Fundamentals of database systems*. Addison Wesley.

[5] Guzman, E., Conejo, R., Galvez, J. (2010) A data-driven technique for misconception elicitation. In De Bra, P., Kobsa, A., Chin, D. (eds.) *User Modeling, Adaptation, and Personalization*, *Lecture Notes in Computer Science*, *6075*, 243-254. Springer Berlin / Heidelberg.

[6] Han, J., Pei, J., Yin, Y., & Mao, R. (2004) Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, *8*, 53-87.

[7] International Working Group on Educational Data Mining (2009) Educational data mining. http://www.educationaldatamining.org/

[8] Koedinger, K., Anderson, J., Hadley, W., & Mark, M. (1997) Intelligent tutoring goes to school in the big city. *Artificial Intelligence in Education, 8*, 30-43 (1997)

[9] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006) Yale: rapid prototyping for complex data mining tasks. In: *Proc of the 12th ACM SIGKDD Int. Conf. Knowledge discovery and data mining*, 935-940. ACM, New York, NY, USA

[10] Mitrovic, A. (2012) Fifteen years of constraint-based tutors: what we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, *22*(1-2), 39-72.

[11] Ohlsson, S. (1992) Constraint-based student modelling. *Artificial Intelligence in Education*, *3*(4), 429-447.

[12] Ohlsson, S. (1996) Learning from performance errors. *Psych. Review*, *103*, 241-262.

[13] Rapid-i: Rapidminer 4.4 class documentation. (2009)
http://rapid-i.com/api/rapidminer4.4/com/rapidminer/operator/learner/associations/fpgrowth/FPGrowth.html

[14] Sison, R., Numao, M., & Shimura, M. (2000) Multistrategy Discovery and Detection of Novice Programmer Errors. *Machine Learning*, *38*, 157–180

[15] Sleeman, D., Hirsh, H., Ellery, I., & Kim, I. (1990) Extending domain theories: two case studies in student modeling. *Machine Learning*, *5*, 11–37.

[16] Suraweera, P., Mitrovic, A. (2002) Kermit: A constraint-based tutor for database modeling. In Cerri, S., Gouardres, G., Paraguacu, F. (eds.) *Proc. Intelligent Tutoring Systems, LNCS*, *2363*, 377-387. Springer Berlin / Heidelberg.

[17] Suraweera, P., & Mitrovic, A. (2004) An Intelligent Tutoring System for Entity Relationship Modelling. *Int. J. Artificial Intelligence in Education*, *14*(3-4), 375-417.

[18] Weerasinghe, A. & Mitrovic, A. (2005) Supporting deep learning in an open-ended domain. In Gh. Negoita, M., Reusch, B. (eds.) *Real World Applications of Computational Intelligence*, *Studies in Fuzziness and Soft Computing*, *179*, 4-75. Springer Berlin / Heidelberg.

[19] Weerasinghe, A., Mitrovic, A., Martin, B. (2008) A Preliminary Study of a General Model for Supporting Tutorial Dialogues. *16th International Conference on Computers in Education*, 125-132.

[20] Zakharov, K., Mitrovic, A., & Ohlsson, S. (2005) Feedback Micro-engineering in EER-Tutor. In C-K Looi, G. McCalla, B. Bredeweg, J. Breuker (Eds.) *Proc. Artificial Intelligence in Education*, IOS Press, 718-725, Amsterdam.