Evaluating the effectiveness of feedback in SQL-Tutor

Antonija Mitrovic and Brent Martin

Intelligent Computer Tutoring Group
Department of Computer Science, University of Canterbury
Private Bag 4800, Christchurch, New Zealand
{tanja,bim20}@cosc.canterbury.ac.nz

Abstract: We present an evaluation of various kinds of feedback in SQL-Tutor. Our initial hypothesis was that low-level feedback, containing all the details of a correct solution would be contra-productive, and that high-level feedback referring to the general principles of the domain that the student's solution violates would be highly effective. The evaluation study performed in 1999 confirmed our hypothesis.

1. Introduction

We have developed **SQL-Tutor**, an Intelligent Teaching System (ITS) that teaches the SQL database language to university students. The system offers various levels of feedback to its students: a complete solution of the current problem, high-level advice on mistakes made, or messages informing the student about the correctness of the solution.

In this paper, we present an evaluation study, the goal of which was to determine the effectiveness of various kinds of feedback. The next section presents the main features of **SQL-Tutor**, while section 3 describes the feedback the system presents to its students. We present our hypothesis in section 4, and describe the evaluation in section 5. The results of the study are given in the next three sections. The final section concludes the paper and discusses future work.

2. SQL-Tutor

SQL-Tutor is a practice environment for students who have learnt about databases in lectures. There are three functionally identical versions for Solaris, MS Windows and the Web. Here we give only a brief description of the system, and the interested reader is referred to other papers [1-4] and the system's Web page[1] for details. Figure 1 illustrates the architecture of **SQL-Tutor**. The system consists of an interface, a pedagogical module that determines the timing and content of pedagogical actions, and a student modeller (CBM), which analyzes student answers. The system contains definitions of several databases, and a set of problems and the ideal solutions to them.

[1]http://www.cosc.canterbury.ac.nz/~tanja/sql-tut.html

At the beginning of a session, **SQL-Tutor** selects a problem for the student to work on. When the student enters a solution, the pedagogical module sends it to the student modeller, which analyzes the solution, identifies mistakes (if there are any) and updates the student model appropriately. On the basis of the student model, the pedagogical module generates an appropriate pedagogical action (i.e. feedback). When the current problem is solved, or the student requires a new problem to work on, the pedagogical module selects an appropriate problem on the basis of the student model.

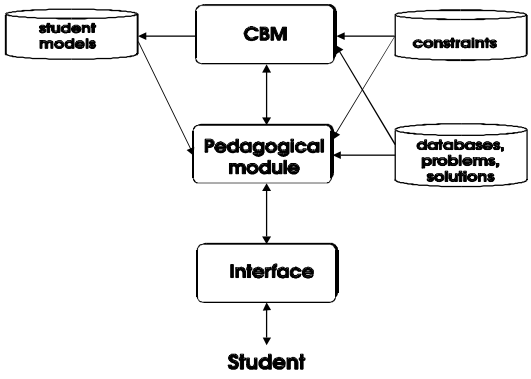

Figure 1. Architecture of SQL-Tutor

SQL-Tutor contains no problem solver. In order to check the correctness of the student's solution, **SQL-Tutor** compares it to the correct solution, using domain knowledge represented in the form of constraints. The system uses Constraint-Based Modeling (CBM) [5] to diagnose students' solutions. The conceptual domain knowledge is represented by over 500 constraints. A student's solution is matched to constraints to identify any that are violated. Student's long-term knowledge is represented as an overlay model, by having a tally for each constraint showing the percentage of correct use.

3. Feedback types

The level of feedback determines how much information is provided to the student. There are six levels of

feedback in **SQL-Tutor**: positive/negative feedback, error flag, hint, all errors, partial solution and complete solution. At the lowest level (*positive/negative* feedback), the message simply informs the student whether the solution is correct or not and, in the latter case, how many errors there are. An *error flag* message informs the student about the clause[2] in which the error occurred. A *hint*-type message gives more information about the type of error, by specifying the general principle that has been violated. This description is directly taken from the constraint. A message of type *all errors* presents the hint messages for all errors the student has made. *Partial solution* feedback displays the correct content of the clause in question, while the *complete solution* simply displays the pre-specified ideal solution of the current problem.

The level of feedback is adjusted in the following way. When a student starts working on a new problem, he/she receives only feedback of the *positive/negative* type. If the student goes through several unsuccessful solution attempts, the feedback is upgraded to the *error flag* level and then to the *hint* level. The system never volunteers more than a *hint*, but the student can ask for *partial* and *complete* solutions by clicking on a *feedback* button and selecting the desired level.

4. The hypothesis

The described mechanism of selecting feedback is overly simple, and is not adaptive. One of our goals is to develop an adaptive mechanism for selecting feedback types. As an initial step towards this goal, we performed an evaluation of effectiveness of various types of feedback available to students. Our initial hypothesis was that the constraint-level feedback (called *hint* or *all-errors* in the context of **SQL-Tutor**) would be most effective (that is, best support students' learning). We hypothesized that *positive/negative* and *error-flag* feedback would be too general to be informative for students, and that *partial-solution* and *complete-solution* feedback would be contra-productive in many cases. Although the student might directly copy latter types of feedback, which maps onto correct solutions in the next submission, we thought that such feedback would not help students to correct misconceptions in the long term.

5. Evaluation study

The evaluation study was carried out in the Computer Science department at the University of Canterbury, on May 4 and 5, 1999, with senior students taking a database course. The students had listened to six lectures about SQL and they all had at least eight hours of hands-on experience of query definition prior to the study. The students used **SQL-Tutor** in a single, two-hour session, and were randomly allocated to one of two versions of the system: one version gave *positive/negative* and *error flag* feedback only, and the other version generated all levels of feedback. All students' actions were recorded and the students filled out a questionnaire at the end of the session.

6. Probability of constraint violation

The first analysis we performed focused on the learning performance. A previous study done on **SQL-Tutor** [4] showed that the degree of mastery of a given constraint is a function of the amount of practice on that unit. We wanted to determine whether feedback would also influence mastery of constraints, and analysed the probability of violating a given constraint C for the nth problem for which the constraint is relevant. To estimate this quantity, we computed, for each student, the proportion of all constraints that he/she violated in the first problem, the second problem, and so on. These proportions were averaged across all subjects and all constraints.

We looked at the learning performance of the students in the two groups (we refer to the group which was offered only two levels of feedback as *limited*, and to the other group as *full*). Figure 2 illustrates the learning performances for the two groups. An explanation of the choice of the cut-off point for this graph is necessary. The students' interactions vary greatly, as students tried various problems at different stages in the interaction. The lengths of sessions were also highly variable. As the number of problems attempted increases, the set of constraints that are relevant diminishes in size. At n=10, the constraint set has, on average, dropped to 32% of the size of the original set at n=1, while at the end of each

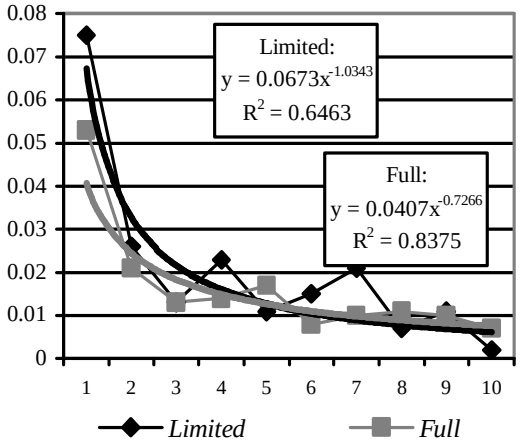

Figure 2. The probability of constraint violation for the two groups

[2] In case that there are several mistakes in various clauses, the pedagogical module will select one of them to start with.

series the set can be as low as 3% of the original. Hence, a single failure will have 30 times the impact on probability as at the start of the curve. We have chosen n=10 to reduce this statistical effect.

When the *full* group is compared to *limited*, the latter has the higher learning rate. However, the existence of the two groups does not allow us to evaluate our hypothesis, as the *full* group received *partial/complete* solution (the detrimental feedback according to our hypothesis) as well as the "good" feedback (*hints/all-errors*). We therefore post hoc split the *full* group into two groups: the *detailed* group used *partial* and *complete* feedback predominantly, while the *general* group used the *hint* and *all-errors* messages.

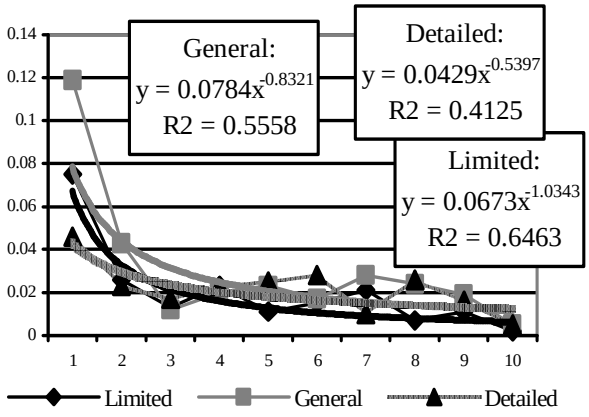

Figure 3. The probability of constraint violation for the three groups

The analysis of learning of the three groups (*limited, general* and *detailed*) is given in figure 3. These results suggest that detailed feedback (i.e. being shown a solution) is detrimental to the rate of learning. However, it is important here to consider possible sources of extraneous effects. In most cases, the group that begins with the highest error rate also has the highest learning rate. The group with the highest initial error rate (which is independent on the feedback) will therefore display the highest initial learning rate.

Furthermore, for the *general* and *detailed* groups, the feedback level was chosen by them, while for the *limited* group the level was artificially determined. It is possible that any trends observed are not because of the effects of feedback, but reflect a characteristic of the students that choose that feedback level.

We also gathered a few statistics on the three groups, given in Table 1. The detailed group solved most problems on average; however, this is due to the fact that the solutions were given to students in this group. It is much more important that the students in the *general* group needed only 2.16 attempts per problem, compared to 2.21 and 2.24 attempts on average for the *detailed* and *limited* groups. Also, the amount of time per attempt is shortest for the *general* group, which is in favour of our hypothesis. This suggests that the "good" feedback, i.e. the feedback messages provided from the constraints, was easier to absorb, and so the time required to understand the feedback and make the necessary changes is substantially reduced. The variation in time required is also heavily reduced, so the worst examples from both the *limited* and *detailed* groups lie many standard deviations outside the distribution for the general group.

Group	Solved	NoAttempts	Time/attempt
Detailed	87.07%	2.21	65.26s
General	83.49%	2.16	47.80s
Limited	84.10%	2.24	78.05s

Table 1. Statistics for the three groups

7. Effect of the feedback on violated constraints

Our hypothesis has a corollary that effective feedback on a violated constraint will increase the chance of that constraint being used successfully the next time. We therefore focused on the effect of feedback received on a violated constraint on the *next* attempt/problem for which the same constraint is relevant. If a particular type of feedback is better than another, we expect to see an increase in the probability that the constraint is used correctly the next time, because the student is more likely to have learned the constraint.

We determined the frequency of a constraint being used successfully after being violated, with respect to a particular level of feedback received on it. Because some feedback types are intended to refer only to the first violated constraint (*error flag, hint, partial solution*), the other violated constraints were treated as having received a level of feedback higher than positive/negative, but lower than any of the other feedback types. This is because the other constraints may indirectly receive feedback (e.g. if they relate to the same clause, and so the same partial solution applies), but at a level which is unknown and variable.

Table 2 presents the frequencies of successful application of a constraint on the next attempt in solving the current problem, after receiving feedback of a specific type. The *Const* column gives the total number

Feedback	Const	Success	Failure	Learned	Correct
Pos/neg	436	254	636	29%	78.0%
Error flag	116	98	126	44%	81.8%
Hint	43	33	43	43%	74.4%
All errors	64	72	81	47%	80.0%
Partial sol	26	22	10	69%	91.6%
Full sol	18	6	16	27%	44.2%

Table 2. The effect of feedback on whole sessions

of constraints which were violated and on which feedback of certain type was given. *Success* is the number of successful applications of the same constraint in the next attempt, while *Failure* specifies the number of times the same constraint was violated following the feedback. Note that the two columns do not add up to *Const*, as there may be several instances of the same constraint violated in student logs. The most frequent type of feedback was positive/negative (a total of 890 messages), while full solution was only given on 22 occasions.

Learned gives the percentage of successful application of the constraint in the next attempt following the feedback. The highest value of *Learned* is obtained for *partial solution*; however, this does not mean that the students have learnt the constraint from such a feedback message. Instead, students typically retype the given solution and submit it. Therefore, there is no real learning involved. If we ignore partial solution, the next best feedback type is *all errors*, followed closely by *error flag* and *hint*. However, these three types of feedback were offered in vary different proportions, with 224 *error flag* messages, 153 messages of the *all errors* type, and only 76 *hint* messages. Only 27% of the solutions made in the attempt following the full solution are correct, and therefore this type if feedback is contra-productive.

The last column *(Correct)* gives the percentage of correct applications of the constraint following the feedback in any future problem. *Partial solution* again has the highest percentage here, but it has only been offered 32 times, which is much less than the number of messages generated for other types of feedback.

8. Focusing on single feedback type

In the previous section, we divided all the students logs into three groups (limited, general and detailed) according to the predominant type of feedback used. The three groups were then compared. Due to the problems in the experimental design, we cannot reach definite conclusions from such an analysis, as the students in *general* and *detailed* groups received messages of other types in addition to the predominant ones. The other problem encountered was the shortness of sessions, resulting in a relatively small number of ocassions were the same constraint was used.

In this section we report on another kind of analysis, performed on the level of individual attempts. Instead of taking the whole session as a unit, we now take each attempt at solving a problem, and classify all attempts in accordance to the type of feedback obtained on it. Therefore, the set containing all instances of hint messages consists of attempts made by various students, with no regard to the version of the system they used in the study.

We then performed the same kind of analysis reported in section 6: we analysed the probability of violating a constraint each time that it was relevant, for different types of feedback obtained on the constraint on previous occasions. Some feedback types (*error flag*, *hint*, *partial solution*) apply only to a subset of the constraints, and are intended to target the first constraint failed. In such cases, any other constraints failed during this attempt are assumed a feedback level of *positive/negative*. The cut-off points are set at 33% of the original number of instances (at n=1).

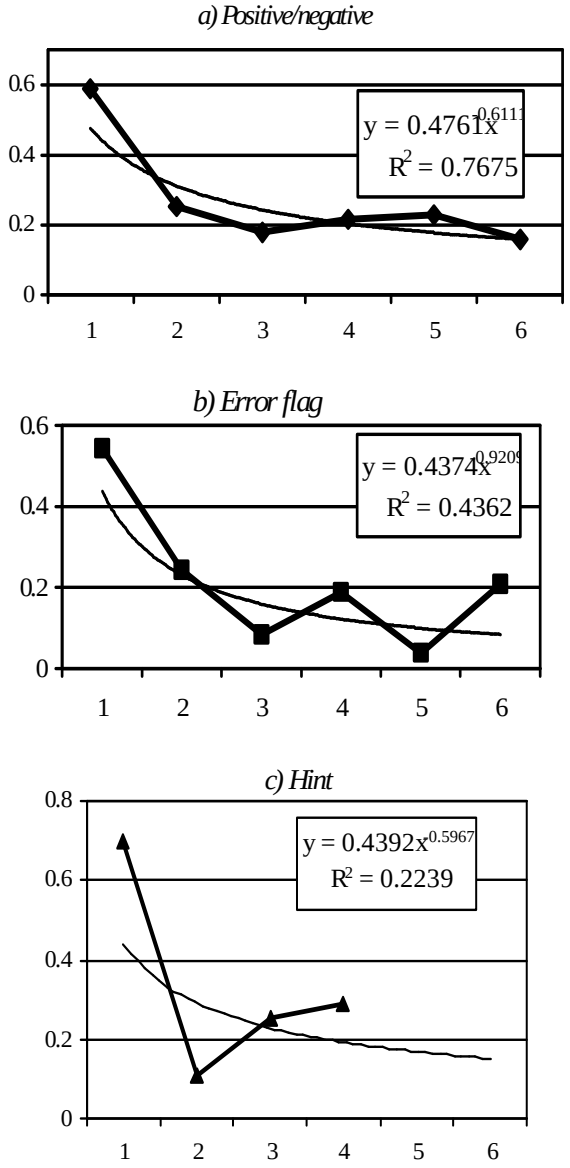

Figure 4. The probability of constraint violation after feedback

The initial learning rate is highest for *all errros* (0.44) and *error flag* (0.40) messages, closely followed by *positive/negative* (0.29) and *hint* (0.26). The learning rates for *partial* (0.15) and *full solution* (0.13) are low. This confirms our hypothesis.

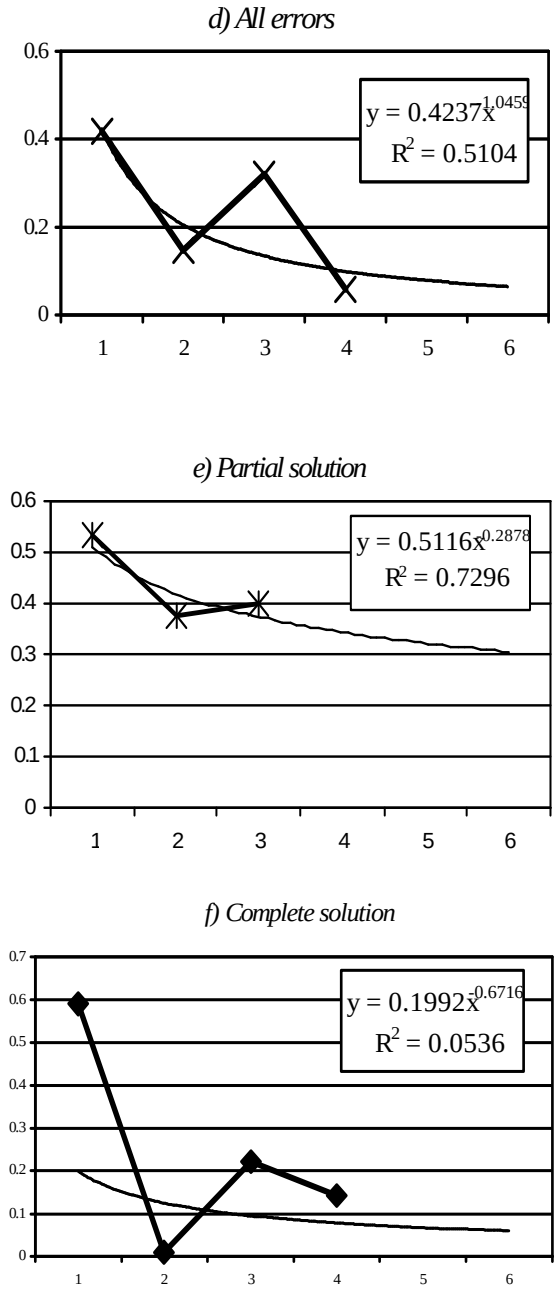

Figure 4. The probability of constraint violation after feedback

9. Conclusions

This paper presented a study of the effectiveness of various kinds of feedback available in the SQL-Tutor system. The level of details in feedback ranges from information about the correctness of the solution, information about the part of the solution that is incorrect (error flag), a hint, a list of hint messages for all errors, a partial solution to the complete solution of the problem. We looked at whether a particular level of feedback enables students to learn faster. The evidence gathered prefers feedback that presents information about the general domain principles that are violated by student's solution (e.g., *hint* and *all errors*). The same feedback levels give the shortest time per attempt and the fewest attempts per solved problem. When analysing the individual attempts, these two feedback levels also give the highest rate of learning.

Due to some problems in experimental design and the high level of uncertainty, which is inherent in all projects dealing with human subjects, our conclusions are not irrefutable. However, we believe that it is absolutely critical to perform evaluations of this kind in all ITS-related projects and that we have made a small contribution in identifying and dealing with the caveats that await researchers.

We plan to extend SQL-Tutor with an adaptive mechanism that will monitor the student during interaction, and adapt the level of feedback automatically, based on the observations of the study presented here.

Acknowledgements

The work presented here was supported by the University of Canterbury research grant U6242.

References

[1] Mitrovic, A., 1998, *A Knowledge-Based Teaching System for SQL*. Proc. ED-MEDIA'98, T. Ottmann, I. Tomek (eds.), 1027-1032.

[2] Mitrovic, A., 1998, A. *Experiences in Implementing Constraint-Based Modeling in SQL-Tutor*, Proc. ITS'98, B. Goettl, H. Halff, C. Redfield, V. Shute (eds.), 414-423.

[3] Mitrovic, A., Hausler, K., 2000, *Porting SQL-Tutor to the Web*. To be presented at the ITS'2000 workshop on *Adaptive and Intelligent Web-based Education Systems*.

[4] Mitrovic, A. Ohlsson, S. 1999. Evaluation of a Constraint-Based Tutor for a Database Language. *Int. J. on Artificial Intelligence in Education*, 10(3-4), 238-256.

[5] Ohlsson S., 1994. Constraint-based Student Modeling. In: Greer, J.E., McCalla, G.I. (eds.): *Student Modeling: the Key to Individualized Knowledge-based Instruction*, 167-189.