Evaluating ConceptGrid: An Authoring System for Natural Language Responses

Stephen B. Blessing[1], Shrenik Devasani[2], Stephen B. Gilbert[2]

[1] University of Tampa, 401 Kennedy Blvd., Tampa, FL 33606 USA

[2] Virutal Reality Applications Center, Iowa State University, 1620 Howe Hall, Ames, IA 50011 USA

sblessing@ut.edu, shrenik@iastate.edu, gilbert@iastate.edu

Abstract

Using natural language as a way for students to interact with an ITS has many advantages. However, creating the intelligence with which the tutor evaluates a student's natural language input is challenging. We describe a system, ConceptGrid, that allows non-programmers to create the instruction for checking natural language input. Three tutor authors used the system to develop answer templates for conceptual-based questions in statistics. Results indicate ConceptGrid is a viable system for non-programmers to use to allow students to use natural language to interact with a tutor.

Introduction

Allowing a student to enter natural language for their responses in an ITS moves the student experience closer to one with a human tutor. Natural language is the main way human tutors interact with students. It makes sense, then, for researchers in ITSs to investigate ways to incorporate responses to natural language in their repertory of student input. Aleven and his colleagues (Aleven et al., 1999) found that students who provided explanations for their solution steps in the Geometry Tutor later showed greater understanding of the concepts than those students who did not, a finding reminiscent of the self-explanation effect found in cognitive psychology (Chi et al., 1994).

A number of intelligent tutors allow students to enter responses in natural language. WHY2-Atlas (Jordan et al., 2001) represents a physics tutor that facilitates dialogue with students concerning conceptual-based physics problems. Medical students have used CIRCSIM to provide natural language input (Glass, 2001). AutoTutor has been deployed in a number of domains to check student's natural language input (Graesser et al., 2004). However, each of these ITSs required significant effort to create the information used to check student responses for their respective domains. This paper explores a domain-independent response authoring system usable by non-programmers.

The Authoring System Challenge

A current challenge in ITS systems is developing ways to make the authoring process easier (Murray et al., 2003). Much time and expertise are required to create these systems, which add to the expense and the ability to deploy them in a wide variety of contexts. The expertise required, particularly the cognitive science and programming knowledge, puts the ability to construct even a simple or limited ITS out of the ability of even the most dedicated instructors or subject matter experts.

Researchers have begun to develop systems that allow non-programmers, non-cognitive scientists to develop ITSs. Aleven and his colleagues (2009) have developed the Cognitive Tutor Authoring Tools (CTAT) that allow such users to develop the instructional components of an ITS (e.g., the hints and just-in-time messages) using a graphical user interface. They have met with success in this endeavor. In our own work we have worked on the xPST system to allow non-programmers to quickly develop an ITS in a more text-based format (Gilbert et al., 2009).

Allowing students to enter natural language responses represents an obstacle to these authoring systems. Systems such as WHY2-Atlas and AutoTutor have demonstrated their effectiveness, but allowing natural language input often involves sophisticated techniques involving machine learning. For example, AutoTutor makes use of latent semantic analysis (LSA; Landauer et al., 1998) to assist in checking student responses. WHY2-Atlas does have an authoring tool component that's usable by non-linguists

and reportedly non-programmers, but still has a bit of a learning curve (one week of full-time use).

Our challenge was to develop an addition to our xPST authoring system that would allow instructor-level non-programmers to create tutoring on natural language input. The learning curve for the system needed to be minimal, on the order of a couple of hours, while still maintaining a strong ability to evaluate student responses. We refer to our approach as ConceptGrid.

The ConceptGrid Approach

Our goal is to provide evaluation of short-answer style questions that may ask the student a definitional style question or a simple comparison, rather than evaluation of longer (paragraph) answers like AutoTutor or of a discursive dialog like WHY2-Atlas. Within our own work, both research and teaching, we have developed a need to have a tool to check such shorter responses. To achieve this goal, ConceptGrid evaluates student input based on the presence or absence of concepts in the answer. The tutor author defines these concepts by using a browser-based interface that allows them to define the textual pattern that the concept should follow in the student's answer. Multiple templates for a concept can be defined, to represent the multiple ways in which different students might phrase a particular concept (see Figure 1). In addition to authoring templates for expected correct concepts, tutor authors can

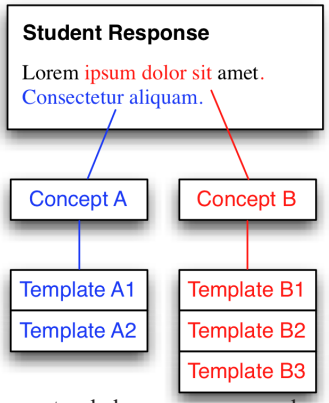

Figure 1: The natural language approach of ConceptGrid. Expected answers are divided into concepts that may occur in any order. The author creates a series of templates to identify each concept.

also develop templates for expected errors so that appropriate feedback can be provided in those circumstances as well. This template solution was chosen to balance the ease of use for the tutor author—it seems natural to think of student answers as coming in these patterns—versus the power of the system. While there are limitations in the expressiveness of these templates, and for some inputs they may be cumbersome, our belief is that they will prove powerful enough to allow for a reasonable evaluation of short (1-2 sentences) student answers. The templates are much more powerful than simple word matching, but less powerful than techniques employed by other tutors that do natural language processing. This approach was chosen with the specific intent to sacrifice some expressive power in a move towards ease-of-use on the flexibility-usability tradeoff.

A template contains one or more atomic checktypes. A checktype tests for the presence of a particular set of words. The current system contains the five checktypes seen in Table 1.

Table 1. ConceptGrid checktypes.

Checktype	Description
Almost(n, wordList)	Returns true if the least Levenshtein distance between a word in wordList and matched word is <= n
Any(m,n)	Allows a match of an arbitrary sequence of words between m and n characters within a template, when the words themselves are not important
Exact (wordList)	Returns true if a literal word match with any of the words in wordList is found
Not(wordList, direction)	Checks to make sure a word in wordList does not appear in the sequence in the given direction; allows for a check of negation
Synonym(wordList)	Uses the WordNet corpus to match synonyms, with an implied Levenshtein distance of 2

ConceptGrid allows for the easy creation of these concept templates through a web-based GUI (see Figure 2). Tutor authors can either type a sample student response or provide the dimensions of the template to define the initial template size. Once defined, rows and columns can be added or deleted from the template by clicking the appropriate "+" (to add a row or column) or "X" (to delete a row or a column). The checktype for a column (i.e., a word position in the template) is chosen from the drop-down menu. The arguments for the chosen concept, if needed, is entered to the drop-down's right, and the relevant words are listed beneath the checktype, one per row, if there are multiple words that might appear in that position.

The template that appears in Figure 2 would match such student inputs as "True experiments allow for causal statements" or "After a true experiment one can make statements of causality." It would not match, "True experiments means no causality," because the "Not"

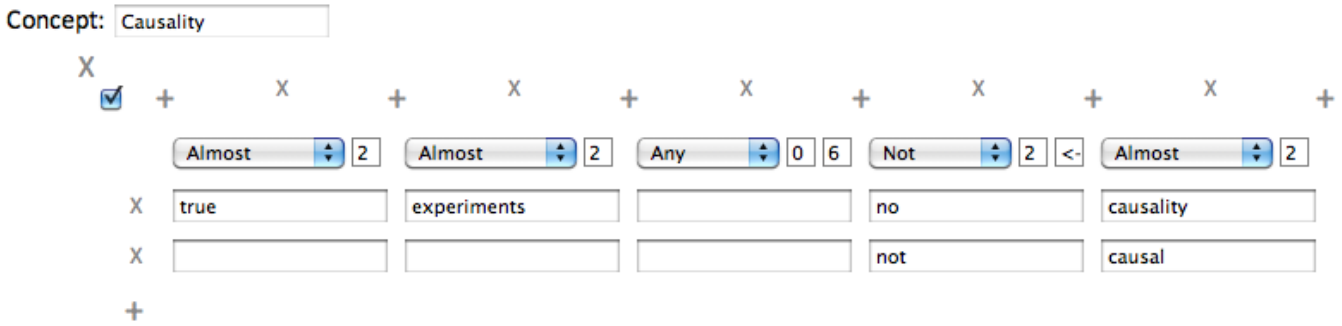

Figure 2. An example template for the concept "Causality."

checktype checks for negation. Note that these templates can match any subpart of the total student input, and do not have to match just the whole of the student input. As long as the pattern contained within the template matches any part of the student input, then that concept is considered to have matched.

Tutor authors can define as many concept templates as needed to adequately check a student response. Once all concepts and templates have been defined, the tutor author fills out the Feedback Table, a ternary truth table specifying feedback to give the students based on their input, given the presence or absence of the relevant concepts in their answer. For example, consider an answer that might be needed as part of a statistics problem, one where the student needed to say if the null hypothesis is rejected or not, and then provide a related statement of significance (e.g., "We should fail to reject the null hypothesis. There is not a significant difference in these data."). There are two concepts, one is the rejection of the null hypothesis, and one regarding significance. Figure 3 shows a Feedback Table for this example based on these two concepts.

Each row in the Feedback Table corresponds to a possible state a student answer might be in, given the presence of absence of concepts. The green checks in the table correspond to when the student input contains that concept (present), and the red X's to when the student input does not contain that concept (absent). Clicking on these icons cycles between them. There is also a yellow hyphen icon to indicate that it does not matter if the concept is present or absent for the system to consider if the student input matches that state (ignore, or "don't care"). For each student answer that uses ConceptGrid, the student answer is run through the concepts, and then the matching line or lines are found in the Feedback Table to provide feedback to the student (perhaps a statement of correctness, or maybe a statement indicating that a concept is missing or a concept is there but should not be). At this point we provide no training or instruction to tutor authors concerning what feedback to provide students, but to rely on their own pedagogical knowledge.

While using ConceptGrid, with its templates, checktypes, and ternary logic Feedback Table likely requires computational thinking (Wing, 2006) it was designed with a GUI in the spirit of visual programming tools such as Scratch (Maloney et al., 2008) and Alice (Pausch et al., 1995) so that non-programmers might be able to use it.

We used ConceptGrid to evaluate student responses to a conceptual-based statistics question (Devasani et al.,

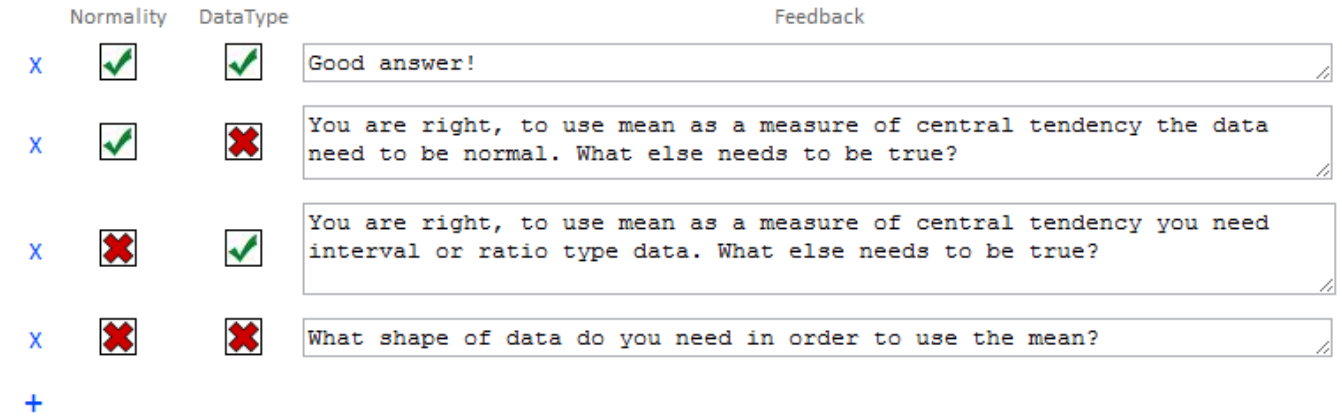

Figure 3. An example Feedback Table.

2011). The second author of this paper used ConceptGrid to create the templates, and checked the work against a corpus of 554 responses provided by 41 students on 6 different problems (some students had multiple attempts per problem). All questions had a similar form, which was to provide a conclusion statement to a multiple-step hypothesis test problem (e.g., "We fail to reject the null hypothesis. There is not a significant difference in empathy between males and females."). The overall accuracy rate was 97% (Devasani et al., 2011).

We want to further examine the effectiveness of ConceptGrid to see if a high accuracy rate was possible with non-programming authors and a variety of problems. The present study is our first attempt at that endeavor.

Methods

Participants

Two current instructors of a college-level statistics course participated as tutor authors for this study. Both have taught statistics multiple times in the past. While both have Ph.D.'s within psychology, neither had cognitive psychology or computer science as their specialty, nor use a symbolic processing language like R or command line SPSS to perform statistics. Neither received compensation for their participation. In addition, the first author of this paper provided responses to the questions, serving as an intermediate user reference point, as he had never used ConceptGrid to score actual student answers.

Materials and Procedures

We had the tutor authors create tutoring using ConceptGrid around statistics-based content. We constructed a set of 6 questions that a student halfway through their first semester in a statistics course should know. After initial construction of the question set, the final set of 6 questions was finalized in consultation with the participants to confirm that these were questions that most students should know. We wanted a set of somewhat easy questions to ensure a good pool of answers to check the tutor authors' work against. The final list of 6 questions is shown in Table 2.

In addition we had the tutor authors create a ConceptGrid for a seventh question, one used in the previous experiment, where the answer is to write a conclusion statement for a hypothesis test (like the example given above concerning rejecting the null hypothesis). As can be seen, the questions are in roughly increasing order of complexity, from one where there is a single correct answer, to ones where there are multiple parts (i.e., concepts) but limited answers, and ones with multiple parts and open-ended phrasings.

Table 2. Questions used for the study.

Questions
1. What statistic is the square root of variance?
2. What are the 3 main measures of central tendency?
3. What is at least one aspect that differentiates a true experiment from a descriptive experiment?
4. What two things must be true for *mean* to be preferred over *mode* or *median*?
5. What is the difference between nominal and ordinal style data?
6. What does parsimony mean?
7. What do you conclude based on these results? [this came after a hypothesis test had been conducted]

The college instructors learned about ConceptGrid in one 45-min face-to-face training session conducted by the first author of this paper. A four-page document was given to the instructors that contained an overview of ConceptGrid, login procedures for the ConceptGrid website, interface instructions on creating concepts and using the FeedbackTable, and the list of 7 questions to be tutored. The 4-page instruction contained one simple ConceptGrid example (one that involved two concepts), and during the short training session another example was developed. Neither example involved statistical content.

In order to test the ConceptGrids created by the tutor authors, we needed a corpus of responses to these questions. For Question 7 we had the 112 responses generated by real students for the previous study. To generate responses to the other six questions we had 87 current students in a first semester college statistics course answer all questions using an online form. These students were from 5 different classes. The responses from one of the classes were given to each participant as they were creating their initial ConceptGrids.

The participants had 2 weeks to complete their initial ConceptGrids after the training session. They were told to plan on it taking about 2 hours and were encouraged to email or ask any questions they might have as they went along. We spent about 10 min with both instructors answering questions of both a technical nature (e.g., what exactly the argument for Almost means) but also of a more conceptual nature (e.g., the best approach for making a template, rather to make it more broad or more specific).

After completion of their initial set of ConceptGrids, the participants' solutions were tested against the entire corpus of student responses. They were provided feedback concerning their accuracy rate, and given the text of half the student responses their ConceptGrids miscategorized. We gave them only half so that they would not be tempted to overfit the data. We then gave them two days to modify their ConceptGrids to see how much they might improve upon their accuracy rate, at which time their ConceptGrids

were again tested against the entire corpus. We expected to see an overall high accuracy rate, over 90%, but decreasing with the increasing level of question complexity.

Results

The website kept track of how long participants worked on each of their ConceptGrids. The two beginning authors spent an average of 1.11 hours editing their first set of ConceptGrids. The intermediate author spent 0.39 hours. This is the time spent actually editing. The website also logged the total amount of time logged in, which would account for planning time. Unfortunately, one beginner user kept logged in while doing off task behavior, so an accurate measure cannot be obtained. For the other two users, there is a 2:1 ratio of total time to editing time. Precise timing data is not available for the second iteration due to a technical issue. Anecdotal evidence indicates users spent 45 min on average for this phase.

Table 3. Average accuracy results (percentages).

Question	First Iteration	Second Iteration
Q1 Overall	97.70	99.62
Q2 Overall	100.0	100.0
Concept 1 (“mean”)	100.0	100.0
Concept 2 (“median”)	100.0	100.0
Concept 3 (“mode”)	100.0	100.0
Q3 Overall	70.50	77.78
Concept 1 (“manipulation”)	71.26	74.33
Concept 2 (“control”)	88.12	85.44
Concept 3 (“causality”)	98.47	97.70
Q4 Overall	96.17	96.93
Concept 1 (“normality”)	78.16	89.66
Concept 2 (“data type”)	96.17	96.17
Q5 Overall	65.90	67.82
Concept 1 (“nominal”)	62.07	70.50
Concept 2 (“ordinal”)	65.13	73.18
Q6 Overall	99.23	98.47
Q7 Overall	67.86	71.13
Concept 1 (“rejection”)	56.85	96.13
Concept 2 (“significance”)	63.10	63.99
Overall	82.98	87.31

A research assistant scored all the student responses for correctness. The first author of this paper also scored all the responses. The 7 total discrepancies (1.3% of the corpus) were resolved by verbal agreement. After each participant indicated he or she was done working on the initial ConceptGrid for each question, the ConceptsGrids were checked against the student responses. In such a way we obtained an accuracy score for each participant, indicating the percentage of time his or her ConceptGrids correctly rejected and correctly accepted the student responses. Table 3 displays the mean accuracy. The Concepts for the overall questions were derived from discussions with the participants concerning what a correct answer for these questions should contain.

Examining individual accuracy results, the two beginners scored an overall average of 77.29% and 86.14% on the first iteration, and then increased to 84.07% and 86.61% on the second iteration, respectively. The intermediate user went from 85.52% to 91.24%. Questions 5 and 7 proved most difficult, due to the wide variability of student responses. Considering the improvement across all of the patterns that the participants authored, where the average went from 82.98% to 87.31%, a significant difference was observed ($t(56) = 2.76$, $p < .05$, $d = 0.37$).

Investigating the actual ConceptGrids themselves, all users tended towards short templates. This indicates their strategy across questions was to zero in on a particular phrase that indicates student understanding and create a concept template for that phrase. This makes logical sense for some questions where the concept is a single word (e.g., all three concepts for Q2), but the participants adopted a minimalist approach, to largely successful effect, for the other questions as well. The average number of atomic checktypes used per concept in the first iteration was 2.90, and that decreased slightly to 2.85 in the second iteration, though the two beginning users increased their average number of checktypes, while the intermediate user decreased his. All the different atomic checktypes were used at least once, but the Almost and Any checktypes were used the most often. The Any checktype is useful when two crucial words appear in a concept, but they may be separated by an unknown number of words.

In informal discussions after the experience, both beginning authors indicated that they felt ConceptGrid was easy to use, with a short learning curve. The first two questions were easy for them, as the data suggest, with the others being more challenging. However, once they hit upon the proper level of specificity for creating their concept templates, the task became much easier. Both saw value in the tool, and saw how it could be applied more generally. Both also admitted frustration with doing the second iteration task, for three different reasons. Due to the technical issue and the timing within the semester, the amount of time available to perform the edits was short, just two days, making it challenging to perform the task. Perhaps more importantly though, the feedback given the participants was hard to decipher. The feedback took the form of a spreadsheet that contained half of their incorrect responses. A false positive was indicated by a “1” and a false negative was indicated by a “-1.” It was a lot of data, and there was no way to get immediate feedback based on an existing student corpus as to how good their edits were

within the website (they had to wait until the second author of this paper could generate the statistics).

Discussion

Overall, we were pleased with the result. The final accuracy was not above our hoped-for 90% for the beginning authors, but the issues we had with the second iteration did not help. With the results here and our observations from Devasani et al. (2011), experience with the concepts and interface results in improved templates. Bettering the authoring environment to allow for more immediate and easier to interpret feedback as the tutor author made edits would assist greatly, for both beginning and more advanced authors. We did observe a slight decrease in accuracy with increasing complexity, though some of the more complex questions enjoyed a high accuracy. Also, some concepts enjoyed a nice improvement between iterations. In considering these differences, and in discussions with the participants, we considered these observations and what we could do to ensure high success rates in the future.

One issue that arose as the participants went from their first iteration to their second was that answers that previously matched correctly either became false positives or false negatives in the second iteration. Part of this was beyond the participant's control. For example, in defining a concept for Q6, the one about parsimony, one participant used the Synonym atomic checktype with "simplicity" in the wordlist. A WordNet synonym for "simplicity" is "ease," and with a Levenshtein distance of 2, several false acceptances occurred. One solution here is to make the Levenshtein distance a function of the length of the word. This suggestion may also be appropriate for the Almost atomic checktype, getting rid of one of its numerical arguments. The tutor author would then not need to worry with it. One of the participants in her initial iteration had the argument set high for some of her concepts.

Based on our observations and conversations, we are now considering two design improvements in addition to further usability testing. First is an automatic way to score the correctness of ConceptGrids. One could imagine the online tool being able to accept a scored set of student responses. Once loaded, a ConceptGrid could then be scored against it by a simple click of a button, allowing for an immediate check to see how a change affected the accuracy. Second, once the ConceptGrid has been scored against the student input, a better way to present the results (the percentages and matches of correct and incorrect acceptances and rejections).

We plan on using ConceptGrid with a wider set of users and domains. We feel it offers a good, domain general, solution to checking short natural language answers in an ITS environment.

References

Aleven, V., Koedinger, K., Cross, K. (1999). Tutoring answer explanations fosters learning with understanding. Proceedings of Artificial Intelligence in Education, AIED 1999, 199-206.

Aleven, V., McLaren, B.M., Sewall, J., Koedinger, K.R. (2009). A new paradigm for intelligent tutoring systems: Example-tracing tutors. *International Journal of Artificial Intelligence in Education*, *19*, 105-154.

Chi, M.T.H, de Leeuw, N., Chiu, M.H., LaVancher, C.: (1994). Eliciting self-explanations improves understanding. *Cognitive Scienc,. 18*, 439—477.

Devasani, S., Aist, G., Blessing, S. B., & Gilbert, S. (2011). Lattice-based approach to building templates for natural language understanding in intelligent tutoring systems. In G. Biswas, S. Bull & J. Kay (Eds.)*, Proceedings of the Fifteenth International Artificial Intelligence in Education Conference* (pp. 47-54), Auckland, NZ. Berlin, Germany: Springer.

Gilbert, S., Blessing, S. B., & Kodavali, S. (2009). The Extensible Problem-specific tutor (xpst): Evaluation of an api for tutoring on existing interfaces. In V. Dimitrova et al. (Eds.), *Proceedings of the 14th International Conference on Artificial Intelligence in Education* (pp. 707-709), Brighton, UK. Amsterdam, Netherlands: IOS Press.

Glass, M. (2001). Processing language input in the CIRCSIM-tutor intelligent tutoring system. In: Moore, J.D. et al. (eds.), *Artificial Intelligence in Education*. pp. 210—221.

Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H., Ventura, M., Olney, A., & Louwerse, M.M. (2004). AutoTutor: A tutor with dialogue in natural language. *Behavioral Research Methods, Instruments, and Computers*, *36*, 180-193.

Jordan, P., Rosé, C., & VanLehn, K. (2001) Tools for authoring tutorial dialogue knowledge. In J. D. Moore, C. L. Redfield, & W. L. Johnson (Eds.). *AI in Education : AI-ED in the Wired and Wireless Future* (pp. 222-233). Amsterdam: IOS Press.

Landauer, T.K., Foltz, P.W., Laham, D. (1998). Introduction to latent semantic analysis. *Discourse Processes*. *25*, 259-284.

Maloney, J.H., Peppler, K., Kafai, Y., Resnick, M. & Rusk, N. (2008). Programming by choice: Urban youth learning programming with scratch. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*.

Murray, T., Blessing, S., & Ainsworth, S. (2003). *Authoring Tools for Advanced Technology Educational Software*. Kluwer Academic Publishers, Dordrecht, The Netherlands.

Pausch, R., Burnette, T., Capeheart, A.C., Conway, M., Cosgrove, D., DeLine, R., Durbin, J., Gossweiler, R., Koga, S., & White, J. (1995). Alice: Rapid prototyping system for virtual reality. *IEEE Computer Graphics and Applications*, 15, 8-11.

Wing, J.M. (2006). Computational thinking. *Communications of the ACM*, *49*, 33-35.