Intelligent Tutoring Systems for Ill-Defined Domains: Assessment and Feedback in Ill-Defined Domains.

Proceedings of a workshop held during ITS-2008.

The 9th international Conference on Intelligent Tutoring Systems,

Montreal Canada,

June 23rd 2008.

Vincent Aleven.
Kevin Ashley.
Collin Lynch.
Niels Pinkwart.

Workshop co-chairs

Preface.

Assessment is a central pillar of tutoring. Both human tutors and intelligent systems use assessment of students'problem-solving habits and performance (both immediate and long-term)in order to provide feedback and guidance. The development of good assessment and feedback techniques is essential to the development of reliable tutoring systems for all domains.

Intelligent tutoring systems have shown great success in providing assessment and useful feedback. These gains, however have come chiefly in well-defined domains. In well-defined domains tutors may rely on a strong domain theory to structure the relevant domain knowledge and to validate student actions. This has permitted the development of strong, domain-general methodologies such as model-tracing systems and constraint-based tutors. In ill-defined domains no such theory exists to structure the relevant knowledge and validation, where it occurs, often depends on reasoned argument rather than formal proofs. As a result, assessment and feedback structures that rely exclusively on a strong domain theory have limited application in truly ill-defined domains.

The papers in this volume focus on the central question of how best to provide for assessment and feedback in ill-defined domains. How does one distinguish between successful and unsuccessful students, and how does one explain concepts to, motivate, and guide student behavior. These papers present a range of promising approaches, each one suited to the authors' unique problems and vantage points. Our goal in collecting these works is to motivate a discussion of general approaches to assessment and feedback. What techniques have shown success in ill-defined domains? What techniques have not? And to what extent may one generalize?

This volume is a continuation of our two prior workshops on ill-defined domains. The first was held in Jhongli, Taiwan in 2006. Papers there covered language learning, the adaptation of traditional tutoring techniques to ill-defined domains and work on group support, guidance, and collaboration. At the 2007 workshop in Los Angeles on AIED applications in ill-defined domains, authors presented work on tutoring in medical diagnosis and argumentation. We believe that these papers represent the increasing maturity of research in ill-defined domains and help to guide further exploration in this increasingly important subfield of ITS.

Vincent Aleven, Kevin D. Ashley, Collin Lynch & Niels Pinkwart.
June 2008.

Organizing Committee.

Table of Contents.

1. Two Approaches for Providing Adaptive Support for Discussion in an Ill-Defined Domain
Erin Walker, Amy Ogan, Vincent Aleven, Chris Jones

13. Interactive Narrative and Intelligent Tutoring for Ethics Domain
Rania Hodhod and Daniel Kudenko

22. Selection Strategy to Improve Cloze Question Quality
Juan Pino, Michael Heilman, and Maxine Eskenaz

35. Generating and Evaluating Object-Oriented Designs for Instructors and Novice Students
Sally Moritz and Glenn Blank

46. A Sequential Pattern Mining Algorithm for Extracting Partial Problem Spaces from Logged User Interactions
Philippe Fournier-Vige1, Roger Nkambou and Engelbert Mephu Nguifo

56. What do argument diagrams tell us about students' aptitude or experience? A statistical analysis in an ill-defined domain
Collin Lynch, Niels Pinkwart, Kevin Ashley and Vincent Aleven

Two Approaches for Providing Adaptive Support for Discussion in an Ill-Defined Domain

Erin Walker, Amy Ogan, Vincent Aleven, Chris Jones

Carnegie Mellon University, Pittsburgh, PA, USA
aeo@andrew.cmu.edu, erinwalk@andrew.cmu.edu, aleven@cs.cmu.edu,
cjones@andrew.cmu.edu

Abstract. Intelligent tutoring systems have increased student learning in many domains with well-structured tasks such as math and science. As these successes may be applicable to ill-defined domains, is important to investigate how to apply tutoring paradigms for tasks that are ill-defined. We focus on the domain of intercultural competence, where asynchronous online discussion is a typical instructional task. We explore two methods of assessing student contributions and delivering feedback: an adaptive support system that provides individual feedback, and a peer moderator from the class supported by adaptive assistance. We report on our initial experience in two pilots of these systems, where we found promise that students may benefit from each approach.

1 Introduction

Intelligent tutoring systems (ITS) have been shown to be effective at increasing student learning in domains with well-structured tasks, where criteria for correctness are relatively easy to quantify. As researchers move towards domains with ill-defined tasks, however, it is important to identify how to apply tutoring paradigms in these new domains. Whereas most ITS (see [1] for review) provide correctness feedback on student solutions, similar definitive feedback is difficult to give in ill-defined tasks. Also, classroom methods for teaching these domains, such as discussion with peers or Socratic dialogue, are not congruent with a typical tutoring system interface [2]. Intercultural competence, the ability to consider and understand events in a foreign culture in terms of cultural differences, is one such ill-defined domain. In displaying intercultural competence, students are asked to demonstrate that they have mastered various *savoirs*, such as the ability to explain cultural behavior (see [3] for a detailed discussion of the domain]. However, solutions in this domain may depend on an interpretation of events or language. For example, a student asked to evaluate immigration issues in France could draw on historical, journalistic or experiential knowledge from various perspectives to formulate a valid response. While there may be rules that describe cultural norms or behavior, they tend to have many exceptions depending on the circumstances or on individual traits of the actors in a situation. Therefore, in learning intercultural competence there is often an emphasis on the critical analysis of cultural events.

Intercultural competence is often taught using group discussion of multimedia materials such as newspapers or film. For example, students might read or watch *Les Misérables* and discuss the meaning of justice and authority in French society. The process of giving students input (such as culturally-relevant video), having them notice and reflect on relevant features, and produce written or verbal output under the guidance of an instructor is a primary technique for getting students to acquire skills like taking the perspective of another culture [4]. While classroom discussion plays a prominent role in this instruction, in general much in-class discussion has moved to asynchronous online discussion forums, where students post on course topics and read contributions by other students [5]. In these environments, students need support in conducting a good discussion [6, 7].

Intelligent tutoring support may be an appropriate way of helping students have productive online discussions. In asynchronous online discussion, adaptive feedback has played a role, but the feedback is generally delivered by an instructor [8]. However, instructors often have too many responsibilities to actively moderate every forum [9].It is conceivable that an adaptive collaborative learning system could help to provide additional support to the forum, giving students guidance and feedback as they participate in asynchronous discussion. Adaptive collaborative learning systems (ACLS) have been shown to be effective at increasing learning compared to non-adaptive support for collaboration and individual work [10]. However, to the best of our knowledge such systems have focused on dyadic rather than classroom collaboration [see 11 for review]. Further, implementation of these systems is generally still at an early stage.

In order to investigate how to apply techniques taken from intelligent tutoring system research to ill-defined domains, we explore two approaches to supporting cultural discussion. First, we develop an ACLS for asynchronous discussion in intercultural competence (*an individual support system*). This ACLS has a simple model of a good discussion post, compares a given student post to the model, and provides private feedback to individual students based on the model. This form of adaptive support draws on traditional intelligent tutoring paradigms. Second, we use a *moderator support system* to deliver adaptive support to a student from the class who moderates the discussion forum (a *peer moderator*), based upon the same model used by the individual support system. While feedback given by an intelligent system might be more consistent than feedback given by a peer moderator, students might feel more accountable to address a human moderator's feedback. Further, a moderator might be able to detect post quality better than an intelligent system. Although peer moderator interventions have been implemented in discussion forums [12], their effectiveness has not been empirically demonstrated, and there is some question about how effectively peers can deliver feedback. Therefore, providing intelligent support to the peer moderator might be an effective approach. This concept draws heavily on the IHelp model of "helping the peer helper" [13], but our system goes beyond IHelp's brokerage service in that it actively helps the peer moderator do his or her task. In this paper, we explore two methods of assessing student contributions and delivering feedback – an individual adaptive support system, and a peer moderator from the class aided by adaptive support. We report on our initial experience in deploying these systems.

2 Adaptive Feedback in a Cultural Discussion

2.1 Learning Environment

The discussion that we are trying to support takes place as part of an e-learning environment created by Ogan et al. [7] for helping students to acquire intercultural competence skills. Students are instructed to watch a video clip from a French film that includes cultural content related to a given theme (e.g., a video clip from the movie *Monsieur Ibrahim* that touches on immigration issues). They follow a sequence in which the video is paused, they predict the next event, and then they reflect on the result. Next, students are given a prompt for forum discussion, and asked to contribute at least three times to the board. For *Monsieur Ibrahim*, the prompt was:

> Post with questions, analysis, or other thoughts about the immigration issues in France you've seen. Think about the racial and ethnic stereotypes in France that you have seen depicted in this film to get started.

In this exercise, students are given authentic material and scaffolding to help them process domain material deeply. They then produce output in the form of discussion posts, helping them to further analyze the material.

2.2 Building an Expert Model

In intelligent support systems, a model is generally used as a basis for feedback. This constraint is particularly challenging in ill-defined domains as, by definition, they are more difficult to model. To develop a model for our system, we conducted a theoretical analysis based on existing literature, examining quantitative measures of discussion quality (e.g., [6]), qualitative coding schemes for discussion (e.g., [14]), and specific aspects of good cultural discussion (e.g., [15]). We also conducted an empirical analysis based on data from a previous study, looking at both positive and negative aspects of student discussion.

We distilled our findings into five initial dimensions that we hoped to target for support. To help explain the dimensions, we address them using the following post as an example (but see [16] for more detail):

> I think that the question is a mixture of the culture and the stereotypes. During my experiences in France and my French studies, I found that the differences between the cultures in France are the result of stereotypes. There is a big difference between the French European/Catholic and the French of Middle East/Africa/Muslim. Really, the negative stereotypes are the result of that cultural schism and that is the origin of the cultural inequality in France.

Two of the dimensions were task dimensions, relating to what makes a good discussion post in general.

1. *OT: Is the post on-topic?*
   The above post refers to concepts from the discussion prompt, thus is on-topic.
2. *GA: Does it contain a good argument (conclusion and supporting evidence?)*
   The above post does not contain sufficient evidence to support its conclusions.

We also included two dimensions relating directly to intercultural competence. These dimensions represent a necessary simplification of the domain for prototype systems:

3. *MP: Does the post show awareness of multiple points of view or an alternate point of view?*
   Although the above post does not elaborate on multiple points of view, it does acknowledge that they exist
4. *NF: Does the post introduce relevant new facts not directly from the video?*
   The above post talks about the poster's experiences in France.

Finally, we included a dimension to check whether the discussion includes correct and relevant cultural knowledge from the video:

5. *CF: Does the post reference at least one correct cultural element specific to the themes of the video, and no incorrect cultural elements?*
   The above post does not include any facts seen in the video clip.

In order to produce concise and meaningful feedback to students, we organized these five dimensions in a theoretical hierarchy of importance (see Figure 1). The large diamonds in the diagram represent the primary decision points for assessing posts, and are organized from most to least important, left to right. The "on topic" dimension is at the highest level of priority, because if the content of the post was not relevant to the cultural issue, it makes no sense to examine whether, for example, the post contains a good argument. Novel facts are the lowest-priority dimension, because they add to a post but are not necessary for a good post. The small diamonds in the diagram are secondary decision points that augment the assessment; e.g., once we ascertain that the post does not contain multiple perspectives, we then check whether it has correct facts. Feedback (represented by squares) focuses mainly on the primary lacking dimension, but incorporates a second lacking dimension as needed.

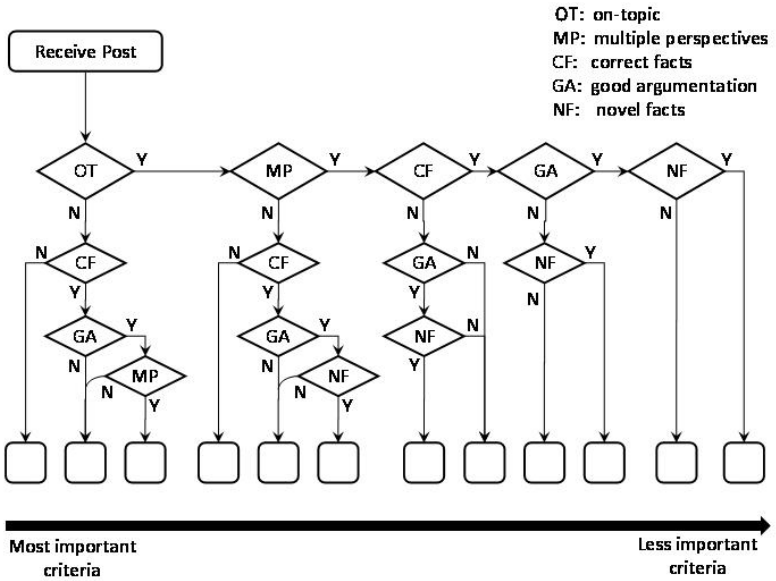


Figure 1: Model used to assess a discussion post and provide feedback

2.3 Assessing post quality

In order to develop an adaptive support system to deliver feedback based on this model, we needed to assess where student posts required assistance. We explored two methods of assessing post quality, both based on the model described above. First, we attempted to design an individual support system to detect student performance on the model dimensions. Because building an entire natural language processing system for a new domain is very difficult and perhaps unnecessary, the goal for our prototype was to use simple algorithms to rate the posts on the five dimensions. With only a keyword analysis based on words which appeared in the corpus and were highly correlated with human coding of the dimensions, we achieved reasonable accuracy (kappa values between .5 and .9) on three of the five dimensions: on topic, good argument, and multiple perspectives. We also took a first step towards assessing the correct facts dimension of the model; the keywords were used to characterize the particular topic the students were discussing, based on several pre-defined topics for the assignment.

Because automated assessment is not as complete as human assessment can be, we also explored the use of student moderators from the class to assess the discussion posts. After reading the board and choosing a post to address, moderators were asked to answer 4 yes/no questions, mapping to the first four dimensions of the model. Moderators were also asked to select the topic of the facts that appeared in the post, and indicate whether those facts were correct (see Figure 2). The moderator then submitted the ratings to the moderator support system.

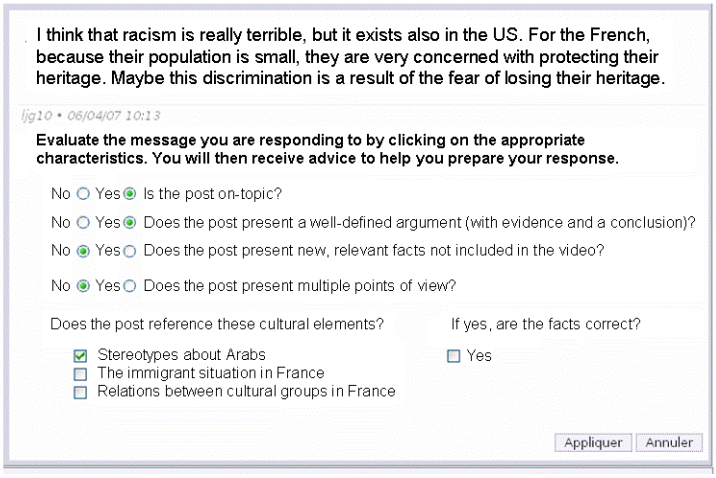


Figure 2. Posting interface for the moderator (translated into English)

2.4 Providing Feedback

We then used the assessment of the post to provide feedback to students, again exploring the individual support and moderator support approaches. In the individual

support scenario, after students created a post on the discussion board, they submitted it to the system. The support system ran the analysis module to determine where the post fell along each of the three dimensions. It then followed the decision tree in Figure 1 to determine which type of feedback to provide. The system randomly selected one feedback message from three with alternate wordings that were associated with the location of the post in the tree. To develop these feedback strings, we interviewed an expert to determine what experts focused on in writing feedback to posts. The system then added several correct facts related to the categorization of the post topics from a collection of facts about the issues presented in the video. If the post was off-topic, the system simply suggested several facts from a variety of different relevant topics. This feedback was presented to the student (see Figure 2), who was required to make at least one modification to the post before being allowed to submit the final version to the discussion board. Once the post was submitted, it appeared on the discussion board and was available for the rest of the group to read.

In the moderator support scenario, the system used the moderator ratings to navigate through the model hierarchy and identify which feedback type is appropriate. The moderator was then presented with a feedback template to help in constructing appropriate feedback. The template was again developed by consulting an expert and consisted of both praise for the positive elements of the post and constructive criticism for the negative elements of the post. The system also suggested three facts about the culture that the moderator might want to incorporate into his or her post, using the same algorithm as the individual adaptive support. Moderators were then encouraged to fill the template in with specific details from the post they were replying to, and use the additional facts that were provided. They then submitted their feedback as a post to the discussion forum.

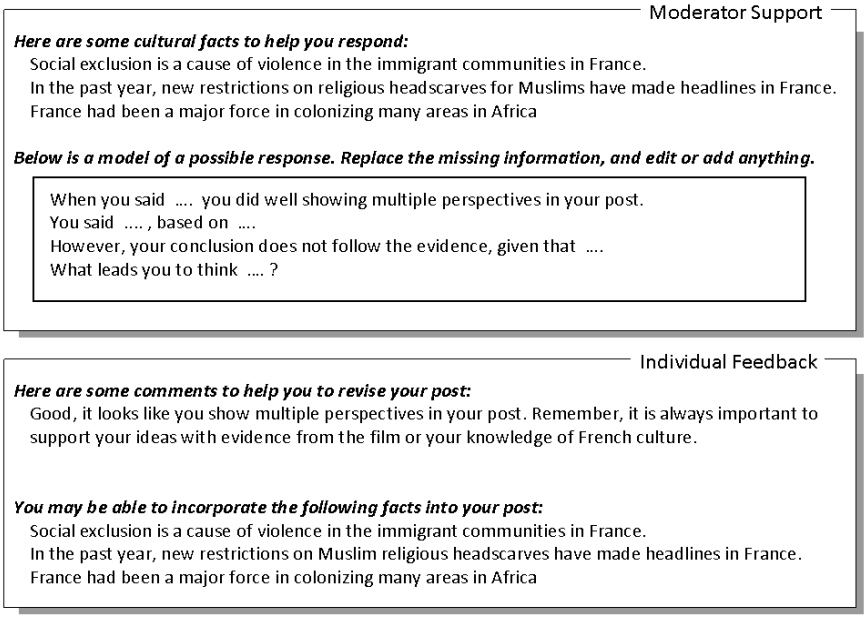


Figure 3. Feedback examples for the moderator and individual support (translated)

3 Preliminary Evaluation

3.1 Individual Support System

We evaluated the individual support system in a small pilot study in an upper level French language class. Participants were 8 students who were relatively fluent in French. The study took place over the course of a week, during which students watched the film clip and posted to the discussion board. Students were asked to post in French at least three times over the course of the week (specifically on a Monday, Wednesday, and Friday), and to read the other posts before posting. The 8 students posted a total of 25 times, which was slightly more than the 3 posts per student which was required in the assignment. Posts on average were around 48 words long. Two coders then rated all posts on the five dimensions of our model based on a coding manual and resolved differences through consensus.

Our first research question was whether the feedback that students received from the adaptive system was appropriate. It is clear that students needed feedback, as posts submitted to the system were at a fairly low level. Almost half the posts were missing all dimensions of good discussion other than being on-topic, and no post contained all five dimensions. When students submitted their posts, the feedback provided by the individual support system was fully relevant in 64% of cases, in that for those posts the model generated an assessment that matched human ratings. For the other cases, the feedback may not have been as accurate. However, feedback was worded tentatively, so when the assessment was faulty it still appeared to offer a valid suggestion.

The next step was to examine whether student posts tended to improve as a result of receiving feedback (note that numbers here are not significant due to low sample size in this exploratory pilot, but the trends reported warrant further study). Almost a third of student posts improved their score on at least one dimension as a result of their revisions, and several addressed the feedback directly in their posts. The most improvement tended to be on the correct facts and good argumentation dimensions, possibly because of the additional facts students received (see Table 1). Adding facts can provide supporting evidence for a conclusion. For example, the following post received appropriate feedback and improved on those two dimensions:

> *Version 1*: There is tension because of the difference between cultures, but I think that M Ibrahim wants to reduce the tension by being familiar with Momo
>
> *Version 2*: I think that there is tension between cultures because of the difference in cultures, but M Ibrahim wants to reduce the tension by being familiar with Momo. I don't think that the tension is because of hate, and this is explained because Momo still does his shopping at M Ibrahim's

Another third of posts made non-trivial changes upon receiving feedback but did not improve on the ratings, such as this post:

> *Version 1:* What religion is Momo? Because he is named Moses

Version 2: What religion is Momo? Because he is named Moses. Also, he has excluded M Ibrahim

The other 10 posts simply made trivial changes so that their new posts would be accepted by the system. We investigated why certain posts may have been revised and why others were left as they were. There appeared to be two factors. First, students were more likely to revise their earlier posts than their later posts. Second, unsurprisingly, students were more likely to revise posts that received relevant feedback.

Table 1. Percent of posts in individual support study containing each dimension (human rated)

	On topic	Multiple perspectives	Correct facts	Novel facts	Good argument
Posts that made no revisions	1	.0	.30	.10	.30
Posts that made revisions – before feedback	1	.13	.27	.27	.20
Posts that made revisions - after feedback	**1**	**.27**	**.53**	**.33**	**.53**

3.2 Moderator Support System

We evaluated the moderator support system in an upper-level French classroom using the same materials as the adaptive support study. Participants were 5 students: 3 posters, and 2 students who volunteered to be moderators. The instructions for this study were similar to those of the previous study, but the moderators were given separate instructions to read posts and reply on Tuesday and Thursday, so that there would be time for students to read the moderators' posts before writing to the board the second and third time. The moderator received adaptive support in moderating. The board received 12 posts total over the week (4 from moderators and 8 from the other students), which was slightly less than the assignment required. Moderator posts were on average 62 words, longer than the regular posts which averaged 50 words. Two coders then rated all posts on the five dimensions of our model and resolved differences through consensus.

Our first research question related to the nature of the feedback given by the moderators. On the surface, moderators did not appear to utilize the support. One of the moderators did use the rating system and arrived at ratings similar to our ratings of the posts, and the other didn't follow the rating system at all, choosing instead to give the posts she replied to a perfect score. Both moderators deleted the feedback tem-

plate we provided and wrote their own feedback. While the support was underutilized, it did appear to have a positive effect on the moderator posts. The 4 moderator posts in this condition communicated elements of the model, like multiple perspectives, prior personal experience, and good argumentation. For example, one moderator chose to respond to the following post:

> "Do you think that, at the end, he made a little joke about the word Arab? For me, it's like he is commenting on the big problem of stereotypes of race and religion in France; that all the Arabs have stores and work all the time"

This post was rated as being on-topic and having multiple perspectives, due to the recognition of the stereotypes existing in France. The moderator replied:

> "It is not a joke, it seems to me, but a stereotype that comes from the manner in which the French think. Here, in the US, it's the people who call storekeepers Koreans. Do you think for the same reason? Also, for the Catholics, Sunday is a sacred day, for God, but the Arabs are a different religion. Is Sunday also exceptional? If yes, why is it that they work?"

In the first line, the moderator acknowledged a conclusion the poster made and clarified the point. The moderator then compared a US perspective to the French perspective, using prior knowledge about the US perspective. The moderator also brought up an important point for understanding the movie, which is that Sunday is a sacred day for Catholics but possibly not for Muslims. Finally, the moderator asked thought-provoking questions to keep the discussion going. Although the moderator's understanding of the situation was not perfect, the end result was an improvement in the overall quality of the discussion.

Next, we attempted to look at the quality of the regular student posts before and after they received feedback given by the moderators, a difficult task due to the low number of students who posted. We divided posts into two categories based on student exposure to moderator feedback. We placed posts that were made before any moderator feedback appeared in the thread in one category, and posts that were made after feedback appeared in the thread in a second category. Posts on the moderator board on average across students rated fairly low at the outset, but had a trend towards improvement after feedback from the moderator. While this result is initially promising, more investigation is necessary due in part to the low number of participants.

6 Discussion

In this exploratory work, we compared the relative advantages and disadvantages of two approaches for supporting discussion in an ill-defined domain. The first question in an assessment of whether these interventions could be successful is whether the feedback provider (human or computer) understands what is occurring in the discussion. In the individual support condition, the ratings generated by the system matched human raters in the majority of cases, even though the technology used to process the post was relatively simple. However, the system only made ratings on a subset of the dimensions. In the supported moderator condition, only one of the two moderators used the ratings appropriately, but those ratings matched our codes for the posts. The

peer moderators had the ability to accurately code along all the model dimensions, but there was less control over whether they would perform their task thoughtfully. One possible issue in the peer moderator condition was that the questions in the ratings system were in French, which may have increased the perceived amount of effort to complete the task.

Secondly, we assessed the quality of feedback provided based on the moderator or individual system ratings. A majority of the individual system ratings matched human ratings of the posts. When posts received less accurate ratings, we observed that the feedback was not out of place, perhaps because it was designed to apply in multiple situations. Compared to the individual system, the feedback provided by the peer moderators was more varied. Peer moderators asked more questions about specific issues in the video rather than offering suggestions about general improvement along the dimensions and presenting additional correct facts as did the adaptive support. In high quality moderator posts, the moderator feedback was richer than the adaptive support, and was able to identify and address misconceptions. In general, the moderator feedback at its best was better than the ITS feedback, but was much more inconsistent in quality.

Finally, we would like to know whether students actually used the feedback to improve the discussion. In the individual support condition, students did appear to use the feedback they received to modify their posts. The majority of these posts improved, most often when the feedback was relevant to their initial post. Students improved the most on using correct facts. This is a positive outcome because one of the major concerns with an unmoderated discussion board is that students may simply discourse at length using faulty information and never arrive at a correct understanding. While we decided to focus on two lacking dimensions while providing feedback, to reduce the amount of cognitive load while students revised their post, an area for future exploration would be to examine the effects of more or less feedback on student posts. On the peer moderator discussion boards, it was more difficult to draw conclusions about improvement because there were not as many posts as in the individual support, and in particular there were not as many responses to moderator posts as we had hoped. It did not look like the moderators had as visible of an impact here.

One reason why the adaptive support may have lead to the immediate improvement is the difference in the manner of presentation of the feedback. When the feedback is individual and private, as in the adaptive support condition, students may pay more attention because it is targeted directly at their writing for which they feel some ownership. The peer moderators, although they were asked to address issues in the posts that they felt needed the most assistance, tended to make their feedback general and applicable to the whole thread rather than a single post. While this may have positive effects on the discussion in general, it is likely that it would take a longer discussion for the effects to be visible. In the short term, it may have lessened the accountability for individual students to incorporate the feedback into their own posts.

We have applied this technique to the ill-defined domain of intercultural competence, and seen some initial promise in both providing private adaptive feedback to posters in a discussion forum and adaptive feedback to a peer moderator of the discussion. There are two key elements of our work that are particularly relevant to ill-defined domains. First, an individual support system that imperfectly models the domain might still form the basis for feedback that leads to student improvement. Sec-

ond, using peers to perform tasks that an adaptive system is unable to do completely might augment the system's effectiveness, and helping the peers using a moderator support system might eliminate inconsistency in their performance. Alternatively, the approach may provide a stepping stone for implementing full intelligent support for a discussion forum, as a corpus of moderator ratings might serve as training data for an automated system that can independently rate student posts. If further evaluation demonstrates that these approaches are successful, this work would generalize to other ill-defined domains, particularly where asynchronous discussion forums are critical elements of classroom instruction.

Acknowledgements. This research is supported by the Graduate Training Grant awarded to Carnegie Mellon University by the Department of Education (#R305B040063). Thanks to the anonymous reviewers for their helpful suggestions.

References

1. VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16(3), 227–265.
2. Lynch, C., Ashley, K., Aleven, V., & Pinkwart, N. (2006). Defining ill-defined domains; a literature survey. In V. Aleven, K. Ashley, C. Lynch, & N. Pinkwart (Eds.), *Proceedings of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th International Conference on Intelligent Tutoring Systems* (pp. 1-10). Jhongli (Taiwan), National Central University.
3. Byram, M. (1997). Teaching and Assessing Intercultural Communicative Competence. Clevedon: Multilingual Matters.
4. Liddicoat, A. J., & Crozet, C. (2001). Acquiring French interactional norms through instruction. In K. R. Rose & G. Kasper (Eds.), Pragmatic development in instructional contexts. Cambridge: Cambridge University Press.
5. Hara, N., Bonk, C., & Angeli, C., (2000). Content analyses of on-line discussion in an applied educational psychology course. Instructional Science. 28(2), 115-152.
6. Guzdial, M., & Turns, J. (2000). Effective discussion through a computer-mediated anchored forum. Journal of the Learning Sciences, 9, 437–469.
7. Ogan, A., Aleven, V., and Jones, C. 2008. Pause, predict, and ponder: use of narrative videos to improve cultural discussion and learning. In Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems(Florence, Italy, April 05 - 10, 2008). CHI '08. ACM, New York, NY, 155-162.
8. Mazzolini, M. & Maddison, S. (2003). Sage, guide or ghost? The effect of instructor intervention on student participation in online discussion forums. Computers and Education, 40, 237-253.
9. Beaudin, B. P. (1999). Keeping online asynchronous discussions on topic. Journal of Asynchronous Learning Network, 3(2), 41–53 [On-line]. Available at: www.aln.org/alnweb/journal/vol3_issue2/beaudin.htm.
10. Kumar, R., Rosé, C. P., Wang, Y. C., Joshi, M., Robinson, A. (2007). Tutorial dialogue as adaptive collaborative learning support. Proceedings of the 13th International Conference on Artificial Intelligence in Education (AI-ED 2007), Amsterdam: IOSPress
11. Soller, A.L. (2001). Supporting Social Interaction in an Intelligent Collaborative Learning System. International Journal of Artificial Intelligence in Education, 12, 40-62.

12. Nachmias, R., Mioduser, D., Oren, A., & Ram, J. (2000). Web-supported emergent collaboration in higher education courses. Educational Technology and Society, 3(3), 94–104.
13. Vassileva, J., McCalla, G., and Greer J. (2003). "Multi-Agent Multi-User Modeling", User Modeling and User-Adapted Interaction, 13 (1-2), 179-210.
14. Henri, F. (1992). Computer conferencing and content analysis. In A. Kaye (Ed.), Collaborative learning through computer conferencing: the Najaden papers (pp. 117–136). Berlin: Spinger.
15. Steglitz, I. (1993). Intercultural perspective-taking: The impact of studying abroad. Unpublished dissertation. University of Minnesota.
16. Ogan, A., Walker, E., Aleven, V., Jones, C. (in press). Using a Peer Moderator to Support Collaborative Cultural Discussion. In E. Blanchard & D. Allard (Eds.), *Proceedings of the Workshop on Culturally-Aware Tutoring Systems at the 9th International Conference on Intelligent Tutoring Systems*. Montréal (Canada).

Interactive Narrative and Intelligent Tutoring for Ethics Domain

Rania Hodhod and Daniel Kudenko

Computer Science Department, University of York, Heslington, York, YO10 5DD, UK
{Poprona, Kudenko}@cs.york.ac.uk

Abstract. This paper presents an interactive narrative system (AEINS) that incorporates intelligent tutoring techniques for teaching in ill defined domains, specifically ethics and citizenship. The version AEINS presented in this paper targets students between the age of 9 and 12. In order to educate students about ethics, the idea is to present and involve students in different moral dilemmas (called *teaching moments*) which can act as an initiator for developing new or deeper thoughts about different ethical situations. The system uses Kohlberg's moral dilemmas as part of its domain. Incorporating teaching moments in the story generation context raises the issue of student agency. We discuss the tradeoff between the level of agency necessary for the student to feel in control of the story progression, and the level of system control necessary to avoid that the story digresses from the educational targets. Our goal is to allow the student/player to participate in an engaging and entertaining drama that is directly influenced by his intentional actions in a consistent and meaningful way, without affecting the educational targets or losing control over the learning process.

Keywords: Interactive narrative, intelligent tutoring, ill defined domains, Kohlberg's moral dilemmas.

1 Introduction

Interactive Narrative (IN) is an approach to interactive entertainment that enables the player to make decisions that directly affect the direction and/or outcome of the narrative experience being delivered by the computer system. In the last ten years there has been a noticeable increase in the application of narrative and drama to a variety of contexts broadly in the area of education (see for example [1, 2, 3, 4, 9]). These applications have drawn on theories that are both relevant to the design and generation of learning environments and to the ways in which learners construct their own understanding of their experience. The benefits should apply to children, teenagers and adults. Not only do INs present learners with interesting and interactive story-like experiences, but concepts from narrative can also be used to highlight important aspects of the content, such as moral dilemmas. In addition, INs can contain

dialogues which help in improving problem solving as well as declarative knowledge acquisition by students [5, 6].

The nature of ill-defined domains, where there typically is a lack of clear distinctions between "right" and "wrong" answers, makes the process of assessing students' progress and giving them appropriate feedback very difficult, if not impossible. Despite of this obvious reality, ethics are taught at schools. If we look on how ethical dilemmas are taught in face to face education, the teachers aim to lead the student to the desired ethical choice by presenting facts and evidences that contradicts his choice. By this way the student will be able to reason about his primary choice. The mentioned technique shows that the teacher sets/identifies certain educational goals and their target is to make the students reach these goals. At this point, the teachers, from their points of view, have identified the right actions to be performed within the presented dilemmas and try through discussion to convince the students about these actions. Adopting this technique makes the use of a cognitive tutor feasible in the ethical domain. AEINS is able to follow the Socratic Method in teaching this type of dilemmas, similar to classroom education but within the context of an evolved story.

In a domain like ethics, not all dilemmas can be treated in the same way. There are many dilemmas where no right or wrong answer can be identified. For these dilemmas we aim to discuss with the student the ideas behind the dilemma without giving any specific answer at the end, the final judgment is up to the student. Examples of such dilemmas are Kohlberg's dilemmas [7]. These dilemmas have the form of stories and can be easily incorporated in a narrative learning environment with minor adaptations to fit in the whole system framework. This makes them highly suited to serve as teaching moments in the educational IN.

In a way, the features of IN environments allow them to act as learning environments for ill defined domains; they are able to provide the educational content and facilitate the feedback process by incorporating it in the context of the story itself. The idea of using a cognitive tutor in educational IN environments, i.e., a module that selects problems for students in a personalized way, has not been sufficiently investigated till now. We claim that including intelligent tutoring techniques in an IN environment will:

1. Allow the student to act freely and take decisions that are incorporated, i.e., the system acts adaptively to continue the ongoing narrative,
2. Monitor the student's actions and follow the educational process, in addition to giving appropriate feedback.

The cognitive tutor includes a *domain model* and *student model* that make use of the model tracing and the knowledge tracing algorithms mentioned in [8]. The first algorithm uses the model to interpret each student action and to follow the student's strategies as they work through problem scenarios. The results of model tracing are used to provide the student with individualized teaching and feedback which is part of the story context. The second algorithm, the knowledge tracing algorithm, is used to estimate how well an individual student has mastered each concept. The results of knowledge tracing are used to determine (a) the selection of problems relevant to individual student needs and (b) when the student has mastered all the knowledge

components, concepts and skills, in a curriculum unit. The cognitive tutor also presents the student with a similar activity if it doesn't seem that the student did well on the first activity.

The paper is organized as follows. Section 2 motivates the work. Section 3 illustrates the AEINS architecture and the main idea. Student/player free agency versus controlled agency is discussed in section 4. Teaching moments and Kohlberg's dilemmas are demonstrated in section 5. Finally, concluding remarks and future work are presented in Section 6.

2 Background

Interactive narrative has been used in education for the last decade. Systems like FEARNOT! [1], TEATRIX [2], IN-TALE [3], TIME [4], are examples of these investigations. The main aim of these systems is to challenge the student to think about the problem more deeply and to assimilate a new view via social emersion. The success of the system is measured by the change of the student's behavior and/or his cognition towards the presented problems. The main drawback in these systems is the lack of a cognitive tutor.

Our approach to address the cognitive tutor problem is to incorporate intelligent tutoring techniques to interactive narrative in order to teach ill defined domains. Intelligent tutoring provides direct customized instruction or feedback to students and helps to maintain immersion and engagement with the learning process. In this case, the student will be able to act freely within the narrative environment influencing the path of the story, and at the same time be monitored and guided by the cognitive tutor. In addition, the system will be able to present the suitable educational dilemma and offer personalized feedback, according to the student's actions. The feedback will be within the context of the story emphasizing the consequences of the student's actions and decisions in the provided safe learning environment. This kind of social interaction associated with the appropriate feedback can spark new thoughts and can provide deeper insights for the student.

Having a tutor model side by side with a narrative model seems promising in well-defined domains. Mott et al. in [9] presented a guided exploratory learning environment, Crystal Island. The authors take the same approach as us regarding the way the tutoring and narrative components interact together. The learning environment is created in the domain of microbiology for middle school students. The results of this work proves that students exhibited learning gains (learning gain, $M = 0.07$, $SD = 0.14$) [10].

An outstanding example of moral dilemmas that can be used in education is the series of Kohlberg's moral dilemmas. These dilemmas have not been used before in any educational digital system. We think those dilemmas are appropriate to be used as teaching moments which are part of the system educational domain, ethics and citizenship. These dilemmas offer the student the chance to face some social dilemmas, which otherwise the student might not have or faced. Interacting with these dilemmas offers the chance to create and develop new and deeper thoughts.

3 AEINS (Adaptive Educational Interactive Narrative System)

A major challenge in interactive narrative is the creation of a large story space while managing the student‘s experience in a believable way so that it remains within the boundaries of that story space. This problem has been considered in the architecture of AEINS where a story space is created with authored contents, which describes the space of possible story experiences for the student/player that are explicitly detached from player/student actions. This idea has been presented before in [11].

AEINS is an inquiry-based learning environment that aims to spark new ideas about ethics at the user’s mind and make the user think more deeply about them. Students of all ages use questions in their learning of topics, and the skill of having learners creating "investigatable" questions is a central part of inquiry education. The version of AEINS presented in this paper targets students between the age of 9 and 12. AEINS is loosely based on GADIN [12] and has the following properties:

- Separation of the story generator (planner) and the tutor model;
- Existence of a domain model and a student model in the learning environment;
- Design the student model in a way that reflects the changes of the student’s behavior/actions in the learning environment and/or changes of his beliefs.

The separation of the story generator (or planner), from the rest of the system components gives great flexibility in adding or removing feasible actions to the story world. The story planner can be used in different worlds without any modifications. The planner’s main role is to create continuities between player/student actions in the story world and how it relates to the story space. The goal of the tutor is to teach the student and avoid frustration, while the goal of story director is to use the narrative to provide a continuous coherent story. A debate about unifying both the story director and the tutor has been presented in [13] and it showed the advantages of having both models separated rather than having them unified in one model. The story director is successful if the user is never aware of the intervention.

AEINS consists of a knowledge base, a domain model, a student model, a pedagogical model, a planner, and a presentation model as shown in figure 1. The knowledge base contains rules and constraints that describe the story world. It also contains all information about the characters and objects, such as their description and their state in the game world. The domain model is structured in two hierarchical levels of abstraction, the educational concepts such as: stealing, drinking, lying ...etc. and the teaching moments. Each concept is attached to a group of teaching moments. Sometimes more than one concept is addressed within a single teaching moment.

The student model is constructed for each particular learner. During the student’s interaction with the system, all information is recorded and is used to provide a description of the learner’s knowledge, characteristics and preferences. According to the student model and the current state of the world, the pedagogical model decides which appropriate teaching moment to present. A STRIPS representation planning algorithm is used, where actions have preconditions and effects. The planning is carried out for each dilemma presented to the user, with the user choosing their

actions and re-planning accommodating these actions if they deviate from the previously computed plan. The presentation model has the four key instructional events according to Gagne which are: awakes, explain, reinforce and transfer as discussed in [14].

To sum up, the roles of the planner, the knowledge base and the story world are: represent and track the student's progression experience as high-level narrative and guide semi-autonomous agents so they facilitate plot development. The role of the cognitive tutor maintained through domain, pedagogical and student models are: assess, guide, support, and reinforce student's problem-solving without giving away story progression.

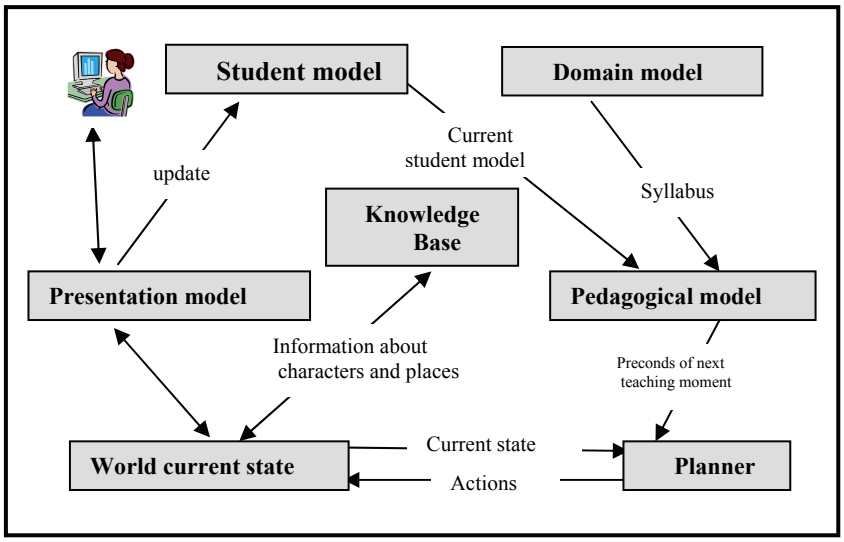


Fig. 1. AEINS architecture

4 Agency versus Control

AEINS is a learning environment which combines interactive narrative and intelligent tutoring techniques in order to deal with ill defined domains. AEINS aims to incorporate the student in moral dilemmas where he has to take decisions and see the consequences in a safe learning environment. A big challenge is to find the appropriate level of student agency in order to ensure, on one hand, that the student is able to perform actions which affects and change the world in a non superficial way and on the other hand, the student has experienced a coherent educational narrative experience that has a dramatic effect, which in turn leads to a recognizable educational outcome.

Within our learning environment, we propose two types of agency. The first kind is *complete free agency* by which the student is able to influence and control the direction of the story (i.e., before reaching or after finishing a teaching moment). The second type is *restricted agency* which exists in the entire interaction within a

teaching moment (see figure 2). Restricting the agency in order to preserve the educational targets is acceptable because the teaching moments themselves are relaxed by varying the places and characters that can participate in their worlds; this part is illustrated in the following section. In addition, every teaching moment is able to provide a different outcome according to the student's interaction within the teaching moment.

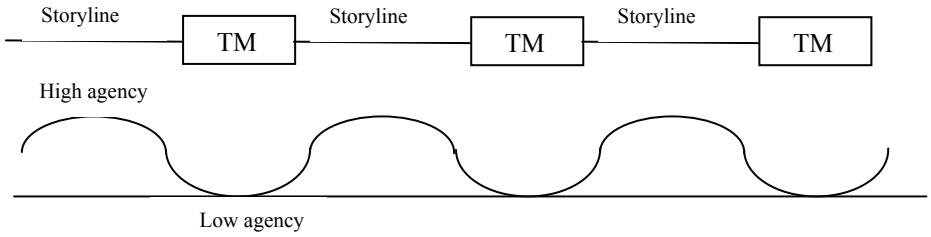


Fig. 2. Representation of student's agency in the learning environment

5 Kohlberg's Dilemmas

The teaching moments in AEINS are scripted and authored according to general moral situations. They are a crucial part of the story generation process. They aim to create new thoughts and/or develop deeper thoughts about some moral situations. Kohlberg's moral dilemmas are examples of moral stories; he was interested in how people would justify their actions if they were put in a similar moral crux [15]. The idea is to incorporate these dilemmas as the teaching moments which are part of the story and part of the active characters' lives. The student is able to interact with these dilemmas and influence the outcome.

Kohlberg was mainly interested in the reasoning behind the answers to ethical questions and tried to achieve this through discussions and questions with the interviewed students. We incorporated the questions Kohlberg used in his interviews as part of teaching moments to enquire about the student's reasoning. By monitoring student's actions and responses through the story and the different teaching moments and with a continuous update of the student model, conclusions on the student's reasoning can be extracted from the attributes of the student model. For example, we incorporated one of Kohlberg's dilemmas in a teaching moment as follows. The dilemma has been expressed as the story:

> *"Judy was a twelve-year-old girl. Her mother promised her that she could go to a special rock concert coming to their town if she saved up from baby-sitting and lunch money to buy a ticket to the concert. She managed to save up the fifteen dollars the ticket cost plus another five dollars. But then her mother changed her mind and told Judy that she had to spend the money on new clothes for school. Judy was disappointed and decided to go to the concert anyway. She bought a ticket and told her mother that she had only been able to save five*

dollars. That Saturday she went to the performance and told her mother that she was spending the day with a friend. A week passed without her mother finding out. Judy then told her older sister, Louise, that she had gone to the performance and had lied to her mother about it. Louise wonders whether to tell their mother what Judy did".

An example of the interview questions for the above dilemma is:

1. *Should Louise, the older sister, tell their mother that Judy lied about the money or should she keep quiet?*
2. *In wondering whether to tell, Louise thinks of the fact that Judy is her sister. Should that make a difference in Louise's decision?*
3. *Does telling have anything to do with being a good daughter?*

AEINS starts by asking the student 3 questions about how he considers himself; for example do you consider yourself honest person? If the student answers yes, then the honesty attribute will be updated from *studentmodel (honesty, 0)* to *studentmodel (honesty, 10)*. Then the student starts acting by choosing his friends and/or invites a friend to a certain place. In response to his actions the agents are free to decide to be the user's friends or not and to accept his invitation or not. Some actions like *asking person to cheat* have another role in updating the student model. The student can then tell the system to act. The system chooses the appropriate teaching moment that best suits the current student model. For example: The teaching moment below is chosen if *studentmodel (loyalty, ?Value$_1$) and ?Value$_1$>θ_1 and ?Value$_1$<θ_2; studentmodel (honesty, ?Value$_2$) and ?Value$_2$>θ_1*, where θ_1 and θ_2 are pre-specified thresholds. In the story context, the teaching moment has the following preconditions ('?' denotes a variable): *character (?x, ?user); character (?Y, ?Char);place(?Place1), place(?Place2), at (?X, ?Place1) and at (?Y, ?Place2) and ?Place1=?Place2.*

These preconditions are sent to the planner as goals where its role is to find the suitable actions that can lead to satisfy these preconditions. In this example getting a character called Gina and the player in the same place. The director tells Gina to phone the user and asks him to meet at certain place. If the user agrees then he and Gina are now together at the same place, the preconditions of the dilemma are satisfied and the system starts the teaching moment. If the user refused to join Gina so the director will choose another action where Gina will reply that she will join the student at his location. The story generated for this dilemma is shown below. The system output is in normal font, comments are in italic, and student/player responses are in bold.

Your little sister Gina wants to talk to you; she seems to have a big secret
Gina is ready to tell you her secret;
I saved money to go to the concert and mum asked me to buy school clothes with this money, so I lied to mum and told her I earned half the money only, and I did go to the concert without telling anyone, this is between me and you. DON'T TELL ANYONE.

Your friend John heard the conversation between you and your sister accidentally; he is coming to discuss it with you.
John: I am sorry, I have heard your sister story accidentally. Do you think you should tell your mum that your sister lied about the money?
Yes

John: By telling your mum, do you think this makes you a good son?
Yes
John: What If Gina's was not your sister; will this make a difference in your decision?
No
John: Do you think by telling your mum you betrayed your sister's trust?
No
The dialogue continues this way, at the same time the student model is updated after each answer, but it is not used until the whole dilemma finishes and then it is used by the pedagogical model to decide the next appropriate dilemma.

The main challenge at this stage is the feedback which is not possible to be in the form of "your answer is right" and "your answer is wrong". As we discussed before the narrative environment offers the chance for the feedback to be tailored within the context of the story. So the consequences of the student's interactions with the teaching moment are implicitly the kind of feedback needed in this educational domain. The system is able to present the student with one of the possible outcomes for this dilemma. For example, the above situation will end with the sister Gina upset with the student and the mother being upset from the daughter and the daughter being grounded. If the student chooses not to tell their mother, a guilty feeling will accompany him and/or he may choose to advise the little sister not to repeat such an action again. Through this experience the student will get the chance to reason more deeply about those dilemmas and see what outcome satisfies him most. We intend to extend the student model to include student beliefs that are updated through his interaction with the system. These beliefs will help to evaluate what the students have learnt from the activity.

6 Remarks and future work

In this paper we proposed an adaptive educational interactive narrative system. The system is mainly a narrative-based system that incorporates intelligent tutoring. The system is able to monitor the student's actions and provide personalized teaching and feedback. The system uses Kohlberg's moral dilemmas as part of its domain. An important issue is the agency of the student within the learning environment. We proposed a technique that offers two types of agencies; a full and complete agency outside the teaching moments and semi-controlled agency inside the teaching moments.

As the next step in this research, we plan to evaluate AEINS internally and externally. The internal evaluation will assess the inner workings of the intelligent interactive narrative, i.e., it will check that the system is capable of offering a continuous coherent story with the inclusion of the suitable teaching moments according to the individual student/player. We intend to do external evaluation in the near future via a user study with respect to the following two hypotheses: (1) Students feel that an adaptive story is more engaging than a fixed story. (2) Students gain a

deeper understanding of the domain than they had before, and are able to reason more deeply about moral dilemmas.

References

[1] Watson, S., Vannini, N., Davis, M., Woods, S., Hall, M., Hall, L., Dautenhahn K.: Fear Not! An Anti Bullying Intervention. In: Patrick Olivier and Christian Kray (eds.) Evaluation of an Interactive Virtual Learning Environment, Artificial & ambient Intelligence, Eds. Proceedings of the AISB Annual Convention, Uk (2007)

[2] Prada, R., Machado, I., Paiva, A.: Teatrix: A Virual environment for story Creation, Intelligent Tutoring syatems,. G. Gauthier, C. frasson & K. van Lehn (eds.), Springer (2000)

[3] Riedl, M., Stern, A.: Believable Agents and Intelligent Story Adaptation for Interactive Storytelling. 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment, Darmstadt, DE (2006)

[4] Harless, W. G.: An Interactive Videodisc Drama: The Case of Frank Hall. Journal of Computer-Based Instruction, http://www.idrama.com/Media/jcbi_86.htm

[5] Mitrovic, A.: An Intelligent SQL Tutor on the Web. International Journal of Artificial Intelligence in Education, 13(2-4), 171-195, 2003.

[6] A. Weerasinghe, A. Mitrovic: Supporting Self-Explanation in an Open-Ended Domain. KES 2004: 306-313, 2004.

[7] http://www.haverford.edu/psych/ddavis/p109g/kohlberg.dilemmas.html

[8].Koedinger, K. R., Aleven, V.: Exploring the Assistance Dilemma in Experiments with Cognitive Tutors, Educational Psychology Review, Springer Netherlands. Volume 19, Number 3, September (2007)

[9] Mott, B., McQuiggan, S., Lee, S., Lee, S. Y., and Lester. J.: Narrative-Centered Environments for Guided Exploratory Learning. In Proceedings of the Agent Based Systems for Human Learning Workshop at the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (ABSHL-2006), Hakodate, Japan, May 9, 2006.

[10] McQuiggan, S., Rowe, J., Lee, S., and Lester, J.: Story-Based Learning: The Impact of Narrative on Learning Experiences and Outcomes. In Proceedings of the Ninth International Conference on Intelligent Tutoring Systems (ITS-2008), Montreal, Canada, 2008.

[11] Magerko, B.: Story Representation and Interactive Drama. 1st Artificial Intelligence and Interactive Digital Entertainment Conference, Marina del Rey, CA (2005)

[12] Barber, H. and Kudenko, D.: Adaptive Generation of Dilemma-based Interactive Narratives, In Proceedings of SAB'06 Workshop on Adaptive Approaches for Optimizing Player Satisfaction in Computer and Physical Games, Rome, October 2006.

[13] Riedl, M., Lane, H. C., Hill, R., Swartout, W.: Automated Story Direction and Intelligent Tutoring: Towards a Unifying Architecture. Technical Report. University of Southern California Marina Del Rey Ca Inst For Creative Technologies (2006)

[14] Gagne, R., Dionne, G.: Measuring Technical Change and Productivity Growth Varying Output Qualities and Incomplete Panel Data, Cahiers de recherche 9201, Centre interuniversitaire de recherche en économie quantitative, CIREQ (1992)

[15] Crain, W. C.: Kohlberg's Stages of Moral Development. W.C. Theories of Development Prentice-Hall. pp. 118-136 (1985), http://faculty.plts.edu/gpence/html/kohlberg.htm

A Selection Strategy to Improve Cloze Question Quality

Juan Pino, Michael Heilman, and Maxine Eskenazi

Language Technologies Institute
Carnegie Mellon University, Pittsburgh PA 15213, USA
{jmpino,mheilman,max}@cs.cmu.edu

Abstract. We present a strategy to improve the quality of automatically generated *cloze* and *open cloze* questions which are used by the REAP tutoring system for assessment in the ill-defined domain of English as a Second Language vocabulary learning. Cloze and open cloze questions are fill-in-the-blank questions with and without multiple choice, respectively. The REAP intelligent tutoring system [1] uses cloze questions as part of its assessment and practice module. First we describe a baseline technique to generate cloze questions which uses sample sentences from WordNet [2]. We then show how we can refine this technique with linguistically motivated features to generate better cloze and open cloze questions. A group of English as a Second Language teachers evaluated the quality of the cloze questions generated by both techniques. They also evaluated how well-defined the context of the open cloze questions was. The baseline technique produced high quality cloze questions 40% of the time, while the new strategy produced high quality cloze questions 66% of the time. We also compared our approach to manually authored open cloze questions.

1 Introduction

This paper describes a strategy to generate cloze (fill-in-the-blank) and open cloze (without multiple choice) questions, which are part of the assessment and practice module in the REAP system [1]. REAP is an intelligent tutoring system for English as a Second Language (ESL) vocabulary learning that provides a student with authentic documents retrieved from the web that contain target vocabulary words. These documents go through text quality filtering and are annotated for readability level [3] and topic. The system uses these annotations to match documents to the student's level and interests. After each reading the system provides a series of practice exercises for focus words that were in that reading. Following the practice session, the system updates the learner model for the student based on his or her performance. In the following reading session, the system searches for texts according to the updated model, in order to give the student individualized practice. The REAP system uses several types of questions, for example *synonym* or *related word* questions [4], and in particular cloze questions. A cloze question consists of a *stem*, which is a sentence

with a target word removed, and of *distractors*, which are words semantically or grammatically related to the target word. Examples of good and bad quality cloze questions are shown in Fig. 1 and 2. Brown et al. [5] successfully generated several different types of questions, for example synonym and cloze questions, by using WordNet [2]. However, teachers whose students were using the tutor judged the overall quality of the cloze questions to be insufficient to be used in their class. As a result, teachers generated the cloze items manually and without authoring support, which is time-consuming. This paper presents a strategy to improve the quality of automatically-generated cloze and open cloze questions.

Vocabulary learning can be considered an ill-defined domain due to the fact that there is no absolute measure of vocabulary knowledge. The exact nature of individual word knowledge is an unresolved research question, with various possible theories [6–8]. Although there exist ways of encoding lexical knowledge such as WordNet, Latent Semantic Analysis [9], and thesauri which can in some sense be viewed as models, there is no clear consensus on how to model lexical information. Even when assuming that an individual word is known, it is still challenging to make inferences about other, possibly related, words, because words appear in very different contexts [12]. On the contrary, well-structured domains have fully-developed cognitive models which have been applied successfully in intelligent tutoring. Examples include the work of VanLehn et al. [13] in physics and Anderson and Reiser in computer programming [10]. Also, many practice tasks for vocabulary learning, such as writing a sentence containing a vocabulary word (sentence production), lack single, definite correct answers. In fact, the sentence production problem can have an infinite number of correct answers: words can be combined in novel ways since human language is productive.

Although sentence production is a very good exercise for vocabulary learning and provides a rich measure of the knowledge of a word, it is very hard to assess automatically, precisely because it has many solutions. One could imagine having a model solution and compare to it a student's solution as in the entity relationship modelling tutor KERMIT [11]. Again this is hard to do automatically because the student's solution could be very far, for example semantically, from the model and yet correct. On the other hand, the reverse problem, posed by cloze questions, of completing a sentence with a missing word is less open-ended but still valuable for practice and assessment. One advantage is that cloze questions can be scored automatically. Even though the process of grading is simpler, cloze questions can still assess various aspects of word knowledge. Nation divides the concept of word knowledge into *receptive knowledge* and *productive knowledge* [6]. He lists ten criteria for receptive knowledge and nine criteria for productive knowledge. Multiple-choice cloze questions appear to involve five of the categories for receptive knowledge: students have to identify the written form of the target word and of the distractors. They also have to know at least one meaning of the word, namely the meaning of the word in a given context. They can make use of related words if such words are found in the stem. Finally, they need to recognize the correct grammatical usage of the word. However, there is no test of morphological knowledge. Students can also answer the question if

they know the meaning of the word in one specific context only. Finally, they do not have to check if the usage (register, frequency of use) of the word fits in the stem. Furthermore, *open* cloze questions also involve several categories of productive knowledge. To answer an open cloze question, students need to produce a word expressing a certain meaning in the given context. This word could be related to, or form collocations, that is pairs of frequently co-occurring words, with certain words in the stem, it should form a grammatically correct sentence, it should be spelled correctly and its level of formality should be chosen carefully. Therefore, open cloze questions also assess productive knowledge. Having a reliable assessment of individual word knowledge allows the system to know which words it should focus on.

Existing work on cloze question generation such as that of Liu et. al [14] has focused on lexical items regardless of their part-of-speech. Lee and Seneff [15] focused on generating cloze questions for prepositions with a technique based on collocations. Others have generated cloze questions for common expressions. For instance, Higgins [16] generated cloze questions for structures such as "not only the" that assess grammatical skills rather than vocabulary skills. Hoshino and Nagakawa [17] generated questions for both vocabulary and grammatical patterns. Mitkov et al. [18] generated cloze questions about concepts found in a document by converting a declarative sentence into an interrogative sentence.

This work first focused on generating cloze questions for adverbs and was then extended to other parts of speech. Adverbs are considered to stand between open class and closed class words[1], which is why our strategy should be extensible to any part of speech. Additionally, it seems that adverbs with the suffix **-ly** (for example "clearly"), which are the most frequent kind of adverb, can be easily replaced in a sentence by other adverbs of the same kind without producing a grammatically or semantically incorrect sentence, if this sentence does not sufficiently narrow the context. As a consequence, it is important to avoid having several possible answers for cloze questions on adverbs.

This paper concentrates on the quality of the stem and the quality of the distractors. Sumita et al. [19] generate distractors thanks to a thesaurus; if the original sentence where the correct answer is replaced by a distractor gets more than zero hit on a search engine, the distractor is discarded. In [14], the authors use a technique based on word sense disambiguation to retrieve sentences from a corpus containing a target word with a particular sense. Their strategy also uses collocations to select suitable distractors. Our strategy makes use of collocations too by applying it to both stems and distractors in order to ensure their quality.

[1] An open word class is a class of words that tends to evolve rapidly, for example the nouns form an open class because new nouns are often created while other nouns become obsolete; a closed class is stable, for example the set of prepositions is not likely to evolve very quickly.

2 Cloze Question Generation

We first describe the baseline technique based on WordNet sample sentences, which allowed us to investigate linguistic features for good cloze questions.

WordNet is a lexical database in which nouns, verbs, adjectives and adverbs are grouped in synonym sets, or *synsets*. Synsets have a semantic relationship such as synonym, antonym, etc. For each synset, a definition and, most of the time, sample sentences, are provided. The latter are used to produce the stems. We want to generate a cloze question for an adverb w. We assign its most frequent synset as an adverb to w. This synset may have sample sentences, either involving the target word w itself or synonyms in the same synset. If sentences involving the target word are available, we select them. After this first selection, if there are still several possible sentences, the longest one is preferred, because longer sample sentences in WordNet tend to have richer contexts. The distractors are chosen randomly from a list of adverbs. As a result, they have the same part of speech as the target word which is recommended by Haladyna et al. [20] as a way of avoiding obviously wrong answers. Furthermore, Hensler and Beck [21] have shown that higher proficiency readers among students from grade 1 to 6 can use part of speech as a clue to answer a cloze question. Since the REAP system is used by ESL learners who are able to use parts of speech in their native language, we believe they may be able to use them in English as well. Thus all the distractors have the same part of speech.

2.1 Stem Selection

Similar to the baseline, our strategy aims to select the most suitable sentences from a set of sentences containing the target word. However, for both for the stem and the distractors, our selection criteria are more fine-grained and we expect to produce a better quality output.

First, to apply a selection strategy we need to choose from several sample sentences per word. However, WordNet has zero or one sample sentence per word in any given synset. Therefore, we used the Cambridge Advanced Learner's Dictionary (CALD) which has several sample sentences for each sense of a word. We retained the same selection criterion as for the baseline, namely the length of the sentence, and added new linguistically relevant criteria. Our approach employs the following selection criteria: complexity, well-defined context, grammaticality and length.

Each sample sentence was given a weighted score combining these four criteria. We assessed the complexity of a sentence by parsing it with the Stanford parser [22, 23] and counting the resulting number of clauses. The Stanford parser uses the Penn Treebank syntactic tag set described in [24]. By *clause* we mean the sequences annotated with the following tags: S (simple declarative clause), $SBAR$ (clause introduced by subordinating conjunction), $SBARQ$(direct question introduce by *wh*-word or *wh*-phrase) and $SINV$ (declarative sentence with subject-auxiliary inversion). We chose this selection criterion through analysis of a dataset of high quality manually-generated cloze questions. We noticed that

high quality stems often consist of two clauses, one clause involving the target word, and the other clause specifying the context, as in the following sentence: "We didn't get much information from the first report, but *subsequent* reports were much more helpful." (the target word is italicized). We believe that more clauses tend to make the context more well-defined.

The context of a sentence is considered to be well-defined if it requires the presence of the target word in the sentence and rejects the presence of any another word. A way to assess how well-defined the context is in a sentence with respect to a target word is to sum the collocation scores between the target word and the other words in the sentence. Collocations are pairs, or sometimes larger sets, of words that frequently co-occur, often despite the absence of clear semantic requirements. An example is that tea is much more likely to be called "strong" than "powerful". Conversely, a car is more likely to be "powerful" than "strong". An "argument", however, can be either. The strength of the context around a word is in some sense determined by the presence of very informative collocations. For example, the sentence "I drank a cup of strong (blank) with lemon and sugar" has a very well-defined context for "tea" because of the collocations with "strong", "lemon", "sugar", and "drink". We followed the method described by Manning and Schütze [25] to estimate a set of collocations for each target word. We used a corpus consisting of approximately 100,000 texts appropriate for ESL learners, which are a part of the REAP database of potential readings. The system calculated the frequency of co-occurrence of content words by counting co-occurences within a window of two adjacent words. It then identified salient collocations by calculating a likelihood ratio for each possible pair. For this last step, other metrics such as point-wise mutual information and Pearson's chi-square test were also tried and produced similar estimates of collocations.

We also used the Stanford parser to assess the grammaticality of the sentence. Each parsed sentence was assigned a score corresponding to the probabilistic context-free grammar score. Longer sentences generally have poorer scores and thus we normalized this score with the length of the sentence[2]. Although the parser is applied to any sentence, even ungrammatical ones, the latter receive a lower score than grammatical sentences. Knight and Marcu [26] use a similar technique to estimate the probability of a short sentence in a noisy channel model for sentence compression.

Finally, we used the sentence length as a quality criterion. Indeed, we noticed that the sentences generated by the baseline technique were too short (6 tokens on average) to provide enough context for the target word to be guessed (see Fig. 2), although there were some rare exceptions (see Fig. 1).

The weights for each criterion were determined manually by the authors by examining their effects on performance on training data. We randomly chose 30 adverbs as training data and modified the weights so as to obtain the best sample

[2] The scores are log-probabilities, they are therefore negative, for example a correct sentence of ten words could be assigned a score of -100, a correct sentence of five words could be assigned a score of -50, and after normalizing, both sentence would be assigned a score of -10

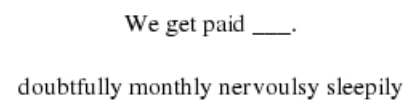


Fig. 1. A rare case of a short sentence with a sufficiently well-defined context for the target word.

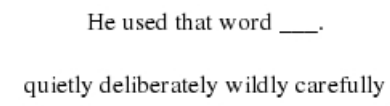


Fig. 2. A short sentence with multiple answers due to an ill-defined context.

sentences for these adverbs. Our criteria to judge the best samples were the same criteria used by ESL teachers for their assessment of the cloze questions (see Sect. 3). Our final weights are 1 for length, 8 for complexity, 0.3 for grammaticality and 0.2 for the collocation criteria[3]. In addition, we filtered out misspelled sentences with the Jazzy spell checker [27]. Although this is not very relevant for the Cambridge Dictionary, it is relevant when using a large corpus of text.

2.2 Distractor Selection

The quality of distractors was also scored. In order to produce a score for each distractor, we replaced the target word with a distractor in the sentence, then rescored the sentence. Only the grammaticality and collocation criteria measured how well the distractors fit in the sentence, both from a syntactic and from a semantic point of view. We selected the distractors with the highest scores, that is the ones that fit best in the sentence. Thus we prevented them from being obviously wrong answers. However we also wanted to avoid having several possible answers, which would happen if the distractors fit too well in the sentence. We therefore selected distractors that were semantically "far enough" from the target word. To compute semantic similarity between two words, we used Patwardhan and Pedersen's method [28]. In this method, two words w_1 and w_2 are associated with their definition in WordNet. Each word d of the definition is associated with a *first order context vector*, computed by counting the co-occurrences of d with other words in a corpus[4]. Then, computing the resultant (i.e. the sum) of these *context vectors* produces a *second order context vector*, which represents the meaning of the word. Finally the dot product of the *second order context vectors* associated with w_1 and w_2 give the semantic similarity between w_1 and w_2. This method, unlike several other methods based on the WordNet hierarchy, handles all parts-of-speech, not just verbs and nouns.

Using distractors that are semantically different from the target word does not guarantee that they will not fit in the sentence. Figure 2 shows that unrelated words can fit in the same sentence. Therefore, this technique will work with a sentence that has few possible answers.

[3] These values do not accurately reflect relative importance of the criteria; they are intended for the reader's information

[4] Patwardhan and Pedersen use the glosses in WordNet as a corpus

2.3 Stem selection using several dictionaries and a raw text corpus

We also applied the technique for stem selection to several dictionaries and a raw text corpus, with all parts of speech included. A corpus provides many sentences per word, therefore producing many good quality cloze questions per word. However, unlike dictionaries where sentences are carefully crafted, a large text corpus will contain many useless sentences that the algorithm ought to discard. We did not generate distractors. This test was mainly designed to evaluate the quality of the sentences independently of the quality of the distractors.

Dictionaries In order to compare sentences from dictionaries and from raw text corpus, we extracted the sample sentences provided by several dictionaries [5] for each word in the Academic Word List [29].

Corpus preprocessing A ten million word subset of the REAP documents, which are gathered from the Web, was filtered for text quality by running a part-of-speech (POS) tagger on each document and computing the cosine similarity between the vectors of POS trigrams in the document and a corpus of English literature, known to be grammatical and of high quality. HTML tags of the documents were stripped out. The raw text was chunked into sentences using the sentence detector of OpenNLP toolkit [30]. Only the sentences containing words from the Academic Word List were retained.

Parameters We used two different sets of weights for dictionaries and for the raw text corpus, shown in Tab. 1. In dictionaries, sentences tend to be too short, therefore the length weight is positive. On the contrary, sentences found in raw text are very long. To avoid long sentences, the length weight is negative. This way, we expect to find a trade-off between well-defined context and length. Furthermore, sentences of more than 30 words were discarded as not suitable for a practice exercise. A nonlinear function of the length could also have discarded sentences that are too short or too long. Dictionary sentences often lack a verb, which is why the complexity weight is high. The grammaticality weight is slightly higher for raw text because dictionary sentences are usually well-formed.

3 Evaluation

A series of three experiments were conducted. In each of them, five ESL teachers, two of whom are native speakers, assessed the quality of a set of questions. This set consisted of manually-generated and automatically-generated questions displayed in random order. Open cloze questions were generated to measure how well-defined the context was independently of the choice of distractors.The experiment settings are detailed in Tab. 2.

[5] Cambridge Advanced Learner's Dictionary, Longman Dictionary of Contemporary English, Dictionary.com, Merriam-Webster OnLine, MSN Encarta, RhymeZone.

Table 1. Different choice of weights for dictionaries and raw text corpus

	Dictionaries	Raw Text Corpus
length	0.4	-0.7
complexity	6	3
grammaticality	0.3	0.4
collocations	0.3	0.3

Table 2. Experiment Settings

Strategy	Baseline	Linguistically Enriched	Linguistically Enriched	
Question Type	Cloze	Cloze	Open Cloze	
Corpus for automatically generated questions	WordNet	CALD	Dictionaries	REAP
Number of automatically generated questions	30	30	34	33
Number of manually generated questions	30	30	30	

For each cloze question, the teachers answered a series of questions, illustrated in Fig. 3. The questions targeted specific reasons regarding the suitability of cloze questions.

For open cloze questions, the teachers assessed the quality of the context on a one-to-four scale. They also indicated if the sentence was too long, too short or of the right length, and if the sentence was too simple, too difficult or at the appropriate level of English for upper-intermediate ESL students.

3.1 Cloze Questions

The same teachers evaluated randomly ordered questions for both the baseline technique and the proposed technique. The target words for these questions were randomly sampled with replacement from a pool a word in both cases. Overall, the teachers judgments had a Fleiss' *kappa*[6] agreement of 0.3. We therefore consider the agreement between the teachers to be "fair" [31]. We expect that agreement would have been higher if better training procedures had been in place, such as allowing teachers to practice on an independent set of items and then discuss their differences in order to better calibrate their judgments. It should be also noted that these teachers teach at a variety of levels, in France and the United States.

Table 3 summarizes the results of the first two experiments about cloze questions. 40.14% of the questions generated by the baseline technique were judged acceptable, while our proposed strategy generated 66.53% of suitable questions.

[6] Fleiss' *kappa*, unlike Cohen's *kappa* measure, is a statistic for measuring agreement between more than two raters.

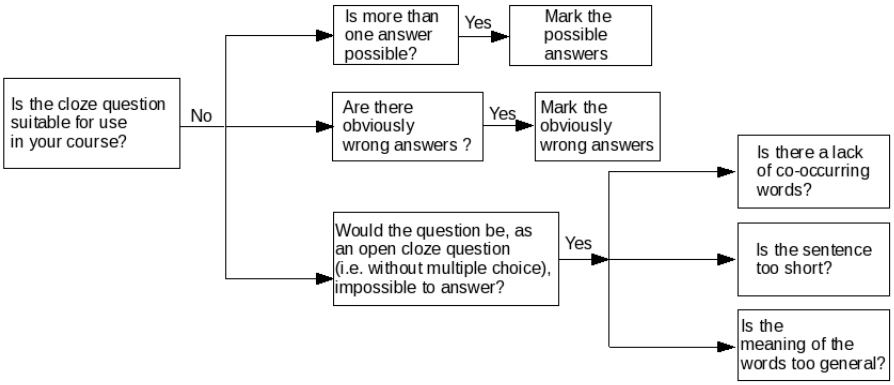


Fig. 3. Flow chart of the cloze question evaluation process

The difference was statistically significant ($t(29) = 2.048$, $p < 0.025$). This improved performance is still too low to plug the generated questions directly in a tutoring system; however, it appears to be high enough to allow us to build an authoring tool to improve authoring efficiency.

The most often cited reason for unacceptable questions was that several answers were possible. It was given for 34.01% of the baseline questions and 12.08% of the questions generated with the new strategy. In the baseline technique, there was no verification to determine if the distractors were semantically far enough from the target word. This flaw was corrected in the new strategy, but there is still room for improvement of the quality of the distractors. The second reason for unacceptable questions was that these questions, as open cloze questions, were impossible to answer, either because of a lack of co-occurring words, overly short sentences, or words with too general meaning. Again, the proposed strategy made a significant improvement over the baseline for all these aspects. However, the *kappa* values for inter-rater reliability were much lower for these points. The presence of obviously wrong answers as a reason for not acceptable questions was not often cited by the assessors either in the baseline strategy, or in the proposed strategy, probably because the distractors had the same part of speech as the correct answer and fit in the sentence at least grammatically.

3.2 Open Cloze Questions

Figures 4 and 5 show the distribution of the questions at levels 3 and 4 of context quality and for each set of question (manual, generated from dictionaries or from raw text). We consider that a sentence at level 3 and 4 of context quality is acceptable for an open cloze question. 61.82% of the automatically-generated questions for dictionaries are at levels 3 and 4 while 71.23% of the manually-generated questions were at levels 3 and 4.

Table 3. Assessment of cloze questions

Strategy	Manual (1st experiment)	Baseline	Manual (2nd experiment)	Linguistically Enriched
Suitable question	90.13%	40.14%	80.67%	66.53%
Several possible answers	3.95%	34.01%	10.67%	12.08%
Obviously wrong answers	0.66%	4.76%	1.33%	3.36%
Multiple choice necessary	1.32%	32.65%	0.67%	6.04%
Lack of co-occurring words	0%	16.33%	0%	2.68%
Too short	0.66%	19.93%	0%	5.37%
Words with too general meaning	0.66%	18.37%	0%	2.01%

When applied to dictionaries, our strategy is comparable to the human method for generating stems. The main difficulty encountered when generating cloze questions was the choice of the distractors. We can therefore focus on this task since the first one, namely generating stems, is relatively well mastered. However, the strategy did not perform as well on raw text. In raw text, the context is usually well-defined over several sentences but it is hard to find a single sentence that contains a rich context in itself. Analysis of length and level of difficulty shows that dictionary sentences tend to be too short but with the right level of difficulty and that raw text sentences tend to be too long and too difficult, as shown in Tab. 4.

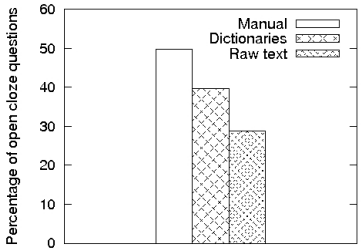


Fig. 4. Distribution of open cloze questions at context level 3 (well-defined context, two or three words can fit in the sentence)

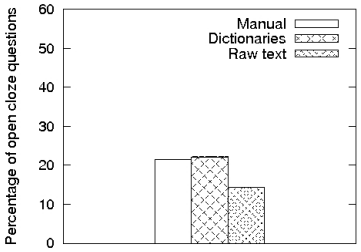


Fig. 5. Distribution of open cloze questions at context level 4 (very well-defined context, only one word can fit in the sentence)

4 Conclusion and Future Work

We have developed a strategy for selecting high quality cloze questions. This strategy was prompted by a baseline technique developed within the framework

Table 4. Length and difficulty level for open cloze questions

	Manual	Raw text	Dictionaries
Too long	0%	18.38%	2.29%
Too short	5.88%	10.29%	23.66%
Right length	94.12%	71.32%	74.04%
Level too difficult	4.31%	29.46%	9.37%
Level too simple	0.86%	1.55%	5.47%
Right level	94.83%	68.99%	85.16%

of the REAP system. It was developed by taking into account the weak points of the baseline technique shown by a first evaluation. The evaluation of our strategy showed that we are able to generate stems of good quality, and therefore open cloze questions of good quality. We believe that by generating distractors of better quality, for example using the technique described in [19], we will be able to improve the automatic generation of cloze questions. Our technique can be extended to other languages by using different parsers, dictionaries and corpora. It can also be used as a measure for the amount of context, or information, a sentence provides.

We face two main challenges. First, distractors that fit in a sentence grammatically often also fit semantically. Choosing distractors with a large semantic distance from the correct answer does not always solve this problem. Similarly, open cloze questions rarely have only one possible answer. Second, corpus sentences inherently differ from dictionary sample sentences because the same amount of context is defined in several corpus sentences and in one dictionary sentence only.

At a higher level, we have demonstrated the utility of linguistically motivated, statistical approaches for generating assessment and practice materials in the ill-defined domain of English vocabulary learning. Ill-defined domains often deal with processes, especially linguistic ones, that are probabilistic in nature or at least possess a degree of complexity which make them challenging to model with deterministic algorithms. As such, statistical methods, such as those applied in our approach to generating vocabulary assessments, can be particularly effective for developing educational technology for ill-defined domains.

5 Acknowledgments

We would like to thank Gregory Mizera, Stacy Ranson, Catherine Jutteau, Renée Maufroid, Geneviève Jaslier and Annie Nottebaert for assessing the questions. We also would like to thank Catherine Jutteau and Sarah Carvalho for reviewing this paper and Le Zhao and Anagha Kulkarni for scientific input. This material is based on work supported by NSF grant SBE-0354420, the research reported here was supported in part by the Institute of Education Sciences, U.S. Department of Education, through Grant R305B040063 to Carnegie Mellon University. Any

opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsor.

References

1. Heilman, M., Collins-Thompson, K., Callan, J., Eskenazi, M.: Classroom success of an Intelligent Tutoring System for lexical practice and reading comprehension. Proceedings of the Ninth International Conference on Spoken Language Processing (2006)
2. Fellbaum, C.: WordNet. An electronic lexical database. Cambridge, MA: MIT Press (1998)
3. Collins-Thompson, K., Callan., J.: Predicting reading difficulty with statistical language models. Journal of the American Society for Information Science and Technology, vol. **56**, no. **13** (2005) 1448–1462.
4. Heilman, M., Eskenazi, M.: Application of Automatic Thesaurus Extraction for Computer Generation of Vocabulary Questions. Proceedings of the SLaTE Workshop on Speech and Language Technology in Education. (2007)
5. Brown, J., Frishkoff, G.,Eskenazi, M.: Automatic question generation for vocabulary assessment. Proceedings of HLT/EMNLP. Vancouver, B.C. (2005)
6. Nation, I. S. P.: Learning Vocabulary in Another Language. Cambridge, England: Cambridge University Press (2001) 26–28
7. Dale, E., O'Rourke, J.: Vocabulary building. Columbus, Ohio: Zaner-Bloser (1986)
8. Stahl, S. A.: Three principals of effective vocabulary instruction. Journal of Reading, vol. **29** (1986)
9. Landauer, T. K., Foltz, P. W., Laham, D.: An Introduction to Latent Semantic Analysis. Discourse Processes, vol **25** (1998) 259–284
10. Anderson, J. R., Reiser, B. J.: The LISP tutor: it approaches the effectiveness of a human tutor. Lecture notes in computer science, vol. **174** (1985) 159–175
11. Suraweera, P., Mitrovic, A.: An Intelligent Tutoring System for Entity Relationship Modelling. International Journal of Artificial Intelligence in Education, vol. **14** (2004) 375–417
12. Heilman, M., Eskenazi, M.: Language Learning: Challenges for Intelligent Tutoring Systems Proceedings of the Workshop of Intelligent Tutoring Systems for Ill-Defined Domains (2006)
13. VanLehn, K., Lynch, C., Schulze, K., Shapiro, J.A., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., and Wintersgill, M.: The Andes Physics Tutoring System: Lessons Learned. International Journal of Artificial Intelligence and Education, 15 (3). (2005)
14. Liu, C. L., Wang, C. H., Gao, Z. M., Huang, S. M.: Applications of Lexical Information for Algorithmically Composing Multiple-Choice Cloze Items. Proceedings of the Second Workshop on Building Educational Applications Using NLP. Ann Arbor, Michigan (2005) 1–8
15. Lee, J., Seneff, S.: Automatic Generation of Cloze Items for Prepositions. Proceedings of Interspeech (2007)
16. Higgins, D.: Item Distiller: Text retrieval for computer-assisted test item creation. ETS (2006)
17. Hoshino, A., Nakagawa, H.: Assisting cloze test making with a web application. Proceedings of Society for Information Technology and Teacher Education International Conference. San Antonio, Texas, USA (2007) 2807–2814

18. Mitkov, R., An Ha, L., Karamanis, N.: A computer-aided environment for generating multiple-choice test items. Natural Language Engineering, Cambridge University Press (2006) 1–17
19. Sumita, E., Sugaya, F., Yamamoto, S.: Measuring Non-native Speakers' Proficiency of English by Using a Test with Automatically-Generated Fill-in-the-Blank Questions. Proceedings of the Second Workshop on Building Educational Applications Using NLP. Ann Arbor, Michigan (2005)
20. Haladyna, T. M., Downing, S. M., Rodriguez, M. C.: A Review of Multiple-Choice Item-Writing Guidelines for Classroom Assessment. Applied Measurement in Education. Lawrence Earlbaum (2002)
21. Hensler, B. S., Beck, J.: Better student assessing by finding difficulty factors in a fully automated comprehension measure. Proceedings of the 8th International Conference on Intelligent Tutoring Systems. Jhongli, Taiwan (2006)
22. Klein, D., Manning, C. D.: Fast Exact Inference with a Factored Model for Natural Language Parsing. Advances in Neural Information Processing Systems **15**. Cambridge, MA: MIT Press (2002) 3–10
23. Klein, D., Manning, C. D.: Accurate Unlexicalized Parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics (2003) 423–430
24. Marcus, M. P., Santorini, B., Marcinkiewicz, M. A.: Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics **19** (1993) 313–330
25. Manning, C. D., Schütze, H.: Foundations of Statistical Natural Language Processing. Cambridge, MA: MIT Press (1999) Chap. 5
26. Knight, K., Marcu, D.: Summarization Beyond Sentence Extraction: A Probabilistic Approach to Sentence Compression. Artificial Intelligence, vol. **139**(1) (2002) 91–107
27. http://jazzy.sourceforge.net/
28. Patwardhan, S., Pedersen, T.: Using WordNet Based Context Vectors to Estimate the Semantic Relatedness of Concepts. Proceedings of the EACL Workshop Making Sense of Sense - Bringing Computational Linguistics and Psycholinguistics Together. Trento, Italy (2006) 1–8
29. Coxhead, A.: A new academic word list. TESOL Quarterly, vol. **34** (2) (2000) 213–238
30. http://opennlp.sourceforge.net/
31. Landis, J. R., Koch, G. G.: The Measurement of Observer Agreement for Categorical Data. Biometrics, vol. **33**, no. **1** (1977) 159–174

Generating and Evaluating Object-Oriented Designs for Instructors and Novice Students

Sally Moritz and Glenn Blank[1]
Computer Science and Engineering Department
Lehigh University
Bethlehem, PA 18015 USA
001-610-758-4085, 001-610-758-4867
sally.moritz@gmail.com, glennblank@gmail.com

Abstract. Creating object-oriented designs for even simple problems of the type assigned to beginning students is a challenging task. While rules can be applied to identify key design components, multiple acceptable solutions often exist, and instructors may have different preferences for which variation(s) are best. We describe a tool we developed to assist instructors in designing solutions for problems to be assigned to novice CS1 students. The tool generates a design template of acceptable solutions which the instructor may modify and add comments. Given an updated template, the Expert Evaluator component of DesignFirst-ITS assists novice students as they design a class diagram that models the assigned problem.

Keywords: intelligent tutoring system, unified modeling language, object-oriented design, CS1, natural language parsing.

1 Introduction

Learning object-oriented design and programming is a challenging task for beginning students. Data collected from CS1 courses in numerous studies [9, 14] have shown that, in spite of new curricula and tools that focus on object concepts, many students still have difficulty understanding and applying these concepts to solve problems. High school teachers find gaining the necessary mastery of the subject challenging. Reaching high school students is a must if we are to attract able students to the field.

Our approach introduces the design of UML (Unified Modeling Language) very early in the CS1 curriculum. In the "design-first" curriculum [11], students learn how to create a class diagram that represents the solution for a problem. Students learn procedural code only after they generate code stubs from the class diagram, as they implement each design element. Many tools generate code stubs from class diagrams. But students often need guidance as they translate a problem description into a design. We developed DesignFirst-ITS, an intelligent tutoring system (ITS) that observes students as they create UML class diagrams and offers assistance when they need it.

[1] This work was supported by the National Science Foundation under Grant No. EIA-02317687 and the Pennsylvania Infrastructure Technology Association (PITA).

This paper focuses on two components of DesignFirst-ITS: 1) an Instructor Tool and Solution Generator, which let an instructor enter a problem description in English, then generates a solution template, and 2) an Expert Evaluator, which compares a student's actions to the template. The solution template represents class diagrams that model multiple valid designs for the problem. Through a web-based front end, instructors can modify the template and add comments to be displayed to students in DesignFirst-ITS. The Instructor Tool allows multiple valid design variations, incorporating instructor preferences into the final model of acceptable solutions as well as feedback in the instructor's own voice.

Object-oriented design is an ill-structured domain in terms of the characteristics described in [7]: 1) it lacks a definitive solution, since many possible solutions are possible (it is so unusual that when students produce the same or too similar solutions it smacks of plagiarism); 2) the solution depends on the concepts that underlie the problem; and 3) problem solving requires both retrieving relevant concepts and mapping them to the task at hand. Furthermore, object-oriented design involves open-textured, abstract concepts, in the sense also described in [7], such as class, method and parameter and the way these concepts interact in relationships such as when to use an attribute rather than a parameter [15]. Both the Expert Evaluator and Solution Generator interpret free form input using NLP technology and domain specific expertise.

2 Related Work

Like DesignFirst-ITS, Collect-UML helps students who create class diagrams from problem description [3]. When a student adds a design element to her diagram, she selects a name by highlighting a word or phrase in the problem description. No free-form entry of element names is allowed. This approach avoids the need to understand the student's intent from natural language terms. A pedagogical drawback of this approach is that the student must eventually learn how to transfer to a less structured problem solving environment. DesignFirst-ITS, on the other hand, lets a student use natural language terms and abbreviations to develop her own UML class diagram.

Collect-UML evaluates student solutions by applying a set of constraints which must hold. Feedback text that explains the error in general terms is stored with each constraint. Since constraints do not model problem-solving knowledge, Collect-UML offers feedback offered only when a student submits a complete or partial solution, which may be too late to identify the flaw in the student's process. Constraint Based Modeling researchers have shown that a system can include constraints to be applied at intermediate steps in the problem-solving process; the NORMIT tutor uses this technique [9]. However, this approach cannot easily provide feedback after each solution step or suggest a next step to a student who is stuck. We believe that modeling the problem-solving process is crucial to help novices.

We studied software engineering research to define a process for translating a problem description to an object-oriented design. A long popular technique has been to identify potential classes, attributes and methods using parts-of-speech analysis. Abbott defined a model for identifying parts of speech in a problem specification and

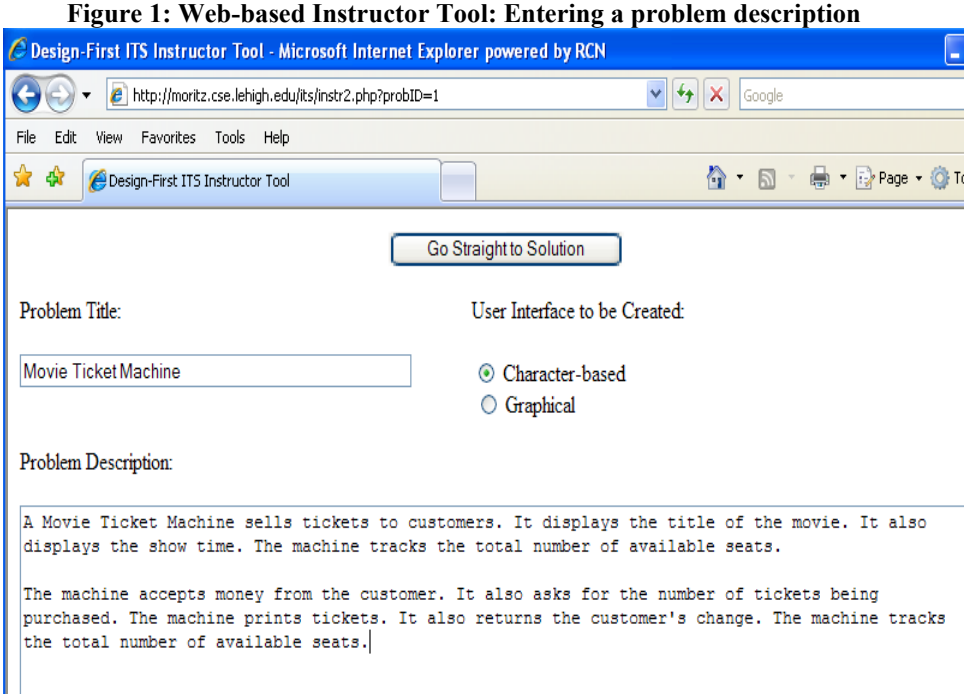
using them to derive the components of an Ada program [1]. Booch suggested that parts of speech, such as nouns and verbs, were often good candidates for design elements such as objects and methods [4]. Object Modeling Technique [14] defined a process to build a list of candidate classes from the nouns in a problem description and apply a set of rules to identify those that did not belong. It went on to define how to identify attributes and methods for the remaining classes.

These ideas have been used in automated tools to assist experienced analysts and developers in the design process. Examples of such tools include Circe [2], LIDA [11] and Metafor [6]. Although used as brainstorming tools, none were applied to educating students in learning object-oriented design.

3 The Instructor Tool and Solution Generator

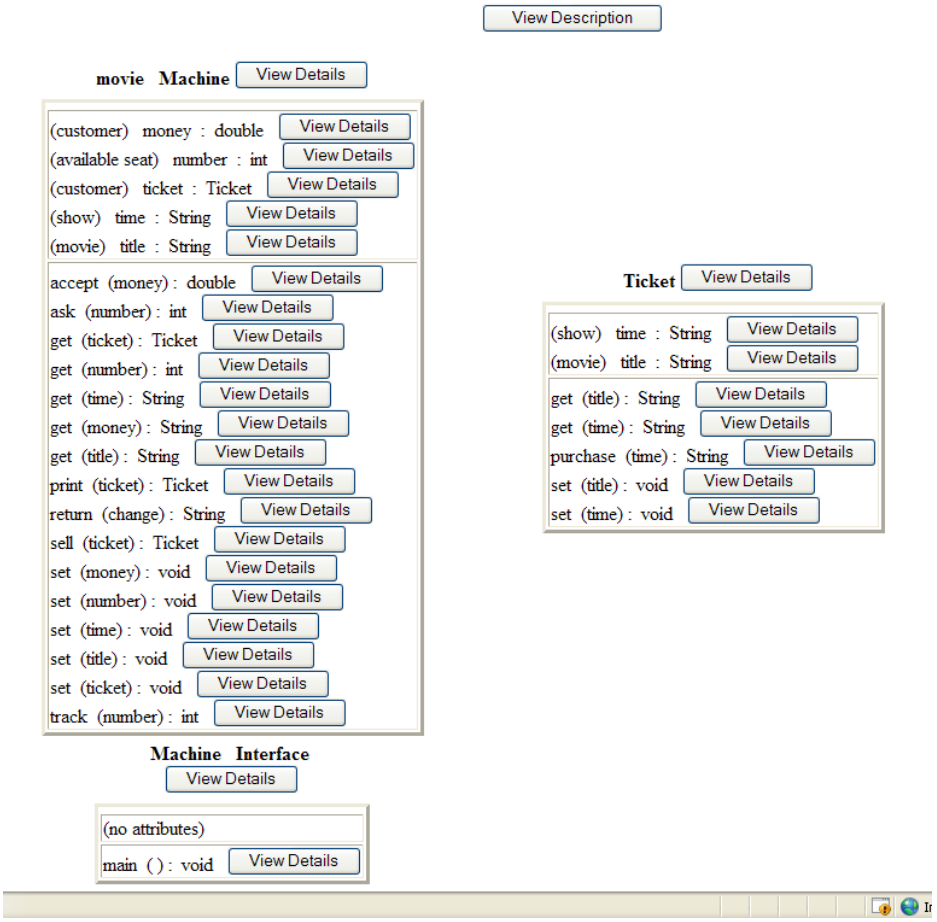
The Instructor Tool lets teachers enter problem descriptions in English, as in figure 1:

Figure 1: Web-based Instructor Tool: Entering a problem description



The Solution Generator then automatically generates class diagrams as potential solutions. The Instructor Tool then displays the results and lets the teacher view details (the source of each method or attribute in the problem description) and delete elements (see figure 2). The Tool aids the teacher in creating clear and concise problem descriptions for students. It also allows her to revise the solution based on her preferences, and to add her own voice to the ITS by entering comments and explanations to be used as feedback to the student.

Figure 2: Instructor Tool displays a solution template



The Solution Generator incorporates two external software packages: the MontyLingua natural language processor [6] and the WordNet lexical database [5]. MontyLingua identifies verb/subject/object tuples, which are in turn the basis for extracting classes and their attributes and methods. It also performs part-of-speech tagging, which the Solution Generator uses to determine the role each word should play in a design. The parts of speech are also passed to WordNet to find the definition that matches the word's use within the text. Natural language processing is difficult, and the current state of the art still has shortcomings. Thus it is not surprising to find a number of limitations in MontyLingua. The most notable are compound clauses, dependent clauses and appositives. Rules for structuring the problem text so as to avoid these shortcomings are outlined in an Instructor Tool User Manual.

WordNet provides services similar to a dictionary and thesaurus. It groups nouns, verbs, adjectives and adverbs into synonym sets. The Solution Generator uses WordNet to identify semantic information, such as nouns that are Actors, by

searching for phrases in a WordNet definition that indicate a person, such as "someone who…" or "a person who…" When multiple definitions exist for a word, the Instructor Tool chooses the most common use; the teacher can select a different definition and regenerate the solution from the problem description based on the new definition. WordNet also identifies synonyms within the context of the problem, which the Instructor Tool suggests as synonyms to the teacher as she reviews the generated solutions. The synonyms that are accepted by the instructor are saved as "related terms" for the component in the solution template. The Expert Evaluator compares a student entry against these related terms when looking for a match in a solution template. JWordNet, a Java-based WordNet implementation developed at George Washington University by Kunal Johar, was used to integrate WordNet into the Instructor Tool (www.seas.gwu.edu/~simhaweb/software/jwordnet/).

After using MontyLingua to parse tuples from text, the Solution Generator algorithm tags sentence subjects and objects as actors, attributes (based on the WordNet definition, pluse a database of simple values, such as money, amount, balance), methods (if a noun that is a verb form) and potential classes. Verbs are tagged as potential methods. Classes for which no attributes or methods are defined are removed. It adds optional get and set methods for each attribute that doesn't already have one plus standard classes for character-based or graphical user interface.

The generated design typically has more attributes and methods than required. The teacher can delete such elements with the Instructor Tool, providing explanations why they are not appropriate. The instructor can mark an element optional, supplying comments explaining why it is allowed but not optimal. Thus the Instructor Tool can extend variability and feedback beyond what the Solution Generator provides.

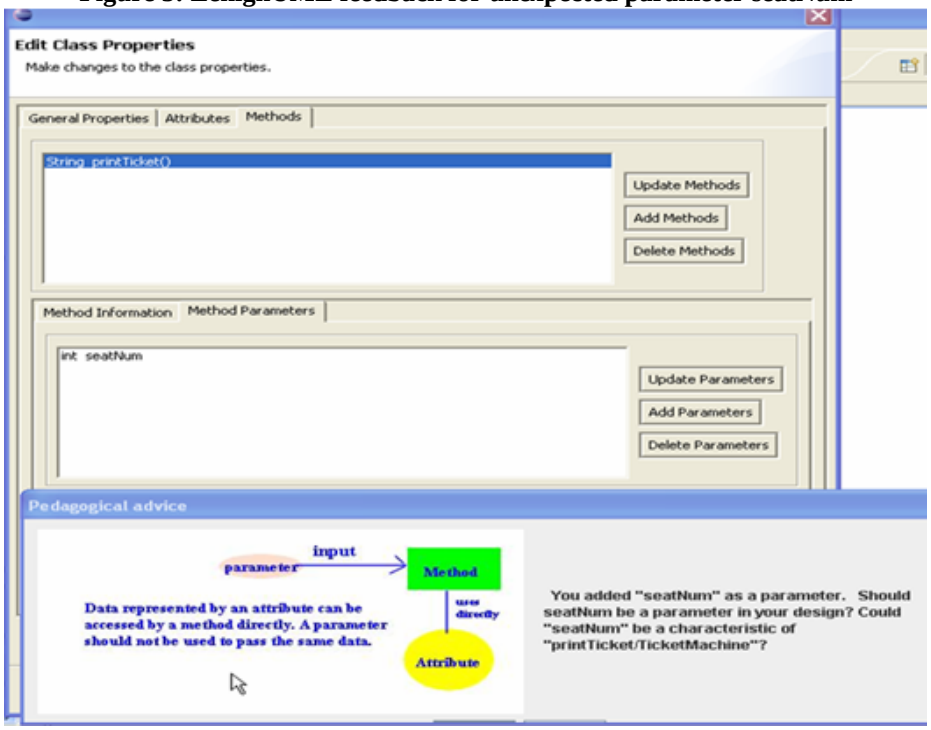
4 The Expert Evaluator

DesignFirst-ITS provides a relatively unstructured environment that allows students freedom to build their designs. Students design classes, methods and attributes in the LehighUML editor, which we developed as a plug-in for the Eclipse integrated development environment. (A stand-alone version of LehighUML has also been created, for use outside of the complex Eclipse environment.) As the student designs a solution for a given problem, LehighUML reports each student action to a database on a server. DesignFirst-ITS analyzes student actions and provides feedback to the student as necessary.

The Expert Evaluator (EE) evaluates each of the student's steps in the background by comparing it with the solution template, and generates an information packet for a correct student action and an error packet for an incorrect action. The packet also contains the concept applied, the type of error committed, and a recommended alternative to the incorrect action. The Student Model analyzes these packets to determine the knowledge level of the student for each concept and attempts to find reasons for student errors [15, 16]. The Student Model updates the student profile and passes the EE's packets along with a reason packet that contains possible reasons for the student's error to the Pedagogical Advisor [12]. In Figure 3, a student enters a

method printTicket() with an unexpected parameter seatNum. The Expert Evaluator diagnoses a possible error and the Pedagogical Advisor provides visual feedback.

Figure 3: LehighUML feedback for unexpected parameter seatNum



The EE uses the following algorithm to match the student's component name to a name in the solution template. First, it compares the student's component with each solution component and its related terms. A solution component name may be all or part of the student's name. Abbreviations and spelling errors for each term are considered by applying two string similarity metrics from the SimMetrics open source algorithms (www.dcs.shef.ac.uk/~sam/stringmetrics.html). If none exceed the similarity criteria, there is no match. If there is more than one match, then we consider adjectives to break ties; for example, distinguishing between attributes such as "customer balance" and "total balance." If the student's element matches an expert's deleted component, mis-matches a components (e.g., a student's attribute matches an expert's method) or if it has the wrong data or return type, generate an error packet; otherwise generate an info packet. If the student element does not match anything, generate an unknown element error packet.

Consider an example scenario. Suppose a student has successfully added a class she calls TicketMachine. She has also added valid attributes movie and price. Now she adds an attribute cash with data type int to the class TicketMachine. The EE attempts to match cash to attributes of TicketMachine not yet matched, which now includes (customer) money. It then compares cash to money and its related terms. WordNet did not return cash as a synonym when the Solution Generator added the attribute money. So what happens next depends on whether the instructor entered cash

as a related term of money. If cash does match, then the EE makes sure that it is not a duplicate in the student's complete solution. Then it compares the data type for the matched element (in this case double) to the student's data type (int). In this case, int is not listed as an acceptable data type, so the EE creates an error packet with the code INC_DATATYPE. On the other hand, if cash does not match as a related term, then the EE compares cash to attributes for other classes, then to other classes, parameters, actors and methods. If there were a match, an error packet would indicate any match that is found (there is none); otherwise, an error packet indicates an unknown element.

5 Evaluation

Evaluation of student learning with DesignFirst-ITS is reported in [16]. In summary, in a study of 42 high school students, there is a significant gain ($p<.001$) from pre-test to post-test for students getting feedback from DesignFirst-ITS. The rest of this section focuses on the Instructor Tool, Solution Generator and Expert Evaluator.

Five computer science teachers evaluated the Instructor Tool and Solution Generator. The experts were given the Instructor Tool User Manual, an overview of DesignFirst-ITS and the Instructor Tool, rules for writing a problem description and examples. The evaluators then each entered one or two problems of their own invention, using the tool to refine the description and the solution until it met with their own satisfaction. They then completed an interview and questionnaire.

The teacher evaluations of the Solution Generator focused on how well the software's interpretation of the problem description matched the teacher's intent and the quality of the resulting class diagram. The problem descriptions they entered provided examples of different types of problems as well as different language styles, which was invaluable in measuring and improving the Solution Generator's ability to interpret a range of voices and perspectives.

The teachers listed specific difficulties or bugs they found in solution generation. These most frequently listed had to do with MontyLingua's sentence parsing. Sometimes MontyLingua correctly parsed subordinate clauses such as "The team that accumulates the most points wins" but sometimes it did not. The most common result was that the information contained in the subordinate clause was dropped. The teachers were able to work around the problem by rewording their text. However, avoiding subordinate clauses completely is not always possible, even for the most simple subjects. Natural language parsing is where the most improvements can be made as better software becomes available. A second source for bugs was pronoun resolution: the simple strategy of using the subject of the prior sentence as a pronoun's antecedent worked well except for one case in which the antecedent was taken from the subject of a subordinate clause. A third source of difficulties was interpretation or word meanings. In one case, WordNet did not return a valid definition. This happened for the word "alarm" – a simple word which one would expect to be found even in an abridged version of the WordNet database. WordNet matched another word. One remedy is to download a larger version of the WordNet database; another remedy is that the instructor can add terms with definitions and related terms. In a second case, WordNet returned a set of suitable definitions for the

word, but the most common definition was not appropriate for the context. This occurred in a problem description of a basketball game: "The team to accumulate the most points wins." WordNet's most frequently used sense for point is "A geometric element that has position but no extension." But the appropriate definition in this context is "The unit of counting in scoring a game or contest." This difficulty isn't truly a bug; understanding the context of the problem is beyond the scope of the Solution Generator. In this case, the instructor was able to choose the correct sense for the word and regenerate the related terms, adjectives and data type for the element.

The bugs detected represented a small portion of the teachers' text. They did not prevent successful use of the tool, as all of the evaluators were able to create an acceptable solution. The evaluators all rated the quality of the solution as a 4 on a scale of 1 to 5 (1 being poor, 5 being excellent), and all said the solution generation met or exceeded their expectations. One commented that learning how to properly word the problems for optimal interpretation would take some time, but the learning curve was not too cumbersome. Another suggested adding other commonly used methods, such as toString and equals. They observed that the tool encouraged instructors to think more deeply about how the problem should be worded so that students have a clear and complete understanding and that the tool created a better solution than the teacher had initially developed.

The EE was tested with 33 students in a CS1 course at Lehigh University in November 2006. These students attended an optional workshop in which they used the DesignFirst-ITS to create a class diagram for the Movie Ticket Machine problem. Every action taken by each student was recorded in the ITS database.

Table 1: Expert Evaluator Results

Number of Actions	Number EE Marked Correct	Number EE Marked Correct in Error	Number EE Marked Incorrect	Number EE Marked Incorrect in Error	Number EE Marked Unknown
1035	540	16	454	79	41

For 1035 student actions, the EE had 16 false positives (3%), 79 false negatives (17%) and 41 unknowns (4%), for an overall error rate of 13%. An accuracy rate of 87% is quite high and acceptable for the overall performance of DesignFirst-ITS.

The EE marks an action as unknown when it cannot match the action to any component within the solution template. In this population of data, this included the use of words not anticipated by WordNet or the instructor ("pay" for an "enter money" method, "dollars" for "money"); the use of single-letter or nonsense component names ("p", "asdf"); and the use of abbreviations not recognized by the string similarity metrics ("tix" for "tickets").

When the EE identified an action as correct, it was right 97% of the time; only 3% of the actions marked correct by the EE were not. This makes sense; most matches were on terms taken directly from the problem description. All of the instances in which actions were erroneously marked correct involved ambiguous names used by the student. "Display" or "enter" are only partial method names; they should indicate what is to be displayed or entered. When the EE encounters ambiguous names, it tries

to match on data type or return type. If that does not resolve the ambiguity, it chooses a match at random. A better strategy may be to mark the action unknown to allow the Pedagogical Advisor to ask the student for clarification.

When the EE identified an action as incorrect, 17% of the time the action should have been accepted as correct. There are two main causes of an incorrect diagnosis: a misinterpretation of the student's action, or a failure to recognize an acceptable solution that the action represents. There were only a few unique errors that occurred within this population of students; those same errors occurred many times, and in some cases triggered other errors (such as when an error with an attribute leads to errors interpreting methods involving the attribute).

All of the misinterpretation errors occurred with components that had very similar names or were similar in purpose. For example, there was some confusion over "number of available seats/tickets" as an attribute and the number of tickets requested by the customer. These are difficult even for a human instructor to resolve, especially when students would need to use long names to distinguish between them. This trickled down to confusion with methods similar to "get" and "set" functions for these attributes—such as "enterNumberTickets," which the student may have intended as a method to request the number of tickets the customer wants rather than set the attribute representing the number of tickets available for sale. A better way to handle these cases might be to mark such components as unknown and allow the Pedagogical Advisor to ask the student for clarification. The EE can store the closest match in the recommended action fields, so the Advisor can ask "Did you mean…?"

There were three distinct errors classified as failure to recognize alternate solutions. One involved the return type of the "print tickets" method. Fourteen students coded String as its return type, which was not allowed as an optional value in the solution template. In this case the teacher who taught the class had not prepared the solution template in the Instructor Tool, and the question of what data is returned from the print ticket method is wide open to interpretation. It is the instructor's prerogative to allow more or less flexibility in the finer details of a solution.

A second variation entered by one student includes two attributes related to the number of seats in the theater. A student had "seatsAvailable" and "seatsRemain" as attributes; the EE marked the second as a duplicate of the first. An instructor might agree that there is no need for two attributes. Or an instructor might want to accept both attributes, and could do so by adding wording to the description such as "The machine keeps track of the total number of seats in the theater, and the number of seats remaining." Thus using the tool, an instructor can add more verbal and structural variability than the algorithm automatically generates.

Overall, the error rate in the data collected does not significantly hinder the useful operation of the ITS. The error rate presented could be improved by using the Instructor Tool to accept simple variations and synonyms that were rejected. In actual classroom use, this would be a routine and expected task for the instructor. There were four simple changes that would reasonably be made for the Ticket Machine problem: 1) adding alternate return data types for the printTickets method; 2) adding "tix" as a synonym for "tickets"; 3) adding "cash" as a synonym for "money"; 4) adding "dollars" as a synonym for "money." The first change alone would yield 14 fewer errors; the second 7, the third and fourth one each. The result for the input student actions would be 56 actions marked in error incorrectly, instead of 79. These

changes would improve the overall error rate to 11%, or an 89% accuracy rate, including unknowns (93% accuracy excluding unknowns).

A simple revision to the EE's logic that would also improve its accuracy would be to pass ambiguous elements to the Pedagogical Advisor for clarification instead of making best guess matches. This should not be too annoying for the student. Indeed, asking the student for clarification more often might be beneficial in inducing the student to think more deeply about her design elements, and thus improve learning.

A richer enhancement would create a feedback loop in which the tutor learns from student work. That way the Instructor Tool could alter its model to allow for acceptable solutions that the instructor did not anticipate. Such solutions might involve adding additional elements or modifying required elements to optional. An interesting question is whether the instructor must remain in this loop or a machine learning algorithm could recognize novel student solutions.

6 Conclusions

The research summarized in this paper applied software engineering principles to develop tools that assist both students and teachers in learning object-oriented design. The Instructor Tool analyzes problem descriptions in English and generates a class diagram representing a design template that supports multiple acceptable solution variations. Expert instructors found that the tool generated valid design elements, including some which they had not anticipated. Experts rated both the tool's ease of use and completeness of functionality as a 4 on a scale of 1 to 5.

The EE analyzes student actions with an accuracy of 87%. DesignFirst-ITS lets students solve problems naturally, using their own terms in their UML design. Students are unencumbered by scaffolding they don't need; students who have difficulties are provided with assistance as they need it. Students are not locked into a single solution; multiple solutions that are represented in the problem description are recognized. Feedback on student errors is relevant to the design process, since recommended actions are based on the Solution Generator's analysis.

The tools, curricula and dissertation are available at designfirst.cse.lehigh.edu.

References

1. Abbott, R.: Program Design by Informal English Descriptions. In: Communications of the ACM, November 1983, pp. 882-894. (1983)
2. Ambriola, V., Gervasi, V.: Processing Natural Language Requirements. In: Proceedings of the 12th International Conference on Automated Software Engineering, pp. 36-45 (1997)
3. Baghaei, N., Mitrovic, A., & Irwin, W.: Problem-Solving Support in a Constraint-based Intelligent System for Unified Modeling Language. In: Technology, Instruction, Cognition and Learning Journal, Vol 4, No 1-2 (2006)
4. Booch, G.: Object-Oriented Development. In: IEEE Trans. on Software Engineering, 12(2), pp. 211-221 (1986)
5. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)

6. Liu, H., ieberman, H.: Metafor: Visualizing Stories as Code. Proceedings of the ACM International Conference on Intelligent User Interfaces, pp. 305-307 (2005)
7. Lynch, C., Ashley, K., Aleven, V., Pinkwart, N: Defining "Ill-Defined Domains"; A literature survey. In: Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at 8th International Conference on Intelligent Tutoring Systems (2006).
8. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagen, D., Kolikant, Y., Laxer, C., Thomas, L., Utting, I., Wilusz, T.: A Multi-National Study of Assessment of Programming Skills of First Year CS Students. In: SIGCSE Bulletin 33(4), pp. 125-140 (2001)
9. Mitrovic, A., Martin, B., & Suraweera, P.: Intelligent Tutors for All: The Constraint-Based Approach. In: Intelligent Systems: IEEE, Vol. 22, No. 4, pp. 38-45 (2007)
10. Moritz, S., Blank, G.: A Design-First Curriculum for Teaching Java in a CS1 Course. In: ACM SIGCSE Bulletin (inroads), June 2005, pp. 89-93 (2005)
11. Overmyer S.P., Lavoie B., & Rambow O.: Conceptual Modeling through Linguistic Analysis Using LIDA. In: Proceedings of the 23rd International Conference on Software Engineering (ICSE 2001), pp. 401-410 (2001)
12. Parvez, S., Blank, G.: Individualizing Tutoring with Learning Style Based Feedback. In: Proc. 9th International Conference on Intelligent Tutoring Systems (2008)
13. Ratcliffe, M., Thomas, L.: Improving the Teaching of Introductory Programming by Assisting Strugglers. In: Proc: 33rd ACM SIGCSE Technical Symposium (2002)
14. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W.: Object-Oriented Modeling and Design. Prentice Hall (1991)
15. Wei, F, Blank, G.: Student Modeling with Atomic Bayesian Networks. In: Proc. of 8th International Conference on Intelligent Tutoring Systems, pp. 491-502 (2006)
16. Wei, F., Blank, G.: Atomic Dynamic Bayesian Networks for a Responsive Student Model. In Proc. 13th International Conference on Artificial Intelligence in Education (2007)

A Sequential Pattern Mining Algorithm for Extracting Partial Problem Spaces from Logged User Interactions

Philippe Fournier-Viger[1], Roger Nkambou[1] and Engelbert Mephu Nguifo[2]

[1]University of Quebec at Montreal (Canada),
[2]Université Lille-Nord de France
fournier_viger.philippe@courrier.uqam.ca

Abstract. Domain experts should provide relevant domain knowledge to an Intelligent Tutoring System (ITS). For many ill-defined domains, the domain knowledge is hard to define explicitly. In previous works, we showed how sequential pattern mining can be used to extract a partial problem space from logged user interactions, and how it can support tutoring services during problem-solving exercises. This article describes our current work where we propose an extended sequential pattern mining algorithm to extract a problem space that is richer and more adapted for supporting tutoring services.

1 Introduction

Domain experts should provide relevant domain knowledge to an Intelligent Tutoring System (ITS) so that it can guide a learner during problem-solving learning activities. One common way of acquiring such knowledge is to observe experts and novices, to capture different ways of solving problems. However, this process is time-consuming and it is not always possible to define a satisfying complete or partial problem-space, in particular when a problem is ill-structured. According to Simon [2], an ill-structured problem is one that is complex, with indefinite starting points, multiple and arguable solutions, or unclear strategies for finding solutions. Domains that include such problems and in which, tutoring targets the development of problem-solving skills are said to be ill-defined (within the meaning of Ashley et al. [3]).

To address the problem of domain knowledge acquisition in procedural and ill-defined domains, we recently proposed a framework for learning a partial problem space from user's interactions [1]. This framework takes as input sequences of user actions performed by expert, intermediate and novice users, and consists of applying two knowledge discovery techniques. First, sequential pattern mining (SPM) is used to discover frequent action sequences. Second, association rules discovery find associations between these significant action sequences, relating them together. This framework was applied in the RomanTutor tutoring system [4] to extract a partial problem space that is used to guide users, and thus showed to be a viable alternative to the specification of a problem-space by hand for the same domain [1,5]. The proposed framework differs from other works that attempt to construct a problem space from logged student interactions such as [6] and [7], since these latter are devoid of learning, reducing these approaches to simple ways of storing or integrating raw user solutions into structures. In this paper, we discuss limitations of our initial

framework and present an extended SPM algorithm to extract a problem space that is richer and more adapted for supporting tutoring services. The paper first introduces RomanTutor and the problem of SPM from user actions. Then it presents extensions to SPM, preliminary results, future work and a conclusion.

2 The RomanTutor Tutoring System

RomanTutor [2] is a simulation-based tutoring system to teach astronauts how to operate Canadarm2, a 7 degrees of freedom robotic arm deployed on the International Space Station. The main exercise in RomanTutor is to move the arm to a goal configuration. To perform this task, an operator must select at every moment the best cameras for viewing the scene of operation among several cameras mounted on the manipulator and on the space station. In a previous work, we attempted to model the Canadarm2 manipulation task with a rule-based knowledge representation model [5]. Although, we described high-level rules such as to set the parameters of cameras in a given order, it was not possible to model how to rotate arm joint(s) to attain a goal configuration. The reason is that there are many possibilities for moving the robot to a goal configuration, and there is no simple "legal move generator" for finding all these possibilities. As a solution, in our previous work [1], we identified 155 actions that a learner can take, which are (1) selecting a camera, (2) performing a small/medium/big increase or decrease of the pan/tilt/zoom of a camera and (3) applying a small/medium/big positive/negative rotation value to an arm joint. Then, we applied SPM to mine frequent action sequences from logged users' interactions. The resulting knowledge base served principally in RomanTutor to track the patterns that a learner follows, and to suggest the next most probable actions that one should execute.

3 Sequential Patterns Mining from User Actions

The problem of mining sequential patterns is stated as follows [8]. Let D be a transactional database containing a set of transactions (here also called plans) and a set of sequence of items (here called actions). An example of D is depicted in figure 1.a. Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of actions. We call a subset $X \subseteq A$ an actionset and $|X|$, its size. Each actions in an actionset (enclosed by curly brackets) are considered simultaneous. A sequence $s = (X_1, X_2, \ldots, X_m)$ is an ordered list of actionsets, where $X_i \subseteq A$, $i \in \{1,\ldots,m\}$, and where m is the size of s (also noted $|s|$). A sequence $s_a = (A_1, A_2, \ldots, A_n)$ is contained in another sequence $s_b = (B_1, B_2, \ldots, B_m)$ if there exists integers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ such that $A_1 \subseteq B_{i1}$, $A_2 \subseteq B_{i2}$, . . . , $A_n \subseteq B_{in}$. The relative support of a sequence s_a is defined as the percentage of sequences $s \in D$ that contains s_a, and is denoted by $supD(s_a)$. The problem of mining sequential patterns is to find all the sequences s_a such that $supD(s_a) \geq$ *minsup* for a database D, given a support threshold *minsup*.

Consider the dataset of figure 1.a. The size of the plan 2 is 6. Suppose we want to find the support of S2. From figure 1.a, we know that S1 is contained in plan 1, 2 and 6. Hence, its support is 3 (out of a possible 6), or 0.50. If the user-defined minimum

support value is less than 0.50, then S1 is deemed frequent. To mine sequential patterns several algorithms have been proposed [8], [9]. Initially [1], we chose PrefixSpan [9] as it is one of the most promising approach for mining large sequence databases having numerous patterns and/or long patterns, and also because it can be extended to mine sequential patterns with user-specified constraints. Figure 1.b shows some sequential patterns extracted by PrefixSpan from the data in figure 1.a using a minimum support of 25%. In RomanTutor, one mined pattern is to select the camera 6, which gives a close view of the arm in its initial position, slightly decrease the yaw of camera 6, select the elbow joint and decrease a little bit its rotation value. Although, the set of patterns extracted for RomanTutor constitute a useful problem space that capture different ways of solving problems, we present next the limitations of SPM encountered, and extensions to PrefixSpan to address these issues.

ID	Sequences of actions
1	1 2 25 46 48 {9 10 11 31}
2	1 25 46 54 {10 11 25} 48
3	1 2 3 {9 10 11 31} 48
4	2 3 25 46 11 {14 15 48} 74
5	4 1 25 27 46 48
6	1 3 44 45 46 48

→

ID	Seq. patterns	Support
S1	1 46 48	66 %
S2	1 25 46 48	50 %
S3	1 25 46 {10 11}	33 %
S4	1 {9 10 31}	33 %
S5	1 {9 11 31}	33 %
…	…	…

Fig. 1. (a) A Data Set of 6 Plans (b) Example of Sequential Patterns Extracted

4 Extending Sequential Pattern Mining with Time Intervals

A first limitation that we encountered is that extracted patterns often contain "gaps" with respect to their containing sequences. For instance, in the example of figure 1, action "2" of plan 1 has not been kept in S1. A gap of a few actions is ok in a tutoring context because it eliminates non-frequent learners' actions. But when a sequence contain many or large gap(s), it becomes difficult to use this sequence to track a learner's actions and to suggest a next relevant step. Thus, there should be a way of limiting the size of the gaps in mined sequences. Another concern is that some patterns are too short to be useful in a tutoring context (for example, sequences of size 1). In fact, there should be a way of specifying a minimum sequential pattern size.

An extension of SPM that overcomes these limitations is to mine patterns from a database with time information. A time-extended database is defined as a set of time-extended sequences $s = <(t_1,X_1), (t_2,X_2),\ldots, (t_n,X_n)>$, where each actionset X_x is annotated with a timestamp t_x. Each timestamp represents the time elapsed since the first actionset of the sequence. Actions within a same actionset are considered simultaneous. For example, one time-extended sequence could be <(0, a), (1, b c), (2, d)>, where action *d* was done one time unit after *b* and *c*, and two time units after *a*. The time interval between two actionsets (t_x,X_x) and (t_y,X_y) is calculated as $|t_x - t_y|$. In this work, we suppose a time interval of one time unit between any adjacent actionsets, so that $|t_x - t_y|$ become a measure of the number of actionsets between (t_x, X_x) and (t_y, X_y). The problem of Generalized Sequential Pattern Mining with Time Intervals (GSPM) [10] is to extract all time-extended sequences s from a time-

extended database, such that supD(s) ≥ minsup and that s respect all time constraints. Four types of constraints are proposed by Hirate and Yamana [10]. The constraints C1 and C2 are the minimum and maximum time interval required between two adjacent actionsets of a sequence (gap size). The constraints C3 and C4 are the minimum and maximum time interval required between the head and tail of a sequence. For example, for the sequence <(0, a), (1, b c), (2, d)>, the time interval between the head and the tail is 2 and the time interval between the actionset (0, a) and (1, b c) is 1.

Hirate and Yamana [10] have proposed an extension of the PrefixSpan algorithm for the problem of GPSM. We present it below –with slight modifications- as it is the basis of our work. The algorithm finds all frequent time-extended sequences in a database ISDB that respect *minsup*, C1, C2, C3 and C4, by performing a depth-first search. The algorithm is based on the property that if a sequence is not frequent, any sequence containing that sequence will not be frequent. The algorithm proceeds by recursively projecting a database into a set of smaller projected databases. This process allows growing patterns action by action by finding locally frequents actions.

In the following, the notation ISDB|(t,i) represents the time-extended database resulting from the operation of projecting a time-extended database ISDB with a pair (timestamp, item). ISDB|(t, i) is calculated as follow.

```
ISDB((t,i))
  ISDB|(t,i) := Ø.
  FOR each sequence σ=<(t1,X1),(t2,X2)…(tn,Xn)> of ISDB.
  FOR each actionset (tx,Xx) of σ containing i.
    IF Xx/{i} = Ø
      s :=<(tx+1-tx,ax+1), … (tn-tx,Xn)>
    ELSE
      s :=<(0, Xx/{i}), (tx+1-tx, ax+1), … (tn-tx, Xn)>
    IF s ≠ Ø  and s satisfies C1, C2 and C4
      Add s to ISDB|(t,i).
  Return ISDB|(t,i).
```

The Hirate-Yamana algorithm (described below) discovers all frequent time-extended sequences.

```
algoHirate(ISDB, minsup, C1, C2, C3, C4)
  R := ø.
  Scan ISDB and find all frequent items with support
   higher than minsup.
  FOR each frequent item i,
    Add (0, i) to R.
    algoProjection(ISDB|(0, i), R, minsup,C1,C2,C3,C4).
  RETURN R;

  algoProjection(ISDB|prefix, R, minsup, C1, C2, C3, C4)
    Scan ISDB|prefix to find all pairs of item and
     timestamp, denoted (t, i) satisfying minsup, C1 and C2.
    FOR each pair (t, i) found
      newPrefix := Concatenate(prefix, (t, i)).
      IF newPrefix satisfies C3 and C4
        Add newPrefix to R.
        IF (size of ISDB|newPrefix) >= minsup
         algoProjection(ISDB|newPrefix, R, minsup,C1,C2 C3,C4).
```

To illustrate the Hirate-Yamana algorithm, let's consider applying it to the database ISDB depicted in figure 2, with a *minsup* of 50 % and the constraint C2 equals to 2 time units. In the first part of *algoHirate*, frequent actions *a*, *b* and *c* are found. As a result <(0,a)>, <(0,b)> and <(0,c)> are added to *R*, the set of sequences found. Then, for *a*, *b* and *c*, the projected database ISDB|(0,a), ISDB|(0,b) and ISDB|(0,c) are created, respectively. For each of these databases, the algorithm *algoProjection* is executed. *algoProjection* first finds all frequent pairs (timestamp, item) that verify the constraints C1, C2 and *minsup*. For example, for ISDB| (0,a), the frequent pair (0,b) and (2,a) are found. These pairs are concatenated to (0,a) to obtain sequences <(0,a), (0,b)> and <(0,a), (2,a)>, respectively. Because these sequences respect C3 and C4, they are added to the set R of sequences found. Then, the projected database ISDB|<(0,a), (0,b)> and ISDB|<(0,a), (2,a)> are calculated. Because these databases contain more than *minsup* sequences, *algoProjection* is executed again. After completing the execution of the algorithm, the set of sequences R contains <(0,a)>, <(0,a), (0,b)>, <(0,a), (2,a)>, <(0,b)> and <(0,c)>.

5 Extending SPM with Automatic Clustering of Valued Actions

A second limitation that we encountered when applying PrefixSpan to extract a problem space is it relies on a finite set of actions. As a consequence, if some actions were designed to have parameters or values, they have to be defined as one or more distinct actions. For example, in our previous work in RomanTutor, we categorized the joint rotations as small, medium and big, which correspond respectively to 0 to 60, 60 to 100, and more than 140 degrees. The disadvantage with this categorization is that it is fixed. In order to have dynamic categories of actions, we extended the Hirate-Yamana algorithm to perform an automatic clustering of valued actions.

We propose to define a valued sequence database as a time-extended sequence database, where sequences can contain valued actions. A valued action *a{value}*, is defined as an action *a* that has a value *value*. In this work, we consider that a value is an integer. For example, the sequence <(0,a{2}), (1,b), (2,bc{4})> contains the valued action *a*, and *b* with values 2 and 4, respectively. The left part of figure 3 shows an example of a sequence database containing valued actions.

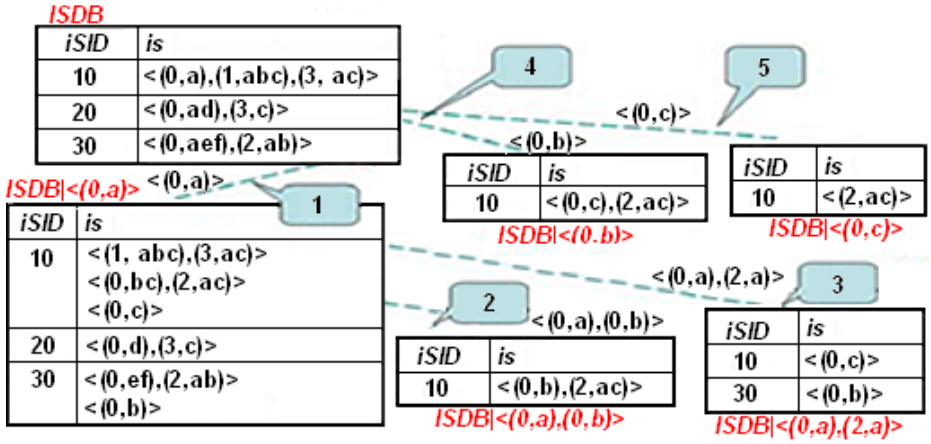


Fig. 2. An application of the Hirate-Yamana Algorithm (adapted from [10])

ID	Time-extended sequences
1	<(0,a{2}), (1,bc{4})>
2	<(0,a{2}), (1,c{5}))>
3	<(0,a{5}), (1,c{6}))>
4	< (0,f), (1,a{6}))>
5	<(0, f b{3}), (1,e) , (2,f))>
6	<(0,b{2}), (1,d))>

→

Mined valued seq. patterns	Supp.
<(0,a{2})>	33 %
<(0,a{5})>	33 %
<(0,a{2}), (1, c{5})>	33 %
<(0,c{5})>	50 %
<(0,f)>	33 %
…	…

Fig. 3. Applying Hirate-Yamana with Automatic Clustering of Valued Actions

To mine patterns from a valued database, we added a special treatment for valued actions. We modified the action/pair counting of the Hirate-Yamana algorithm to note the values of the action being counted, and their sequence ids. We modified the database projection operation ISDB|(t,i) so that it can be called with a valued action *i* and a set of values V={v_1, v_2…, v_n}. If the support of *i* in ISDB is higher or equals to 2 * *minsup*, the database projection operation calls the K-Means algorithm [11] to find clusters. The K-Means algorithm takes as parameter K, a number of clusters to be created, and the set of values V to be clustered. K-Means first creates K random clusters. Then it iteratively assigns each value from V to the cluster with the nearest median value until all clusters remain the same for two successive iterations. In the database projection operation, K-Means is executed several times starting with K=2, and incrementing K until the number of frequent clusters found (with size >= *minsup*) does not increase. This larger set of frequent clusters is kept. Then, the sequences of ISDB|(t,i) are separated into one or more databases according to these clusters. Afterward, *algoProjection* is called for each of these databases with size equal or greater than *minsup*.

Moreover, if ISDB|(t,i) is called from *algoHirate* and *n* clusters are found, instead of just adding <(o,{i})> to the set R of sequences found, <(0, i{v_{x1}})>, <(0, i{v_{x2}})> … <(0, i{v_{xn}})> are added, where v_{x1}, v_{x2}, .. v_{xn} are the median value of each cluster. Similarly, we have adapted *algoProjection* so that sequences are grown by executing Concatenate with (t, i{v_{x1}}), (t, i{v_{x2}}) … (t, i{v_{xn}}, instead of only <(t,{i})>.

The right part of figure 3 shows some sequences obtained from the execution of the modified algorithm with a *minsup* of 32 % (2 sequences) on the valued sequence database depicted in the left part of figure 3. The advantage of the modified algorithm over creating fixed action categories is that actions are automatically grouped together based on their similarity and that the median value is kept as an indication of the values grouped. Note that more information could be kept such as the min. and max. values for each cluster, and that a different clustering algorithm could be used.

A test with logs from RomanTutor permitted extracting sequential patterns containing valued actions. One such pattern indicates that learners performed a rotation of joint EP with a median value of 15º, followed by selecting camera 6. Another pattern found consists of applying a rotation of joint EP with a median value of 53º followed by the selection of the camera 4. In this case, the dynamic categorization enhanced the quality of the extracted patterns, since otherwise both patterns would be considered as starting with a "small rotation" of the joint EP.

6 Extending Sequential Pattern Mining with Context Information

A third limitation that we encountered when applying the PrefixSpan algorithm is that it does not consider the context of each sequence. In a tutoring system context, it would be useful, for instance, to annotate sequences with success information and the expertise level of a user and to mine patterns containing this information. Our solution to this issue is to add dimensional information to sequences. Pinto et al. [12] originally proposed Multi-dimensional Sequential Pattern Mining (MDSPM), as an extension to SPM. A Multidimensional-Database (MD-Database) is defined as a sequence database having a set of dimensions $D=\{D_1, D_2,\ldots D_n\}$. Each sequence of a MD-Database (an MD-Sequence) possesses a symbolic value for each dimension. This set of value is called an MD-Pattern and is noted $\{d_1, d_2\ldots d_n\}$. For example, consider the MD-Database depicted in the left part of figure 4. The MD-Sequence 1 has the MD-Pattern {"true", "novice"} for the dimensions "success" and "expertise level". The symbol "*", which means any values, can also be used in an MD-Pattern. This symbol subsumes all other dimension values. An MD-Pattern $P_x=\{d_{x1}, d_{x2}\ldots d_{xn}\}$ is said to be contained in another MD-Pattern $P_y=\{d_{y1}, d_{y2}\ldots d_{ym}\}$ if there exists integers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ such that $d_{x1} \subseteq d_{y1}$, $d_{x2} \subseteq d_{y2}$, . . . , $d_{xn} \subseteq d_{yn}$. The problem of MDSPM is to find all MD-Sequence appearing in a database with a support higher than *minsup*. Figure 4 shows an MD-Database with time information and some patterns that can be extracted from it, with a *minsup* of 2 sequences.

Pinto et al. [12] proposed three algorithms for MDSPM. The first one cannot be applied in combination with the Hirate-Yamana algorithm, as it required embedding dimensions as additional actions in sequences. The two other algorithms are similar. In our implementation, we chose SeqDim and integrated it with our extended Hirate-Yamana algorithm. SeqDim is executed as follow. First, frequent sequences are found by SPM. Then, for each sequence, the containing MD-Sequences are used to mine frequent MD-Patterns which are then combined with the sequence to form MD-Sequence(s). For MD-Pattern mining we chose the algorithm described in [12]. In our experiment with RomanTutor, MDSPM showed to be useful as it allowed to

successfully identify patterns common to all expertise level that lead to failure ("*, failure"), for example. Currently, we have encoded two dimensions: expertise level and success. But additional dimensions can be easily added. In future work, we plan to encode skills involved as dimensional information (each skill could be encoded as a dimension). This will allow computing a subset of skills that characterize a pattern by finding common skills demonstrated by users who used that pattern. This will allow diagnosing missing and misunderstanding skill for users who demonstrated a pattern.

7 Dividing Problems into Sub-problems

The last improvement that we made is to how we apply the algorithm to extract a partial problem space. Originally [1], we mined frequent patterns from sequences of user actions for a whole problem-solving exercise. But, we noticed that in general, after more than 6 actions performed by a learner, it becomes hard for the system to tell which pattern the learner is doing. For this reason, we added the definition of "problem states". For example, in the RomanTutor, where an exercise consists of moving a robotic arm to attain a specific arm configuration, the 3D space was divided into cubes, and a problem state is defined as the set of cubes containing the arm joints. An exercise is then viewed as going from a problem state P_1 to a problem state P_F. When a learner do an exercise, we log (1) the sequence of problem states visited by the learner $A=\{P_1, P_2 \ldots P_n\}$ and (2) the list of actions performed by the learner to go from each problem state to the next visited problem state (P_1 to P_2, P_2 to P_3, … P_{n-1} to P_n). After many users performed the same exercise, we extract sequential patterns from (1) the sequences of problems states visited, and (2) from the sequences of actions performed for going from a problem state to another.

Dividing long problems into sub-problems allow a better guidance of the learner, because at any moment, only the patterns starting from the current problem state have to be considered. We describe next how the main tutoring services are implemented. To recognize a learner's plan, the system proceeds as follow. The first action of the learner is compared with the first action of each pattern for the current problem state. The system discards the patterns that do not match. Each time the learner makes an action, the system compares the actions done so far by the learner with the remaining patterns. When the problem-state changes, the system considers the set of patterns associated to the new problem state. If at any given moment a user action does not match with any patterns, the algorithm ignores the last user action or the current action to match for each pattern. This makes the plan recognizing algorithm more flexible. At a more coarse grain level, a tracking of the problem states visited by the learners is achieved similarly as the tracking for actions. One utility of the plan recognizing algorithm for actions/problem states is to assess the expertise level of the learner (novice, intermediate or expert) by looking at the patterns applied. The plan recognizing algorithm also allows suggesting to the learner the possible actions from the current state. In this case, the tutoring service selects the action among the set of patterns that has the highest relative support and that is the most appropriate for the estimated expertise level of the learner. When no actions can be identified, RomanTutor relies on a path-planner [4] to generate an approximate path to the goal.

An MD-Database		
ID	**Dimensions**	**Sequences**
1	true, novice	<(0,a),(1,bc)>
2	true, expert	<(0,d) >
3	false, novice	<(0,a),(1,bc)>
4	false, interm.	<(0,a),(1,c), (2,d)>
5	true, novice	<(0,d), (1,c)>
6	true, expert	<(0,c), (1,d)

→

Mined MD-Sequences	
Dimensions	**Sequences**
*, novice,	<(0,a)>
*, *	<(0,a)>
*, novice	<(0,a), (1,b)>
true, *	<(0,d)>
true, novice	<(0,c)>
true, expert	<(0,d)>

Fig. 4. An Example of SPM with Dimensions and Time Information

8 A Preliminary Experiment

We conducted a preliminary experiment in RomanTutor for the two manipulation problems depicted in [1]. We asked 12 users who participated in the first experiment to record new plans. The expertise level and success failure was added manually to sequences. From this data, we extracted sequential patterns for problem-states and actions with the extended SPM algorithm. In a subsequent work session, we asked the users to compare the tutoring services offered in [1] with those offered with the newly extracted knowledge base. On the whole, users preferred the newer version, as the hints offered were generally more precise and more appropriate, and help could be provided in more situations. We also observed that the system inferred more often correctly the estimated expertise level of learners.

9 Conclusion

In this paper, we reported several limitations of SPM for extracting problem spaces, and proposed an extended SPM algorithm that combines (1) time intervals, (2) multi-dimensional pattern mining and (3) the automatic clustering of valued actions. We also suggested dividing problems into problem states to enhance the relevance of the tutoring services. The algorithm was used to extract a problem space, and support tutoring services in RomanTutor. We are currently working on adapting our algorithm to eliminate redundancy among patterns by mining only closed patterns (those that are not included in another pattern having the same support) [13]. We are also working on including skills as dimensional information, and on conducting a larger experiment.

References

[1] Nkambou, R., Mephu Nguifo, E. & Fournier-Viger, P. (2008), Using Knowledge Discovery Techniques to Support Tutoring in an Ill-Defined Domain. *Proc. 9th Int. Conf. Intelligent Tutoring System (ITS 2008).*

[2] Simon, H. A. (1978). Information-processing theory of human problem solving. In W.K. Estes (Ed.), *Handbook of learning and cognitive processes: Vol. 5. Human information.*

[3] Lynch, C., Ashley, K., Aleven, V. and Pinkwart, N. (2006). Defining Ill-Defined Domains; A literature survey. *Proc. of the Intelligent Tutoring Systems for Ill-Defined Domains Workshop*. pp.1-10. ITS'2006.

[4] Kabanza, F., Nkambou, R. & Belghith, K. (2005), Path-planning for Autonomous Training on Robot Manipulators in Space, *Proc. of IJCAI 2005.*

[5] Fournier-Viger, P., Nkambou, R. & Mayers., A. (2008). A Framework for Evaluating Semantic Knowledge in Problem-Solving-Based Intelligent Tutoring Systems. *Proc. of FLAIRS 2008*, AAAI press, pp. 409-414.

[6] McLaren, B. & al. (2004). Bootstrapping Novice Data: Semi-Automated Tutor Authoring Using Student Log Files. *Proc. of the Workshop on Analyzing Student-Tutor Logs*. ITS'2004.

[7] Jarivs, M., Nuzzo-Jones, G. & Heffernan, N.T. (2006) Applying Machine Learning Techniques to Rule Generation in Intelligent Tutoring Systems. *Proc. of ITS'2006*, pp. 541-553.

[8] Agrawal, R. & Srikant, R.: (1995). Mining Sequential Patterns. *Proc. Int. Conf. on Data Engineering*, pp. 3-14.

[9] Pei, J., Han, J. et al. (2004). Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE Trans. Knowledge and Data Engineering*, 16(10), 1-17.

[10] Hirate, Y., Yamana, H. (2006). Generalized Sequential Pattern Mining with Item Intervals, *Journal of Computers*, 1(3), pp. 51-60.

[11] MacQueen, J.B. (1967). Some Methods for Classification and Analysis of Multivariate Observations, *Proc. 5th Berkeley Symposium on Mathematic Statistics and Probability*, pp. 281-297.

[12] Pinto, H et al. (2001), Multi-Dimensional Sequential Pattern Mining, *Proc. Int. Conf. Information and Knowledge Management (CIKM2001)*, pp. 81-88.

[13] Wang, J., Han, J & Li, C. (2007). Frequent Closed Sequence Mining without Candidate Maintenance, *IEEE Transactions on Knowledge and Data Engineering*, 19(8), pp.1042-1056

Acknowledgment. Our thanks go to the FQRNT and the NSERC for their logistic and financial support. The authors also thank the current/past members of GDAC/PLANIART for their work in RomanTutor, and Severin Vigot for integrating the algorithm in RomanTutor.

What Do Argument Diagrams Tell Us About Students' Aptitude Or Experience? A Statistical Analysis In An Ill-Defined Domain*

Collin Lynch[1], Niels Pinkwart[2], Kevin Ashley[3] and Vincent Aleven[4]

[1]University of Pittsburgh, Intelligent Systems Program, Pittsburgh, PA, USA
[2]Clausthal University of Technology, Department of Informatics, Germany
[3]University of Pittsburgh, School of Law, Pittsburgh, PA, USA
[4]Carnegie Mellon University, HCI Institute, Pittsburgh, PA, USA

Abstract. In ill-defined domains, argumentation skills are essential in order to define problems and to present, justify, and evaluate solutions. In well-defined domains there exist accepted methods of characterizing student arguments as good or bad. This is not always possible in ill-defined domains, where competing arguments are often acceptable. In this paper, we use a set of statistical analysis methods to investigate whether, despite the lack of an "ideal solution,", student-produced argument diagrams can be diagnostic in that they can be used to reliably classify students into novices and experts or high and low aptitude. Our analysis, based on data collected during three studies with the LARGO ITS, suggests that indeed, argument graphs created by different student populations differ considerably, particularly with respect to the completeness and "connectedness" of graphs, and can thus potentially be used to adapt the system to a particular student's needs.

Keywords: ill-defined domains, assessment, feedback, graph comparison

1 Introduction

Argumentation is a fundamental mode of reasoning and analysis in many domains, such as law, ethics, public policy, and science, that typically involve ill-structured problems.

Ill-structured problems are characterized by an incompletely stated goal, relevant constraints not stated in the problem, and competing, possibly inconsistent, reasonable solutions [19]. By contrast, well-structured problems have clearly stated goals and a strong well-supported domain theory that may be used to validate solutions [19].

In addressing ill-structured problems, argumentation is essential in order to define the problem and to present, justify, and evaluate solutions. The solver must frame the problem, refining the goal and inferring constraints [5]. Solvers may frame

* This research is supported by NSF Award IIS-0412830.

the problem in different ways depending on their knowledge, values, and interests, and often there is less consensus on the "right" way to frame the problem [19]. A solution "usually is justified by verbal argument that indicates why the solution will work, and provides a rebuttal by attacking a particular constraint or barrier to the solution or by attempting to refute an anticipated opposing position." [19].

Argumentation skills are therefore essential for students to learn how to address ill-structured problems not only in domains like law or ethics. Although mathematics and science are typically taught using well-structured problems, when it comes to discovering new knowledge, the problems are ill-structured. Skilled mathematicians and scientists engage in argument schema that are similar to those used by attorneys or ethicists, such as evaluating a proposed definition or decision rule by testing how it works on hypothetical examples as described below [6,7]. Even in solving a well-structured problem, a skillful student can state an argument to justify his solution approach. One expects such arguments to be more uniform, however, because the goal and constraints are stated in the problem and there is more of a consensus about how to proceed.

Increasingly, argument diagrams are used to help students acquire argumentation skills. Argument diagrams are graphical formats that reify aspects of the arguments' structure [1]. They can be a simple tree, whose root contains the argument's conclusion, each child of which can be supported by additional nodes that represents an argument premise. [13]. Often, argument diagrams are based on Toulmin's model of argumentation in which the argument structure represents data moving through a warrant to support a claim [15]. As described in [13] and at http://www.phil.cmu.edu/projects/argument_mapping/, other schemes for diagramming arguments have been developed, some of which have been employed in teaching.

The use of argument diagramming appears to be most widespread in courses on critical thinking and in philosophy [4,16,18]. To a lesser extent, they have been applied in teaching legal and scientific reasoning [2,11,14]. Students use them to represent their own arguments or to reconstruct arguments in some source text [18] such as, in our present research, transcriptions of oral arguments before the U.S. Supreme Court.

Whatever the diagramming scheme, software is often used to assist students in constructing the diagrams [4,13,18]. Initially, the diagrams served primarily as representational tools; the important thing was the diagramming process and its impact on learning, not the details of the resulting diagrams. In a software environment, however, the diagrams can also convey diagnostic information about the student's understanding. A small but increasing number of intelligent tutoring systems (ITS) focus on argumentation skills, employ diagrams, and even evaluate student-generated diagrams in order to provide feedback [11,14].

Automatically evaluating argument diagrams for providing feedback necessarily presents a challenge, especially when problems are ill-structured. For reasons discussed above, arguments supporting correct solutions to a well-structured problem are unlikely to diverge widely. The problem constraints are clearly specified, and the solution is deterministic and may be derived by more or less algorithmic means. As a result, such arguments can take relatively few plausible paths and one would expect to see detailed similarities across the alternatives (e.g., invoking the same physics

formulas or geometry theorems). Thus, one may readily construct criteria for assessing argument quality. For example, ActiveMath [9] can check the structure of mathematical arguments (proofs) on a fairly detailed level. By contrast, with ill-structured problems, given the need to frame the problems and the divergence of plausible frames, it is far less likely that all good arguments will look alike, at least in detail of the framings and justifications. Good arguments about ill-structured problems may have similarities, indeed they probably do, but the similarities may be manifest *only* at a more abstract level in the form of more generalized patterns in the argument diagrams.

Some ITS developers have manually developed rules for identifying patterns in the diagrammed arguments that are the occasions for meaningful feedback [11,14]. These patterns range from avoiding loops in the arguments to checking whether the graph reflects the important moves in the argument text being reconstructed to identifying portions of the argument diagram that are complete enough for the student to reflect on their significance.

As databases of argument diagrams accumulate, the question arises whether they might disclose other diagnostic patterns. Given the relative novelty of computer-supported argument diagramming and the difficulty of manually extracting the patterns, pedagogically valuable patterns may not yet have been discerned and the criteria for assessing argument diagrams in terms of such patterns may not yet be known. The problem-solving tasks involved in ill-structured problems likely give rise to distinctive patterns that reflect students' understanding of the tasks. For example, given the various argument schemes for evaluating a proposed solution, such as posing a hypothetical to critically evaluate a proposed decision rule, one would expect to see patterns involving proposed rules, hypotheticals, responses, and the links among them. The existence and nature of such patterns and criteria is a matter of research.

Manually analyzing the diagrams for such patterns is tedious. As noted above, we have developed automated techniques so that an ITS can identify and analyze known patterns. In this paper, we focus on how much information may be obtained from diagrams using standard statistical measures. In the context of our research teaching law students about hypothetical reasoning, we operationalize the question as follows. When students reconstruct a real-life argument graphically, are these graphs diagnostic of (a) general aptitude for legal reasoning (as measured by the Law School Admission Test (LSAT) a preparatory exam used by law schools not unlike the GRE) and (b) experience as law students (as measured by years in law school), and are these graphs predictive of (c) the gains in argumentation skill / knowledge that result from making the graphs (as measured by performance on the LARGO study pre- and post-tests)? Similarly, we want to understand better the differences in graphs that correlate with post-test differences relating to hypothetical reasoning skills. One's belief that argument diagramming activity reflects something real about legal argument skills would be bolstered to the degree that more advanced students create graphs that are recognizably different from those created by beginning students; and likewise, though perhaps not as strongly, to the degree that high LSAT students create recognizably different graphs from those of low LSAT students. In this paper, we investigate the research questions using statistical analysis techniques in order to detect significant

differences in the sets of argument diagrams created by the participants in our studies with LARGO.

ITSs have long used methods to infer, from student-system interactions, characteristics of students' current knowledge (e.g., through knowledge tracing), and these methods have been applied successfully to predict post-test scores based on how well students were able to solve problems within the tutor [3]. The current effort is focused differently, as a consequence of the ill-definedness of our domain: the focus is on discovering and validating measures by which to assess student argument diagrams.

2 Study Context

The LARGO ITS [11] for legal argumentation supports students in the process of analyzing oral argument transcripts taken from the U.S. Supreme Court. These are complex, real-world examples of argumentation of the kind in which professors seek to engage students in class. Since U.S. Supreme Court oral arguments tend to be more complicated than classroom arguments, students probably need support in order to understand and reflect on them.

Students annotate oral arguments using a graphical markup language with Test, Hypothetical and Fact nodes and relations between them. The test and hypothetical nodes represent assertions of tests and hypotheticals in the arguments. They may be linked to relevant portions of the transcript and contain a student-authored summary of the assertion.

LARGO provides support by analyzing student diagrams for "characteristics". These are associated with phases (1=orientation, 2=transcript markup, 3=diagram creation, 4=analysis, and 5=reflection). Characteristics in phases 1-3 can be thought of as diagram "weaknesses" (i.e., areas of potential problems or errors), while characteristics in phases 4 and 5 are opportunities for reflection. The system provides feedback in the form of self-explanation prompts which (in the later phases) encourage reflection about the diagram and the argument or (in the earlier phases) inform the student about misconceptions. Failing to link a test or hypothetical node to the transcript triggers the UNLINKED_TEST or UNLINKED_HYPO characteristics, both phase 1, and advice suggesting that the student link them. TEST_REVISION_SUGGESTED, phase 5, is triggered when the student has rated other students' test formulations using collaborative filtering and his own formulation was rated poorly indicating that a change might be needed. For more information on the characteristics see [10,11].

We conducted three studies with LARGO. In the Fall of 2006 we tested it with 28 paid volunteers from the first year Legal Process course at the University of Pittsburgh School of Law. Students were randomly assigned to analyze a pair of cases using LARGO or a text-based note-taking tool without feedback. We found no overriding differences in terms of post-test scores or system interactions between the conditions. However, lower aptitude students, as measured by LSAT score, showed higher learning gains than their low-LSAT text peers. Also, the use of the help was strongly correlated with learning [11].

Since participation was voluntary, the students self-selected for their interest in the curriculum, system, and pay. A second study was necessary to further examine and substantiate the findings with non-volunteers. We developed a LARGO curriculum covering three personal jurisdiction cases integrated into one section (85 students) of the 2007 first-year Legal Process course at the University of Pittsburgh School of Law. Participation was mandatory. The students were not paid but were given coffee gift cards as a token of appreciation. The curriculum was structured as preparation for a graded writing assignment on personal jurisdiction, worth 10% of their grade. As in 2006, students were randomly assigned to LARGO and text conditions, balanced by LSAT scores. The curriculum consisted of six weekly two-hour sessions, one more than in the first study. In this study, we found no significant differences between conditions [12]. Post-hoc analysis revealed that students in the first study made far more use of the advice functions than students in the second study, which may explain the difference between the study outcomes. We are presently conducting a third (2008) study with LARGO involving experienced (third-year) law students at the University of Pittsburgh. Students in this study are assigned to mark up the same set of cases used in the fall 2007 study. At the time of this paper writing, a total of 17 third-year students have completed this study. Their data is used below.

Analysis of student's pre- and post-test scores in the three studies shows that the experienced students performed significantly higher than the novices in terms of post-test score ($t(5.03)=48.91, p < 0.001$). There was no pre-test score difference between the groups ($t(1.65)=34.00, p < 0.1$).

Figure 1 shows two example graphs created by a novice (left) and experienced student (right). These graphs describe the same oral argument transcript, but clearly differ from another. The novice student has produced a linear series of notes with few structural interrelationships (indicated by arcs) nor any link to the transcript. When the student generates a test or hypothetical node with no link to the transcript the box is labeled with a hand symbol  . When a link is made the symbol changes to a highlight  . The experienced student, by contrast, has produced a linked graph representing the relationships among the elements with commentary on the arcs. He has also linked the tests and hypotheticals to the transcript. He has similarly made use of additional And-Clause and Outcome fields in the test and hypo nodes to produce more structured representations. These differences may just be due to the fact that in the ill-defined domain of legal argumentation there are no "ideal" diagrams – thus all diagrams will likely be different. On the other hand, there may well be typical diagram aspects that are characteristic of specific student groups. We now analyze the extent to which certain differences between diagrams are diagnostic and thus allow for a prediction about the student who created them.

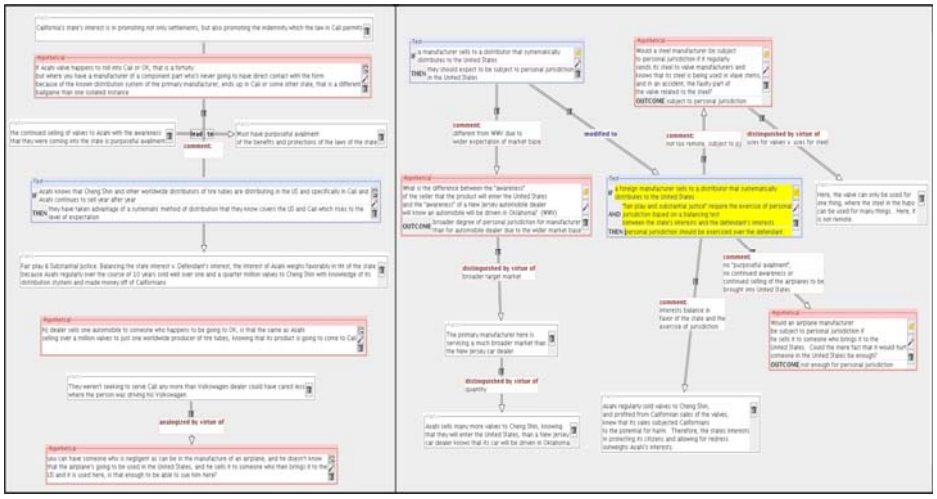


Figure1: Selected novice (left) and experienced (right) student graphs.

3 Results

Our first analysis of the diagrams was based on simple statistical measures such as the number of contained nodes, or relations, or the ratio of relations to nodes. The results can be summarized as follows:

- In none of our studies, did we observe a statistically significant correlation between the number of diagram elements and either the LSAT score, post-test score, or pre/post test gain.
- In 2007, the number of graph relations correlated positively with students' LSAT scores ($r=.32$, $p<.05$), yet the other studies do not support this (2006: $r=-.21$, $p>.4$; 2008: $r=.02$, $p>.9$).
- In 2007, the ratio of relations per nodes correlated positively with students' LSAT scores ($r=.32$, $p<.05$). A similar trend could be observed in 2006 ($r=.37$, $p<.2$), but not with the experienced students in 2008 ($r=.1$, $p>.7$).
- An ANOVA of the number of elements (nodes and relations) revealed significant differences between the three studies ($(F2,72)=9.3$,$p<.001$ for nodes, $(F2,72)=23.3$,$p<.001$ for relations). Also the ratio "relations per nodes" was different in the three studies ($F(2,72)=21.2$, $p<.001$). A post-hoc Tukey test revealed that experienced students produce significantly ($p<.05$) more relations ($m=12.3$) than the volunteer novice students ($m=7.9$), who produced significantly more than the non-voluntary novices ($m=5.2$). For the elements in graphs, there is a significant difference ($p<.05$) between both experienced students ($m=10.5$) and the volunteer novices ($m=9.6$) as compared to the non-volunteers ($m=7.5$); the first two did not differ significantly. All novices (volunteer or not) had a significantly ($p<.05$) smaller link-to-node ratio than experienced students. The averages were 1.14 for the experienced students,

and .82 and .67 for the two groups of first semester students.

This indicates that some (even very simple) measures are characteristic of specific student groups or aptitudes. However they are not sufficient as diagnostic tools that could distinguish the work of good students from that of poor students: None of the simple statistics discussed above is sufficient to accurately predict post-test scores or pre/post gains.

We therefore conducted a second set of analyses on the characteristic patterns that LARGO detects in student graphs and uses to give feedback. We analyzed the graphs created by students in all three studies for the occurrence of characteristics as detected by LARGO's graph grammar engine. A first analysis showed that the groups did not differ in terms of the **total** number of characteristics triggered by the graphs ($F(2,65)=1.2$, $p>.3$): on average, a student graph had between 14.3 (2008 study) and 18.4 (2006 study) characteristics. Thus LARGO can give as much feedback to an advanced student as to a first semester student.

We then investigated if the **types** of characteristics in the diagrams varied between the studies. A plausible heuristic here is that better (or more advanced) students will produce graphs with fewer weaknesses (i.e., characteristics of phase 1 – 3), and more opportunities for reflection (i.e., characteristics of phase 4 and 5). Figure 2 shows the relative frequencies of detected characteristics by phase. As the figure illustrates in the 2008 study, almost 50 percent of all detected characteristics were of the "phase 5" (reflection phase) type. For novice students, these frequencies are only 30 and 41 percent, respectively. This difference between the average amount of graph characteristics that indicate "opportunities for reflection" is not statistically significant, though ($F(2,65)=2.4$, $p=.09$). For none of the other phases (1-4), is the difference significant either.

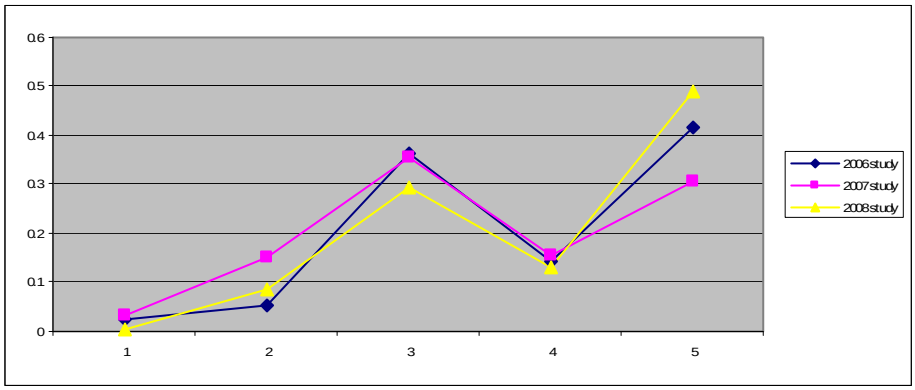


Figure 2: Relative frequencies of characteristics in student graphs by phase

Next, we analyzed whether the characteristics detected in students' graphs correlate with their pre/post gains or LSAT scores: does, for instance, a larger number of "weaknesses" mean that the student will perform poorer on the post-test? We compared the average number of phase 1-3 and phase 4+5 characteristics to the student's post-test score, the student's pre/post gain score, and the student's LSAT score. In the 2006 and 2008 studies, we observed no significant correlations between these variables. In the 2007 study, the number of "reflection characteristics" was

negatively correlated to the student's gain scores ($r=-.35$, $p<.05$) – i.e., students who have learned more during tool usage tend to produce final graphs that contain fewer opportunities for reflection.

A third analysis investigated if any of the individual diagram characteristics are by themselves diagnostic – out of the many types of characteristics that LARGO can detect, which ones are most predictive? We conducted a chi^2 analysis on the sets of characteristics (as detected by LARGO) contained in the final student-created diagrams. Due to training differences between the 2006 and 2007+8 studies we used only the 2007 novice and expert data. Our analysis demonstrated that three of the individual characteristics (the three described above) can be used to classify students into "above-median" and "below-median" in terms of their post-test score [8]. These characteristics are "UNLINKED_HYPO" ($c^2(10.40,N=51)=1.00$, $p<0.01$, precision=47/51), "UNLINKED_TEST" ($c^2(10.88,N=51)=1.00$, $p<0.001$, precision=38/51) and "TEST_REVISION_SUGGESTED" ($c^2(7.04,N=51)=1.00$, $p<0.01$, precision=35/51).

We further examined to what extent the three study populations differed. Our findings show that the following characteristics can be used to predict the group membership: NO_FACTS: ($c^2(8.61,N=51)=1.00,p < 0.01$, precision=32/51), UNLINKED_TEST: ($c^2(4.46,N=51)=1.00,p < 0.1$, precision=32/51), TEST_REVISION_SUGGESTED: ($c^2(12.40,N=51)=1.00,p < 0.001$, precision=41/51) and TEST_FACTS_RELATION_SPECIFIC ($c^2(7.44,N=51)=1.00,p < 0.01$, precision=39/51). The novice subjects exhibited more occurrences of NO_FACTS and UNLINKED_TEST than the expert subjects, while the expert subjects exhibited more instances of TEST_REVISION_SUGGESTED and TEST_FACTS_RELATION_SPECIFIC.

4 Discussion

Overall, the analysis results confirm our hypothesis. While – due to the ill-defined nature of the domain of legal argumentation – the diagrams created by students vary considerably and do permit a direct classification of "good" and "poor" diagrams, the differences between diagrams are not random. Even with rather simple statistical measures, it is possible to detect systematic differences between novice-produced and expert-produced diagrams, between diagrams of students with different aptitudes, and between those of students who use the system voluntarily or on a mandatory basis. Also, our results show that there are some aspects in diagrams that are suitable as diagnostic measures for learning gains.

Our first set of analyses, focusing on the number of elements and relations in student graphs, showed that graphs created by non-volunteers have fewer elements and relations than graphs created by volunteers. This may be caused by motivational factors: volunteers might have been more willing to use the system, resulting in more complex diagrams. A different analysis [12] provides further support for this. Also, experts create more relations and elements than novices, and produce more links per node than the novices resulting in more "connected" diagrams. "Connectedness" is important: if multiple parts of a larger argument graph are connected to each other

more frequently, this may be an indication of deeper reflection on the argument transcript that the graph represents, since the "relations" in LARGO diagrams can only be drawn reasonably if one thinks about the argument and understands the argument model. Connectedness can also be used to distinguish between novices with higher and lower LSAT scores. This further supports the hypothesis that "highly connected" graphs are an indication of more advanced or talented students.

Our second set of analyses indicates that even though the total number of diagram characteristics (as detected by LARGO) does not differ greatly between student groups, advanced students tend to produce diagrams that have more "high level" characteristics which indicate opportunities for reflection. This supports the classification of characteristics as implemented in LARGO. A surprising finding is that for first-semester students who used the system on a mandatory basis, the *fewer* opportunities for reflection in the final student-created diagrams LARGO detects, the *higher* the student learning gains. As such, a diagram with a small number of these high-level diagram characteristics would predict a high learning gain. This finding was not reproduced in the other two studies, though. A possible explanation for this effect may be that the diagrams we analyzed are the result of students' one-hour sessions with the tool. Many of the students who used the system in 2007 on a mandatory basis appeared not to be highly motivated to do so [12]. The few who did use the system intensively have received more feedback on their diagram than their peers, resulting in diagram changes (responding to feedback) that reduce the number of detected characteristics in the final diagrams. In that sense, fewer diagram characteristics in the final diagrams can be an indication of more activity with the system, which in turn leads to higher learning gains. An in depth-investigation of these relationships still remains to be done. An analysis of the log files of the student's activities with the system over the whole usage time (as opposed to analysis of students' final argument diagram, as we report in the current paper) will enable us to investigate, for example, whether relations between detected diagram characteristics and learning differed depending on the amount of help use.

The third set of analyses we conducted focused on individual characteristics (of the type that LARGO can detect using its built-in graph grammar engine). This analysis revealed that apparently, the linking behavior of students (i.e., whether they connect their argument diagram elements to the argument text or not) can be used to distinguish students by aptitude, as measured by LSAT score: better students tend to link their elements to the text more consistently. Also, the results of this analysis confirm the utility of LARGO's algorithm for peer-review of test formulations: a standing recommendation that the students change their test formulation correlates with the aptitudes of the students. This indicates that the students made use of the peer-review process but, paradoxically, that they did not always reformulate their test in response. The results of this analysis also substantiated that that the graphs created by experts are different from those created by novices, and gave a further and more specific means of distinction (in addition to the general "graph connectivity" aspect discussed above). In other words, some "key characteristics" are suitable as heuristic diagnostic tools – their presence (or absence) indicates that the diagram was created by a more (or less) advanced student. Consistent with our expectations, the characteristics typical of experts belong to "higher level" feedback messages, while

the typical "lower aptitude student" characteristics correspond to "beginner's mistakes".

5 Conclusion

No "ideal argument diagram" can be defined for the task of legal argumentation that LARGO is designed to teach. Consequently, there is a rich variety of diagrams that would be called of good (or poor) quality, but that are substantially dissimilar. Still the question, whether some properties of an argument diagram created by a student are diagnostic of that student's skills, is interesting: is the variation between argument diagrams created by different students purely random, and a result of the ill-definedness of the domain, or are there properties of diagrams that are characteristic of specific types of students?

We applied statistical analysis techniques to analyze argument diagrams created by students in three studies with LARGO. The subject populations in these studies differed in terms of their experience (beginning law students vs. advanced), aptitude (as measured by LSAT score), mode of participation in the studies (voluntary vs. mandatory), and learning gains. We found clear evidence that the graphs created by these different populations differ from each other. For some distinctions, rather simple statistical measures (such as the number of relations in a graph) are sufficient. Others require more advanced analysis methods, such as counting the different "graph characteristics" as detected by the feedback mechanism in LARGO.

These findings are an important step in our research with LARGO, and they have implications for other researchers in the field of ITSs in ill-defined domains. Even in domains where it is impossible to make sharp distinctions between "good" and "bad" solutions due to the lack of ideal solutions or a domain theory, the solution differences are meaningful. The diagrams created in LARGO contain diagnostic information about the student's understanding – i.e., whether he is likely to be of lower aptitude, or if he is more likely to be an advanced student or a beginner. This knowledge can potentially be used to adapt the system to the particular needs of a student.

An important related question is how to induce the distinguishing characteristics of good argument diagrams, that is, how to identify new diagnostic patterns. We pursue this line of inquiry in a different paper using machine learning techniques [8].

References

1. Buckingham Shum, S. 2003. The Roots of Computer Supported Argument Visualization. In Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making. Springer, London, UK.

2. Carr, C. 2003. Using Computer Supported Argument Visualization to Teach Legal Argumentation. In Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making, 75–96. Springer, London, UK.

3. Corbett, A.T., Anderson, J.R. (1995) Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. User Modeling and User-Adapted Interaction, 4, 253-278.

4. Harrell, M. 2007. Using Argument Diagramming Software to Teach Critical Thinking Skills. In Proceedings of the 5th International Conference on Education and Information Systems, Technologies and Applications.

5. Goldin, I., Ashley, K., & Pinkus, R. 2006. Teaching Case Analysis through Framing: Prospects for an ITS in an ill-defined domain. In Proceedings of the ITS 2006 Workshop on ITS for Ill-Defined Domains. Jhongli, Taiwan.

6. Hurley S. L. 1990. Coherence, Hypothetical Cases, and Precedent. 10 Oxford J. of Legal Studies 221, 230-234.

7. Lakatos, I. 1976. Proofs and Refutations. London: Cambridge University Press.

8. Lynch, C., Ashley, K., Pinkwart, N., & Aleven, V. 2008. Argument graph classification with Genetic Programming and C4.5. To appear in Proceedings of 1st International Conference on Educational Data Mining.

9. Melis, E., & Siekmann, J. 2004. ActiveMath: An Intelligent Tutoring System for Mathematics. In Proceedings of Seventh International Conference 'Artificial Intelligence and Soft Computing (ICAISC), 91-101. Springer.

10. Pinkwart, N., Aleven, V., Ashley, K., & Lynch, C. 2006. Using Collaborative Filtering in an Intelligent Tutoring System for Legal Argumentation. In Proceedings of Workshops held at the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Lecture Notes in Learning and Teaching, 542-551. Dublin (Ireland), National College of Ireland.

11. Pinkwart, N., Aleven, V., Ashley, K., & Lynch, C. 2007. Evaluating Legal Argument Instruction with Graphical Representations using LARGO. In Proceedings of the 13th International Conference on Artificial Intelligence in Education (AIED2007), 101-108. Amsterdam, IOS Press.

12. Pinkwart, N., Lynch, C., Ashley, K., & Aleven, V. 2008. Re-evaluating LARGO in the Classroom: Are Diagrams Better than Text for Teaching Argumentation Skills? To appear in Proceedings of the 9th International Conference on Intelligent Tutoring Systems.

13. Reed, C., & Rowe, G. 2007. A pluralist approach to argument diagramming. Law, Probability and Risk 6, 59−85.

14. Suthers, D., Weiner, A., Connelly, J., & Paolucci, M. 1995. Belvedere: Engaging students in critical discussion of science and public policy issues. In Proceedings of the 7th World Conference on Artificial Intelligence in Education, 266 -273.

15. Toulmin, S. E. 1958. The Uses of Argument. Cambridge: Cambridge University Press.

16. Twardy, C. 2004. Argument maps improve critical thinking. Teaching Philosophy 27, 95–116.

17. van den Braak, S. W., van Oostendorp, H., Prakken, H., & Vreeswijk, G. A. W. 2006. A critical review of argument visualization tools: do users become better reasoners? In Working Notes of the 6th Workshop on Computational Models of Natural Argument (CMNA2006).

18. van Gelder, T. 2007. The Rationale for Rational™. In Law, Probability and Risk 6, 23-42.

19. Voss, J. (2006) “Toulmin's Model and the Solving of Ill-Structured Problems” in Hitchcock D. and Verheij, B. (eds.) Arguing on the Toulmin Model: New Essays in Argument Analysis and Evaluation. Springer.