Using Problem-Solving Context to Assess Help Quality in Computer-Mediated Peer Tutoring

Erin Walker[1], Sean Walker[1], Nikol Rummel[2], Kenneth R. Koedinger[1]

[1] Human-Computer Interaction Institute, Carnegie Mellon University, USA
[2]Institute of Psychology, University of Freiburg, Germany
erinwalk@andrew.cmu.edu, walker.sean.m@gmail.com, rummel@psychologie.uni-freiburg.de, koedinger@cmu.edu

Abstract. Collaborative activities, like peer tutoring, can be beneficial for student learning, but only when students are supported in interacting effectively. Constructing intelligent tutors for collaborating students may be an improvement over fixed forms of support that do not adapt to student behaviors. We have developed an intelligent tutor to improve the help that peer tutors give to peer tutees by encouraging them to explain tutee errors and to provide more conceptual help. The intelligent tutor must be able to classify the type of peer tutor utterance (is it next step help, error feedback, both, or neither?) and the quality (does it contain conceptual content?). We use two techniques to improve automated classification of student utterances: incorporating domain context, and incorporating students' self-classifications of their chat actions. The domain context and self-classifications together significantly improve classification of student dialogue over a baseline classifier for help type. Using domain features alone significantly improves classification over baseline for conceptual content.

Keywords: intelligent tutoring, computer-supported collaborative learning, adaptive collaborative learning systems, peer tutoring

1 Introduction

Student participation in online collaborative classroom activities can increase both group performance and individual learning outcomes. However, these positive effects are not always found [1], in part because when left to their own devices students may not interact in ways that lead them to benefit from collaboration. One common technique for improving computer-mediated collaborative interaction is *scripting*, where the collaboration is structured so that students take on particular roles and go through designated phases in order to increase the effectiveness of their collaboration [see 2 for review]. Although scripts have been shown to be effective, they have been criticized for over-structuring the collaboration for some students, decreasing these students' motivation, while under-structuring the collaboration for others, failing to provide them with sufficient support [3]. It has been theorized that an intelligent tutor for collaboration that is responsive to student needs and to the current interaction state might be more effective [4], and, in fact, adaptive support for collaboration has been demonstrated to be an improvement over no support and fixed support at increasing

learning [5, 6]. Despite these potential advantages, intelligent tutoring for collaborative learning is still at an early phase, and few classroom-ready systems have been developed and evaluated.

A key component of improving collaborative interactions is supporting students in conducting productive dialogues. Thus, developing mechanisms for assessing student dialogues has been a focus of research in this area. One way of assessing student dialogue is through *self-classification*, where students are asked to indicate the type of statement that they are making before or after they compose it. For example, students may select a sentence starter like "We need to work together on this…" to begin their utterance. Based on the starters that students select, the system can make inferences about what students are saying, and use these inferences to provide feedback [e.g., 7]. However, students do not consistently select sentence starters or classifiers that match the content of their utterances, and therefore the inferences that the system makes based on those labels can be inaccurate [8]. Consequently, researchers have been moving towards using *machine classification* to assess student dialogue as it occurs in order to provide students with assistance. So far this technology has been used in limited ways in intelligent tutoring for collaborative learning; for instance, for classifying the topic of conversation [6], or for assessing student accuracy when they use sentence starters [8]. As the quality of student dialogue contributes to how students benefit from collaboration, improving our ability to automatically classify student utterances would increase our ability to target support to those utterances.

We have developed an intelligent tutor for collaborative learning by extending the Cognitive Tutor Algebra, an existing successful intelligent tutoring system, to support two students of similar abilities in tutoring each other in algebra. Within the context of providing intelligent support for peer tutoring, we explore two different approaches for improving the accuracy of dialogue classification: incorporating information about the domain context, and incorporating student self-classifications. Firstly, we use information about the domain context of the interaction as additional features for a machine learning classifier. This context includes information directly taken from the students' problem-solving behavior (e.g., a student has just taken a incorrect step in the problem), information about how student dialogue relates to the problem-solving context (e.g., a student has referred to another student's incorrect step), and information about the history of the interaction (e.g., a student has referred to another student's incorrect steps 10 times over the course of the whole interaction). There is a precedent for this approach: The few adaptive collaborative learning systems that have used domain information have shown gains both in the variety of support that those systems provide and in the effects of support [e.g., 9], but they have not applied these models to the classification of collaborative dialogue. However, this approach has been applied successfully in asynchronous collaborative contexts [10], and domain features have been successfully used to enhance the ability of automatic classifiers in other fields [e.g., 11]. Secondly, we use student self-classifications of their own chat dialogue as a potential feature for improving the accuracy of the machine classifier. As described above, it is common in adaptive collaborative learning systems to ask students to classify their own utterances. While these classifications are not always accurate, they may still be relevant for assisting the machine classifier in labeling the utterance.

In this paper, we explore how incorporating information on the domain context and self-classification features might improve the classification of peer tutor dialogue. We describe the details of our collaborative learning system, the model of peer tutoring that we are trying to support, and our data collection procedure. We then describe the classification approaches we compare: baseline classification of student dialogue based solely on text features (*B*), baseline classification with additional domain features (*B+D*), baseline classification with additional self-classification features (*B+SC*), and baseline with problem-solving and self-classification features (*B+D+SC*). We discuss the results of comparing the classifiers and their implications.

2 Context

We attempt to automatically classify student dialogue within the context of an intelligent tutoring system for reciprocal peer tutoring, called APTA (the Adaptive Peer Tutoring Assistant). APTA provides an interface for one student to tutor another student on algebra problems, and then provides the peer tutor with assistance on how to be a better tutor. To do so, APTA maintains a model of good peer tutoring and compares student actions to the model. Accurately assessing the quality of student chat actions enables this comparison to be made and effective assistance to be given. In the following section, we describe the functionality of APTA in more detail, to make it clear which aspects of student dialogue we are trying to assess and why.

2.1 APTA: Adaptive Peer Tutoring Assistant

Reciprocal peer tutoring is a collaborative learning activity where two students of similar abilities take turns tutoring each other. It has been shown to improve student learning over unscripted collaboration and individual learning [12], and is an effective intervention even for low-ability students [13]. We have constructed a peer tutoring addition to the Cognitive Tutor Algebra (CTA), a successful intelligent tutoring system for high school mathematics [14]. In our peer tutoring script, students are given a task like "Solve for x," for an equation like "ax + by = c." Students go through two phases: a preparation phase and a collaboration phase. In the *preparation phase*, peer tutors solve problems using the CTA, receiving hints and error feedback when necessary. During the *collaboration phase*, students are grouped into pairs and collaborate at different computers, taking turns being peer tutors and peer tutees. Peer tutees solve the same problems as their tutor solved in the preparation phase, using the same equation solver interface. Peer tutors can see their peer tutee's actions, but cannot solve the problem themselves. Instead, they can mark the peer tutee's actions right or wrong, and interact with the tutee in a chat tool, where they can give help and feedback. We augmented the chat tool with sentence classifiers, asking peer tutors to label their utterances as a prompt ("asks for explanation"), error feedback ("explains why wrong"), a hint ("gives hint"), or an explanation ("explains next step"). Peer tutors had to select a classifier before they typed in an utterance, but they could also choose to click a neutral classifier ("comments").

As most students are novice tutors and need support to collaborate effectively, they receive two different types of assistance from the system as part of the script (see Figure 1). First, we have augmented the script with adaptive *domain support* for the peer tutor. If the peer tutor marks a correct tutee action wrong in the interface, or an incorrect action right, the cognitive tutor will intervene by indicating that the step was marked incorrectly, and providing feedback on what to do next. The peer tutor can also request a domain hint from the cognitive tutor at any time. Second, we augmented the script with adaptive *interaction support* for the peer tutor. This support primarily takes the form of reflective prompts delivered to both students in the chat window such as, "Tutor, you might want to explain that further", and "Tutee, did you understand what the tutor just said?" For these prompts to be effective, they must contain relevant information and be presented at moments when the peer tutor can apply them to the interaction. Therefore, we maintain both a model of good peer tutoring and a running assessment of the actual quality of the students' tutoring.

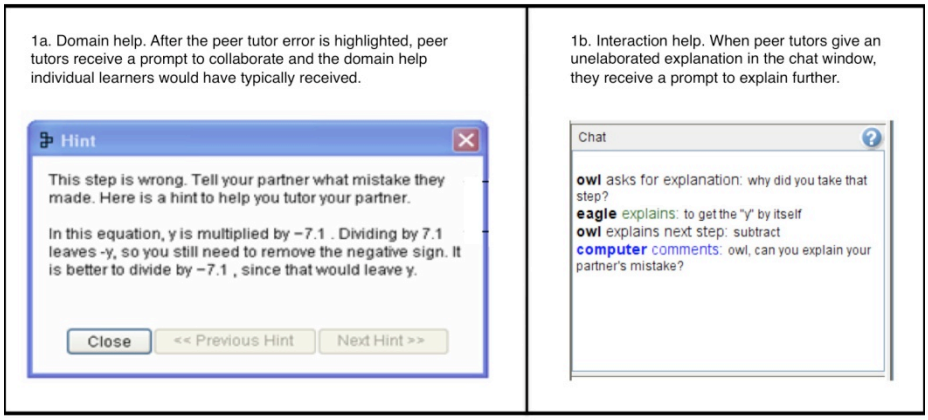

Figure 1. Two forms of assistance in the Adaptive Peer Tutoring Assistant (APTA). Peer tutors are provided with domain support and interaction support.

2.2 Modeling Reciprocal Peer Tutoring

To provide support for the peer tutor role in reciprocal peer tutoring, we have constructed a production-rule model for peer tutor help-giving. Our production rule model focuses on three help-giving skills, derived from the following research. Tutors have been found to benefit from the tutoring activity by reflecting on their existing knowledge as they observe tutee problem-solving steps and errors, and then constructing new knowledge as they compose explanations [15]. In order for tutees to benefit from the activity, peer tutor help should be given at impasses, should target tutee level of understanding, should explain errors, should provide assistance for the next-step, and should be conceptual & elaborated [16]. The first skill in our model prescribes that the peer tutor should give help when needed, which we operationalize as giving help after tutee help requests and errors, but not after correct steps (*S1: Help When Needed*). The second skill prescribes that the peer tutor should give relevant

help for the tutee's specific needs. For example, after a tutee error, the peer tutor should respond by prompting tutees to self-explain and then, if necessary, explaining the tutee mistake (*S2: Appropriate Help*). Finally, the peer tutor should explain the rationale behind problem-solving steps, rather than simply saying what to do. In particular, when peer tutors give help on the next step, they are expected to use hints and explanations that reference relevant domain concepts (*S3: Conceptual Help*).

We assess these collaboration skills using Bayesian knowledge tracing, and provide feedback based on the assessment [17]. In order to determine whether peer tutors are displaying the above three skills, we need to classify two aspects of the help peer tutors give in their dialogue with tutees:

1. *Help type*. Are peer tutors giving next-step help, error feedback, both, or no help at all? Using the classified help type in conjunction with the problem-solving context (e.g., knowing whether the tutee has just made a correct step, incorrect step, or help request) can help us decide whether tutors are giving the appropriate kind of help (*S2*) when it is needed (*S1*).
2. *Conceptual content*. Are peer tutors giving help that explains concepts rather than simply stating what to do next? Being able to identify this aspect lets us know whether tutors are providing enough conceptual help (*S3*).

By accurately classifying these aspects of student dialogue, we can develop intelligent support for peer tutoring that enables us to improve peer tutor performance on the above three help-giving skills.

2.3 Corpus & Data Coding

We used a corpus drawn from a classroom study we conducted comparing adaptive support for peer tutoring to fixed support for peer tutoring. As part of the study, students participated in two supported peer tutoring sessions; one in which they acted as the tutor, and one in which they acted as the tutee. We have a total of 84 tutoring sessions from both conditions, consisting of an average of 21.77 tutor lines of dialogue per session (SD = 10.25). Two raters coded tutor utterances for help type and conceptual content. We computed interrater reliability on 20% of the data, and the remainder of the data was coded by one rater and checked by the second. All disagreements were resolved through discussion. We segmented the dialog by chat messages, creating a new segment every time students hit enter. First, each help segment was coded for help type by determining whether it consisted of *previous-step help* relating to an action tutees had already taken (e.g., "no need to factor because there is only one g"; kappa = 0.83), and whether it consisted of *next-step help* relating to a future action in the problem (e.g., "how would you get rid of 2*h*?"; kappa = 0.83). If the help segment contained both categories, its help type was labeled "both", and if it contained neither category (e.g., "on to the next problem"), its help type was labeled "none". Second, each help segment was coded for whether it contained a concept (e.g., "add *ax*" was purely instrumental help, while "add *ax* to cancel out the –*ax*" was conceptual). Kappa for conceptual help was 0.72. In our dataset, 935 tutor instances were coded as "none", 764 were coded as "next-step help", 83 were coded as "previous-step help", and 47 were coded as "both"; 1654 instances were coded as non-conceptual help, and 165 were coded as conceptual help.

3 Method

3.1 Baseline Classification

We generated baseline machine classifiers for *help type* and *conceptual content* using Taghelper Tools, state of the art text-classification technology designed for coding collaborative dialogue [18]. Taghelper automatically extracts several dialogue features for use in machine classification, including unigrams, bigrams, line length, and punctuation. In our dataset, Taghelper generated 641 features. We used a chi-squared feature selection algorithm to rank the most predictive features, and selected 150 features for help type and 125 features for conceptual content. We used 10-fold cross validation to train a support vector machine classifier for each dimension.

3.2 Incorporating Domain Features

We augmented the dialogue features generated by Taghelper with domain context features. After assembling the problem-solving context, text substitution, and history features described below, we again used a chi-squared feature selection algorithm to rank the most predictive features. We used 10-fold cross validation to train a support vector machine classifier for help type and conceptual content.

Problem-Solving Context. In general, features describing the tutee's problem-solving progress may provide information about the type and quality of the help peer tutors tend to give (e.g., peer tutors may be more likely to give error feedback after the tutee has made an error). Thus, we added a total of 10 features for the classifier, created using information from the CTA models of student problem-solving. This information included whether the last step taken on the problem was correct or incorrect, the student's progress in the problem (i.e. the number of correct, incorrect, and total steps taken), and the student's problem-solving momentum (e.g. the number of incorrect steps the student had made in a row).

Text Substitutions. We then added features representing whether peer tutors *referred* to problem-solving elements in their utterances (e.g., "subtract x" refers to a specific problem-solving action). By treating different references to the problem as members of a higher-level category, we can compensate for a lack of training data and enable the classifier to transfer between different units of the problem. More specifically, we extracted a list of problem-related actions from the CTA menu options that tutees were able to select in the unit (e.g., {factor, distribute, add, subtract}), and a list of problem-related variables from the problem-statement (e.g., $x = a + b$ would return $\{x,a,b\}$). We then substituted specific occurrences of a problem-related action or term in the text with general terms (see the "Substituted Text" column in Table 1), and used the new text as input into Taghelper. We also added a feature that indicated that a substitution had been made ("Action Present" and "Term Present" in Table 1).

Further, by tracking which specific aspects of the problem tutee utterances referred to, we hoped to be able to better identify the target of the help given by the peer tutor.

Thus, we added a feature representing whether the substituted terms referred to the tutee's last correct step or last incorrect step (e.g., in the second row of Table 1, "add x to the left side" sets the Term Present feature to "last-correct", indicating that there is a term in the problem which refers to the last correct step). We also made substitutions based on whether peer tutors referenced terms that appeared in the problem-solving hints generated by the cognitive tutor. We created a list of verbs and nouns found in the hints, and then substituted a generic "concept" word for these words (as in the third row of Table 1). We added an additional feature representing whether a concept term had been substituted. Finally, we created substitution features to indicate whether multiple substitutions of the different types had occurred. The presence of multiple substitutions in an utterance makes it more likely that a reference to the problem actually occurred. This approach emphasized those utterances where multiple substitutions were done, while deemphasizing utterances where only a single substitution took place. Overall, we added 7 text substitution features.

Table 1. Selected features created from particular tutor utterances. If the tutee's last action was "factor x", and this action was correct, the following are the substitutions that would be made.

Chat Text	Substituted Text	Action Present	Term Present	Term-Concept Present	Action-Term Present
now factor	now action	last-correct	no	no	no
add x to the left side	action term to the left side	yes	last-correct	no	yes
isolate the p	concept the term	no	yes	yes	no

Substitution History. Finally, we added 6 history features in an attempt to provide holistic information about the overall quality of the interaction. The history features were based on the numbers of each type of substitution made; features were created for what percent of the peer tutor's total number of utterances referred to a concept, what percent referred to a correct or incorrect action, and what percent referred to a correct or incorrect term. We also included a simple yes/no feature as to whether or not a substitution of a specific type was made at any point, under the rationale that somebody who has given a certain kind of help in the past would be more likely to give that kind of help in the future. All history features were updated with each tutor utterance; that is, history features were only computed based on all utterances that had occurred *prior* to the current utterance, so that the algorithm could be applied to a learning situation as it unfolds. We based the history features on the substitutions rather than on the machine classifications to avoid being stuck in a state where, for example, because the machine has not yet classified an utterance as conceptual help, it is likely to never classify an utterance as conceptual help.

3.4 Adding Self-Classification

In addition to creating domain features, we also added two features that involved students' self-classification of their actions. As described in Section 2, before composing an utterance peer tutors were asked to label their utterance as a prompt,

error feedback, a hint, an explanation, or a comment. The label selected by the peer tutor, as well as their overall use of sentence classifiers, may be predictive of the type of help the peer tutor gave in a particular utterance. We added the self-classification specified by the tutor and the number of non-comment sentence classifiers used by the tutor in total as features in the machine classification.

4 Results

We hypothesized that both the domain features (B+D) and the self-classification (B+SC) would lead to an improvement over the baseline classification (B), with a classifier containing all three sets of features being the most effective (B+D+SC). We compared Cohen's kappa for all classifiers in the Weka Experimenter using 10 repetitions of 10-fold cross-validation (see Table 2) [19]. We use kappa instead of percent accuracy due to the imbalanced frequency distribution between categories (for example, there was over 10 times more non-conceptual help utterances than conceptual help utterances). Weka uses paired t-tests corrected for dependence between samples to compare classifiers. For help type, B+D+SC was significantly better than the B classifiers ($p < .05$). For conceptual help, only B+D was significantly better than baseline ($p < .05$). It is encouraging that the help type kappa for BS+D+SC approached the kappa we achieved for human interrater reliability, and that the conceptual help kappa improved substantially between the B and B+D.

Table 2. Kappas for the baseline (B), baseline plus self-classification (B+SC), baseline plus domain (B+D), and baseline plus self-classification plus domain feature sets (B+D+SC). Kappas are reported for both the help type and conceptual help classifications.

	Help Type Kappa		Conceptual Help Kappa	
Classifier	M	SD	M	SD
B	.78	.04	.59	.10
B + SC	.78	.04	.60	.10
B + D	.80	.04	.66*	.10
B + D + SC	.81*	.04	.65	.11

Examining which features were ranked highly by the chi-squared feature selection algorithm for the B+D+SC feature set, we can see that our domain context features consisted of 7 of the top 10 features for the help type classification, and 7 of the top 10 features for the conceptual help classification (see Table 3). In addition, one highly ranked feature for help type was the sentence classifier used, part of the SC feature set. Overall, for the help type classifier, only 3 of the domain context features created were not selected to be part of the machine classifier, and 2 of these features related to the number of correct steps that had recently been taken by the tutee. It is interesting that while incorrect problem-solving actions appeared somewhat predictive of the type of help given, correct problem-solving actions did not. This result makes sense, as it is more likely that tutors would refer to a previous incorrect step than to a previous correct step. For the conceptual help classifier, 14 of the 25 conceptual help features were not selected, suggesting that conceptual help classification is less dependent on domain context.

Table 3. The top ten ranked features in chi-squared feature selection for help type and conceptual help for the baseline plus problem-solving plus self-classification feature set.

Rank	Help Type Kappa	Conceptual Help Kappa
1	action present	concept present
2	"action"	"concept"
3	term present	concept & term present
4	"term"	"concept_term"
5	"BOL_action"	line length
6	action & term present	"you_concept"
7	classifier used	percent concepts used
8	"term_EOL"	"how_do"
9	"BOL_undo"	"you"
10	"undo"	"term_by"

5 Conclusion

The focus of this paper was to increase the accuracy of automated classification of peer tutor utterances in order to improve the ability of an intelligent tutoring system for peer tutoring to provide appropriate support. To do so, we explored the use of domain context features, extracted from individual domain models found in the Cognitive Tutor Algebra, as input into dialogue classifiers to augment automatically extracted text features. We also examined whether student self-classifications of their own utterances might improve the machine classification. We found that domain context features in combination with self-classifications significantly improved the accuracy of an automated classifier with respect to help type, but only domain context improved the accuracy of conceptual content classification.

We incorporated three different types of domain context features into the machine classifier: problem-solving context, text substitutions, and substitution history. Of those features, relevant text substitution and substitution history features were highly related to the machine classification for each dimension; for example, substitutions of references to tutee actions were highly predictive of help type, while substitutions of references to concepts were highly predictive of conceptual help. In contrast, self-classifications were less effective; they appeared to augment the results of the help-type classifications, but inhibit the results of the conceptual help classification. This result is not unexpected, as the self-classifications that students made were far more relevant to the help type dimension than to the conceptual help dimension. Perhaps student self-classifications that were more related to whether the utterance included conceptual help would have a positive effect on a conceptual help classifier.

These results make the argument for an emphasis on designing adaptive support for collaboration that is rooted in problem-solving context. If domain context information can improve the accuracy of automated collaborative dialogue classification, it would make sense for intelligent tutoring systems for collaborative learning to incorporate domain models. While domain models are difficult to build from scratch, integrating adaptive collaborative learning systems with existing individual intelligent tutoring

systems may be a way to leverage sophisticated domain information in order to improve the effectiveness of intelligent tutoring for collaborative learning.

Acknowledgments. This research is supported by the Pittsburgh Science of Learning Center, NSF Grant #SBE-0836012. Thanks to Carolyn Rose, Ruth Wylie, & Amy Ogan.

References

1. Lou, Y., Abrami, P. C., d'Apollonia S.: Small group and individual learning with technology: A meta-analysis. R. of Ed. Res., 71(3), 449-521. (2001)
2. Kollar, I., Fischer, F., Hesse, F. W.: Computer-supported collaboration scripts – a conceptual analysis. Ed. Psych. Review, 18(2), 159-185. (2006)
3. Dillenbourg, P.: Over-scripting CSCL: The risks of blending collaborative learning with instructional design. In: P. A. Kirschner (ed.), Three worlds of CSCL. Can we support CSCL, pp. 61-91. Heerlen: Open Univeriteit Nederland. (2002)
4. Rummel, N., Weinberger, A. New challenges in CSCL: Towards adaptive script support. In: G. Kanselaar, V. Jonker, P.A. Kirschner, & F. Prins: (eds.), International perspectives of the learning sciences: Cre8ing a learning world. Proceedings of the Eighth International Conference of the Learning Sciences (ICLS 2008), Vol 3 (pp. 338-345). ISLS. (2008)
5. Gweon, G., Rosé, C., Carey, R. & Zaiss, Z.: Providing support for adaptive scripting in an on-line collaborative learning environment. Proc. of SIGCHI Conference on Human Factors in Computing Systems (pp. 251-260). ACM Press. (2006)
6. Kumar, R., Rosé, C. P., Wang, Y. C., Joshi, M., Robinson, A.: Tutorial dialogue as adaptive collaborative learning support. In: R. Luckin, K. R. Koedinger, & J. Greer (eds.) Proceedings of AIED 2007 (pp. 383-390). IOS Press. (2007)
7. Tedesco, P.: MArCo: Building an artificial conflict mediator to support group planning interactions. IJAIED, 13(1), 117-155. (2003)
8. Israel, J., Aiken, R.: Supporting collaborative learning with an intelligent web-based system, IJAIED, 17(1), 3-40. (2007)
9. Baghaei, N., Mitrovic, T., Irwin, W.: Supporting Collaborative Learning and Problem Solving in a Constraint-based CSCL Environment for UML Class Diagrams. IJCSCL, 2 (2-3), 159-190. (2007)
10. Wang, Y. C., Joshi, M., Rosé, C. P., Fischer, F., Weinberger, A., Stegmann, K.: Context Based Classification for Automatic Collaborative Learning Process Analysis. Poster in AIED 2007. (2007)
11. Dybowski, R., Laskey, K. B., Myers, J. W., Parsons, S.: Introduction to the Special Issue on the Fusion of Domain Knowledge and Data for Decision Support. J. of Machine Learning Research. *4*. 293-294. (2003)
12. Fantuzzo, J. W., Riggio, R. E., Connelly, S., Dimeff, L. A.: Effects of reciprocal peer tutoring on academic achievement and psychological adjustment: A component analysis. *Journal of Educational Psychology, 81*(2), 173-177. (1989)
13. Robinson, D. R., Schofield, J.W., Steers-Wentzell, K.L. Peer and Cross-age Tutoring in Math: Outcomes and Their Design Implications, Educational Psychology, 17, 327-361. (2005)
14. Koedinger, K., Anderson, J., Hadley, W., Mark, M.: Intelligent tutoring goes to school in the big city. IJAIED, 8, 30-43. (1997)
15. Roscoe, R. D., Chi, M.: Understanding tutor learning: Knowledge-building and knowledge-telling in peer tutors' explanations and questions, R. of Ed. Res. 77(4), 534-574. (2007)
16. Webb, N. M.: Peer interaction and learning in small groups. I. J. of Ed. Res., 13, 21-39. (1989)
17. Beck, J., and Sison, J.: Using Knowledge Tracing in a Noisy Environment to Measure Student Reading Proficiencies, IJAIED, vol. 16, no. 2, pp. 129-143. (2006)
18. Rosé, C. P., Wang, Y. C. , Cui,Y., Arguello, J., Fischer, F., Weinberger, A., Stegmann, K. Analyzing Collaborative Learning Processes Automatically: Exploiting the Advances of Computational Linguistics in Computer-Supported Collaborative Learning. IJCSCL. (to appear)
19. Hall, M. Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H.: The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. (2009)