Edinburgh Research Explorer

Combining Semantic Interpretation and Statistical Classification for Improved Explanation Processing in a Tutorial Dialogue System

Citation for published version:
Dzikovska, MO, Farrow, E & Moore, JD 2013, Combining Semantic Interpretation and Statistical Classification for Improved Explanation Processing in a Tutorial Dialogue System. in HC Lane, K Yacef, J Mostow & P Pavlik (eds), *Artificial Intelligence in Education: 16th International Conference, AIED 2013, Memphis, TN, USA, July 9-13, 2013. Proceedings.* Lecture Notes in Computer Science, vol. 7926, Springer-Verlag GmbH, pp. 279-288. https://doi.org/10.1007/978-3-642-39112-5_29

Digital Object Identifier (DOI):
10.1007/978-3-642-39112-5_29

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Artificial Intelligence in Education

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 22. Sep. 2021

Dzikovska, M. O., Farrow, E., & Moore, J. D. (2013). Combining Semantic Interpretation and Statistical Classification for Improved Explanation Processing in a Tutorial Dialogue System. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), Artificial Intelligence in Education. (pp. 279-288). (Lecture Notes in Computer Science). Springer Berlin / Heidelberg. 10.1007/978-3-642-39112-5_29

Combining semantic interpretation and statistical classification for improved explanation processing in a tutorial dialogue system

Myroslava O. Dzikovska, Elaine Farrow, and Johanna D. Moore*

School of Informatics, University of Edinburgh, Edinburgh, United Kingdom
{m.dzikovska,elaine.farrow,j.moore}@ed.ac.uk

Abstract. We present an approach for combining symbolic interpretation and statistical classification in the natural language processing (NLP) component of a tutorial dialogue system. Symbolic NLP approaches support dynamic generation of context-adaptive natural language feedback, but lack robustness. In contrast, statistical classification approaches are robust to ill-formed input but provide less detail for context-specific feedback generation. We describe a system design that combines symbolic interpretation with statistical classification to support context-adaptive, dynamically generated natural language feedback, and show that the combined system significantly improves interpretation quality while retaining the adaptivity benefits of a symbolic interpreter.

Keywords: Tutorial dialogue, natural language processing, Intelligent Tutoring System (ITS), parsing, semantic interpretation

1 Introduction

In recent years, there has been considerable research on tutorial dialogue systems that accept natural language input and engage in dialogue with students to help them improve their answers [1, 4, 12, 13, 15, 17, 20, 23]. Such systems are designed to allow students to express their answers in their own words, thus encouraging knowledge construction and harnessing the power of self-explanation [3].

One of the challenges in developing effective natural language processing (NLP) modules for tutorial dialogue is finding the right balance between level of detail and robustness. Tutorial dialogue systems aim to provide help and feedback in natural language using a wide range of tutoring tactics. Ideally, system responses will be generated dynamically, taking into account multiple factors, including the current answer diagnosis, dialogue history, and information from the student model such as student ability and motivation. In practice, a system's ability to produce such responses depends on the level of detail provided by the NLP component in its analysis of the student answer.

* This work has been supported in part by US Office of Naval Research grant N000141010085. We would like to thank Natalie Steinhauser, Gwendolyn Campbell, Charlie Scott and Simon Caine for help with data collection and annotation.

Many existing tutorial dialogue systems use hand-crafted semantic interpreters to link natural language input with their domain models, in order to produce fine-grained representations of student input [1, 2, 4, 11, 20]. Such symbolic NLP systems can support dynamic feedback generation by implementing a library of abstract tutorial strategies, and then, for each new problem or situation, producing a feedback message tailored to the context by choosing a strategy to use and instantiating it from the information gathered from the student answer (see Section 2). However, while such systems offer high precision in interpreting user input, they also suffer from recall and robustness problems, and often struggle to achieve adequate performance in large domains.

In contrast, statistical NLP systems use classifiers based on semantic similarity or textual entailment methods to assign student answers to classes corresponding to possible states in a finite-state machine [12, 13, 17, 23]. The classifiers are trained on large corpora, making these methods more robust to unexpected input – an advantage when building systems for large domains. However, the classes they use typically do not provide the fine-grained detail needed to generate natural language feedback dynamically. Therefore, system designers must pre-author feedback messages for each problem and tutoring tactic combination (see Section 3.1), which often limits the range of implemented feedback actions.

In this paper, we investigate how the robustness of a semantic interpreter within a symbolic NLP system can be improved with the addition of a similarity-based statistical classifier. Our goal is to address the robustness issues common in symbolic NLP architectures, making such systems more reliable and easier to use in larger domains. This is the first attempt to integrate statistical classification into an architecture built around dynamic natural language generation. Previous work on combining deep and shallow processing methods in tutorial dialogue [14, 21] targeted finite-state systems with manually authored feedback.

We show that our combined system achieves significantly higher performance than the semantic interpreter alone. The best results are achieved by using the classifier to label sentences that the interpreter cannot handle, thus combining the strengths of the two techniques to improve overall system robustness.

The rest of the paper is organized as follows. In Section 2 we describe how semantic interpretation is implemented in the Beetle II tutorial dialogue system. In Section 3 we examine how statistical classification can be integrated into a system architecture based on symbolic NLP. We then describe the semantic-similarity based classifier we developed and report the results of experimental evaluation in Section 4. We discuss future system improvements in Section 5.

2 Background

As our test environment, we use the Beetle II tutorial dialogue system [4], developed to teach concepts in basic electricity and electronics to students without prior knowledge of the domain. The system provides a three-hour self-contained course where students read pre-prepared instructional materials and interact with a circuit simulator. During the interaction, they are asked questions about circuit behavior that require one- to two-sentence answers. For example, students may be asked to explain what they observed in the simulator (e.g., "Why was bulb A on when switch Y was open?") or to describe general principles (e.g., "Why does a damaged bulb impact a circuit?"). Over the duration of the course, the system asks 56 different explanation questions, each followed by a remediation dialogue if the student's initial answer is flawed.[1]

The system was designed to support fully automatic feedback generation in a dynamically changing context. Each student answer is parsed by a robust wide-coverage dialogue parser and then mapped into a domain-specific semantic representation using a set of hand-crafted rules [9]. For example, if the student responds to "Why was bulb A on when switch Y was open?" by answering "Bulb A was in a closed path", the representation will be (with some details simplified for exposition purposes) `(Bulb A) (Path p) (is-closed p TRUE) (contains p A)`. This representation is first passed on to the circuit simulator to verify that the named bulb is indeed contained in a closed path. Next, the system checks the explanation content for correctness by matching it against a pattern based on the reference explanation supplied by expert tutors, in this instance `(Bulb ?b) (Battery ?bt) (Path ?p) (is-closed ?p TRUE) (contains ?p ?b) (contains ?p ?bt)`. The resulting diagnosis breaks down the representation of the student answer into correct, missing, contradictory and irrelevant parts [7]. In our example, for a bulb to be lit, it is not enough for it to be in a closed path; there must be a battery in the same path. Therefore, the resulting diagnosis will identify all the objects and relationships mentioned by the student as correct, nothing as contradictory or irrelevant, and will report the missing parts as `(Battery ?bt) (contains p ?bt)`.

The tutorial planner uses the diagnosis to choose from a range of remediation strategies and to instantiate them automatically in context. Most strategies rely on the fine-grained details of the answer analysis for their instantiation; for example, confirming the correct parts of the answer ("Right. The bulb is in a closed path."), hinting at missing bits ("Here's a hint. Your answer should also mention a battery."), or (in another example) explicitly identifying problematic parts ("You said that switch X was closed, but it was open."). But there is also a subset of strategies that require less specific information, such as content-free prompts ("Right, but is that everything?") and suggestions for additional reading. At most points in the interaction, the system can instantiate at least two content-free strategies, and two which require information from the student answer diagnosis and dialogue history. Currently, the system chooses which strategy to use based on past student performance. The general policy is to apply content-free prompts initially, to encourage the students to construct the answer themselves, and provide increasingly more specific remediations if the student is struggling [9]. More complex policies are possible in the future, e.g., adapting the choice of feedback to information in the student model.

The use of deep parsing and semantic interpretation provides significant benefits in this application with its dynamically changing simulation environment,

[1] In this paper, we use "flawed" to denote any answer class other than "correct".

because it enables the system to diagnose student input and generate context-specific natural language feedback on the fly. To mitigate robustness issues associated with rule-based processing, the system uses a robust interpretation algorithm and a set of error recovery strategies [6]. This approach is successful on the whole in helping students learn, resulting in significant learning gains between pre- and post-tests [4]. However, natural language interpretation failures are correlated with lower learning gains and lower user satisfaction, and there is substantial room for improvement in interpretation quality [8]. In this paper, we investigate how the quality of natural language interpretation can be improved through a combination of deep and shallow processing without sacrificing the benefits of detailed semantic analysis.

3 System Design

3.1 Answer Classification Approach

The first challenge in developing a statistical classifier to use in a combined system is determining the set of classes to use, balancing the level of detail provided against the feasibility of acquiring training data. It is possible to induce a semantic parser from annotated data [14, 16]. However, annotating a large number of sentences with domain-specific logical forms is extremely labor-intensive, and even more complicated when dealing with vague and ill-formed student answers.

Classification approaches that have been implemented in existing tutorial dialogue systems typically map student propositions to classes or "correct answer aspects" [12, 18, 21], with each class expressing a single complex idea such as "a bulb is in a closed path with a battery"[2]. Such classes are represented by one or more exemplar strings, and student answers are assigned to classes based on the closest match, using semantic similarity and textual entailment methods. Because the classes are represented by textual strings and not by structured symbolic representations, class assignment cannot be used directly to generate natural language feedback. Instead, manually authored remediations are associated with each class (i.e., correct answer aspect), and multiple such remediations are needed for the system to adapt to context and dialogue history.

Since we intend to use statistical methods to complement symbolic interpretation, we chose to use a set of problem- and representation-independent classes that support the high-level decision-making structure embedded in the BEETLE II tutorial planner. Student answers can be flawed in different ways. They may contain explicit errors, contradicting the expected answer or the state of the world (e.g., saying that a switch is closed when it is open); they may correctly include part of the explanation but miss some crucial aspects; or they may state facts that, while true, are not relevant in explaining the phenomenon in question (e.g., stating that a bulb has two terminals does not explain why it is lit).

[2] A finer-grained, generalizable classification approach has been proposed in [19]. This is a promising avenue of research, but it has not yet been integrated into a running system. We defer further discussion of its applicability until Section 5.

These different types of flaws are associated with different tutoring strategies in the BEETLE II tutorial planner, based on analysis of human-human tutoring data and strategies suggested in the literature. In general, the system rejects answers containing explicit errors and asks students to try again; provides positive feedback on incomplete answers but requests more information; and redirects students' attention through hints if their explanations lack relevance. For every flaw type, the system provides both detailed feedback strategies and the content-free prompts described in Section 2.

We therefore defined an annotation scheme with 5 classes, to be used in answer classification: "correct", "partially-correct-incomplete", "contradictory", "irrelevant" and "non-domain"[3]. If the fine-grained analysis is unavailable, the tutorial planner can use the class to select an appropriate content-free prompt as a fall-back strategy, thus improving its robustness.

3.2 Combining Semantic Interpretation and Classification

Once a suitable classifier is built, we need to decide how to combine its results with the output of the semantic interpreter. To better understand the performance of the BEETLE II interpreter, we previously conducted a system evaluation based on a corpus of paid volunteers interacting with the system. Every student answer was manually annotated using our five class coding scheme ($\kappa = 0.69$), and the associated semantic interpretation and diagnosis output from the BEETLE II system was automatically mapped to the same scheme [5]. This annotation enables us to directly compare the performance of the semantic interpreter with that of the classifier, and identify areas for improvement.

In our previous work, we devised a classifier based on lexical similarity and evaluated it alongside the BEETLE II semantic interpreter [5, 10]. The interpreter had a higher precision but substantially lower recall than the statistical classifier, indicating that the two approaches have complementary strengths and weaknesses.

Based on the evaluation results in [10], we identified two key performance issues with the semantic interpreter that we would particularly like to address. First, the interpreter fails to find any interpretation at all for a large proportion of answers to explanation questions (865 out of 2729 instances, or 32%, according to the confusion matrix reported in [10]). We will refer to those cases as "uninterpretable utterances". Second, out of the answers that the system can interpret, a large proportion of "correct" and "contradictory" answers are misinterpreted as "partially-correct-incomplete". Students can feel frustrated if their correct answers are misinterpreted or rejected, and in general when their answers

[3] Students make help requests, social statements and other utterances that do not contribute any domain content to the dialogue, although the tutor has to respond to them nevertheless. These are labeled as "non-domain".

are not understood. Therefore, we attempted to address these issues by testing three combinations of semantic interpretation and statistical classification:[4]

1. `OptimisticCorrect`: if the classifier labels the answer as correct, then the classifier's label is used; otherwise, the label from the semantic interpreter is used. This combination creates a more lenient system that aims to avoid misidentifying correct answers, a known cause of student frustration.
2. `NoReject`: if the semantic interpreter fails to arrive at an interpretation, then the classifier's label is used; otherwise, the label from the semantic interpreter is used. This combination creates a system that never rejects student answers as uninterpretable.
3. `NoRejectCorrect`: if both of the previous conditions hold (the classifier labels the answer as correct and the semantic interpreter fails to find an interpretation), then the classifier's label is used; in all other cases, the label from the semantic interpreter is used. This combination is a more conservative version of the `NoReject` system.

These three different ways of combining the output of the semantic interpreter and the classifier each have advantages and disadvantages. Being more lenient in grading student answers as correct may help improve user satisfaction but risks missing opportunities to correct misconceptions and provide useful remediation. Never rejecting answers as uninterpretable can reduce student frustration. However, uninterpretable utterances often arise from incorrect uses of terminology, and learning to speak in the way expected for the domain has been positively correlated with learning outcomes [22]. The semantic interpreter provides information about the nature of interpretation failures that supports generation of targeted help messages, pointing out problematic wordings not consistent with the domain, such as "Paths cannot be broken, only components can be broken." [6]. Some students may benefit from seeing such rejection messages.

Choosing the best trade-off may depend on the high-level tutoring policy and the application domain. However, it is important to evaluate how different combinations affect the overall quality of natural language interpretation, which affects interaction quality as a whole. This is the focus of the rest of the paper.

4 Evaluation

4.1 Experimental Setup

For this experiment, we used the Beetle portion of the Student Response Analysis task corpus[5], which is an updated version of the gold standard evaluation corpus from [10]. This dataset consists of 3426 student answers to explanation questions

[4] In addition to these rule-based combinations, we also attempted to learn the best combinations directly from the data. Our experiments so far have not resulted in improved performance, so this remains a topic for future work.

[5] `http://www.cs.york.ac.uk/semeval-2013/task7/index.php?id=data`

collected from the interactions of 35 paid undergraduate volunteers working with the Beetle II system.

The Beetle II semantic interpreter was developed based on transcripts from an earlier version of the system which were not included in our evaluation corpus. Thus, this corpus constitutes unseen data for the semantic interpreter.

We used 10-fold cross-validation to evaluate the performance of the standalone classifier and the combined systems. At every iteration, we used 9 folds to train the statistical classifier, and the 10th fold as a test set for the system using it. We report the per-class precision, recall and F1 scores as evaluation metrics, following [10]. We use the macro-averaged F1 score as the primary evaluation metric because it is suitable for evaluating unbalanced class distributions, requiring that the system performs well on identifying all possible classes and does not only focus on the most frequent cases.

In all our combined systems, we use the simple lexical similarity classifier described in [5]. While more sophisticated approaches are available [18, 21], the simple features that we use are fast to compute and do not require additional external resources. Our goal is to produce a lightweight approach that complements the more resource-intensive symbolic interpretation. In future, more advanced features can be considered to further enhance system performance.

4.2 Results

Table 1 shows the performance of the semantic interpreter and our classifier taken alone. Both perform at the same overall level (0.45 macro-averaged F1), but the semantic interpreter has substantially higher precision and lower recall. Thus, the systems have complementary strengths and weaknesses, suggesting that improved performance may be possible by combining the approaches.

Table 2 presents evaluation results for the three combination systems described in Section 3.2. The performance of each of the combined systems differs significantly from the standalone semantic interpreter, with $p < 0.001$ on an approximate randomization test with 10,000 permutations [24].

The best performance improvement is achieved by the `NoReject` system, where the classifier's label is used whenever symbolic interpretation fails, raising the system's macro-averaged F1 from 0.45 to 0.54. Performance improves across all classes, with the largest improvements in "contradictory" and "non-domain". Although this system experiences a drop in precision, resulting in more misidentified classes, it is accompanied by a significant increase in recall, since no utterances are rejected as uninterpretable.

In contrast, the `OptimisticCorrect` system, which always accepts a student answer as correct if the classifier judges it correct, results in significantly reduced performance compared to the semantic interpreter alone (0.43 F1), with precision on identifying correct answers dropping from 0.94 to 0.65, and recall not increasing sufficiently to compensate for the drop. Finally, the more conservative `NoRejectCorrect` system, which only overrides the semantic interpreter if both the interpretation fails and the classifier judges the answer correct, provides a

	Semantic interpreter			Statistical classifier		
	P	R	F1	P	R	F1
correct	0.94	0.50	0.66	0.64	0.78	0.70
pc_inc	0.45	0.51	0.48	0.42	0.35	0.38
contra	0.54	0.18	0.27	0.44	0.36	0.40
irrlvnt	0.21	0.21	0.20	0.09	0.03	0.05
nondom	0.90	0.51	0.65	0.63	0.84	0.73
macro avg	0.60	0.38	0.45	0.45	0.47	0.45

Table 1. Evaluation results for the semantic interpreter alone and the classifier alone.

	`OptimisticCorrect`			`NoReject`			`NoRejectCorrect`		
	P	R	F1	P	R	F1	P	R	F1
correct	0.65	0.85	0.74	0.75	0.66	0.70	0.76	0.66	0.70
pc_inc	0.54	0.31	0.40	0.43	0.64	0.51	0.45	0.51	0.48
contra	0.56	0.09	0.16	0.56	0.40	0.46	0.54	0.18	0.28
irrlvnt	0.22	0.19	0.21	0.20	0.24	0.22	0.21	0.21	0.21
nondom	0.93	0.51	0.66	0.76	0.89	0.82	0.90	0.51	0.65
macro avg	0.58	0.39	0.43	**0.54**	**0.57**	**0.54**	0.57	0.42	0.46

Table 2. Evaluation results for three different system combinations.

small (though still significant) boost in performance compared to the semantic interpreter alone.

These results show that symbolic interpretation and statistical classification can be effectively combined in a system architecture geared towards automatic generation of targeted feedback. We discuss the trade-offs involved and future improvements in the next section.

5 Discussion and Future Work

This paper presents a first attempt at combining a symbolic semantic interpreter and a statistical classifier in the context of a tutorial dialogue system which generates natural language feedback dynamically based on detailed semantic analysis of student contributions. In our evaluation, the rule-based semantic interpreter and the lexical similarity-based statistical classifier perform similarly as stand-alone systems, but can be combined to improve performance significantly by using the statistical classifier to label utterances rejected as uninterpretable by the semantic interpreter.

Unlike previous approaches to statistical natural language understanding in tutorial dialogue, we use a simple set of five correctness classes that apply to all questions, and do not depend on "correct answer aspects" specific to the problem. Assigning one of these classes is sufficient to allow the system to employ a subset of its tutoring strategies, namely, content-free prompts, in situations where the semantic interpreter cannot reliably provide the fine-grained semantic representations necessary for instantiating more specific strategies.

Nielsen et al. [19] show how to obtain more fine-grained information about correct, incorrect and missing parts of student answers using a statistical classification approach. This presents an interesting avenue for future work, as such an approach could potentially enable the system to use a wider range of dynamically generated strategies. However, the finer-grained classification scheme also requires correspondingly more annotation effort, since each student answer must be annotated with 10 labels on average. Our approach is less labor-intensive with respect to annotation, at the cost of having less specific information available.

In the three combination systems that we tried, we found the greatest improvement in language interpretation accuracy when using the classifier only on utterances which the symbolic interpreter rejected as having no interpretation. In contrast, relying on the classifier's "correct" label, which was an attempt to compensate for the large number of correct answers mislabeled by the interpreter, did not improve system performance. This system combination might become more effective if more sophisticated approaches, especially textual entailment methods, were used in the classifier. We are considering the best techniques to use as part of our future work.

The next step in system development is to test the new robust interpreter with users, to see whether improved robustness translates into improvements in end-to-end system performance. While there is clearly a link between interpretation quality and both learning gain and user satisfaction [8], intrinsic evaluation metrics alone are not always good predictors of final outcomes [5]. We are planning to use our robust interpretation module in an upcoming user evaluation, and will assess its contribution by comparing the learning outcomes obtained with the new system to the results of the previous evaluation where less sophisticated NLP was used.

References

1. Aleven, V., Popescu, O., Koedinger, K.R.: Pilot-testing a tutorial dialogue system that supports self-explanation. In: Proc. of ITS-02 conference. pp. 344–354 (2002)
2. Callaway, C., Dzikovska, M., Matheson, C., Moore, J., Zinn, C.: Using dialogue to learn math in the LeActiveMath project. In: Proc. of ECAI Workshop on Language-Enhanced Educational Technology. pp. 1–8 (2006)
3. Chi, M.T.H., de Leeuw, N., Chiu, M.H., LaVancher, C.: Eliciting self-explanations improves understanding. Cognitive Science 18(3), 439–477 (1994)
4. Dzikovska, M., Bental, D., Moore, J.D., Steinhauser, N.B., Campbell, G.E., Farrow, E., Callaway, C.B.: Intelligent tutoring with natural language support in the Beetle II system. In: Proc. of ECTEL-2010. pp. 620–625 (2010)
5. Dzikovska, M.O., Bell, P., Isard, A., Moore, J.D.: Evaluating language understanding accuracy with respect to objective outcomes in a dialogue system. In: Proc. of EACL-12 Conference. pp. 471–481 (2012)
6. Dzikovska, M.O., Callaway, C.B., Farrow, E., Moore, J.D., Steinhauser, N.B., Campbell, G.E.: Dealing with interpretation errors in tutorial dialogue. In: Proc. of SIGDIAL 2009 Conference. pp. 38–45 (2009)
7. Dzikovska, M.O., Campbell, G.E., Callaway, C.B., Steinhauser, N.B., Farrow, E., Moore, J.D., Butler, L.A., Matheson, C.: Diagnosing natural language answers to support adaptive tutoring. In: Proc. of 21st Intl. FLAIRS Conference (2008)

8. Dzikovska, M.O., Moore, J.D., Steinhauser, N., Campbell, G.: The impact of interpretation problems on tutorial dialogue. In: Proc. of ACL 2010 Conference Short Papers. pp. 43–48 (2010)
9. Dzikovska, M.O., Moore, J.D., Steinhauser, N., Campbell, G., Farrow, E., Callaway, C.B.: Beetle II: a system for tutoring and computational linguistics experimentation. In: Proc. of ACL 2010 System Demonstrations. pp. 13–18 (2010)
10. Dzikovska, M.O., Nielsen, R.D., Brew, C.: Towards effective tutorial feedback for explanation questions: A dataset and baselines. In: Proc. of 2012 Conference of NAACL: Human Language Technologies. pp. 200–210 (2012)
11. Glass, M.: Processing language input in the CIRCSIM-Tutor intelligent tutoring system. In: Papers from the 2000 AAAI Fall Symposium, Available as AAAI technical report FS-00-01. pp. 74–79 (2000)
12. Graesser, A.C., Wiemer-Hastings, K., Wiemer-Hastings, P., Kreuz, R.: Autotutor: A simulation of a human tutor. Cognitive Systems Research 1, 35–51 (1999)
13. Jordan, P., Makatchev, M., Pappuswamy, U., VanLehn, K., Albacete, P.: A natural language tutorial dialogue system for physics. In: Proc. of 19th Intl. FLAIRS conference. pp. 521–527 (2006)
14. Jordan, P.W., Makatchev, M., VanLehn, K.: Combining competing language understanding approaches in an intelligent tutoring system. In: Proc. of Intelligent Tutoring Systems Conference. pp. 346–357 (2004)
15. Khuwaja, R.A., Evens, M.W., Michael, J.A., Rovick, A.A.: Architecture of CIRCSIM-tutor (v.3): A smart cardiovascular physiology tutor. In: Proc. of 7th Annual IEEE Computer-Based Medical Systems Symposium (1994)
16. Kwiatkowski, T., Zettlemoyer, L., Goldwater, S., Steedman, M.: Inducing probabilistic CCG grammars from logical form with higher-order unification. In: Proc. of EMNLP-2010 Conference. pp. 1223–1233 (2010)
17. Litman, D.J., Silliman, S.: ITSPOKE: an intelligent tutoring spoken dialogue system. In: Demonstration Papers at HLT-NAACL 2004. pp. 5–8. Boston, Massachusetts (2004)
18. McCarthy, P.M., Rus, V., Crossley, S.A., Graesser, A.C., McNamara, D.S.: Assessing forward-, reverse-, and average-entailment indices on natural language input from the intelligent tutoring system, iSTART. In: Proc. of 21st Intl. FLAIRS conference. pp. 165–170 (2008)
19. Nielsen, R.D., Ward, W., Martin, J.H.: Learning to assess low-level conceptual understanding. In: Proc. of 21st Intl. FLAIRS Conference. pp. 427–432 (2008)
20. Pon-Barry, H., Clark, B., Schultz, K., Bratt, E.O., Peters, S.: Advantages of spoken language interaction in dialogue-based intelligent tutoring systems. In: Proc. of ITS-2004 Conference. pp. 390–400 (2004)
21. Rosé, C., Vanlehn, K.: An evaluation of a hybrid language understanding approach for robust selection of tutoring goals. Int. J. Artif. Intell. Ed. 15(4), 325–355 (2005)
22. Steinhauser, N.B., Campbell, G.E., Taylor, L.S., Caine, S., Scott, C., Dzikovska, M.O., Moore, J.D.: Talk like an electrician: Student dialogue mimicking behavior in an intelligent tutoring system. In: Proc. of 15th international conference on Artificial Intelligence in Education. pp. 361–368 (2011)
23. VanLehn, K., Jordan, P., Litman, D.: Developing pedagogically effective tutorial dialogue tactics: Experiments and a testbed. In: Proc. of SLaTE Workshop on Speech and Language Technology in Education. Farmington, PA (October 2007)
24. Yeh, A.: More accurate tests for the statistical significance of result differences. In: Proceedings of the 18th International Conference on Computational linguistics (COLING 2000). pp. 947–953. Association for Computational Linguistics, Stroudsburg, PA, USA (2000), `http://dx.doi.org/10.3115/992730.992783`