Dealing with Interpretation Errors in Tutorial Dialogue

Myroslava O. Dzikovska, Charles B. Callaway, Elaine Farrow, Johanna D. Moore
School of Informatics
University of Edinburgh, Edinburgh, United Kingdom
mdzikovs,ccallawa,efarrow,jmoore@inf.ed.ac.uk

Natalie Steinhauser, Gwendolyn Campbell
Naval Air Warfare Training Systems Division
Orlando, Florida, USA

Abstract

We describe an approach to dealing with interpretation errors in a tutorial dialogue system. Allowing students to provide explanations and generate contentful talk can be helpful for learning, but the language that can be understood by a computer system is limited by the current technology. Techniques for dealing with understanding problems have been developed primarily for spoken dialogue systems in information-seeking domains, and are not always appropriate for tutorial dialogue. We present a classification of interpretation errors and our approach for dealing with them within an implemented tutorial dialogue system.

1 Introduction

Error detection and recovery is a known problem in the spoken dialogue community, with much research devoted to determining the best strategies, and learning how to choose an appropriate strategy from data. Most existing research is focused on dealing with problems in an interaction resulting from speech recognition errors. This focus is justified, since the majority of understanding problems observed in current spoken dialogue systems (SDS) are indeed due to speech recognition errors.

Recovery strategies, therefore, are sometimes devised specifically to target speech recognition problems - for example, asking the user to repeat the utterance, or to speak more softly, which only makes sense if speech recognition is the source of trouble. However, errors can occur at all levels of processing, including parsing, semantic interpretation, intention recognition, etc. As speech recognition improves and more sophisticated systems are developed, strategies for dealing with errors coming from higher (and potentially more complex) levels of processing will have to be developed.

This paper presents a classification of *non-understandings*, defined as the errors where the system fails to arrive at an interpretation of the user's utterance (Bohus and Rudnicky, 2005), and a set of strategies for dealing with them in an implemented tutorial dialogue system. Our system differs from many existing systems in two ways. First, all dialogue is typed. This was done in part to avoid speech recognition issues and allow for more complex language input than would otherwise be possible. But it is also a valid modality for tutoring - there are now many GUI-based tutoring systems in existence, and as distance and online learning have become more popular, students are increasingly familiar with typed dialogue in chat rooms and discussion boards. Second, different genres impose different constraints on the set of applicable recovery strategies - as we discuss in Section 2, certain help strategies developed for task-oriented dialogue systems are not suitable for tutorial dialogue, because tutoring systems should not give away the answer.

We propose a targeted help approach for dealing with interpretation problems in tutorial dialogue by providing help messages that target errors at different points in the pipeline. In our system they are combined with hints as a way to lead the student to an answer that can be understood. While some parts of the system response are specific to tutorial dialogue, the targeted help messages themselves can serve as a starting point for developing appropriate recovery strategies in other systems where errors at higher levels of interpretation are a problem.

The rest of this paper is organized as follows. In Section 2, we motivate the need for error handling strategies in tutorial dialogue. In Section 3 we describe the design of our system. Section 4 discusses a classification of interpretation problems and our targeted help strategy. Section 5 provides a preliminary evaluation based on a set of system tests conducted to date. Finally, we discuss how the approach taken by our system compares to other systems.

2 Background and Motivation

Tutorial dialogue systems aim to improve learning by engaging students in contentful dialogue. There is a mounting body of evidence that dialogue which encourages students to explain their actions (Aleven and Koedinger, 2000), or to generate contentful talk (Purandare and Litman, 2008), results in improved learning. However, the systems' ability to understand student language, and therefore to encourage contentful talk, is limited by the state of current language technology. Moreover, student language may be particularly difficult to interpret since students are often unaware of proper terminology, and may phrase their answers in unexpected ways. For example, a recent error analysis for a domain-independent diagnoser trained on a large corpus showed that a high proportion of errors were due to unexpected paraphrases (Nielsen et al., 2008).

In small domains, domain-specific grammars and lexicons can cover most common phrasings used by students to ensure robust interpretation (Aleven, 2003; Glass, 2000). However, as the size of the domain and the range of possible questions and answers grows, achieving complete coverage becomes more difficult. For essays in large domains, statistical methods can be used to identify problems with the answer (Jordan et al., 2006; Graesser et al., 1999), but these approaches do not perform well on relatively short single-sentence explanations, and such systems often revert to short-answer questions during remediation to ensure robustness.

To the best of our knowledge, none of these tutorial systems use sophisticated error handling techniques. They rely on the small size of the domain or simplicity of expected answers to limit the range of student input. They reject utterances they cannot interpret, asking the user to repeat or rephrase, or tolerate the possibility that interpretation problems will lead to repetitive or confusing feedback.

We are developing a tutorial dialogue system that behaves more like human tutors by supporting open-ended questions, as well as remediations that allow for open-ended answers, and gives students detailed feedback on their answers, similar to what we observed with human tutors. This paper takes the first step towards addressing the problem of handling errors in tutorial dialogue by developing a set of non-understanding recovery strategies - i.e. strategies used where the system cannot find an interpretation for an utterance.

In early pilot experiments we observed that if the system simply rejects a problematic student utterance, saying that it was not understood, then students are unable to determine the reason for this rejection. They either resubmit their answer making only minimal changes, or else they rephrase the sentence in a progressively more complicated fashion, causing even more interpretation errors. Even after interacting with the system for over an hour, our students did not have an accurate picture as to which phrasings are well understood by the system and which should be avoided. Previous research also shows that users are rarely able to perceive the true causes of ASR errors, and tend to form incorrect theories about the types of input a system is able to accept (Karsenty, 2001).

A common approach for dealing with these issues in spoken dialogue systems is to either change to system initiative with short-answer questions ("Is your destination London?"), or provide targeted help ("You can say plane, car or hotel"). Neither of these is suitable for our system. The expected utterances in our system are often more complex (e.g., "The bulb must be in a closed path with the battery"), and therefore suggesting an utterance may be equivalent to giving away the entire answer. Giving students short-answer questions such as "Are the terminals connected or not connected?" is a valid tutoring strategy sometimes used by the tutors. However, it changes the nature of the question from a recall task to a recognition task, which may affect the student's ability to remember the correct solution independently. Therefore, we decided to implement strategies that give the student information about the nature of the mistake without directly giving information about the expected answer, and encourage them to rephrase their answers in ways that can be understood by the system.

We currently focus on strategies for dealing with non-understanding rather than misunderstanding strategies (i.e. cases where the system finds an interpretation, but an incorrect one). It is less clear in tutorial dialogue what it means for a misunderstanding to be corrected. In task-oriented dialogue, if the system gets a slot value different from what the user intended, it should make immediate corrections at the user's request. In tutoring, however, it is the system which knows the expected correct answer. So if the student gives an answer that does not match the expected answer, when they try to correct it later, it may not always be obvious whether the correction is due to a true misunderstanding, or due to the student arriving at a better understanding of the question. Obviously, true misunderstandings can and will still occur - for example, when the system resolves a pronoun incorrectly. Dealing with such situations is planned as part of future work.

3 System Architecture

Our target application is a system for tutoring basic electricity and electronics. The students read some introductory material, and interact with a simulator where they can build circuits using batteries, bulbs and switches, and measure voltage and current. They are then asked two types of questions: factual questions, like "If the switch is open, will bulb A be on or off?", and explanation questions. The explanation questions ask the student to explain what they observed in a circuit simulation, for example, "Explain why you got the voltage of 1.5 here", or define generic concepts, such as "What is voltage?". The expected answers are fairly short, one or two sentences, but they involve complex linguistic phenomena, including conjunction, negation, relative clauses, anaphora and ellipsis.

The system is connected to a knowledge base which serves as a model for the domain and a reasoning engine. It represents the objects and relationships the system can reason about, and is used to compute answers to factual questions.[1] The student answers are processed using a standard NLP pipeline. All utterances are parsed to obtain syntactic analyses.[2] The lexical-semantic interpreter takes analyses from the parser and maps them to semantic representations using concepts from the domain model. A reference resolution algorithm similar to (Byron, 2002) is used to find referents for named objects such as "bulb A" and for pronouns.

Once an interpretation of a student utterance has been obtained, it is checked in two ways. First, its internal consistency is verified. For example, if the student says "Bulb A will be on because it is in a closed path", we first must ensure that their answer is consistent with what is on the screen - that bulb A is indeed in a closed path. Otherwise the student probably has a problem either with understanding the diagrams or with understanding concepts such as "closed path". These problems indicate lack of basic background knowledge, and need to be remediated using a separate tutorial strategy.

Assuming that the utterance is consistent with the state of the world, the explanation is then checked for correctness. Even though the student utterance may be factually correct (Bulb A is indeed in a closed path), it may still be incomplete or irrelevant. In the example above, the full answer is "Bulb A is in a closed path with the battery", hence the student explanation is factually correct but incomplete, missing the mention of the battery.

In the current version of our system, we are particularly concerned about avoiding misunderstandings, since they can result in misleading tutorial feedback. Consider an example of what can happen if there is a misunderstanding due to a lexical coverage gap. The student sentence "the path is broken" should be interpreted as "the path is no longer closed", corresponding to the `is-open` relation. However, the

[1]Answers to explanation questions are hand-coded by tutors because they are not always required to be logically complete (Dzikovska et al., 2008). However, they are checked for consistency as described later, so they have to be expressed in terms that the knowledge base can reason about.

[2]We are using a deep parser that produces semantic analyses of student's input (Allen et al., 2007). However, these have to undergo further lexical interpretation, so we are treating them as syntactic analyses for purposes of this paper.

most frequent sense of "broken" is `is-damaged`, as in "the bulb is broken". Ideally, the system lexicon would define "broken" as ambiguous between those two senses. If only the "damaged" sense is defined, the system will arrive at an incorrect interpretation (misunderstanding), which is false by definition, as the `is-damaged` relation applies only to bulbs in our domain. Thus the system will say "you said that the path is damaged, but that's not true". Since the students who used this phrasing were unaware of the proper terminology in the first instance, they dismissed such feedback as a system error. A more helpful feedback message is to say that the system does not know about damaged paths, and the sentence needs to be rephrased.[3]

Obviously, frequent non-understanding messages can also lead to communication breakdowns and impair tutoring. Thus we aim to balance the need to avoid misunderstandings with the need to avoid student frustration due to a large number of sentences which are not understood. We approach this by using robust parsing and interpretation tools, but balancing them with a set of checks that indicate potential problems. These include checking that the student answer fits with the sortal constraints encoded in the domain model, that it can be interpreted unambiguously, and that pronouns can be resolved.

4 Error Handling Policies

All interpretation problems in our system are handled with a unified tutorial policy. Each message to the user consists of three parts: a social response, the explanation of the problem, and the tutorial response. The social response is currently a simple apology, as in "I'm sorry, I'm having trouble understanding." Research on spoken dialogue shows that users are less frustrated if systems apologize for errors (Bulyko et al., 2005).

The explanation of the problem depends on the problem itself, and is discussed in more detail below.

The tutorial response depends on the general tutorial situation. If this is the first misunderstanding, the student will be asked to rephrase/try again. If they continue to phrase things in a way that is misunderstood, they will be given up to two different hints (a less specific hint followed by a more specific hint); and finally the system will bottom out with a correct answer. Correct answers produced by the generator are guaranteed to be parsed and understood by the interpretation module, so they can serve as templates for future student answers.

The tutorial policy is also adjusted depending on the interaction history. For example, if a non-understanding comes after a few incorrect answers, the system may decide to bottom out immediately in order to avoid student frustration due to multiple errors. At present we are using a heuristic policy based on the total number of incorrect or uninterpretable answers. In the future, such policy could be learned from data, using, for example, reinforcement learning (Williams and Young, 2007).

In the rest of this section we discuss the explanations used for different problems. For brevity, we omit the tutorial response from our examples.

4.1 Parse Failures

An utterance that cannot be parsed represents the worst possible outcome for the system, since detecting the reason for a syntactic parse failure isn't possible for complex parsers and grammars. Thus, in this instance the system does not give any description of the problem at all, saying simply "I'm sorry, I didn't understand."

Since we are unable to explain the source of the problem, we try hard to avoid such failures. We use a spelling corrector and a robust parser that outputs a set of fragments covering the student's input when a full parse cannot be found. The downstream components are designed to merge interpretations of the fragments into a single representation that is sent to the reasoning components.

Our policy is to allow the system to use such fragmentary parses when handling explanation questions, where students tend to use complex language. However, we require full parses for factual questions, such as "Which bulbs will be off?" We found that for those simpler questions students are able to easily phrase an acceptable answer, and the lack of a full parse signals some unusually complex language that downstream components are likely to have problems with as well.

[3] This was a real coverage problem we encountered early on. While we extended the coverage of the lexical interpreter based on corpus data, other gaps in coverage may remain. We discuss the issues related to the treatment of vague or incorrect terminology in Section 4.

One risk associated with using fragmentary parses is that relationships between objects from different fragments would be missed by the parser. Our current policy is to confirm the correct part of the student's answer, and prompt for the missing parts, e.g., " Right. The battery is contained in a closed path. And then?" We can do this because we use a diagnoser that explicitly identifies the correct objects and relationships in the answer (Dzikovska et al., 2008), and we are using a deep generation system that can take those relationships and automatically generate a rephrasing of the correct portion of the content.

4.2 Lexical Interpretation Errors

Errors in lexical interpretation typically come from three main sources: unknown words which the lexical interpreter cannot map into domain concepts, unexpected word combinations, and incorrect uses of terminology that violate the sortal constraints encoded in the domain model.

Unknown words are the simplest to deal with in the context of our lexical interpretation policy. We do not require that every single word of an utterance should be interpreted, because we want the system to be able to skip over irrelevant asides. However, we require that if a predicate is interpreted, all its arguments should be interpreted as well. To illustrate, in our system the interpretation of "the bulb is still lit" is `(LightBulb Bulb-1-1) (is-lit Bulb-1-1 true)`. The adverbial "still" is not interpreted because the system is unable to reason about time.[4] But since all arguments of the `is-lit` predicate are defined, we consider the interpretation complete.

In contrast, in the sentence "voltage is the measurement of the power available in a battery", "measurement" is known to the system. Thus, its argument "power" should also be interpreted. However, the reading material in the lessons never talks about power (the expected answer is "Voltage is a measurement of the difference in electrical states between two terminals"). Therefore the unknown word detector marks "power" as an unknown word, and tells the student "I'm sorry, I'm having a problem understanding. I don't know the word *power*."

[4]The lexical interpretation algorithm makes sure that frequency and negation adverbs are accounted for.

The system can still have trouble interpreting sentences with words which are known to the lexical interpreter, but which appear in unexpected combinations. This involves two possible scenarios. First, unambiguous words could be used in a way that contradicts the system's domain model. For example, the students often mention "closed circuit" instead of the correct term "closed path". The former is valid in colloquial usage, but is not well defined for parallel circuits which can contain many different paths, and therefore cannot be represented in a consistent knowledge base. Thus, the system consults its knowledge base to tell the student about the appropriate arguments for a relation with which the failure occurred. In this instance, the feedback will be "I'm sorry, I'm having a problem understanding. I don't understand it when you say that circuits can be closed. Only paths and switches can be closed."[5]

The second case arises when a highly ambiguous word is used in an unexpected combination. The knowledge base uses a number of fine-grained relations, and therefore some words can map to a large number of relations. For example, the word "has" means `circuit-component` in "The circuit has 2 bulbs", `terminals-of` in "The bulb has terminals" and `voltage-property` in "The battery has voltage". The last relation only applies to batteries, but not to other components. These distinctions are common for knowledge representation and reasoning systems, since they improve reasoning efficiency, but this adds to the difficulty of lexical interpretation. If a student says "Bulb A has a voltage of 0.5", we cannot determine the concept to which the word "has" corresponds. It could be either `terminals-of` or `voltage-property`, since each of those relations uses one possible argument from the student's utterance. Thus, we cannot suggest appropriate argument types and instead we indicate the problematic word combination, for example, "I'm sorry, I'm having trouble understanding. I didn't understand *bulb has voltage*."

Finally, certain syntactic constructions involving comparatives or ellipsis are known to be difficult

[5]Note that these error messages are based strictly on the fact that sortal constraints from the knowledge base for the relation that the student used were violated. In the future, we may also want to adjust the recovery strategy depending on whether the problematic relation is relevant to the expected answer.

open problems for interpretation. While we are working on interpretation algorithms to be included in future system versions, the system currently detects these special relations, and produces a message telling the student to rephrase without the problematic construction, e.g., “I’m sorry. I’m having a problem understanding. I do not understand *same as*. Please try rephrasing without the word *as*.”

4.3 Reference Errors

Reference errors arise when a student uses an ambiguous pronoun, and the system cannot find a suitable object in the knowledge base to match, or on certain occasions when an attachment error in a parse causes an incorrect interpretation. We use a generic message that indicates the type of the object the system perceived, and the actual word used, for example, “I’m sorry. I don’t know which switch you’re referring to with *it*.”

To some extent, reference errors are instances of misunderstandings rather than non-understandings. There are actually 2 underlying cases for reference failure: either the system cannot find any referent at all, or it is finding too many referents. In the future a better policy would be to ask the student which of the ambiguous referents was intended. We expect to pilot this policy in one of our future system tests.

5 Evaluation

So far, we have run 13 pilot sessions with our system. Each pilot consisted of a student going through 1 or 2 lessons with the system. Each lesson lasts about 2 hours and has 100-150 student utterances (additional time is taken with building circuits and reading material). Both the coverage of the interpretation component and the specificity of error messages were improved between each set of pilots, thus it does not make sense to aggregate the data from them. However, over time we observed the trend that students are more likely to change their behavior when the system issues more specific messages.

Examples of successful and unsuccessful interactions are shown in Figure 1. In (a), the student used incorrect terminology, and a reminder about how the word “complete” is interpreted was enough to get the conversation back on track.

The dialogue fragment in (b) shows how messages which are not specific enough can cause a breakdown in conversation. The system used an insufficiently specific message at the beginning (omitting the part that says that only switches and paths can be closed). This led the student away from an answer which was nearly correct with slightly imperfect terminology to an answer which was insufficient (it’s not enough for the components to be connected, they have to be in a closed path), and then to rephrase it in a more complicated way that was impossible for the system to understand (consistent with findings of Bulyko et al. (2005)).

The next step would be to conduct a formal evaluation of our policy. We are planning to do this using recovery rate after different strategies as our evaluation measure (Bohus and Rudnicky, 2005), which is the percentage of times when the utterance immediately following a non-understanding was successfully interpreted by the system.

6 Discussion

Our error handling policy attempts to balance robustness in interpretation with the need for precise tutorial feedback. Human tutors accept a wide range of variation in terminology and phrasing. However, there is a known trade-off between non-understandings and misunderstandings. We decided that misunderstandings are likely to cause more serious problems with respect to tutoring. Therefore we set strict understanding requirements and focused on developing a policy that would encourage students to use constructions that the system can interpret reliably. This is also supported by our observations in the human-human data – tutors believe that teaching correct terminology is important, and we observed them saying things like “It’s good to put it into your own words like that. Just for the sake of the lesson, could you also use the term *closed path*?”.

Our approach is similar to the TargetedHelp approach of (Hockey et al., 2003), where in the case of an interpretation failure the task assistant produces a report of what the recognizer heard, a description of the problem, and a possible in-domain utterance as an alternative. In a user evaluation such targeted help messages resulted in significantly lower task completion times compared to a system that produced a generic “I didn’t understand” message, and

(a)		
	S:	the open switch makes the path incomplete
	T:	I'm sorry, I'm having a problem understanding. Paths can't be complete. Only circuits can be complete. Please try again.
	S:	the open switch makes the path open
	T:	Right. An open switch creates a gap.
(b)		
	T:	What are the conditions that are required to make a bulb light up?
	S:	The lightbulb has to be in a closed circuit with a battery
	T:	I'm sorry, I'm having a problem understanding. I don't understand when you say that circuits can be closed. Please try again.
	S:	The lightbulb has to be connected to a battery
	T:	Yes. You correctly mentioned a battery and a bulb. Keep trying.
	S:	Both the positive and negative connectors have to be connected to
	T:	I'm sorry, I'm having a problem understanding. Try rephrasing your answer.
	S:	The lightbulb has to be connected to the positive and negative connectors of the battery
	T:	I'm sorry, I didn't understand. Here's the answer. The battery and the bulb must be contained in the same closed path.

Figure 1: Sample student-system interactions. **S:** is student, **T:** is tutor (system). (a) A successful interaction where the student changes their language; (b) a failed interaction where system feedback was unhelpful.

subjects gradually learned how to talk to the system, reducing the number of misunderstandings over time. This gives us reason to believe that our system can achieve similar effects in tutorial dialogue. While we don't suggest alternative domain utterances due to the tutoring reasons described earlier, the progressively more specific hints serve a similar function. To what extent this impacts learning and interaction with the system will have to be determined in future evaluations.

The error handling in our system is significantly different from systems that analyze user essays because it needs to focus on a single sentence at a time. In a system that does essay analysis, such as AUTOTUTOR (Graesser et al., 1999) or Why2-Atlas (Jordan et al., 2006) a single essay can have many flaws. So it doesn't matter if some sentences are not fully understood as long as the essay is understood well enough to identify at least one flaw. Then that particular flaw can be remediated, and the student can resubmit the essay. However, this can also cause student frustration and potentially affect learning if the student is asked to re-write an essay many times due to interpretation errors.

Previous systems in the circuit domain focused on troubleshooting rather than conceptual knowledge. The SHERLOCK tutor (Katz et al., 1998) used only menu-based input, limiting possible dialogue. Circuit Fix-It Shop (Smith and Gordon, 1997) was a task-oriented system which allowed for speech input, but with very limited vocabulary. Our system's larger vocabulary and complex input result in different types of non-understandings that cannot be resolved with simple confirmation messages.

A number of researchers have developed error taxonomies for spoken dialogue systems (Paek, 2003; Möller et al., 2007). Our classification does not have speech recognition errors (since we are using typed dialogue), and we have a more complex interpretation stack than the domain-specific parsing utilized by many SDSs. However, some types of errors are shared, in particular, our "no parse", "unknown word" and "unknown attachment" errors correspond to *command-level errors*, and our sortal constraint and reference errors correspond to *concept-level errors* in the taxonomy of Möller et al. (2007). This correspondence is not perfect because of the nature of the task - there are no commands in a tutoring system. However, the underlying causes are very similar, and so research on the best way to communicate about system failures would benefit both tutoring and task-oriented dialogue systems. In the long run, we would like to reconcile these different taxonomies, leading to a unified classification of system errors and recovery strategies.

7 Conclusion

In this paper we described our approach to handling non-understanding errors in a tutorial dialogue system. Explaining the source of errors, without giving away the full answer, is crucial to establishing effective communication between the system and the student. We described a classification of common problems and our approach to dealing with different classes of errors. Our experience with pilot studies, as well as evidence from spoken dialogue systems, indicates that our approach can help improve dialogue efficiency. We will be evaluating its impact on both student learning and on dialogue efficiency in the future.

8 Acknowledgments

This work has been supported in part by Office of Naval Research grant N000140810043.

References

V. A. Aleven and K. R. Koedinger. 2000. The need for tutorial dialog to support self-explanation. In *Proc. of AAAI Fall Symposion on Building Dialogue Systems for Tutorial Applications*.

O. P. V. Aleven. 2003. A knowledge-based approach to understanding students' explanations. In *School of Information Technologies, University of Sydney*.

J. Allen, M. Dzikovska, M. Manshadi, and M. Swift. 2007. Deep linguistic processing for spoken dialogue systems. In *Proceedings of the ACL-07 Workshop on Deep Linguistic Processing*.

D. Bohus and A. Rudnicky. 2005. Sorry, i didn't catch that! - an investigation of non-understanding errors and recovery strategies. In *Proceedings of SIGdial-2005*, Lisbon, Portugal.

I. Bulyko, K. Kirchhoff, M. Ostendorf, and J. Goldberg. 2005. Error-correction detection and response generation in a spoken dialogue system. *Speech Communication*, 45(3):271–288.

D. K. Byron. 2002. *Resolving Pronominal Reference to Abstract Entities*. Ph.D. thesis, University of Rochester.

M. O. Dzikovska, G. E. Campbell, C. B. Callaway, N. B. Steinhauser, E. Farrow, J. D. Moore, L. A. Butler, and C. Matheson. 2008. Diagnosing natural language answers to support adaptive tutoring. In *Proceedings 21st International FLAIRS Conference*.

M. Glass. 2000. Processing language input in the CIRCSIM-Tutor intelligent tutoring system. In *Proc. of the AAAI Fall Symposium on Building Dialogue Systems for Tutorial Applications*.

A. C. Graesser, P. Wiemer-Hastings, P. Wiemer-Hastings, and R. Kreuz. 1999. Autotutor: A simulation of a human tutor. *Cognitive Systems Research*, 1:35–51.

B. A. Hockey, O. Lemon, E. Campana, L. Hiatt, G. Aist, J. Hieronymus, A. Gruenstein, and J. Dowding. 2003. Targeted help for spoken dialogue systems: intelligent feedback improves naive users' performance. In *Proceedings of EACL*.

P. Jordan, M. Makatchev, U. Pappuswamy, K. VanLehn, and P. Albacete. 2006. A natural language tutorial dialogue system for physics. In *Proceedings of FLAIRS'06*.

L. Karsenty. 2001. Adapting verbal protocol methods to investigate speech systems use. *Applied Ergonomics*, 32:15–22.

S. Katz, A. Lesgold, E. Hughes, D. Peters, G. Eggan, M. Gordin, and L. Greenberg. 1998. Sherlock 2: An intelligent tutoring system built on the lrdc framework. In C. Bloom and R. Loftin, editors, *Facilitating the development and use of interactive learning environments*. ERLBAUM.

S. Möller, K.-P. Engelbrecht, and A. Oulasvirta. 2007. Analysis of communication failures for spoken dialogue systems. In *Proceedings of Interspeech*.

R. D. Nielsen, W. Ward, and J. H. Martin. 2008. Classification errors in a domain-independent assessment system. In *Proc. of the Third Workshop on Innovative Use of NLP for Building Educational Applications*.

T. Paek. 2003. Toward a taxonomy of communication errors. In *Proceedings of ISCA Workshop on Error Handling in Spoken Dialogue Systems*.

A. Purandare and D. Litman. 2008. Content-learning correlations in spoken tutoring dialogs at word, turn and discourse levels. In *Proc.of FLAIRS*.

R. W. Smith and S. A. Gordon. 1997. Effects of variable initiative on linguistic behavior in human-computer spoken natural language dialogue. *Computational Linguistics*.

J. D. Williams and S. Young. 2007. Scaling POMDPs for spoken dialog management. *IEEE Trans. on Audio, Speech, and Language Processing*, 15(7):2116–2129.