Enabling Non-Linguists to Author Advanced Conversational Interfaces Easily

Carolyn P. Rosé, Carol Pai, Jaime Arguello

Language Technologies Institute / Human-Computer Interaction Institute
Carnegie Mellon University
5000 Forbes Avenue Pittsburgh, PA 15213
cp3a,cpai,jarguell@andrew.cmu.edu

Abstract

The study presented in this paper evaluates current progress in a long term effort to provide behavior oriented authoring tools to user interface developers to facilitate rapid development of advanced language understanding interfaces. The results of our study show promise that authors with very little training in computational linguistics can perform nearly as well as graduate students in a language technologies program at tasks that isolate the individual skills required to use our authoring interface for building language understanding interfaces. We discuss implications of these results as well as current directions in this long term project.

Introduction

In this paper we report on research towards the development of an authoring technology for enabling non-computational linguists to build robust language understanding interfaces easily. This technology could arguably have an impact on a wide range of application areas, such as machine translation, question answering, or data bases. In recent years, natural language interfaces have become more prevalent especially in educational applications. Specifically, in the past 7 years, research on building tutorial dialogue systems has become a well established research area in the intelligent tutoring systems community, and work from this community is making a presence in the computational linguistics community as well (Rosé & Hall, 2004; Core & Moore, 2004; Forbes-Riley & Litman, 2004).

The greatest obstacle that currently makes it impractical for tutorial dialogue systems and other education oriented language technology applications to make use of sophisticated approaches to natural language understanding beyond trivial pattern matching approaches is the tremendous expense involved in authoring and maintaining domain specific knowledge sources. The problem is twofold: first, authoring knowledge sources for semantic interpretation for state-of-the-art core understanding systems, such as CARMEL (Rosé & Hall, 2004; Rosé, 2000) or TRIPS (Allen, Byron, Dzikovska, Ferguson, Galescu, & Stent, 2001) just to name a couple, requires tremendous computational linguistics expertise in order to do well. To date, the majority of research groups developing educational technology do not have ready access to this expertise, although they do have access to general computer programming expertise. Next, authoring and maintaining knowledge sources for sophisticated approaches to language processing requires a tremendous amount of time. Since authoring knowledge sources for language processing is only one concern in developing an effective application, the time required to author these knowledge sources by hand is prohibitive even for computational linguists. Our long term goal is to enable interface programmers to develop natural language understanding interfaces as easy and as fast as it is to build a direct manipulation, graphical interface.

The study presented in this paper evaluates current progress in a long term effort to provide *behavior oriented* authoring tools to user interface developers to facilitate rapid development of advanced language understanding interfaces. A behavior oriented approach is one that provides a layer of abstraction between the author and the necessary linguistic knowledge sources, such as those described in (Rosé, 2000) or (Allen, Byron, Dzikovska, Ferguson, Galescu, & Stent, 2001), freeing up the author to focus on the desired behavior rather than the linguistic details of the knowledge sources that would make this behavior possible. Our interface is behavior oriented because it infers knowledge sources from example texts annotated with the behavior the author desires from the run-time system. A current version of these tools has been pilot tested by trained computational linguists building natural language applications at three different universities. The study reported here evaluates the extent to which the current interface design is accessible to programmers with limited *linguistic* training and where the weaknesses remain, which must be addressed in order to enable the target user population to easily author advanced conversational interfaces.

What we mean by advanced conversational interfaces is that they go beyond the capabilities that are possible using state-of-the-art authoring tools. Current authoring

tools for building semantic knowledge sources, such as are included with GATE (Cunningham et al., 2003), Alembic (Jay et al., 1997), and SGStudio (Wang and Arco, 2003), are tailored for information extraction and similar tasks that emphasize the identification of named entities such as people, locations, organizations, dates, and addresses. While regular expression based recognizers, such as JAPE (Cunningham et al., 2000) used in such systems, are not strictly limited to these standard entity types, it is doubtful that they would be able to handle concepts that express complex relationships between entities, where the complexity in the conceptual representation can be encoded in natural language with a much greater degree of variation.

Our authoring system, built using CARMEL (Rosé, 2000) as a back end, achieves a high level of generalization by inducing patterns that match against a more sophisticated underlying linguistic analysis rather than a stream of words, in order to normalize as much of the variation as possible, and thus reduce the number of patterns that the authored rules must account for. Furthermore, our authoring environment offers greater flexibility in output representation than the context-free rewrite rules produced by previous semantic authoring tools, allowing authors to design their own predicate language representations that are not constrained to follow the surface structure of the input text. This flexibility allows a wider range of linguistic expressions that express the same idea to be represented the same way, which then simplifies the task of the modules that must make decisions based on the representations returned by the language interface. An evaluation of the linguistic capabilities of the tools is presented separately (Rosé & Hall, 2004). In this paper we focus on the authoring interface issues.

Sentence: The man is moving horizontally at a constant velocity with the pumpkin.

Predicate Language Representation:
((velocity id1 man horizontal constant non-zero)
(velocity id2 pumpkin ?dir ?mag-change ?mag-zero)
(rel-value id3 id1 id2 equal))

Gloss: *The constant, nonzero, horizontal velocity of the man is equal to the velocity of the pumpkin.*

Figure 1 Example of a meaning representation that abstracts away from the surface form of an expression.

For example, Figure 1 presents an example from the initial evaluation of our authoring technology that demonstrates the capability of the representation to abstract away from the details of the surface form of the sentence, and thus generalize over a wide range of surface variation that can occur. We used for our evaluation a corpus of essays written by students in response to 5 simple qualitative physics questions such as "If a man is standing in an elevator holding his keys in front of his face, and if the cable holding the elevator snaps and the man then lets go of the keys, what will be the relationship between the position of the keys and that of the man as the elevator falls to the ground? Explain why." A predicate language definition was designed consisting of 40 predicates, 31 predicate types, 160 tokens, 37 token types, and 15 abstract types. The language was meant to be able to represent physical objects mentioned in our set of physics problems, body states (e.g., freefall, contact, non-contact), quantities that can be measured (e.g., force, velocity, acceleration, speed, etc.), features of these quantities (e.g., direction, magnitude, etc.), comparisons between quantities (equivalence, non-equivalence, relative size, relative time, relative location), physics laws, and dependency relations.

While the authoring technology presented in this paper could be used in the development of any application where natural language input is desired, this work is particularly motivated by a need within a growing community of researchers working on educational applications of Natural Language Processing to extract detailed information from student language input. The extracted information is then used for formulating specific feedback for students directed at the details of what they have uttered. Such applications include tutorial dialogue systems (Zinn et al., 2002; Popescue et al., 2003) and writing coaches that perform detailed assessments of writing content (Rosé et al., 2003; Carlson and Tanimoto, 2003).

The Authoring Process

The authoring process involves designing a predicate language definition (that specifies constraints on the output representation) and annotating example texts with their corresponding representation in that defined language. From this authored knowledge, CARMEL's semantic knowledge sources are then generated and compiled automatically for use in the run time system that processes novel texts. After the author tests the compiled knowledge sources, the authoring process may continue by updating or expanding the predicate language definition, annotating additional example texts, or modifying already annotated examples. We have identified six high level basic skills required to use the authoring interface, which we systematically test in our study. Our target user population (people with general programming knowledge but not formal linguistics training) will only be capable of using the designed interface to the extent that they possess these skills. Thus, in order to begin to validate our experimental design, we describe the steps involved in the authoring process and how they relate to the six skills tested in our study.

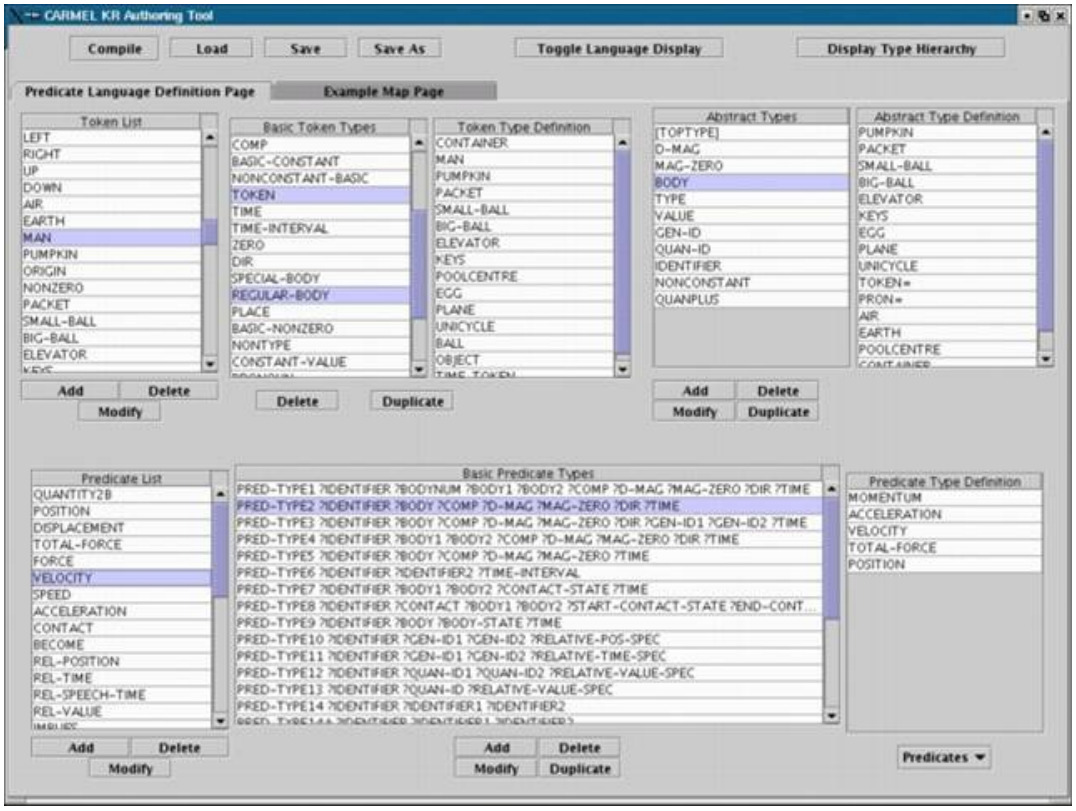

Figure 2 Predicate Language Definition Page

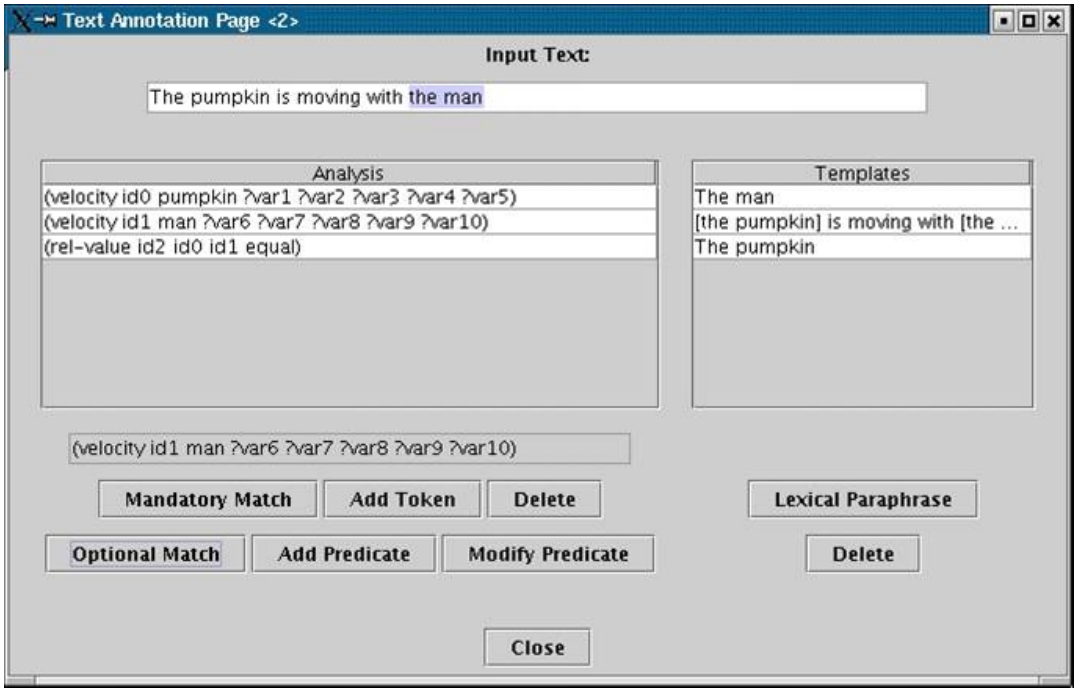

Figure 3 Main Text Annotation Page

The author begins the authoring process by designing the propositional language that will be the output representation from CARMEL using the authored knowledge sources. This is done on a Predicate Language Definition page (displayed in Figure 2) of the GUI interface. The author is completely free to develop a representation language that is as simple or complex as is required by the type of reasoning, if any, that will be applied to the output representations by the back end system. Performing this task requires two of the basic skills we address in our study. First, it requires authors to identify relevant basic units of meaning by identifying alternative phrasings of the same idea as meaning the same thing. Second, it requires authors to formalize the definition of these units of meaning into a predicate logic like language definition.

Figure 3 displays the main page of the interface where individual texts are annotated. The Analysis box displays the propositional representation of the text. This analysis is constructed using the Add Token, Delete, Add Predicate, and Modify Predicate buttons, as well as their subwindows, which are not shown. Once the analysis is entered, the author may indicate the compositional breakdown of the example text by associating spans of text with parts of the analysis by means of the two matching buttons, labeled Optional Match and Mandatory Match. For example, in the very simple example displayed in Figure 3, the phrase "the man" found in the Input Text corresponds to the man token, which is bound in two places in the analysis. The author must highlight the span of text and corresponding span of meaning representation and then click on one of the match buttons. By decomposing example texts in this way, the authoring environment constructs templates that are general and can be reused in multiple annotated examples. These templates contain information from the syntactic analysis of the text that can be used for generating the knowledge sources for the run time system.

The list of templates that form the hierarchical breakdown of this example text are displayed in the Templates list on the right hand side of Figure 3. The templates are displayed in such a way as to hide the details of the linguistic knowledge stored with each template since these are not pertinent to the authoring process. Note that while the author matches spans of text to portions of the meaning representation, the tool stores mappings between detailed linguistic representations and portions of meaning representation, which is a more general mapping that allows the compiled run time system to achieve greater coverage of alternative phrasings of the same idea.

Annotating example texts requires three specific skills. First, it requires authors to recognize how spans of text in example sentences are related to one another, and how the form of the analysis is related to the meaning of the text. Secondly, it requires the author to assign a predicate language representation to the text that expresses its meaning. Finally, authors must be able to match spans of text with portions of the predicate language representation in order to indicate the hierarchical break down of the text.

Other screens not displayed support the maintenance and development process by allowing the author to quickly find already annotated examples in various ways. The GUI interface guides authors in such a way as to prevent them from introducing inconsistencies between knowledge sources. For example, a menu-based interface for entering propositional representations for example texts ensures that the entered representation is well-formed and consistent with the author's predicate language definition. Compiled knowledge sources contain pointers back to the annotated examples that are responsible for their creation. This allows the authoring environment to provide troubleshooting facilities to help authors track down potential sources for incorrect analyses generated from compiled knowledge sources so they can quickly address these problems. A final skill required for effective maintenance and troubleshooting is to be able to predict gaps in the run time system's coverage based on the set of examples that have been annotated. Having this skill would allow authors to be strategic about which new examples they annotate in order to achieve the greatest impact on the overall coverage of the run time system.

Experimental Design

In the previous section we identified six basic skills relevant to the authoring process:

- Skill 1: Identify basic units of meaning.
- Skill 2: Formalize basic units of meaning into predicates, tokens, types, etc.
- Skill 3: Identify how spans of text relate to one another.
- Skill 4: Encode the meaning of a sentence in predicate language.
- Skill 5: Decompose text, matching spans of text with corresponding portions of predicate representation.
- Skill 6: Predict coverage of the compiled run time system from set of annotated examples.

The purpose of our study is to measure how level of formal linguistics instruction is related to ability level on the six identified basic skills. We plan to use this information in several ways. First, it allows us to measure the extent to which we have been successful at insulating authors from linguistic issues that require specialized training to deal with effectively. Second, it allows us to identify which skills are most in need of scaffolding to make the interface easier for non-linguists to use. By observing the types of mistakes non-linguists make we can design scaffolding

with clear vision, knowing which pitfalls to help authors avoid.

11 university students participated in the study with various levels of linguistics training. All 11 subjects had a technical background in computer science or engineering, which is appropriate since our target user population is people with programming expertise but not linguistics or natural language processing training. 4 subjects had never taken any linguistics courses whatsoever. 2 subjects had taken one undergraduate language oriented class in the past (i.e., nature of language or philosophy of language). The remaining 5 subjects were graduate students in a language technologies program. The language technologies students provided an upper bound on expected performance. They represent the current population of users. At the other extreme, the students with no prior linguistics training represent the population of target users. The remaining two students represent a middle ground. We wanted to observe whether their performance would look more like that of the totally inexperienced students or the expert students.

When directly observing users interacting with the tool, it is difficult to separate out the six individual skills because they are not independent. For example, if an author is not able to correctly represent the meaning of a text it could be because of inadequacies in the designed predicate language or in a lack of ability to apply the representation. Thus, in order to test each skill independently, we conducted our study as a pen and paper exercise. For each of the six basis skills we designed a task to assess the ability level of the associated skill independently from the other skills. A description of each task along with an analysis of the results are presented in the next section.

Analysis of Results

Skill 1. For task 1, students were asked to look at a list of 14 sentences and identify spans of text at the sentence level and at the phrase level that are equivalent in meaning. Students were assigned a correct generalization score and an incorrect generalization score to indicate how many equivalences they identified that were either correct or incorrect. Students with no background in linguistics achieved an average of 4.25 for correct generalizations and 3.75 for incorrect generalizations. Students with limited linguistics background achieved an average of 5.5 for correct generalizations and .5 for incorrect generalizations. Graduate students in language technologies achieved an average of 5 for correct generalizations and 0 for incorrect generalizations. Thus, the scores were much more similar between the students with a strong linguistic background and the students with some linguistics background than between those with no linguistics background and those with some, although none of the differences were significant.

Skill 2. For task 2, students were given a meaning representation specification and some examples sentences coded with this representation. They were then given some novel sentences and asked to encode the meaning. Students earned a score between 0 and 1 for each of 5 sentences indicating how close their representation was to an ideal representation. Students with no linguistics background achieved an average score of 3.85, whereas students with some linguistics background achieved a perfect score of 5. The graduate language technologies students achieved an almost perfect score of 4.95. The difference between students with no linguistic background and the other students is significant ($t(9)=3.22$; $p<.05$).

Skill 3. For skill 3, students were asked to select one of two structural analyses for 6 different sentences. Students with no linguistics background achieved an average score of 3.25. Students with some linguistics background achieved an average score of 5. Graduate language technologies students achieved an average score of 5.8. The differences between no background and some background and between some background and expert background were marginal. But the difference between students with no linguistics background and the other students was highly significant. ($t(9)=5.29$, $p<.001$).

Skill 4. For task 4, students were required to design a meaning representation for a new set of sentences on a different topic from the one they were given previously. Students of all backgrounds did surprisingly well at this task. All students designed a representation that was in the correct form and contained most of the required information. The only differences between students were on subtle details pertaining to handling complex comparatives and certain type restrictions on arguments. Surprisingly, the representations designed by the students with limited linguistics background were fully acceptable. However, not all of the graduate language technologies students achieved that level of performance.

Skill 5. For task 5, students were asked to segment texts and match portions of meaning representations to portions of text. There were three sections. In one section students were asked to give a hierarchical bracketing of a sentence. In the second section, students were given a text and its meaning representation. A portion of the meaning representation was highlighted, and the student was required to underline the corresponding piece of text. The third section was the opposite. In other words, part of the text was underlined and students were asked to circle the corresponding portion of the meaning representation. For each section students were assigned a score of 1 to 3 indicating the level of correctness overall. Students whose work was perfect in a section achieved a 3 in that section. If they made no more than 3 mistakes they received a score of 2. Otherwise they received a score of 1. Overall students achieved perfect or nearly perfect in most sections. Students varied as to which of these tasks they

performed better at, although they tended to perform better at the second two tasks, presumably because they are more constrained. Out of 9 possible points, students with no linguistics background achieved a score of 4.5. Students with some linguistics background achieved an average score of 5.5. Graduate language technologies students achieved a score of 6.4. The only significant difference was between students with no linguistics background and the others ($t(9)=2.28$, $p<.05$).

Skill 6. For the final task, students were asked to predict which portions of novel sentences would be covered by a run time system compiled from a given set of annotated examples. Students were universally poor at this task. Out of 7 possible points, students with no linguistic background achieved 2.75, students with some linguistics background achieved 2.5, and graduate language technologies students achieved 3.8. None of the differences were significant.

Conclusions and Current Directions

The results of our study demonstrate that authors with very little training in computational linguistics perform almost identically to the graduate students in a language technologies program at tasks that isolate the individual skills required to use our authoring environment for language interface authoring. Nevertheless, informal user studies involving actual use of the tool had much more disappointing results. The consequence of the general lack of ability to predict how much coverage the knowledge sources inferred from annotated examples left authors without a clear direction or strategy for moving through their corpus. As a result, valuable time was lost from annotating examples that did not yield the maximum amount of new knowledge in the generated knowledge sources. Furthermore, since authors tended not to test the generated knowledge sources as they were annotating examples, errors were difficult for them to track later, despite facilities designed to help them with that task. Finally, although the interface prevented authors from violating the constraints they designed into their predicate language representation, it did not keep authors from annotating similar texts with very different representations, thus introducing a great deal of spurious ambiguity. An additional problem was that authors sometimes decomposed examples in ways that lead to overly general rules, which then lead to incorrect analyses when the overly general rules matched inappropriate examples.

In our current work, we are building upon the insights gained from our formal study and informal observations. We are now working on a new interface design that draws upon the skills that we have learned that our target user population does posses while compensating for the problems we observed in practice.

Acknowledgements

This work was supported in part by Office of Naval Research, Cognitive and Neural Sciences Division Grant N00014-05-1-0043 and NSF ITR EIA-0325054.

References

Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., & Stent, A. 2000. An Architecture for a Generic Dialogue Shell. NLENG: Natural Language Engineering, Cambridge University Press, 6 (3), 1-16.

Burstein, J., Kukich, K., Wolff, S., Lu, C., Chodorow, M., Braden-Harder, L., and Harris, M. D. 1998. Automated scoring using a hybrid feature identification technique. In Proceedings of COLING-ACL'98, pages 206–210.

Carlson, A., and Tanimoto, S. 2003. Learning to identify student preconceptions from text. In Proceedings of the HLT-NAACL 2003 Workshop: Building Educational Applications Using Natural Language Processing.

Core, M. & Moore, J. 2004. Robustness and Fidelity in Natural Language Understanding, Proceedings of SCaNaLu.

Cunningham, H., Maynard, D., and Tablan, V. 2000. Jape: a java annotations patterns engine. Institute for Language, Speech, and Hearing, University of Sheffield, Tech Report CS-00-10.

Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. 2003. Gate: an architecture for development of robust hlt applications. In Recent Advanced in Language Processing.

Forbes-Riley, K. & Litman, D. 2004. Predicting Emotion in Spoken Dialogue From Multiple Knowledge Sources, In the Proceedings of HLT/NAACL '04.

Jay, D., Aberdeen, J., Hirschman, L., Kozierok, R., Robinson, P., and Vilain, M. 1997. Mixed-initiative development of language processing systems. In Proceedings of the Fifth Conference on Applied Natural Language Processing.

Popescue, O., Aleven, V., and Koedinger, K. 2003. A knowledge based approach to understanding students' explanations. In Proceedings of the AI in Education Workshop on Tutorial Dialogue Systems: With a View Towards the Classroom.

Rosé, C. P. 2000. A framework for robust semantic interpretation. In Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics, pages 311–318.

Rosé, C. P., Roque, A., Bhembe, D., and VanLehn, K. 2003. A hybrid text classification approach for analysis of student essays. In Proceedings of the HLT-NAACL 2003 Workshop: Building Educational Applications Using Natural Language Processing.

Rosé, C. P. and Hall, B. 2004. A Little Goes a Long Way: Quick Authoring of Semantic Knowledge Sources for Interpretation, Proceedings of SCaNaLu '04.

Wang, Y. and Acero, A. 2003. Concept acquisition in example-based grammar authoring. In Proceedings of the International Conference on Acoustics, Speech, and Signal Processing.

Zinn, C., Moore, J. D., and Core, M. G. 2002. A 3-tier planning architecture for managing tutorial dialogue. In Proceedings of the Intelligent Tutoring Systems Conference, pages 574–584.