BEYOND STRUCTURED DIALOGUES: FACTORING OUT GROUNDING

Peter A. Heeman, Michael Johnston, Justin Denney and Edward Kaiser

Computer Science and Engineering
Oregon Graduate Institute
PO Box 91000 Portland OR 97291
{heeman,johnston,jdenney,kaiser}@cse.ogi.edu

ABSTRACT

Structured dialogue models are currently the only tools for easily building spoken dialogue systems. This approach, however, requires the dialogue designer to completely specify all dialogue behavior between the user and system, including how information is *grounded* between the user and the system. In this paper, we advocate factoring out the grounding behavior from structured dialogue models by using a general purpose dialogue manager that accounts for this behavior. This not only simplifies the specification of dialogue, but also allows more powerful mechanisms of grounding to be employed, which cannot be implemented within the framework of structured dialogues.

1. INTRODUCTION

As speech recognition and speech synthesis continue to improve, spoken dialogue systems have started to emerge. However, significant barriers remain in building effective spoken dialogue systems. There will always be errors in speech recognition, and unfortunately, natural language understanding and dialogue processing are still quite limited in dealing with such errors. Additionally, these technologies must deal with the large variability in the way users express themselves in spontaneous speech. Hence, it will be impossible to build a system that always understands everything that the user says. Instead, a usable system must deal with its limited abilities in understanding the user. It must collaborate with the user in order to reach mutual understanding of what the user said. This process is referred to as *grounding* [3, 2]. Conversants display their understanding (or lack of understanding) through such means as paraphrasing, making the next relevant contribution, and requesting clarifications or confirmations. We believe that endowing a system with proper grounding mechanisms will make it robust in dealing with limited domain and linguistic knowledge and hence able to converse more effectively and aid the user in accomplishing their goals.

2. STRUCTURED DIALOGUES

Structured dialogues, or finite state dialogue models, provide a formalism for building spoken dialogue systems. In this approach, initiative is kept solely with the system: the system directs the user through a predetermined network of states to accomplish some task. Each state specifies a system prompt and a set of allowed responses that the user can make. The determination of the next state is a function of the user's response and the current state. Figure 1 gives a simple structured subdialogue that allows the user to specify information for finding train information, namely the destination city, the initial city, and the time that the train leaves.

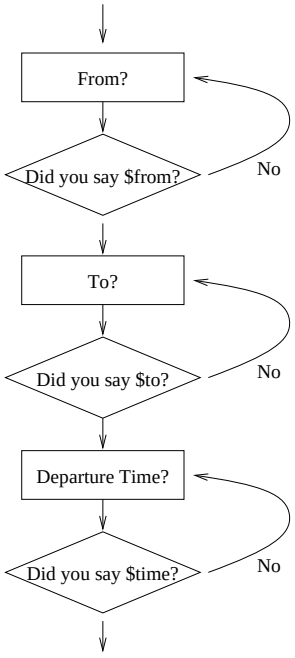

Figure 1: Structured dialogue model for train information

An advantage of structured dialogues is that the system prompts encourage the user to say something from a limited set of possible responses [5, 12]. This simplifies speech recognition and almost eliminates the need for natural language processing. Structured dialogues also simplify the dialogue management component, which is completely specified by the transitions. This technology has proved very popular for building working systems, and the Center for

Spoken Language Understanding at the Oregon Graduate Institute even provides a toolkit for building such systems with this approach [11, 17]. Although such systems seem overly restrictive for the user, Walker *et al.* [19] found that users preferred a system-initiative system over a mixed-initiative system, probably due to the former giving more reliable performance.

When designing a dialogue by means of a finite state dialogue model, the designer must account for all dialogue behavior between the system and the user. As such, they must hand-code the grounding behavior in the dialogue. As a result, grounding is often achieved by asking the user to explicitly verify the recognition and understanding results of the system after each piece of information that the user gives. This is shown in Figure 1 by the three verification questions. Control loops back to the original query if the user indicates that there was a misunderstanding. Another approach to dealing with misunderstandings is to add arbitrary commands, such as "scratch that", which users can utter if they think that a misrecognition or misunderstanding occurred, which will undo the last operation, or "clear history", which will completely reset the system [22].

A second aspect of dialogue that complicates the specification of a structured dialogue model is that there are many different ways that a speaker might want to break what they want to say into individual *contributions* for grounding. For the task of specifying criteria about a train, the user might want to specify the destination, origin and time in a single utterance (e.g. "I want to go to Portland from Chicago at 5 p.m."), or break it down into some combination of contributions, such as first presenting the destination and origin and then presenting the time (e.g. "u: I want to go to Portland from Chicago. s: At what time? u: At 5 p.m."). The way speakers break this down depends on a number of factors, such as "minimization of collaborative effort" [3]. Allowing for all possible contribution patterns for a contribution will add to the complexity of the dialogue structure, making it difficult and laborious to create, edit and maintain. Hence, this variability is typically not supported. Rather, the structured dialogue model prompts the user for a specific ordering of the information as shown in Figure 1.

3. RICHER DIALOGUE MODELS

There are other alternatives to the structured dialogue paradigm. Much work has been done on plan-based models (e.g. [9, 8]) and rational agency theories (e.g. [4, 13]). However, these approaches take the opposite approach from structured dialogues. Such systems aim to understand anything that the user might say about the domain. As such, they require a large amount of linguistic and domain information along with powerful reasoning capabilities in order to function. Moving these research theories into working application systems has been extremely difficult. In fact, of the work mentioned above, only the ARTIMIS system [13] is a implemented spoken dialogue system. However, its current domain is restricted to providing telephone numbers of service providers for employment and weather information. In addition, it has no capabilities to collaborate with the user in the event that misunderstandings arise.

Other working systems, such as those built by Smith *et al.* [16] and Allen *et al.* [1], do display advanced dialogue capabilities, such as employing user-modeling and domain reasoning. However, it is unclear how much of these capabilities have been built in a domain-specific fashion or rely heavily on domain-specific functions.

4. A COMPROMISE

As a compromise, we are pursuing an approach that goes beyond structured dialogues by factoring out the grounding and contribution behavior from the structured dialogue model. Rather than requiring the designer to break down the dialogue into speaker turns, the designer instead partitions the dialogue into *taskboxes*. Each taskbox has a goal along with associated domain knowledge. For example, a taskbox could be used to allow users to schedule an appointment, make a reservation, select a product, make a payment, etc. An example of a dialogue built with the taskbox approach is given in Figure 2. Currently, the goal

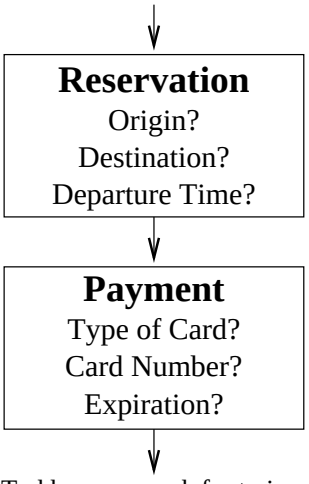

Figure 2: Taskbox approach for train reservation

has been limited to collaborating with the user to find values for slots in a frame. Individual taskboxes are linked together like the states in a structured dialogue. Thus, the main advantage of structured dialogues—providing context for speech recognition and natural language understanding—is preserved. Exactly how much can be included in a taskbox depends on the current state of natural language understanding and dialogue management and can grow accordingly. A further advantage of this approach is that it augments the structured dialogue paradigm. A dialogue can be created as a network with both prompt/ response states and taskboxes.

A key behavior of the dialogue manager is to engage in the grounding process with the user to ensure that everything said is mutually understood, and to collaborate with the user in breaking up what the user wants to say into individual contributions. Our approach draws on work done by Clark and his colleagues and others who have built on their work [3, 2, 18, 7]. In the model of Clark and Schaefer [2], contributions consist of presentation and acceptance processes, and presentations can consist of multiple contributions, which in turn each have a presentation and acceptance process. Their model suggests that after each user presentation, we need to give evidence of our understanding. The amount of evidence should depend on the speech recognition results. If the speech recognition score of the best recognition result is close to the score of the next highest competitor (or a garbage hypothesis), the system should give stronger evidence of its interpretation of what the user said, perhaps paraphrasing it or asking for explicit confirmation [15, 10]. In this case, the system should expect that the user might correct a misunderstanding or make a relevant next contribution. If misunderstandings happen often, the system can adjust its strategy by asking more specific questions that have fewer possible responses, such as yes/no questions. This will ease the speech recognition problem and improve understanding. Such a strategy should be dictated by reasoning about minimization of collaborative effort.

We also draw on the model of Clark and Wilkes-Gibbs [3]. They model how conversants can collaborate in making contributions. Consider the case in which the system misrecognizes "to Boston" as "to Austin", and then responds with "to Austin, from where?" Here, the user will want to correct the misunderstanding, and might respond back with "no, to Boston." Clark and Wilke-Gibbs give a collaborative model in which the conversants iteratively judge and refine the current contribution. After the initial presentation, the other participant would pass judgment on it, either *accepting* it, *rejecting* it, or *postponing* his decision. If it was rejected or the decision postponed, then one participant or the other would *refashion* the contribution. This would take the form of either *expanding* it by adding further qualifications, or *replacing* the original expression with a new expression. The contribution that results from this is then judged, and the process continues until the contribution is acceptable by both participants. For the example above, the utterance "no, to Boston" is a rejection of "to Austin" and a refashioning to "to Boston". Thus the resulting contribution is "to Boston".

We are building a dialogue manager based on the above two models. It takes a specification of the *taskbox*, and engages in the appropriate dialogue behavior to collaborate with the user in realizing the goal of the taskbox. This approach of using a general purpose dialogue manager to collect and ground the information needed for each taskbox will simplify the creation of a spoken dialogue system as well as make the resulting system more user-friendly. This user-friendliness is the result of giving the user more control of how they break their thoughts into individual contributions as well as allowing them to use more natural means to reach mutual understanding.

5. PROTOTYPE

To show the feasibility of this approach, we have implemented a prototype that works with the structured dialogue mechanism in the CSLU speech toolkit [14, 17]. A dialogue state in the toolkit has slots where TCL code can be invoked before and after the recognizer is called. Taskboxes are created by commandeering a dialogue state and using its slots for communicating via sockets with our taskbox dialogue manager (written in SWI-Prolog with a TCP/IP package). The dialogue manager is thus able to control the behavior of the system until it determines that the taskbox is completed.

The general-purpose dialogue manager takes a specification of the taskbox, which includes the goal of the taskbox, the necessary domain information and grammar fragments. The discourse manager tracks the state of the subdialogue, and uses this in formulating what it is going to say. To facilitate the grounding process, the system paraphrases its current understanding of the user's goal. For instance, for the train example, if it thinks the user wants to go to Chicago, it will include this as part of its response. The system will also prompt for missing information. Thus in the above example, it would respond with "To Chicago, from where?" Using its current discourse state and its response to the user, the dialogue manager has expectations of what the user will say next. The user might respond to the question by specifying the origin and optionally specify the departure time as well; or he might reject (or reject and replace) the destination. These expectations, along with the taskbox grammar fragments, are used to dynamically generate the relevant parsing grammar and semantic/discourse interpretation rules for the current discourse state. In the example above, the syntactic rule for the user specifying the origin city has an associated speech act that indicates that the user is adding this information to their goal specification. In addition to using this grammar for parsing and understanding the user's speech, the discourse manager sends a simplified version of this grammar (collapsed to a regular grammar) to the speech recognizer. Thus, the speech recognizer is tightly coupled with the dialogue manager's expectations of the user's upcoming turn (c.f. [21]).

6. ONGOING RESEARCH

There are a number of avenues that we are currently pursuing. First, we are exploring how to make the dialogue manager more sophisticated with respect to grounding. We plan on judging the certainty of the recognition results and

using this to influence the grounding strategy. For instance, if recognition problems occur and continue, we will switch into a more controlled interaction. We are also extending the taskbox specification to include hierarchical domain knowledge representations using 'has-a' and 'is-a' relations. This should allow more complex tasks to be handled inside a single taskbox. For instance, for ordering a pizza and a soft-drink, both have associated sizes with them. In order to allow the user to naturally talk about the sizes of both items, and to repair misunderstandings, the sizes must be associated with the proper item. Rather than force the pizza and soft-drink to be specified in separate taskboxes, we plan on having the dialogue manager understand the hierarchical domain knowledge and keep track of the focus (c.f. [6]). The need for hierarchical reasoning becomes even more apparent for taking multiple orders, where each pizza can have different ingredients. Second, we are planning on incorporating robust parsing techniques, such as used by Phoenix [20]. Rather than use a grammar recognizer for speech recognition, we plan on experimenting with using a statistical language model and feeding its output to a robust parsing with the dialogue expectations. Third, we are integrating the taskbox approach closer to the toolkit and plan on making authoring tools to simplify the specification of the taskbox, which currently must be done in Prolog.

7. CONCLUSION

In this paper, we proposed a new alternative for building spoken dialogue systems that strikes a middle ground between fully structured dialogue models and the richer models of dialogue that employ techniques such as plan-based reasoning and logical inference. Rather than specify the dialogue in terms of a network of prompt/response pairs, the designer specifies a network of taskboxes, which employ a generic dialogue manager that drives the dialogue behavior inside the box. This approach allows much richer dialogue systems to be built while maintaining domain independence and the ease of development that the structured dialogue approach offers.

8. ACKNOWLEDGMENTS

This work was partially funded by the Intel Research Council, ONR under grant N00014-94-1-1154, and DARPA under grant DABT63-95-C-007. The authors would also like to thank the CSLU and CHCC member consortia.

9. REFERENCES

[1] J. Allen, B. Miller, E. Ringger, and T. Sikorski. A robust system for natural spoken dialogue. In *Proceedings of the 34^{th} Annual Meeting of the Association for Computational Linguistics*, 1996.

[2] H. Clark and E. Schaefer. Contributing to discourse. *Cognitive Science*, 13:259–294, 1989.

[3] H. Clark and D. Wilkes-Gibbs. Referring as a collaborative process. *Cognition*, 22:1–39, 1986.

[4] P. Cohen and H. Levesque. Rational interaction as the basis for communication. In P. Cohen, J. Morgan, and M. Pollack, editors, *Intentions in Communication*, pages 221–255. MIT Press, 1990.

[5] R. Cole, D. Novick, P. Vermeulen, S. Sutton, M. Fanty, L. Wessels, J. de Villiers, J. Schalkwyk, B. Hansen, and D. Burnett. Experiments with a spoken dialogue system for taking the U.S. census. *Speech Communications*, 23, 1997.

[6] B. Grosz. The representation and use of focus in a system for understanding dialogs. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 67–76, 1977.

[7] P. Heeman and G. Hirst. Collaborating on referring expressions. *Computational Linguistics*, 21(3):351–382, 1995.

[8] L. Lambert and S. Carberry. A tripartite plan-based model for dialogue. In *Proceedings of the 29^{th} Annual Meeting of the Association for Computational Linguistics*, pages 47–54, 1991.

[9] D. Litman and J. Allan. Discourse processing and commonsense plans. In Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, SDF Benchmark Series, pages 365–388. MIT Press, 1990.

[10] Y. Niimi and Y. Kobayashi. A dialog control strategy based on the reliability of speech recognition. In *Proceedings of ICSLP*, pages 534–537, 1996.

[11] D. Novick and S. Sutton. Building on experience: Managing spoken interaction through library subdialogues. In *Proceedings of Twente Workshop on Language Technology 11*, 1996.

[12] S. Oviatt, P. Cohen, and M. Wang. Toward interface design for human language technology: Modality and structure as determinants of linguistic complexity. *Speech Communications*, 15:283–300, December 1994.

[13] M. Sadek, P. Bretier, and F. Panaget. ARTIMIS: Natural dialogue meets rational agency. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1997.

[14] J. Schalkwyk, J. de Villiers, S. van Vuuren, and P. Vermeulen. CSLUsh: an extendible research environment. In *Proceedings of EUROSPEECH*, 1997.

[15] R. Smith. An evaluation of stratagies for selective utterance verification for spoken natural language dialog. In *Proceedings of the 5^{th} Conference on Applied Natural Language Processing*, pages 41–48, 1996.

[16] R. Smith, D. Hipp, and A. Biermann. An architechure for voice dialog systems based on prolog-style theorem-proving. *Computational Linguistics*, 21(3):281–320, 1995.

[17] S. Sutton, D. Novick, R. Cole, P. Vermeulen, J. de Villiers, J. Schalkwyk, and M. Fanty. Building 10,000 spoken-dialogue systems. In *Proceedings of ICSLP*, 1996.

[18] D. Traum and E. Hinkelman. Conversation acts in task-oriented spoken dialogue. *Computational Intelligence*, 8(3):575–599, 1992.

[19] M. Walker, D. Hindle, J. Frome, G. Di Fabbrizio , and C. Mestel. Evaluating competing agent strategies for a voice email agent. In *Proceedings of EUROSPEECH*, 1997.

[20] W. Ward. Understanding spontaneous speech: The Phoenix system. In *Proceedings of ICASSP*, pages 365–367, 1991.

[21] S. Young, A. Hauptmann, W. Ward, E. Smith, and P. Werner. High level knowledge sources in usable speech recognition systems. *Communications of the ACM*, 32(2), 1989.

[22] V. Zue, S. Seneff, J. Polifroni, M. Phillips, C. Pao, D. Goodine, D. Goddeau, and J. Glass. PEGASUS: A spoken dialogue interface for on-line air travel planning. *Speech Communications*, 15:331–340, 1994.