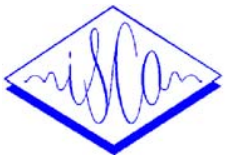

6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

FORM-BASED REASONING FOR MIXED-INITIATIVE DIALOGUE MANAGEMENT IN INFORMATION-QUERY SYSTEMS

Jennifer Chu-Carroll
Lucent Technologies Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974, U.S.A.
jencc@bell-labs.com

ABSTRACT

Previous work on practical dialogue systems focused mainly on developing robust systems that operate in specific domains. However, these systems are often heavily handcrafted and are thus difficult to port to new domains. This paper describes general mixed-initiative dialogue management strategies that are decoupled from domain-dependent task specifications. These strategies are used to select appropriate system actions in response to user utterances, focusing on situations in which user queries are inconsistent, ambiguous, or underspecified. If the dialogue manager fails to resolve the problem using its own knowledge or information from the dialogue history, it adopts one of three query refinement strategies to engage the user in subdialogues to solicit the required information. An evaluation of the dialogue management strategies on a corpus of 263 unseen queries shows that the system is able to correctly answer 90.2% of the queries after redirecting 3% of the calls to a human operator.

1. INTRODUCTION

Most previous work on dialogue systems focused either on developing robust spoken language systems that perform simple tasks in specific domains (e.g., [7, 5, 1]), or on developing algorithms that model specific dialogue phenomena that often take place in naturally-occurring human-human dialogues, such as recognizing and initiating negotiation and clarification subdialogues (e.g.,[6, 9, 4, 2]). Systems in the former category are often heavily handcrafted, and therefore require substantial effort to be ported to perform similar functions in new domains. While systems in the latter category utilize general algorithms that apply across domains, they often require sophisticated modeling of domain knowledge and user beliefs that makes them difficult for actual deployment at the current time.

Our goal is to develop general algorithms for practical dialogue systems that operate across similar applications in different domains to increase portability of such systems. We focus on developing mixed-initiative dialogue management strategies that allow the system to initiate subdialogues for query refinement purposes for the class of form-filling information query applications. We emphasize the generality of our approach by decoupling the domain-independent dialogue management strategies from the domain-dependent specifications with which the dialogue manager reasons. Based on the user queries and task specifications, the dialogue manager may adopt one of three query refinement strategies to resolve inconsistency, underspecification, or ambiguity in the user's queries, resulting either in refined queries that can be automatically answered or in queries redirected to human operators. Evaluation on a corpus on 263 unseen queries shows that, after redirecting 3% of all queries to the operator, the system correctly answers 90.2% of the remaining calls.

Q-type: What
Movie: null
Theater: null
Town: Montclair

Figure 1: Sample Attribute-Value Representation

2. TASK DESCRIPTION

The dialogue management strategies described in this paper will be illustrated by examples taken from the movie information domain, in which a prototype system has been implemented. The system answers user questions about movie showtime and theater playlist information, such as *"What's playing at the Clearviews theater in Montclair?"* and *"When is Antz playing at the Madison theater?"* The characteristic of this application central to the dialogue management strategies is that each user query can be represented as non-recursive attribute-value pairs. For example, the query *"What's playing in Montclair?"* can be represented as attribute-value pairs as shown in Figure 1.

This attribute-value representation allows each question type to be specified by a template indicating, for each attribute, whether a value must, must not, or can optionally be supplied in order for a user query of that type to be considered fully-specified and consistent. For instance, Figure 2 shows that for *when* questions, used to solicit showtime information, the user must provide a movie name and a theater name (e.g.,*"When is <movie> playing at <theater>?"*), whereas for *what* questions, used to obtain theater playlist information, the user must provide a

Q-type:	When	**Q-type:**	What
Movie:	mandatory	**Movie:**	forbidden
Theater:	mandatory	**Theater:**	mandatory
Town:	optional	**Town:**	optional

Figure 2: Sample Specifications for Question Types

theater name (e.g.,"*What is playing at <theater>?*"), but not a movie name. These specifications are determined based on domain semantics, and have to be re-constructed when porting the system to a new domain. However, decoupling these specifications from the rest of the dialogue system allows the dialogue management strategies that make use of these specifications to remain domain-independent.

3. MIXED-INITIATIVE DIALOGUE MANAGEMENT

3.1. Motivation

The purpose of a dialogue manager is to select an appropriate system action based on the system's understanding of the user utterance(s). In our initial prototype system, our main focus is to develop domain-independent dialogue management strategies that 1) allow the user the freedom to provide unrestricted information in query formulation, 2) allow the system to initiate subdialogues to solicit relevant information toward formulating a fully-specified and consistent query, and 3) focus on modeling dialogue phenomena commonly found in simple human-human information query and/or transaction dialogues.

Consider the following dialogue segment based on a naturally-occurring dialogue between a customer (C) and a travel agent (T), taken from [8]:

(1) T: This is a new reservation?
(2) C: New reservation on June twenty first.
(3) T: Okay and he is umm
(4) C: He wants to fly from Boston to BWI on flight, I'm sure it's Piedmont P L P I
(5) T: P I at what time?
(6) C: Seven forty p.m.
(7) T: And this is on the twenty first of June?
(8) C: Yeah, arriving nine oh one p.m.
(9) T: Okay, that's P I flight ten oh five, leaving Boston at seven forty p.m.,....

This dialogue segment illustrates several phenomena that commonly occur in simple information query or transaction dialogues. First, dialogue participants oftentimes specify at the outset the overall goal of the dialogue interaction. This is illustrated by utterance (2) in which the customer specifies the goal of making a new reservation and provides one of the necessary parameters (date of departure). Second, dialogue participants often initiate subdialogues of specific purposes, such as to seek information (utterances (5) and (6)), or to confirm information previously mentioned in the dialogue (utterances (7) and (8)). Third, once all necessary information is obtained, one agent will either provide information (explicitly or implicitly) solicited by the other agent (utterance (9)) or perform a requested transaction. Finally, dialogue participants often make use of their own domain and world knowledge, as well as information in the dialogue history to understand and generate elliptical sentences (e.g. utterance (6)) and anaphoric references (e.g., *this* in (7)).

Based on the above analysis, we adopt an incremental query refinement strategy for processing user utterances. For each user utterance, the dialogue manager compares its attribute-value representation with the specification for the appropriate question type, and responds by 1) providing an answer to the user's query if it is fully-specified and consistent or can become so based on inference, or 2) selecting an appropriate query refinement strategy to initiate a subdialogue to solicit further information from the user.

The rest of this section first briefly outlines the process for interpreting user utterances to obtain their attribute-value representations. We will then focus on the dialogue management strategies for selecting appropriate system actions based on the inferred attribute-value representations for the user utterances.

3.2. Dialogue Management

The dialogue manager takes as input an attribute-value representation of the user's utterance. In the current implementation of the system, this interpretation process is carried out by a vector-based topic identification process, similar to that used in [3]. Briefly, the topic identification process is performed for each attribute in order to determine whether or not, for that attribute, a value is given in the utterance, and if so, the value and the system's confidence in correctly recognizing this value. The outcome of this process is an attribute-value representation for the user utterance similar to that shown in Figure 1.

The dialogue manager reasons with the attribute-value representation and determines how it should respond to the user's utterance. Figure 3 presents our algorithm for response strategy selection. The algorithm, **Select-Strategy**, assumes a specification for each possible question type as discussed in Section 2, and is designed based on the analysis presented in Section 3.1. The algorithm either directly selects one of five response generation strategies, **Provide-Answer**, **Notify-Failure**, **Clarify**, **Confirm**, and **Info-Seek**, or adopts an internal action to modify the utterance representation based on the dialogue history and/or the system's domain knowledge, and recursively applies the algorithm to the modified representation until a response strategy is chosen. This algorithm allow the system to engage the user in dialogue to iteratively refine a query until either the query can be answered or until the system gives up and redirects the call to an operator.

In step 1 of the algorithm, the given utterance representation, *avrep*, is compared with the utterance representation of the previous dialogue turn (which when the algorithm is first invoked is an empty representation). If the two representations are identical, indicating that no progress has been made in query refinement, then the sys-

Select-Strategy(avrep,previous-avrep):
(1) If (avrep == previous-avrep) */* *no progress made* */*
(2) **Redirect** to human operator
(3) Else if fully-specified(avrep) ∧ consistent(avrep)
(4) answer ← database-query(avrep)
(5) If (answer != null) */* *successful query* */*
(6) **Provide-Answer**(answer)
(7) Else
(8) **Notify-Failure**(avrep)
(9) Else if (!consistent(avrep))
(10) relaxed-avrep ← relax(avrep)
(11) Select-Strategy(relaxed-avrep,avrep)
(12) Else if ambiguous(avrep)
(13) ambiguous-attr ← get-ambiguous(avrep)
(14) **Clarify**(ambiguous-attr)
(15) Else if !confident(avrep)
(16) uncertain-attr ← get-uncertain(avrep)
(17) **Confirm**(uncertain-attr)
(18) Else */* *not fully specified* */*
(19) augmented-avrep ← infer-missing(avrep,foci)
(20) If fully-specified(augmented-avrep)
(21) Select-Strategy(augmented-avrep,avrep)
(22) Else
(23) missing-attr ← get-missing(augmented-avrep)
(24) **Info-Seek**(missing-attr)

Figure 3: Strategy Selection Algorithm

tem redirects the call to a human operator. Otherwise, the system attempts to automatically handle the user query by first comparing *avrep* with the specification of the question type in *avrep* to determine if the user's query is fully-specified (i.e., all mandatory attributes are given unique values) and consistent (i.e., no forbidden attribute is given a value and the values given to the attributes do not contradict one another).[1] If so, the system queries its domain knowledge base to find a set of answers that satisfies the user's query (step 4 in algorithm). If an answer is found, then the **Provide-Answer** strategy is chosen to deliver the answer; otherwise, the **Notify-Failure** strategy is selected to notify the user of the invalidity of the query (steps 5-8). The latter case applies, for example, when the user asks *"When is Matrix playing at the Chatham cinema?"* when the movie is *not* playing at that theater.

If the query failed either test in step 3, then the system will modify the original query toward being fully-specified and consistent (steps 9-24 in Figure 3). This modification process may be carried out by one of two types of actions, one that involves internal system actions to modify the attribute-value representation based on information sources such as the dialogue history, and one that involves initiating subdialogues to solicit information from the user. If the user query is inconsistent, i.e., when a forbidden attribute is given a value or when the values given to attributes contradict one another, the system invokes an internal action to relax the utterance representation. The *relax* function removes, of the set of attribute-value pairs that result in the inconsistency, one by one the attribute-value pair with the lowest confidence value from the utterance interpretation process until the utterance representation becomes consistent. The algorithm is then recursively applied to the relaxed representation (steps 9-11). If the query is ambiguous, i.e., when two or more values are recognized for the same attribute, the system adopts the **Clarify** strategy to initiate a subdialogue to ask the user to choose from among the recognized values (steps 12-14). If the confidence value for an attribute-value pair is lower than a given threshold, the **Confirm** strategy is selected to initiate a subdialogue to confirm the recognized value (steps 15-17).

The final case for query refinement occurs when the query is underspecified, i.e., when values are missing for mandatory attributes. In this case, the system either infers the missing values based on its own knowledge or asks the user for them. The system first attempts to infer the missing values from *foci*, a representation of the dialogue history that records the last instantiated value for each attribute (step 19).[2] If the resulting augmented representation is fully-specified, then the algorithm is recursively applied to the augmented representation (steps 20-21); otherwise, the **Info-Seek** strategy is invoked to initiate a subdialogue to solicit the missing information directly from the user (steps 22-24).

The output of the **Select-Strategy** algorithm is a response strategy which the system will adopt to generate its response to the user's utterance. In the current system, each response strategy is mapped to a unique dialogue action, which has a corresponding template to generate natural language utterances that perform the particular dialogue action. For instance, the **Confirm** strategy is carried out by the **Request-If** dialogue action, which is realized by the template *"Did you say X?"*

4. EXAMPLE

In this section, we step through an actual example to illustrate several aspects of our dialogue management strategies. Consider the following dialogue generated by our implemented system, annotated by the sequence of system actions taken for each generated response:

(10) U: What's playing in Montclair?
(11) S: For what theater? **[Info-Seek]**
(12) U: The Wellmont Theater.
(13) S: Affliction, Elizabeth, Saving Private Ryan, and A Simple Plan are playing at the Wellmont theater. **[Answer]**
(14) U: When is a Simple Plan playing?
(15) S: A Simple Plan is playing at the Wellmont theater at 5:15PM. **[Infer, Answer]**
(16) U: How about Saving Private Ryan?

[1] If the system was not able to recognize the question type from the user utterance, then the utterance is considered underspecified, and step 18 of the algorithm is executed.

[2] When a value is inferred from the dialogue history, its confidence value is 0. Thus, if this inferred value results in an inconsistent representation, the inferred value will be the first candidate for relaxation.

(17) S: Saving Private Ryan is playing at the Wellmont theater at 7:00PM. **[Infer, Infer, Answer]**
(18) U: When is Shakespeare in Love playing?
(19) S: Shakespeare in Love is not playing at the Wellmont theater. **[Infer, Notify-Failure]**

The attribute-value representation for utterance (10) is as shown in Figure 1. **Select-Strategy** first compares this representation with the specification for question-type *what* in Figure 2, and finds the representation to be underspecified, since the mandatory attribute *theater* is not given a value. Having no dialogue history from which to infer unspecified attributes, the system adopts the **Info-Seek** strategy to solicit the information from the user (step 24 in Figure 3), resulting in utterance (11). The user's response in (12), when added to the original attribute-value representation, results in a full-specified and consistent representation; thus, the system queries its database and invokes the **Provide-Answer** strategy to inform the user of the result (utterance (13)).

The user's next query in utterance (14) is again underspecified when interpreted in isolation. However, the missing attribute, *theater*, can be inferred from the dialogue history as *the Wellmont theater*, resulting in a fully-specified query. Similarly, when processing the query in utterance (16), the question-type is first inferred to be *when* from the dialogue context, and in the next iteration, the missing theater is inferred as in (14).

5. PERFORMANCE EVALUATION

Our prototype system was evaluated on the transcription of a corpus of 263 unseen queries each of which had been annotated with its attribute-value representation, and may or may not be fully-specified. For each query, the system first generated an attribute-value representation, then applied the **Select-Strategy** algorithm to the utterance representation. The **Select-Strategy** algorithm was applied with one slight modification to simplify the evaluation process, namely, when a query was fully-specified and consistent, instead of selecting between the **Provide-Answer** and **Notify-Failure** strategies (steps 4-8), the utterance representation was returned. This representation was then compared with the annotated representation for that query to obtain a correct/incorrect judgment on the system generated representation.

If a refinement strategy was selected by the algorithm, the system generated an information-seeking, confirmation, or clarification question. If the question solicited information available in the original user query, the answer was provided according to the original query; otherwise, a random answer consistent with the query was provided (for instance, if the system question was *"For what theater?"*, a random theater name was given as an answer). A query was redirected to a human operator if no progress was made in query refinement during two successive system-user exchanges; otherwise, a fully-specified and consistent representation was eventually obtained, which then received a correct/incorrect judgment.

In this evaluation, the system redirected 3% of all calls to the human operator, and correctly obtained the attribute-value representation of 90.2% of the remaining queries. Our baseline system was one without dialogue query refinement capabilities, which correctly recognized 79.8% of all user queries. Compared against the baseline system, our dialogue management strategies for query refinement yielded a 51.5% reduction in relative error rate.

6. CONCLUSIONS

In this paper, we presented general mixed-initiative dialogue management strategies that are applicable to the class of form-filling information query tasks. Our system decouples the task-dependent specifications from the task-independent dialogue management reasoning strategies in order to achieve high portability. The dialogue management strategies allow the system, in addition to providing direct answers to user queries, to engage in dialogue with the user for query refinement purposes by initiating information-seeking, clarification, or confirmation subdialogues. An evaluation of the dialogue management strategies on a corpus of unseen user queries yielded a 51.5% reduction in relative error rate when compared with a baseline system without query refinement capabilities.

7. ACKNOWLEDGMENT

I would like to thank Bob Carpenter and Christine Nakatani for helpful discussions.

8. REFERENCES

[1] H. Aust and O. Schröer. Application development with the Philips dialog system. In *Proc. ISSD*, pages 27–34, 1998.

[2] J. Chu-Carroll and S. Carberry. Collaborative response generation in planning dialogues. *Computational Linguistics*, 24(3):355–400, 1998.

[3] J. Chu-Carroll and B. Carpenter. Dialogue management in vector-based call routing. In *Proc. COLING-ACL*, pages 256–262, 1998.

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

[5] L. Lamel. Spoken language dialogue system development and evaluation at LIMSI. In *Proc. ISSD*, pages 9–17, 1998.

[6] D. Litman and J. Allen. A plan recognition model for subdialogues in conversation. *Cognitive Science*, 11:163–200, 1987.

[7] S. Seneff and J. Polifroni. A new restaurant guide conversational system: Issues in rapid prototyping for specialized domains. In *Proc. ICSLP*, pages 665–668, 1996.

[8] SRI Transcripts. Transcripts derived from audiotape conversations made at SRI International, Menlo Park, CA, 1992.

[9] M. Walker. The effect of resource limits and task complexity on collaborative planning in dialogue. *Artificial Intelligence*, 85:181–243, 1996.