

A FORM-BASED DIALOGUE MANAGER FOR SPOKEN LANGUAGE APPLICATIONS¹

David Goddeau, Helen Meng, Joe Polifroni, Stephanie Seneff, and Senis Busayapongchai[†]

Spoken Language Systems Group
MIT Laboratory for Computer Science
Cambridge, MA 02139 USA
BellSouth Information Systems [†]
Tucker, GA 30084 USA
<http://www.sls.lcs.mit.edu>

ABSTRACT

A popular approach to dialogue management is based on a finite-state model, where user utterances trigger transitions between the dialogue states, and these states, in turn, determine the system's response. This paper describes an alternative dialogue planning algorithm based on the notion of filling in an electronic form, or "E-form." Each slot has associated prompts that guide the user through the dialogue, and a priority that determines the order in which the system tries to acquire information. These slots can be optional or mandatory. However, the user is not restricted to follow the system's lead, and is free to ignore the prompts and take the initiative in the dialogue. The E-form-based dialogue planner has been used in an application to search a database of used car advertisements. The goal is to assist the user in selecting, from this database, a small list of cars which match their constraints. For a large number of dialogues collected from over 600 naive users, we found over 70% compliance in answering specific system prompts.

1. INTRODUCTION

A new generation of spoken-language systems is emerging which combines speech recognition with natural language understanding, discourse resolution, language generation, and dialogue management. These systems engage the user in a multi-utterance conversation in order to complete a goal-directed task. Dialogue management is an important component of such a system. Its role is to direct the conversation along a productive path, making the interaction with the user as natural and efficient as possible while keeping the user within the boundaries and capabilities of the domain. The system needs to respond to user initiative, keeping track of information and goals provided by the user. It can also take the initiative when necessary, prompting for needed information or asking for clarification or corrections when the user's input seems anomalous (perhaps due to a recognition error).

There have been several approaches to dialogue management embodied in recent systems. Early approaches were collections of routines constructed for specific domains [3, 7]. These would often work quite well, but could not easily be ported to a different domain, and their complexity made it difficult to predict system behavior or

modify the algorithm.

An alternative approach is to develop a dialogue manager based on a formal model, such as finite state automata. In this paradigm, the dialogue space is represented as a set of dialogue states. Each state is usually associated with a prompt or question. The user's responses to the questions are categorized and used to determine which state the system should enter next. A number of systems have been developed based on this approach [8] and a number of tools for constructing such dialogue managers have been offered [6, 1].

The advantage of these finite-state scripted systems is that they are relatively straightforward to design and their behavior is predictable. The disadvantages are that the dialogue tends to be inflexible and machine-initiated. They are therefore most suited to applications in which the interaction is well-defined and can be structured as a sequential form-filling task or a tree, preferably of yes/no or short answer questions. The system maintains complete control over the dialogue, requiring the user to follow a predefined scripted path, with limited alternatives at branch points. If the user tries to deviate from the system's plan, the system is unable to understand their queries. Once a tree-based dialogue has proceeded through several levels, it is usually very difficult for a user to back up and change their response to one of the previous questions, thereby sending the dialogue down a different branch of the tree.

This paper presents a dialogue management scheme intended to overcome some of the limitations of these approaches. In particular, it is intended to enable both user-initiated and system-initiated interactions, to improve robustness both to system errors and changes in user goals, and to allow the construction of systems whose behavior is predictable and verifiable. While the system can ask directed questions, it does not insist that the user comply. This gives the user a great deal of flexibility, as constraints or requests can be specified in any order the user chooses based on their requirements.

2. THE WHEELS DOMAIN

We have applied this new dialogue management scheme within the WHEELS [5] domain, which we have been developing over the past year, under the sponsorship of BellSouth Intelliventures. This system provides a spoken language interface to a database of approximately 5,000 classified advertisements for used automobiles from

¹ This research was supported by a research contract from BellSouth Intelliventures.

| | |
|--------|----------------------------|
| E-form | |
| make: | <i>Volkswagen</i> |
| model: | <i>Jetta</i> |
| price: | <i>under 6,000 dollars</i> |
| color: | <i>red</i> |

Figure 1: A graphical representation of an E-form for the WHEELS domain.

the *Atlanta Journal and Constitution*. The task is to assist the user in narrowing the list of ads to a small number (currently, less than five), which can then be read or faxed to the user.

WHEELS is implemented using the infrastructure of the GALAXY [4] system. This is a distributed framework for organizing conversational systems to optimize resource-sharing and extensibility. Using this framework, the WHEELS domain is organized as a server process. The input to the server is a semantic frame representation of the input utterance produced by the speech recognizer and language analyzer which run in separate processes. The output of the server includes a spoken response and an optional tabular representation.

3. DIALOGUE MANAGEMENT IN WHEELS

The basic data structure of this approach can be thought of as an electronic form (or E-form) [2]. The E-form is merely a virtual form which is transparent to the user. It is represented in the same semantic frame language as the input utterances, and contains slots for the relevant constraints of the domain. For example, in the WHEELS domain, there are slots for make, model, price, mileage, year and other features such as color, convertible, etc., as suggested in Figure 1. These constraints have different priorities to different users; for any given user some may be critical while others irrelevant. Furthermore, the constraints are correlated – e.g., a more recent model probably costs more. The user is essentially performing an optimization task with their own preferences as the utility function. Simply requiring the user to fill out each slot in the E-form in turn is clearly unsatisfactory. The user must be allowed to explore multiple combinations to find the result which best matches their desires.

We have developed an E-form based algorithm for processing an input semantic frame derived from a user query, which involves the following five steps:

1. Extracting information from the current semantic frame,
2. Combining this information with the context E-form,
3. Querying the database based on the combined E-form,
4. Updating the context E-form, and
5. Producing a response summarizing the current status and prompting for more information (if needed to achieve the goal).

The first step involves mapping the contents of the input semantic frame to the canonical format of the E-form. A given constraint

can be articulated in a variety of ways, and by a variety of semantic expressions. For example, a constraint on the age of a car can be expressed as a model year “1990 Hondas,” a range of years, “after 1990,” “between 1990 and 1993,” or an age specification, “less than three years old.” Each of these will be represented differently in the semantic frame. The E-form, in contrast, has a canonical representation for each field. Much of this translation can be described by a table mapping semantic predicates to corresponding E-form entries. However, there are often cases which require special handling. For example, number expressions are very prevalent in the WHEELS domain and can be interpreted as prices, years, mileages, etc. In general, we classify inputs of the format “ninteen <decade> <digit>” as year specifications, but other numeric inputs mandate disambiguation. Under such circumstances, the system consults the prompt state, assuming user compliance with the generated prompts. However, if the previous prompt is neither for price nor mileage, then the numeric value is inserted into any existing context constraints, following a prioritized order of price over mileage.

The second step combines the information in the new E-form with the context E-form. At first glance this seems straightforward – add the new information to the context E-form and use the fields from the new form when there is a conflict. In practice, the situation is more complex. One problem is the interaction of multiple constraints on the same slot. Successive constraints on price, for example, could represent refinements of an initial constraint or an override beginning a new avenue of exploration. This ambiguity can be partially resolved with heuristics on the constraints. If the intersection of the previous (context) constraint and the new constraint is a subset of the context constraint, and the new constraint is not a superset of the context constraint, use the intersection as the combined constraint. For example, if a price constraint of “under ten thousand dollars” is followed by “over five thousand,” then the two should be combined to form a new constraint of “between five and ten thousand dollars.” These heuristics can often be generalized across fields (i.e., most range constraints follow similar pragmatic rules).

Another set of problems arise from the interaction between the various slots. For example, if a user has been exploring one model of car, say Honda Accords, and switches to asking about Escorts, clearly the make field must be updated along with the model field (since Escorts are made by Ford and not Honda). Less obvious is which of the other fields should be carried along into the new context, and which should be cleared. For example, perhaps the color and mileage constraints should be relaxed while the price range constraint should be preserved. The E-form framework provides a convenient format for specifying these sorts of policy decisions.

Context update depends on the result of the database query. In particular, if the query returns an empty response, that is, no objects in the database match the current E-form specifications, the context E-form should not incorporate the latest set of constraints.

Response generation is determined based on the current state of the E-form, the most recent prompt, and the number of items retrieved from the database. If no entries matched the constraints, the system asks the user to be more general. If fewer than five entries have been found, the system considers the search to be complete and generates

a concluding prompt. Otherwise, the system cycles through an ordered list of prompts, choosing the first one whose corresponding field in the E-form remains empty, as long as it is not the same as the most recent prompt. If all prompt fields are filled, but there are still too many items to display, the system asks the user to be more specific.

Mechanisms are also in place which allow the user top-level control of the dialogue. For example, the command “scratch that” restores the dialogue context (i.e., the context E-form and prompt) to its state before the preceding utterance. This mechanism is crucial for recovering from some recognition errors.

U: CAN YOU GIVE ME A LIST OF CORVETTES
 S: *There are 25 Chevrolet Corvettes. What price range are you interested in?*
 U: I'D LIKE A CONVERTIBLE
 S: *There are 8 Chevrolet Corvette convertibles. What year or range of years are you interested in?*
 U: HOW ABOUT A BLACK LEXUS
 S: *There are 14 black Lexuses. Which particular model are you interested in?*
 U: I'D LIKE A CONVERTIBLE
 S: *There are 208 convertible cars. Can you provide a make or a make and model?*
 U: SCRATCH THAT
 S: *There are 14 black Lexuses. Which particular model are you interested in?*
 U: ANY COLOR
 S: *There are 39 Lexuses. What price range are you interested in?*
 U: HOW ABOUT ONE WITH AUTOMATIC TRANSMISSION
 S: *There are 3 Lexuses with automatic transmission. Can I help you with something else?*
 U: NO THANK YOU
 S: *Have a nice day!*

Figure 2: Example of a dialogue between a user and WHEELS. ‘S’ stands for system, ‘U’ for user.

4. EXAMPLE DIALOGUES

Figure 2 is an example of a dialogue between a user and the WHEELS system, illustrating the overriding of slot values by new information. The dialogue starts with a query about Chevrolet Corvettes, followed by one about convertibles. In this case, we see that the make and model are inherited in the discourse. When the user switches to asking about a Lexus, the system automatically clears the previous make *and* model slots, as well as the preference for convertibles. When the user asks about convertibles again, the system infers that the previous make and model are to be erased, using the pragmatic constraint that there are no convertible Lexuses. The system also supports removal of existing context constraints upon *explicit* request from the user, such as a request for “any color.” Whenever a request produces no search output, it will not be inherited in the discourse.

S: *Welcome to the Electronic Automobile Classifieds. What car would you like?*
 U: DO YOU HAVE ANY MAXIMAS
 S: *There are 46 Maximas. What price range are you interested in?*
 U: NO MORE THAN TEN THOUSAND
 S: *There are 29 Maximas costing less than 10,000 dollars. What year or range of years are you interested in?*
 U: IN THE NINETIES
 S: *There are 3 Maximas costing less than 10,000 dollars in the 90s. Can I help you with something else?*
 U: WHAT ABOUT EIGHT THOUSAND
 S: *There is 1 Maxima costing less than 8000 dollars in the 90s. Can I help you with something else?*

Figure 3: Example of a dialogue between a user and WHEELS illustrating numeric disambiguation. Again, ‘S’ stands for system, ‘U’ for user.

The second dialogue, as shown in Figure 3 illustrates the disambiguation of number expressions, based on the previous prompt if it is relevant. The final exchange illustrates a complex situation. Here neither the prompt nor pragmatics (eight thousand could represent a price constraint, mileage constraint or model number) is sufficient for disambiguation. Instead, the system searches the context E-form for previously specified constraints, and attempts to override the slots in some prioritized order. Note that this insertion preserves any inequalities in the context.

| Category | Number | Percent |
|----------------|--------|---------|
| No Parse: | 1691 | 12.0 |
| Compliance: | 4645 | 32.9 |
| Noncompliance: | 1824 | 12.9 |
| Undirected: | 5939 | 42.1 |
| Total Utts: | 14099 | 100.0 |

Table 1: Breakdown of distribution of user queries in WHEELS dialogues collected in wizard mode.

| Prompt Type | Number Given | Co-operative Response (%) |
|-------------|--------------|---------------------------|
| Make/Model | 2253 | 76.2 |
| Price | 2804 | 73.9 |
| Year | 890 | 78.4 |
| Mileage | 522 | 30.4 |
| Total | 6469 | 71.8 |

Table 2: Statistics on degree of compliance for naive users responding to specific WHEELS requests.

5. DATA ANALYSIS

We have been developing the WHEELS domain jointly with BellSouth Intelliventures, who have undertaken the task of collecting a large number of dialogues between the WHEELS system and naive users, using a wizard-mode data collection procedure. Dialogues have been collected from over 600 speakers at shopping malls in the Jacksonville and Melbourne areas of Florida.

The data collection initiative yielded over 14,000 user queries. We have processed the resulting log files automatically to determine the percentage of time the system asks a specific question and the relative degree of compliance among users. Results are summarized in Tables 1 and 2. Overall, the system was unable to process about 12% of the utterances, due to parse failures. Over half of the remaining queries were preceded by a specific system request, such as “What price or range of prices are you interested in?” For this subset, we determined in each instance whether the user had actually responded directly to the request. We found that users complied over 70% of the time.² A breakdown by individual system requests is given in Figure 2. Users were far less willing to answer questions about mileage than questions about make, model, year, or price. They were most cooperative with requests for make or model. It is interesting to observe that make/model was always the first question asked, and mileage was always the last. On the other hand, mileage may not be as important to the users as other constraints.

6. SUMMARY AND FUTURE PLANS

The dialogue management paradigm described here was designed with two goals in mind: first, to support mixed-initiated interactions, and second, to promote predictable and verifiable dialogue managers. The E-form structure seems to be an effective mechanism for accumulating and canonicalizing the constraints articulated by the user. It is also a convenient framework for specifying the combination of new information with dialogue context. Furthermore, it is straightforward to specify within this framework an appropriate sequencing for system prompts that helps guide the user along a productive path. By suggesting specific alternatives, the system encourages the user to stay within the range of the system’s capabilities, and subtly directs them towards an efficient execution of their goals.

Future work on this dialogue model is envisioned in two areas – extending the basic model to deal with more complex goals and tasks, and abstracting the domain-independent from the domain-dependent aspects of the dialogue manager. The current system as implemented in the WHEELS domain pursues a single goal. Many interesting applications, however, have a complex structure with multiple subgoals. A good example is the air travel component of our GALAXY system. We are currently exploring the use of a more complex E-form in this domain. A dialogue component in this domain would typically have as its first goal extracting the user’s desired itinerary, where each leg of the journey must be instantiated with a particular flight and fare. The system might later inquire after the user’s need for hotel or rental car reservations. One challenge of such a task is that the various subgoals can interact. For example, the opportunity

² Whenever there were intervening unparseable queries, we assumed the system’s request was still active.

to take advantage of a reduced airfare or hotel rate, which may not become apparent until late in the dialogue, may motivate the user to change the overall structure of their itinerary.

A second area of work in dialogue management is to separate the domain-dependent and domain-independent portions of the system. Ideally, the domain-dependent portion could be expressed in a high-level dialogue scripting language. However, given the amount of domain-dependent code needed in the relatively simple WHEELS domain, much more research needs to be done to reach this point. An intermediate solution for the near term might be to provide a domain-independent shell.

Yet another direction for future work is to explore how dialogue managers can be abstracted and shared among similar domains. Given that it takes considerable work and craftsmanship to construct a good dialogue manager for a given domain, it make sense to try to reuse as much of that work as possible for similar domains. Perhaps dialogue manager shells can be produced for a generic domain class (for example, on-line shopping) which can be easily customized to a particular instance.

7. ACKNOWLEDGEMENT

We would like to acknowledge the assistance and expertise of our collaborators at BellSouth Intelliventures, especially Dewey Anderson and Dave Anderson, for their help in coordinating the project and collecting the large body of field data in Florida.

8. REFERENCES

1. Lindberg, B., B. Andersen, A. Baekgaard, T. Broendsted, P. Dalsgaard, J. Kristiansen, *An Integrated Dialogue Design and Continuous Speech Recognition System Environment*, Proc. ICSLP, pp. 16530–1656, Banff, Alberta, Canada, 1992.
2. Dertouzos, M., “Communications, Computers and Networks,” *Scientific American*, pp. 62-69, Sept. 1991.
3. Glass, J., G. Flammia, D. Goodine, M. Phillips, J. Polifroni, S. Sakai, S. Seneff, and V. Zue, “Multilingual Spoken-language Understanding in the MIT VOYAGER System,” *Speech Communication*, Vol. 17, No.1-2, pp. 1-18, 1995.
4. Goddeau, D., E. Brill, J. Glass, C. Pao, M. Phillips, J. Polifroni, S. Seneff, and V. Zue, “GALAXY: A Human-language Interface to On-line Travel Information,” Proc. ICSLP, pp. 707–710, Yokohama, Japan, September 1994.
5. Meng, H., S. Busayapongchai, J. Glass, D. Goddeau, L. Hetherington, E. Hurley, C. Pao, J. Polifroni, S. Seneff, and V. Zue, “WHEELS: A conversational system in the automobile classifieds domain,” *These Proceedings*, 1996.
6. D. Colton, R. Cole, D. Novick, and S. Sutton, “A Laboratory Course for Designing and Testing Spoken Dialogue Systems,” Proc. ICASSP-96, pp. 1129-1132, Atlanta, GA, 1996.
7. V. Zue, S. Seneff, J. Glass, D. Goddeau, D. Goodine, C. Pao, M. Phillips, and J. Polifroni, “PEGASUS: A Spoken Dialogue Interface for On-Line Air Travel Planning,” *Speech Communications*, Vol. 15, pp. 331–340, 1994.
8. Zeigler, B. and B. Mazor, “Dialog Design for a Speech-interactive Automation System,” Proc. EUROSPEECH-95, pp. 113-116, Madrid, Spain, 1995.