A Form-Based Dialog Manager for Spoken Language Applications

David Goddeau, Helen Meng, Joe Polifroni, Stephanie Seneff, Senis Busayapongchai

Spoken Language Systems Group
MIT Laboratory for Computer Science
Cambridge, MA 02139

ABSTRACT

A new generation of spoken-language applications is emerging which combines speech recognition with natural language understanding, discourse resolution, language generation, and dialog management. These systems engage the user in a multi-utterance conversation in order to accomplish their task. Dialog management is an important component of such a system. Its goal is to make the conversational interaction with the user as natural and efficient as possible while keeping the user within the boundaries and capabilities of the application. A popular approach to dialog management is based on a finite-state model, where user utterances trigger transitions between the dialog states and these states, in turn, determine the system's response. This paper describes an alternative dialog planning algorithm based on the notion of filling in an electronic form, or "e-form". The e-form has slots representing information to be provided by the user. Each slot has associated prompts that guide the user through the dialog, and a priority which determines the order in which the system tries to acquire information. These slots can be optional or mandatory (e.g., the information must be provided before the task can be completed). 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 dialog. Constraints or requests can be specified in any order the user chooses based on their requirements. This approach allows for a range of mixed-initiative dialogs, from primarily user-driven to primarily machine-driven. The e-form-based dialog planner has been used in an application to search a database of used car advertisements. The database contains some 5000 entries indexed by a variety of features. The goal of the application is to assist the user in selecting, from this database, a small list of cars which match their constraints.[1]

1. Introduction

Conversational systems are an emerging paradigm for human-computer interaction in which the user and system participate in a multiple-utterance spoken dialog in order to achieve some goal or complete a task. These systems combine speech recognition and natural language understanding technologies with natural language generation and speech synthesis to provide two-way spoken communication.

A key component of these systems is dialog management. The task of dialog management is to ensure that the conversation remains coherent and keeps moving towards the goal. The system need 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 correction when the user's input seems anomalous (perhaps due to a recognition error).

There have been several approaches to dialog management embodied in recent systems. Early approaches were ad hoc collection of routines constructed for a particular application [**?**] [**?**] [**?**]. These could be highly engineered to the application and would often work fairly well. However, they could not easily be ported to a different application, and their complexity made it difficult to predict system behavior or modify the algorithm.

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

The advantage of these systems is that they are relatively straightforward to design and their behavior is predictable. The disadvantages of these finite-state scripted systems are that the dialog tends to be inflexible and machine-initiated. They are therefore most suited to applications in which the interaction is machine-define and can be structured as a form-filling task or a tree of yes/no or short answer question. In addition, they can be fragile to input errors or user changes. Once a tree-based dialog 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 dialog dow a different branch of the tree.

This paper presents a dialog management scheme intended to overcome some of the limitations of these appraoches to dialog. In particular, it is intended to enable user-initiative and mixed-initiative dialogs, 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.

2. E-Form Based Dialog Management

The basic data structure of this approach can be thought of as an electronic form (or e-form)[1] The e-form is represented in the same semantic frame language as the input utterances. It contains slots for the relevent constraints of the application domain. For example, in a application helping a user shop for a refrigerator, there might be slots for brand name, model, capacity, energy rating, price, reliability, and linear dimensions (ie it must fit through the user's door). These constraints will have different priorities to different users; for any given user some may be critical while other irrelevent. Furthermore, the constraints are related (larger capacity model cost more, some brand have better reliability, etc.). 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 is clearly unsatisfactory. The user must be allowed to explore multiple combinations to find the result which best matches their desires.

–also user may not know domain sufficiently to fill slots - simple form filling does not allow prioritization.

2.1. Control

The basic control strategy

In addition to the constraint slots, the e-form also contains components to keep the dialog moving forward. There are prompts associated with each slot. The system uses these slots to request information from the user.

2.2. Algorithm

The algorithm for processing an input utterance, represented as a semantic frame, has 5 stages.

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

The first step involves mapping contents 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 contraint on the age of a car can be expressed as a model year "1990 Hondas", a range of years, "after 1990", "later than 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 specific handling. The treatment of number fragments "between 1990 and 1993" is linguisticly ambiguous between years and dollar amounts. It often require specific domain knowledge, or reference to the dialog context (e.g. the last prompt) to resolve.

[1] It should be emphasized that the e-f0rm is merely a convenient representation of user constraints. It is not intended for strictly "form-filling" applications, which are typically have machine-directed dialogs and machine-defined goals.

```
{c display
   :topic {q named_auto
             :number "pl"
             :pred {p make
                      :topic "honda" }
             :pred {p model
                      :topic "civic" } }
                  }
```

Figure 1: Semantic frame for the sentence "Show me the Honda Civics."

```
{ eform
   :make {p make
            :topic "honda" }
   :model {p model
             :topic "civic" } }
```

Figure 2: E-form derived from input frame

The second step combines the information in the new e-form with the e-form containing the current context. 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 over-ride beginning a new avenue of exploration. This ambiguity can be partially resolved with heuristics on the constraints. For example, 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. These heuristics can be generalized across fields (ie most range constraints follow similar constraints), but are difficult to express in a dialog scripting language. ***** (we need to introduce the desire to generalize earlier – or only at the end).

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. 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. Attached to each slot is a table specifying which other slots are to be cleared or over-ridden in response to new information for that slot.

—-inheritance subtlties, overrides, etc —

Query generation — straightforward from e-form. Application specific.

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

The response to the user has two functions, to summarize the results of the latest query and to prompt the user for information which might move the dialog closer to the goal. In a display-oriented application, the query summary consists of a spoken response and a tabular or graphical display of the query result. In a displayless application, the spoken response must be expanded to summarize the query results.

control operation –scatch that

3. Implementation in the WHEELS Domain

The e-form based dialog manager has been implemented in an application called WHEELS[**?**]. This application presents the user with a database of approximately 5000 classified ads for used automobiles[**?**]. The user browses the database using spoken language to find ads for vehicles which satisfy their needs and constraints. This application is intended primarily for telephone-only (displayless) delivery. The goal of the interaction is to narrow the list of ads down to a small number (for example, less than 5) which can then be read or fax-ed to the user.

This application is implemented using the infrastructure of the GALAXY[**?**] system. This is a distributed framework for organizing conversational systems to optimize research-sharing and extensability. Under this framework, the WHEELS application 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.

— form description – Take the case of the used car domain, which the five main fields :make, :model, :year, :mileage, and :price, and some secondary fields such as :color.

3.1. E-Form Example

Subtleties and Overrides

4. Experimental Results from the WHEELS Domain

competion time

```
{c what_about
   :subject 1
   :olang "english"
   :pred {p less_than
            :topic {q dollar_amount
                       :name 8000
                       :units "dollar" } } }
```

Figure 3: Input frame for "how about under 8000 dollars"

```
{ eform
   :price {p less_than
             :topic {q dollar_amount
                        :name 8000
                        :units "dollar" } } }
```

Figure 4: E-form derived from input frame

```
{ eform
   :make {p make
            :topic "honda"
            type "make" }
   :model {p model
             :topic "civic"
             type "model" }
   :price {p less_than
             :topic {q dollar_amount
                        :name 8000
                        :units "dollar" } } }
```

Figure 5: Query e-form combined from context and input

percent co-operative – value of flexibility

5. Summary and Conclusions

The dialog management paradigm described above was designed to support two goals:

- to handle user-initiative and mixed-initiative dialogs, and
- to encourage dialog managers that are predictable and verifiable.

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 dialog context.

6. Future Work

Future work on this dialog 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 dialog manager. The current system as implemented in the WHEELS domain pursues a single goal. Many interesting application, however, have a complex structure with multiple subgoals. For example, a travel planning application might have as its first goal extracting the users desired itinerary. Then each leg of the journey must be instantiated with an airline flight, or other mode of transportation. Finally, the system may inquire after the users need for hotel or rental car reservations. One challenge of such a task is that the various subgoals can interact. For example, the opportunity to take advantage of a reduced airfare or hotel rate, which may not become apparent until late in the dialog, may motivate the user to change the overall structure of their itinerary.

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

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

7. REFERENCES

1. D. Goddeau, 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. In *Proc. ICSLP*, Yokohama, Japan, September 1994.