Proc. of the Internat. Conf. Speech Language Processing, Beijing, China, Oct. 2000.

DIALOGUE MANAGEMENT IN THE BELL LABS COMMUNICATOR SYSTEM

Alexandros Potamianos, Egbert Ammicht and Hong-Kwang J. Kuo

Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974, U.S.A.
email: {potam,eammicht,kuo}@lucent.com

ABSTRACT

This paper describes a dialogue manager and its interaction with semantics and context tracking in a spoken dialogue system developed for general information retrieval and transaction applications. The dialogue system supports the following basic functionality: electronic form filling, database query, result navigation, attribute-value pair referencing, and value and reference resolution. General data structures and algorithms for representing and resolving ambiguity in a spoken dialogue system and a parsimonious parameterization for all application-dependent semantic and dialogue information are proposed. Dialogue management algorithms examine the semantics and dialogue state and adapt to the user's needs and task necessities. These algorithms are applied to a travel reservation application developed under the auspices of the DARPA Communicator project. The proposed algorithms are application-independent and facilitate ease of developing new spoken dialogue systems by changing only the semantics encoded in the prototype tree and the domain-dependent templates used by such components as the parser and the prompt generator.

1. INTRODUCTION

Despite the significant progress that has been made in the areas of speech recognition and spoken language processing, building successful dialogue system prototypes still requires large amounts of development time and human expertise. In addition, natural language understanding algorithms and dialogue management algorithms have little generalization power and are not portable across application domains. Our main goal is to reduce prototyping time and effort by creating application-independent tools and algorithms that automate the design process and can be used by non-expert application developers.

Extensive literature exists in application development and graphical user interface design for the keyboard and mouse modalities. One of the basic differences between a spoken language interface and a keyboard and mouse interface is the inherent existence of ambiguity in spoken language. The dialogue manager must have knowledge and access to ambiguity and try to resolve it. In this paper, we attempt to formalize the types of ambiguities present in spoken language and create the appropriate data structures to represent them. By highlighting the main difference in semantics between the spoken and other modalities, our system is more easily extensible to multi-modal input.

A further goal is to create an intelligent adaptive speech interface. The dialogue system should allow for user exploration, provide alternate routes to completing a task, and should never let the user get lost or trapped. Versatility, customizability, cooperation, and supervision (on a need-only basis) are some of the features of a good interface [4].

The proposed system builds on recent efforts in dialogue system design [1, 2, 8, 6, 3, 7]. It uses a hierarchical semantic representation that is dynamically constructed based on user input (see also [6]), as well as an agenda, which is an ordered list of tasks to be accomplished [6, 3]. The adaptive initiative module introduced in [2, 1] is also part of the Communicator system. Additional features of the dialogue system include the clear separation of application-specific (e.g., the artificial intelligence module) and application-independent components, a formal representation of semantic ambiguity, and application-independent context tracking and dialogue management algorithms.

2. ARCHITECTURE AND SYSTEM OVERVIEW

The system is built using a star hub architecture as required by the DARPA Communicator project. The MIT Galaxy hub provides the basic message routing, logging and state maintenance capabilities for the various servers making up the application [7]. We chose to organize the application into three servers in the current system, corresponding to a basic functional decomposition of the application. Each server in turn consists of a controller that interacts with the various component modules and the outside world. We have

- the telephony platform responsible for all message traffic to and from the user. The automatic speech recognition (ASR) and text-to-speech (TTS) modules are incorporated. [9, 10]
- the understanding and dialogue management server that provides the core of the application and is described in further detail below.
- the flight information database server [8] that allows live queries for flight data over an internet connection.

Functionality such as inter-server traffic, shared state data maintenance and time-stamped event logging is handled by the hub.

2.1. System Overview

The objective of the application is to collect and combine information from a user and from various databases. This information must be consistent and complete, in the sense that the application must be able to provide the service expected by the user. In the present application, this implies that all information required to book a travel itinerary has to be assembled. The basic unit for information storage is the attribute-value (AV) pair, (e.g., CITY:Boston). AV pairs are assembled and stored in tree data structures.

The understanding and dialogue management server consists of the following modules: parser/interpreter, context tracking, tree note-pad, dynamic electronic form, initiative tracking, artificial intelligence, agenda, and output generation. In brief, these modules interact as follows.

A user utterance is first analyzed in the semantics and context tracking modules: i) The parser/interpreter extracts a set of attribute-value pairs from the user utterance. ii) The context tracking module next augments this representation with dialogue context information (based on pragmatic information). iii) The tree note-pad maintains the list of current and previously collected semantic data representations, merges those that are consistent, and highlights ambiguities.

The dialogue management and prompt generation functionality is then provided by the remaining modules: iv) The electronic forms contain an ordered list of the attribute-value pairs that can be elicited from the user; the order is based on the relative importance of the AVs for task completion and the system's confidence in their values. These forms are updated at every dialogue turn. v) The initiative module computes a normalized score of the task and dialogue initiative based on system and user cues [1]. vi) The artificial intelligence module contains probabilistic inference rules, consistency checking rules for AV pairs and agenda modification rules; these rules are application-specific. vii) The agenda is a high-level ordered list of tasks to be accomplished, updated at each dialogue turn based on user input. viii) The generation module uses the initiative scores, the top candidates from the electronic forms and the agenda to generate an output semantic representation that is then translated into spoken form before being played out to the user with TTS.

3. SEMANTICS AND CONTEXT TRACKING

The information assembled by the application must conform to a certain structure. In the present work, these relationships are represented in the form of a tree: the *prototype tree*. All data obtained from the user and extracted from database queries are stored in related structures: the *application tree* that maintains the information collected in a given transaction, and the ambiguous trees that maintain all position ambiguous data.

3.1. Prototype Tree and Application Tree

There are two major relationships between the various concepts in the dialogue manager which are expressed using a tree structure: the part-of-whole relationship and the member-of-a-class relationship. Member-of-a-class relationships are represented by a class definition of an object-oriented language; each member of a class is an instantiation of the same class definition. These classes form the tree nodes. Part-of-whole relationships are represented by the tree edges, such that each node is considered a part of its parent node. In the example in Fig. 1, a trip consists of flights and hotels; a flight in turn consists of legs, etc. Flights, hotels, and legs are classes. This tree (referred to as the *prototype tree*) thus represents the semantics of the application; it holds no AV values (see Fig. 1).

The objective for the application is to interact with the user to build a data structure, the *application tree*, that conforms to the semantics. Node specific constraints on the application tree such as multiple instantiations of sub-trees (e.g., a flight may consist of one or more legs) are encoded by the individual classes. These classes are instantiated as necessary and added to the application tree in conformance with the prototype tree structure. In addition to the prototype tree relationships, there are rules that relate two or more concepts or concept instantiations. These relationships are not incorporated in the data structures defined here, but are instead part of the artificial intelligence module. In support of this functionality, information that is specific to the dialogue and application manager, such as e-forms and the agenda are also maintained (see Section 4).

Prototype trees, class definitions, and associated parameterizations are specified from data files exclusively, so the associated algorithms do not depend on the particular semantics of the application. Therefore the application semantics can be modified easily, and new applications can be built without changing the code base.

3.2. Parser and Interpreters

Results from the recognizer (best, n-best, or word-graphs) are recursively composed with a rule transducer to produce the best parse. The parse tree is interpreted and transformed to a representation that is consistent with the semantics of the prototype tree, but encodes ambiguity as described below. Parser-dependent interpreters are used for the date, time, and city concepts. These interpreters can be re-used in other applications that use similar concepts. The parser is described in more detail in [5].

3.3. Context Tracking and Ambiguity

Given an AV pair, two types of ambiguity can arise as illustrated in Fig. 2. A position ambiguity occurs when the mapping of the AV pair onto the prototype tree is not uniquely defined. For example, the pair CITY:Boston could be an arrival or departure city in the current leg. In contrast, a value ambiguity occurs when alternate values exist for a given attribute, such as two different city names. Both types of ambiguity are expressed in the trees produced by the parser by duplicating all branches of the prototype tree from a node holding the AV pair to the node(s) at which the ambiguity occurs.

The branches produced by the parser and interpreters do not in general extend all the way from the root to the leaves of the prototype tree. This ambiguity must be resolved by the context tracker that tries to find the most probable completion of the parser output based on the semantics of the prototype tree and the context of what the user or the system have said. The algorithm for merging context and parser branches is application-independent and is outlined next: If an unambiguous and non-conflicting completion of the branch exists based on context the branch is completed. If there is ambiguity (part of the branch is missing) a corresponding position ambiguous tree is constructed. If there is no ambiguity but there is conflict (overlap between context and parser branch) the conflict is resolved in favor of the parser. Finally, branches that lead to both position and value ambiguities are discarded.

The context also provides the means to assign confidence scores to ambiguous branches. These in turn form the basis for the conflict resolution (a function of the dialogue manager) and branch merging routines (a function of the note-pad).

3.4. Persistent Semantics and Tree Note-Pad

Trees with completed branches may be merged into the application tree. A simple application-independent algorithm is used for merging branches from different trees into the application tree. Consider two leaf nodes that correspond to the same position in the prototype tree. If they have the same value, the nodes are merged; otherwise, they become value ambiguous siblings. Position ambiguous trees are maintained separately in the note-pad.

A snapshot of the note-pad is shown in Fig. 2: The single tree to the left illustrates the application tree, holding all unambiguous and value ambiguous information. The tree to the right shows one of a forest of position ambiguous trees.

4. DIALOGUE MANAGER AND GENERATION

The main functionality of the dialogue manager is to elicit attribute-value pairs from the user, to resolve value and position ambiguity, to inform the user about updates in the application tree, to perform

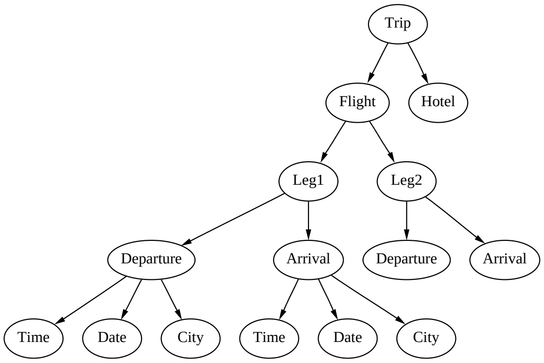

Figure 1: Part of the prototype tree.

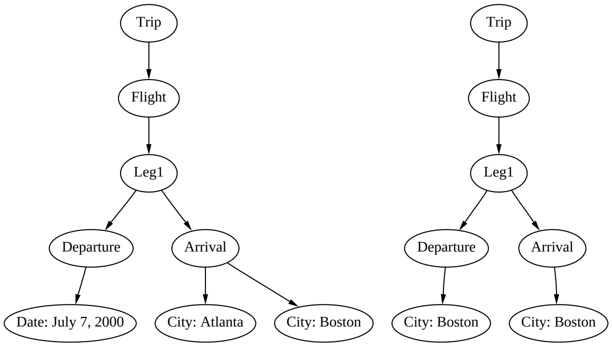

Figure 2: A snapshot of the Note-Pad.

database queries and communicate results to the user, and to help the user navigate the database results. The main components of the dialogue manager are the electronic form, the agenda, and the prompt generation modules. The emphasis again is on designing dialogue management algorithms that can be ported with little or no modification to other information retrieval and transaction applications.

4.1. Dynamic Electronic Forms and Ambiguity Resolution

The electronic form (e-form) module provides the machinery for eliciting attribute-value pairs from the user and resolving ambiguity in the application tree semantics. The e-form consists of a set of attributes that are needed to form a database query (similar to a traditional electronic form), and scores associated with those attributes, e.g., a flight leg is an e-form consisting of departure/arrival city, time, and date attributes. The score (a normalized number between 0 and 1) is attached to each attribute in the application tree and is dynamically updated at each dialogue turn.

The e-form scores encode two pieces of information: the necessity of an attribute for successfully completing the query and the confidence that an attribute has been given an unambiguous value. Attributes can be classified broadly into three categories: mandatory, optional and irrelevant. The task-relevance of attributes is, in our case, a continuous number between 0 and 1[1]. For example, for a flight query, cities and dates are more critical than time or airline carrier information. Conditional importance of attributes is also captured in the e-form, e.g., when the arrival date is specified, the importance of specifying a departure date is reduced.

The second part of the e-form score consists of the confidence scores provided by the understanding module for each attribute-value pair. Confidence scores for each attribute-value pair and task-relevance scores for each attribute are combined to obtain a single e-form attribute score. The combined score is used by the dialogue manager to rank order the attributes of the e-form and decide which attribute (if any) should be in focus for the next dialogue turn. This algorithm is application-independent. For the attribute in focus, the dialogue manager selects the appropriate dialogue act based on the value(s) (if any) and confidence(s) associated with those values: *'prompting'* if the attribute is not instantiated, *'re-prompting'* if the attribute has many values all with low confidence, *'explicit confirmation'* if the attribute has more than one value with high confidence, or *'implicit confirmation'* for an AV with mid to high confidence.

4.2. Agenda

The agenda consists of a sequence of e-forms, dialogue acts, and application actions. Examples of e-forms are flight legs, hotel reservations, and car rentals. Typical application actions include summarization of flight information, confirmation of e-form values, and database query. A focus is associated with each entry in the agenda, e.g., 'first flight leg' can be the focus of an e-form agenda entry. The focus is encoded using the semantics from the application tree (a branch in the tree). Note that no low level information about the attributes in the e-form exists in the agenda since the e-form sub-dialogue manager is automatically and dynamically generated as described in the previous section. There is also machinery to add/delete parts of the agenda as the user requests it (dynamic agenda generation). The artificial intelligence module can also modify the agenda.

4.3. Dialogue Flow: An Example

The typical dialogue flow for the travel reservation application is shown in Fig. 3. Note that although the figure represents the dialogue flow for the travel application, it could very well be the dialogue diagram for any other information retrieval and transaction application. Each module in the diagram represents an agenda entry. The two most important modules are labeled as "fill" and "navigation" in Fig. 3. Both modules are to a large extent application and modality independent. The "fill" module uses dynamic electronic forms to elicit attribute-value pairs from the user as discussed above. The "navigation" module helps the user select the appropriate result from the database. Conditions for transitions between modules are shown on the arcs of the diagram.

4.4. Adaptive Initiative Module

In [1, 2], an algorithm is described for tracking the degree of initiative that should be given to the user at each dialogue turn. The initiative tracking algorithm uses dialogue and semantic cues to identify task and dialogue progress. A different dialogue strategy is selected based on the degree of initiative that should be given to the user. The adaptive initiative strategies have been shown to improve subjective and objective dialogue quality ratings [1]. The set of cues used in the initiative tracking algorithm incorporated in the Communicator system has been enhanced to include position and value semantic ambiguities.

4.5. Prompt Generation

The natural language generation module is template based. For each dialogue act, e.g., "inform," "resolve position ambiguity," "resolve value ambiguity," a set of attributes and values are specified in a template. A module is used that translates generic

[1] The task-relevance of attributes can be computed automatically from the database; typically high-entropy fields are more relevant.

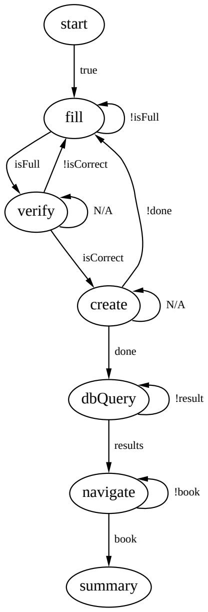

Figure 3: Dialogue Flow.

attributes and values to spoken forms. Certain dialogue acts correspond to multiple application actions, e.g., "database query" and "database results" both correspond to the "inform" dialogue act. In this case, these actions must also be specified to the generation module.

5. RESULTS

In this section, some preliminary results from the June 2000 DARPA Communicator data collection trial are presented. The purpose of this trial is data collection and system debugging for various sites participating in the DARPA Communicator research project. A simple version of the system described in this paper was used for the data collection trial. The system had basic flight reservation capabilities but limited navigation capabilities. A high-level flow diagram of the dialogue manager is shown in Fig. 3. The dialogue manager used static electronic forms and fixed initiative. The speech recognizer and text-to-speech synthesizer used domain-independent acoustic models and synthesis models. No barge-in was available. Flight schedule and price information was provided through a back-end that was connected to the internet. [8]

Naive users were asked to complete flight reservations. In most cases, the caller was given a fixed itinerary. Despite the simplicity of the fielded system, out of a total of 51 calls after excluding database and other platform system failures, over 75% task completion was achieved. User complaints focused on the frustrating experience of being repeatedly misunderstood, the poor quality of the synthetic voice, and the lack of help and task supervision[2]. In this preliminary system, no confidence scores were implemented, resulting in false triggering of the disambiguation routines due to ASR errors.

In addition, a five-question user survey was prepared by the Communicator evaluation committee and was filled out by the callers where a scale of 1 to 5 was used to rate each question. (1 represented the strongest agreement to a favorable statement about the system.) The survey's questions focused on the quality of the spoken interface and the application. The results are currently being analyzed.

We believe the task completion and subjective scores for the system will improve substantially once the major bugs have been resolved and further enhancements to our algorithms are implemented. The system will be formally evaluated in upcoming trials and the results will be reported in future papers.

[2] The dialogue manager was designed with few help messages and no suggestions to gauge the ability of naive users to complete the task with no supervision.

6. CONCLUSIONS

We have developed a new application-independent framework for semantic representation and context tracking and showed how the semantic persistence and ambiguity resolution features are used by the dialogue manager. The major features of the system are its parsimonious domain representations, portability, adaptive dialogue strategies, adaptive system intelligence, elaborate semantic representations, and a clear separation of application-dependent components. The ultimate goal is to provide flexible and upgradeable dialogue systems, with general and reusable components. Designing an interface that is versatile and customizable is essential for enhancing user experience. More research is needed to meet these challenges.

Acknowledgments: This work was partially funded by DARPA under the auspices of the Communicator project. The data collection was designed and organized by NIST. The user survey and scenarios were created by the DARPA Communicator evaluation committee. The authors would like to express their sincere appreciation to Jennifer Chu-Carroll and Andrew Pargellis for many helpful discussions; to Antoine Saad and Qiru Zhou for building the Communicator-compliant audio platform; to Matt Einbinder and Roger Barkan for their help with system design, implementation, and data collection; to the Colorado University team for providing the database backend.

7. REFERENCES

[1] J. Chu-Carroll, "MIMIC: An adaptive mixed initiative spoken dialogue system for information queries," in *Proceedings of the 6th ACL Conference on Applied Natural Language Processing*, (Seattle, Washington), May 2000.

[2] J. Chu-Carroll, "Form-based reasoning for mixed-initiative dialogue management in information-query systems," in *Proc. Eurospeech*, (Budapest, Hungary), pp. 1519–1522, Sept. 1999.

[3] E. Levin, R. Pieraccini, W. Eckert, G. D. Fabbrizio, and S. Narayanan, "Spoken language dialogue: from theory to practice," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, (Keystone, Colorado), Dec. 1999.

[4] A. Potamianos, H.-K. J. Kuo, A. N. Pargellis, A. Saad, and Q. Zhou, "Design principles and tools for multimodal dialog systems," in *Proc. ESCA Workshop Interact. Dialog. Multi-Modal Syst.*, (Kloster Irsee, Germany), June 1999.

[5] A. Potamianos and H.-K. J. Kuo, "Statistical recursive finite state machine parsing for speech understanding," in *Proc. ICSLP'2000*, (Beijing, China), Oct. 2000.

[6] A. Rudnicky and W. Xu, "An agenda-based dialog management architecture for spoken language systems," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, (Keystone, Colorado), Dec. 1999.

[7] S. Seneff, E. Hurley, R. Lau, C. Pao, P. Schmid, and V. Zue, "Galaxy-II: A reference architecture for conversational system development," in *Proc. ICSLP'98*, (Sydney, Australia), Dec. 1998.

[8] W. Ward and B. Pellom, "The CU Communicator system," in *Proc. Workshop on Automatic Speech Recognition and Understanding*, (Keystone, Colorado), Dec. 1999.

[9] Q. Zhou, C.-H. Lee, W. Chou, and A. Pargellis, "Speech technology integration and research platform: a system study," in *Proc. Eurospeech'97*, (Rhodes, Greece), pp. 621–624, Sep. 1997.

[10] Q. Zhou, A. Saad, and S. Abdou, "An enhanced BLSTIP dialogue research platform," in *Proc. ICSLP'2000*, (Beijing, China), Oct. 2000.