ABSTRACT

The development of natural language spoken dialog systems requires collection and labeling of a large set of user-system interactions (knowledge). Ideally, it should be collected from live traffic in the field, since scripted scenarios in a lab typically result in unnatural phraseology. To achieve this, we introduce an extension of the "Wizard of Oz", a hidden human agent who oversees the machine side of the interaction in collaboration with the automated dialog manager, unbeknownst to the user. The Wizard can provide a continuum of supervision ranging from explicitly controlling every step to a Wizard-override mode where the machine operates semi-autonomously and the human overrides only when necessary. All interactions are instrumented and entered directly into a database using a standard interface and SQL. The collected data will typically be analyzed in the laboratory to quantify system performance, and to introduce algorithmic improvements prior to a new set of experiments. Over the course of system development iterations, the nature of the Wizard evolves over a continuum of functions from actual control (in the absence of any system knowledge) to passive observer. The user cannot tell the difference.

1. INTRODUCTION

A very significant area of application for natural language spoken dialog systems is better automation of call-centers maintained by businesses to provide customer care. The development of such natural language spoken dialog systems presents a problem for the system designer. Building such a system requires collection and labeling of a large set of user-system interactions (knowledge) ideally collected from actual live traffic - scripted scenarios in a lab typically result in unnatural phraseology, with users responding differently than found in the field. In the absence of knowledge, however, an automated data collection system that provides users with the services they require cannot be built.

To overcome this problem, we introduce a hidden human agent, the "Wizard of Oz", to oversee the machine side of the interaction in collaboration with the automated dialog manager, unbeknownst to the user. The Wizard can provide a continuum of supervision ranging from explicitly controlling every step of the dialog, to an observer mode where the machine operates autonomously for the most part and the human overrides only when necessary. This guarantees satisfied users who are provided the services that they request.

In this paper we explore the issues of such a Wizard data collection system, and describe some of our experiences in using an earlier prototype for a telecommunications application [1].

Since a Wizard's actions are in general consistent with his/her understanding of the user responses, these responses are in effect labeled with the extra-linguistic information (such as topic identifiers) required for the derivation of appropriate natural language understanding algorithms. All interactions are entered directly into a database using a standard access interface and SQL.

The working system we describe integrates a telephony interface, a real-time full-duplex audio streaming engine, speech recognition and prompt generation engines, a dialog manager, and a Wizard GUI component to allow a human agent to adjust the dialog manager's behavior [2]. Moreover, a fully integrated data collection facility (we use a standard database interface) is used to record all dialog session information. Operation of the system results in databases that fully reflect not only every aspect of a user-machine transaction, but also a possible (but not necessarily required) Wizard intervention. To allow rapid modification of the Wizard capabilities, application dialog, and the schema in the databases, we enabled (but do not require), the use of a scripting language.

In a typical application (e.g., call routing), a service provided by a human agent may already be deployed in the field. If so, this system may simply tap into the audio stream between the agent and the user, rather than provide the complete application. In addition to overseeing the new system, the Wizard also enters the information gathered into the original system, and eventually directs it to provide the automated service.

The development of a working natural language dialog system will typically involve a sequence of experiments. The data collected in any one trial will typically be analyzed in the laboratory to quantify system performance, and to introduce algorithmic improvements prior to a new field data collection experiment. Over the course of system development iterations, the nature of the Wizard evolves over a continuum of functions from full control (in the absence of any system knowledge) to passive observer. The user cannot tell the difference.

2. THE DATA COLLECTION SYSTEM

2.1. System Components

The wizard system uses a modular design approach. Individual components and associated data flows are shown in Figure 1. Overall, the system decomposes into two main sets of modules: the audio component that interacts with the audio data, and the application component that implements the dialogue and natural understanding systems, provides the desired services.
2.2.1 The Audio Modules

The audio components are responsible for all aspects of audio stream processing. No audio data passes between these modules and the application modules. In particular, the application sends commands that allow generation of the audio stream for prompts. Similarly, the results of the analysis of user speech by the automatic speech recognizer are passed back to the application modules.

Audio processing consists of the following subsystems:
- **Audio Subsystem** - This module implements the basic audio data flows. It receives audio input data from the prompt generator and plays them out to the user. The user’s responses are forwarded to the speech recognizer over a TCP/IP socket connection. More generally, the subsystem defines input and output channels that can be replaced at will in response to control inputs. This allows arbitrary audio sources and sinks to be used by the system. In particular, data may be taken directly from the host’s audio ports, or from an ISDN interface. Additional channels provide audio data recording and real-time monitoring capabilities. Audio data flow can be started and stopped at any point as required by the audio control.
- **Automatic Speech Recognizer** - We use the AT&T Watson speech recognition engine. The recognizer provides partial results and other parameters (e.g., acoustic confidences) at regular time intervals. The socket connection to the audio module allows the recognizer to run on a separate processor when desired.
- **Touch-tone Recognizer** - The dialog management allows two different interaction styles, voice and touch-tones. While touch-tone inputs may in principle be handled in the same way as speech (They are subject to a grammar, and require delimiting and time-out handling.) They are not incorporated in the speech recognizer module, however, since they may run in parallel to or in lieu of the recognizer. In our system, this capability is integrated with the audio subsystem.

2.2.2 The Application Modules

The application modules embody all systems not directly concerned with processing of raw audio data. They consist of the following subsystems:
- **Control Subsystem** - the master control connects and interacts with all other subsystems. Audio specific controls are pushed across a TCP/IP socket interface to an audio control module dealing with audio data issues exclusively.
- **Dialog Management (DM) and Natural Language Understanding Subsystem(s)** - responsible for user response analysis and dialog generation, based on inputs received from the control subsystem.
- **Wizard GUI Subsystem** - the Wizard agent graphical user interface. This is the key component of the system. The capabilities change from experiment to experiment.
- **Database Subsystem** - stores all data collected. In particular, the tables contain identifiers for the actual audio data (stored separately by the audio recorder subsystem), and transaction information sufficient to reconstruct dialogs and system behavior. The database schema can change from experiment to experiment.
- **Services Subsystem** - provide the actual service requested by the user. (Since the premise of the system is to collect data on live traffic, user expectations have to be met!) This module is application specific. It may in particular incorporate an Information Retrieval Subsystem to obtain external data required by either dialog management or the service subsystem proper.

2.2 Wizard System Design Issues

The basic function of the Wizard is to guide the human/system dialog in accordance with the experiment objectives. The Wizard is listening to the evolving dialog, and is presented with a graphical user interface that displays current system status and allows the Wizard to direct or override system actions (see Figure 2). The Wizard thus overrides any erroneous functioning of the system. In theory, the Wizard could override ASR output only. While resulting in a system that exhibits perfect recognition, this assumes that the DM modules work to the system designer’s satisfaction.

A much more profitable approach is to allow the Wizard to interact with the DM directly. The module analyzes user responses, following which it offers the Wizard an opportunity to modify its results. The module then integrates Wizard actions (if any), and proceeds with the dialog. If the Wizard were to simply override the DM, all further dialog would have to be carried out by the Wizard directly since the DM model of the interaction no longer corresponds to reality (i.e., the Wizard GUI would have to provide the DM functionality in parallel).
The possible Wizard actions are to a) override the system; b) agree with the system, or c) do nothing (human agents might not be well motivated, or otherwise distracted). Since Wizard actions are in general consistent with his/her understanding of the user responses, these responses are in effect labeled with the extra-linguistic information (such as topic identifiers) required for the derivation of appropriate natural language understanding algorithms. In experiments where errors are introduced intentionally, an effective approach is to present the Wizard with the actual DM analysis results and have the Wizard correct the system results as before. The DM then introduces errors according to some algorithm specified by the system designer. The advantage is that the dialog is labeled correctly, and Wizard idiosyncrasies in error type selection are avoided.

Figure 2 is a schematic representation of a possible GUI screen. The evolving dialog is displayed in a window showing system prompts and ASR output (the underlined words serve as annotation to give the user insight into the functioning of the natural language understanding module). A timer bar displays the time elapsed since the user fell silent, waiting for a response. The radio-button widget shows the default system action and allows the Wizard to override the DM.

2.3 Wizard System / Database Interactions

The primary intent of the database is to provide a structured, searchable representation of user/system interactions. All subsystems can potentially contribute to the information logged. In the interest of modularity, we have chosen to interact with the database through a standard interface from the master control subsystem using SQL. While keeping the recorded audio data as part of the database proper would simplify the required bookkeeping, space considerations lead us to keep them separate: the databases generated in an experimental trial are small, and provide a useful corpus of data for analysis in themselves.

Given the distributed nature of the design, special care must be taken with time stamps. Since many events of interest relate to the audio files, all time information is specified with respect to the audio stream sample clock, measured from the start of a given transaction. Note that related events can originate from different subsystems at different times, e.g., a prompt definition from the DM, prompt start and stop timestamps from the audio subsystem, an abort command from the Wizard GUI, possibly while the prompt is playing. The relationships of events are available in the control subsystem, which can thus collect them and write appropriate records to the database. If events are logged without structure, the later reconstruction of the user/machine transaction becomes very difficult.

Another consideration is that the database may serve as the single point of definition for application data. For example, prompts require a text representation, and possibly an audio file, a touch-tone grammar, an ASR grammar, timeout parameters, etc. This information must be available in the database, if a transaction is to be reconstructed. While in principle the DM could provide this information, a simpler design is to have the DM specify a prompt ID and a response type ID referring to data base records. It is then up to the control subsystem to retrieve all relevant information and forward suitable commands to the relevant subsystems. For the example described, this might entail (among others) i) displaying the prompt on the Wizard GUI, ii) causing the prompt generator to play the prompt to the user, iii) enabling the touch-tone and ASR systems with appropriate commands, iv) initializing the logging system with a record for a new turn.

A related issue is that using the database as a single point of definition for system data allows for more ready modification of the system – even during an experimental trial.

2.4 Preexisting Systems

When designing a natural language dialog system, it is often the case that there already exists a deployed system. This is
particular true for customer care centers maintained by businesses. In the simplest case, this may be a station for a human agent. In the case of a touch-tone based system, a human backup agent station is typically available, as well as a means for routing calls directly to this station. Under such conditions, the system development work required for deployment of a Wizard data collection system can typically be simplified, if a way can be found to tap into the agent/user audio data stream. Rather than replacing the original agent station, the Wizard data collector may run in parallel with it. The agent allows the Wizard system to conduct the interaction with the user. The payoff is that the Wizard system need no longer incorporate a service module; instead the agent uses the original system to provide the services requested.

For services requiring significant data entry, two human agents might be required: one operating the wizard data collection system, while the second operates the system that actually provides the service. Any external information retrieval (e.g., a caller profile) required by the system might also be left to the original station, with Wizard intervention to provide the collection system DM with this additional information.

The configurable audio channels of the audio module take on additional significance in this context: by adapting channels to the actual requirements for the tap, the remaining system remains unaffected.

3. FIELD EXPERIENCES

The Wizard data collection system will in general change between experimental trials: as ASR, Natural Language Understanding and DM are improved, the role of the Wizard changes continuously from ‘director’ to ‘labeler’, eventually to passive observer or supervisor. The dialog changes too, requiring new Wizard displays. Thus, the Wizard GUI must change. The DM/Wizard interaction may change. The database schema are likely to change – desired data for the next trial change. The database may be possible.

In every trial minor changes to the system occurred while the collection was underway, such as changing dialogs and time-out parameters to improve the perceived responsiveness of the system to the user, proving the effectiveness of the scripting language.

4. CONCLUSION

Knowledge acquisition for a natural spoken language dialog system should ideally be collected from live traffic in the field to avoid bias introduced by a user base that is aware of its participation in an experiment. To conduct such experiments, the data collection system must be able to conduct dialogs acceptable to the users of the system, and must ultimately provide the services expected. Since development of the underlying language understanding algorithms does not in general proceed uniformly, branches in a dialog may be automated to various degrees. In the first trial no automation may be possible.

In this paper, we have extended the notion of a hidden human agent, the “Wizard of Oz”, to oversee the machine side of the interaction in collaboration with the automated dialog manager, unbeknownst to the user. This human agent provides the system with a continuum of supervision ranging from explicitly controlling every prompt of the dialog and every service provided by the system, to simply supervising the system and overriding only to insure that users receive appropriate care should a transaction go too far off track.

Such Wizard based systems require frequent modification of the system capabilities, Wizard GUI interfaces, and schema of the databases built up in a sequence of data collection trials. This forces system designers to put a heavy emphasis on modifiability of the code. The system design we have presented here has been used successfully for a series of data collections, starting from full Wizard control to full system automation. Throughout, users were unaware of the human supervision, allowing them to react naturally, but guaranteed of receiving the services that they expected.

5. REFERENCES