MDWOZ: A Wizard of Oz Environment for Dialog Systems Development

Cosmin Munteanu, Marian Boldea

Computer Science Department,
"Politehnica" University of Timişoara,
Bd. V. Pârvan 2, Timişoara 1900, Romania
{cosmin,boldea}@cs.utt.ro

Abstract

This paper describes MDWOZ, a development environment for spoken dialog systems based on the Wizard of Oz technique, whose main goal is to facilitate data collection (speech signal and dialog related information) and interaction model building. Both these tasks can be quite difficult, and such an environment can facilitate them very much. Due to the modular way in which MDWOZ was implemented, it is possible to reuse parts of it in the final dialog system. The environment provides language-transparent facilities and accessible methods such that even non-computing specialists can participate in spoken dialog systems development. The main features of the environment are presented, together with some test experiments.

1. Introduction

During the last years, spoken dialog systems have become an important option in human-computer interfaces, especially for access to various public information systems, but despite this rapid growth, there are yet a lot of constraints that slow down their development. Among these, the long time and high cost of building such a system, and the lack of expertise and data (Sutton et al., 1996).

Regarded as user interfaces, spoken dialog systems should obey the classical software engineering prototyping cycle. But building a prototype dialog system could be as expensive as building a working one, and yet far from the desired final functionality. The main problem is that the interaction models built off-line are poor approximations of the way people actually interact with computers.

To overcome some of these problems, the Wizard of Oz method (Fraser and Gilbert, 1991; Gibbon et al., 1997; Bernsen et al., 1998) was introduced starting from an idea in Frank Baum's story (Baum, 1900): one human "wizard", possibly with some assistance, simulates a dialog system, and collects data to be used for building a real one (not only the interaction model, but also acoustic, language, and semantic models).

For such a simulation to be as close as possible to the final system's behavior, a number of appropriate supporting tools are necessary. In the ideal case, these tools should offer the possibility for the wizard to simulate and/or control all parts of a typical dialog system (Lamel, 1998): speech recognition, semantic analysis, dialog management, domain knowledge base, natural language generation, and text-to-speech conversion. At the same time, they should allow for relevant data to be collected, and new components to be included as they become functional, passing gradually from a pure simulation to a "system in the loop" situation.

Even if lately publicly available software packages that allow to build dialog systems started to appear, like Philips' SpeechMania, or the CSLU Toolkit (Sutton et al., 1996), none of them seems to support the Wizard of Oz technique, although it is very used for spoken dialog systems development, as a large number of papers demonstrate. This means a considerable effort has to be repeatedly spent in preparing the simulations, and was the main incentive for this work.

In an attempt to fill in this vacancy, the MDWOZ development environment provides a set of tools suitable for dialog interaction model design and test, database building and query, system simulation, and data collection. They reduce the time required to prepare simulations, provide easy ways to collect data in realistic conditions, and ultimately facilitate the building of new dialog systems.

2. MDWOZ Structure

MDWOZ was conceived in a modular manner, to provide the possibility of using just parts of it during simulations, and for an easy transition from the simulation phase to the final system, including the reuse of some components and/or the use of external ones. It runs on two computers connected by a local network, and its overall structure, outlined in Figure 1, consists of four modules: Wizard, User, Data Acquisition, and Communication.

2.1. The Wizard Module

As the part where the tools supporting the wizard are located, this is the most complex module in the environment, and includes components for domain knowledge database building and access, dialog control and interaction model building and refinement, and output generation. Although automatic speech recognition, semantic analysis, dialog management, and natural language generation are only simulated at this moment, provisions have been made for them to be easily included as they become available.

2.1.1. Database Subsystem

Whatever the domain for which a dialog system is to be built, the corresponding knowledge has to be stored in a database in order to be used during human-computer interactions, and easily accessible during Wizard of Oz simulations. To support both these aspects, the database subsystem (Figure 2) includes, besides a database engine, a couple of graphic interface tools that facilitate building and consulting domain specific knowledge bases.

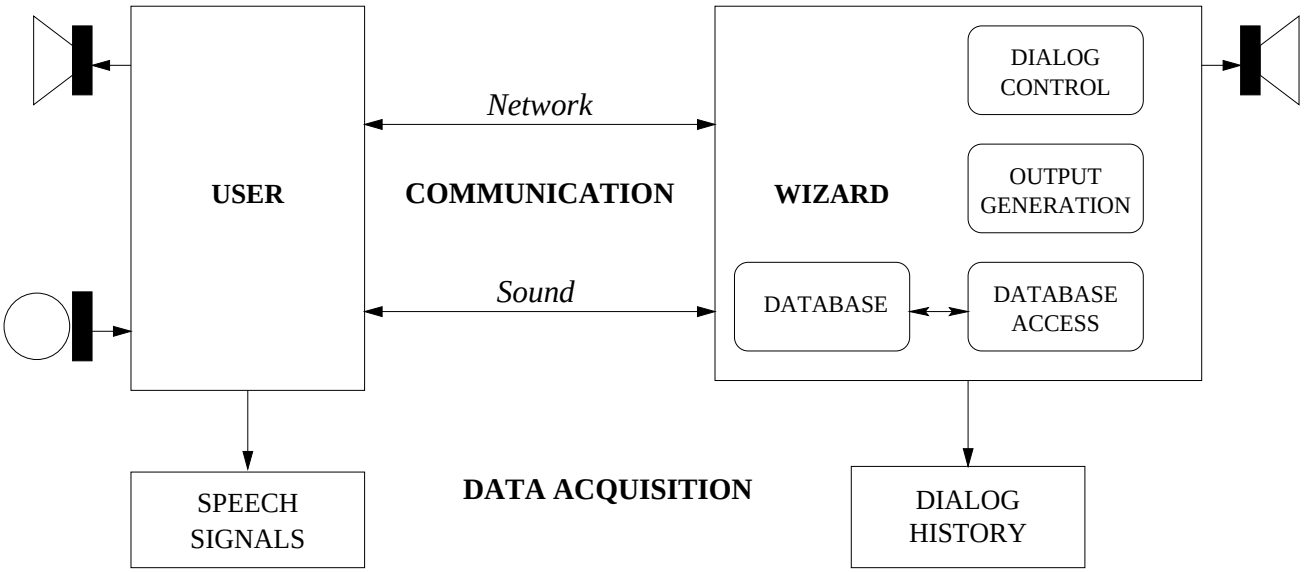

Figure 1: MDWOZ structure

The Database Engine, implemented using the mySQL (mySQL, 2000) package, is connected to the graphic interface tools `DBinput` and `DBask`, both based on the Xforms library (Zhao and Overmars, 1996), via a C-language API, and has a client-server architecture. A knowledge base can be built according to a configuration file using `DBinput`, and queried through `DBask`.

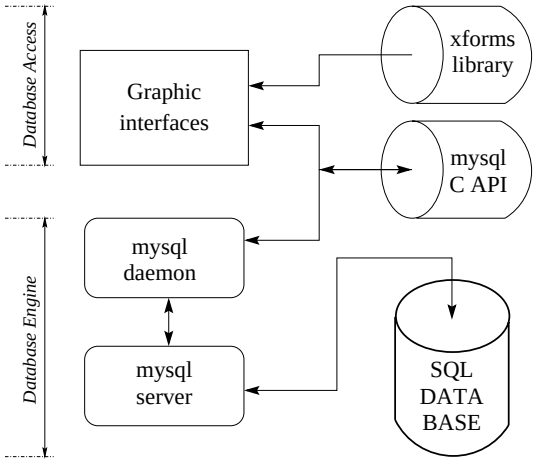

Figure 2: Database subsystem

During simulations, the wizard can search the database using `DBask`: (s)he can select any number of up to 12 displayed fields, and `DBask` will build an SQL query based on the values in the selected fields. Logical, comparison, and wildcard operators are accepted for performing more complicated queries: combining them, especially range values are easier to represent (e.g. *afternoon* could become: `>14 AND <20`). The query results will be displayed using the same interface, and in case of more than one answer, the wizard can choose a specific one using scroll buttons.

`DBinput` is similar to `DBask`, with supplemental capabilities for adding, deleting, and modifying records.

2.1.2. Dialog Control

Multiple solutions have been proposed to model and control the interactions between a dialog system and its users, but most of them can not be easily represented (if at all) in a form suitable to assist a wizard during simulations, so that for the initial version of MDWOZ we chose the least problematic from this point of view – that based on finite state models, representable as oriented graphs.

In the dialog model graph, user and system (wizard) nodes exist, according to the party which is expected to produce an utterance in that state. For an easier use during simulations, each node is uniquely named, and has associated a text, all stored in a graph configuration file indicating this name-text correspondence.

To facilitate both the graphical representation of the interaction model, and wizard's work, the Dialog Control is built around the `daVinci` (daVinci, 1998) graphs visualization package, chosen due to its unique features that allow not only drawing, but also interacting with graph nodes. A dialog model can thus be designed and refined based on a graph configuration file, and using `daVinci`'s graph drawing capabilities.

At the start of a simulation, the entire graph is displayed, in which each node is labeled with its name. Once the wizard selects a node, only that node and its immediate successors are displayed, each labeled with the associated text from the graph configuration file (Figure 3). From now on, the wizard can navigate through the graph, but can choose only from the currently active node's successors.

In a user node, an utterance from the subject is expected, based on which the wizard can formulate a query whose results will be used to generate the next wizard utterance. When the currently selected node is a wizard one, an output utterance is produced.

2.1.3. Output Generation

As mentioned before, there are user and wizard nodes in the dialog description graph. When a wizard node is

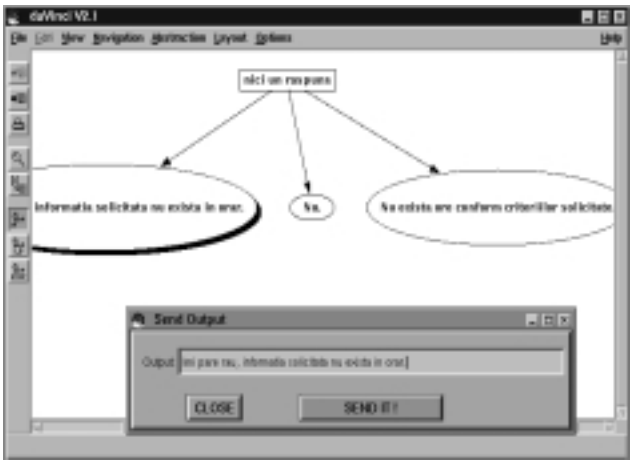

Figure 3: Dialog control and output generation

selected, an `Output` window is started (Figure 3), displaying the text associated to the selected node. This text can be subsequently modified, e.g. by adding certain `DBask` query results: to speed up this editing, database fields can be stored in an abbreviated form (e.g. "/ai" for "Artificial Intelligence"), and expanded according to a configurable set of conventions stored in a file.

In the general case, the output text can be sent to an external text-to-speech (TTS) program, and/or to the user's terminal. In the existing implementation, although a TTS system is available, to avoid potential problems due to the quality of the synthesized signal, pre-recorded words are used to synthesize the speech signal after the text is normalized.

2.2. The User Module

At the opposite end of the MDWOZ environment, the User module runs on a separate computer, and is connected to the wizard's one through the Communication module. It takes wizard outputs (text and/or voice), and displays and/or replays them to the user, which interacts with the system just by voice.

2.3. The Data Acquisition Module

As the main purpose of a Wizard of Oz simulation is the collection of various data concerning human-computer dialogs in conditions as realistic as possible, the Data Acquisition module, running on both the wizard's and user's computers, includes a series of facilities aimed at this end.

On the user side, the speech signal from the subject is recorded, together with the signal from the wizard: although the latter might be useless in most situations, as the text from which it is generated is also recorded, we include it to allow for echo cancellation and barge-in work.

On the wizard side, the dialog history is recorded, consisting of two files: one with wizard's text outputs, the other with the trace of the dialog, i.e. the names of the nodes on the path through the dialog interaction model graph. For reference, the graph configuration file used in every dialog session is also saved.

2.4. The Communication Module

This module insures the synchronization between the user and the wizard using a local network to connect the user's computer to the wizard's one. The communication is done through an audio and a data channel. The audio channel transports the speech signal between the two computers and combines the wizard and user signals into a stereo channel. The data channel synchronizes the beginning and the end of recordings, and optionally transmits the wizard's output texts to the user's terminal.

3. Experiments

The development of MDWOZ and the experiments presented here are both parts of a larger project on spoken dialog systems, and the experiments had two aims: first, to evaluate the environment itself; secondly, to collect data for training and testing a speech recognition system and a stochastic semantic analyzer, also parts of this project.

In developing a dialog system using the Wizard of Oz technique, three phases can be identified (Gibbon et al., 1997): (1) pre-experimental – for defining the dialog structures and restrictions, (2) first experimental phase – for a first system evaluation, and (3) secondary experimental phase – for recordings and improvements. In setting up the experiments, we followed almost the same staging:

1. Domain analysis and definition
2. MDWOZ setup and test
3. System simulations

(1) As a pilot project, we chose to develop a system providing information about classes timetable and examinations schedule, and to facilitate domain analysis and definition an appropriate database was built using `DBinput`.

(2) To setup and test MDWOZ, a smaller dialog system offering information only about the examinations schedule was simulated, using an appropriate interaction model, which after these tests was extended to cover also classes timetable.

3.1. System simulations

Wizard of Oz simulations usually take place in two separate rooms, but this solution implies a very good synchronization between the two sides, so for the first experiments we chose to run the tests in a single room, with a large folding screen between the two computers (which reminds us the alternative name "Pay No Attention to the Man Behind the Curtain" used for such experiments). To prevent users from hearing the wizard typing and clicking, we used a close-talking microphone attached to a pair of headphones.

To make sure the collected data are appropriate for training the semantic analyzer, i.e. at least the concepts in the database are covered, a number of scenarios have been prepared. Besides at least one scenario, each subject could also try some free dialogs, to allow for more variability.

Until the moment of this writing, 17 subjects participated in the simulations, each trying two or three scenarios and at least a free session, for a total of 49 sessions that covered about 3 hours of dialog containing 537 turns, with

an average of 10.9, a minimum of 5 and a maximum of 21 turns/session.

During the simulations period, the interaction model was changed, usually after a dialog session in which the subject asked a valid question, but there were no paths through the dialog graph allowing to answer it. This way, the interaction model size increased from 23 to 32 states. Besides adding states, in certain cases states were removed, and texts associated to wizard nodes were changed.

Data concerning the dialogs and acoustic signals were collected as described in section 2.3. The speech signals are being transcribed to be used for building an automatic speech recognition system, and together with the dialog data – to train and test a stochastic semantic analyzer, and study dialog management aspects.

3.2. Wizard Difficulties

Even if the Wizard of Oz method is a powerful technique for simulating dialog systems, it can raise a lot of problems. The main trouble we faced was "lying" to the subjects: this was particularly difficult as most of them were students in computer science, and some of them very suspicious about the system.

To be technically plausible, we told users that, with a view to a possible deployment in a distributed environment, the dialog system was implemented in a client-server manner, so it needs a person to watch the server and take notes of what is going on/wrong.

A good way to strengthen the belief that the system is real was to systematically simulate the same errors, especially at the semantic level. A few times, some users were particularly keen to keep talking to a computer, so we used some known MDWOZ instabilities to make it crash, which was also useful to increase this belief.

3.3. Users feedback

In evaluating a dialog system, questionnaires are very useful, as shown e.g. in (Andernach et al., 1993), and we collected user feedback both during experiments and immediately after, using such a questionnaire.

The questions asked were: *Have you ever used such a system? Is this the first time you use this system? Did the system supply all the information you wanted? How difficult it was for you to obtain the information you wanted? Do you think the system understood everything that you asked? When the system misunderstood your question, was it difficult to reformulate it?*.

The feedback is summarized in Table 1. After each dialog was over, it was evaluated from two points of view: user and system. From the user point of view, the dialogs were divided in *satisfactory* and *unsatisfactory* (meaning the user got/didn't get the desired information), and from the system point of view, in *failure* and *success*, depending on the system (purposely) crashing or not.

Used other information system before	No	73%
	Yes	27%
First time using this system	No	6%
	Yes	94%
System provided all information	No	0%
	Almost	67%
	Yes	33%
Obtaining information was ...	Difficult	0%
	Acceptable	46%
	Easy	54%
System understood user's questions	No	0%
	Mostly	67%
	All	33%
Questions difficult to reformulate	No	94%
	Yes	6%
User satisfaction	No	26%
	Yes	74%
System performance	Failure	4%
	Success	96%

Table 1: Feedback statistics

4. Conclusions and Future Work

Although MDWOZ is still evolving, we expect it to become fully functional shortly. However, from the tests already run, we can conclude that it provides a simple, non-expensive solution for acquiring knowledge necessary to develop spoken dialog systems, not only by data collection and interaction model definition and refinement, but also observing user behavior.

A future improvement will be a new implementation for the Dialogue Control module using a dialog description language instead of (or optionally along with) the interactive dialog graph design.

As more data will be collected, we also expect it to go from a pure simulation use to a "system in the loop" one, by the inclusion of the automatic speech recognizer and stochastic semantic analyzer trained on these data.

5. References

Andernach, T., G. Deville, and L. Mortier, 1993. The Design of a Real World Wizard of Oz Experiment for a Speech Driven Telephone Directory Information System. In *Proceedings EUROSPEECH'93*.

Baum, F., 1900. *The Wizard of Oz*.

Bernsen, N. O., H. Dybkjaer, and L. Dybkjaer, 1998. *Designing Interactive Speech Systems*. Springer.

daVinci, 1998. www.tzi.de/˜davinci.

Fraser, N. and G. Gilbert, 1991. Simulating speech systems. *Computer Speech and Language*, 5:81–99.

Gibbon, D., R. Moore, and R. Winski, 1997. *Handbook of Standards and Resources for Spoken Language Systems*. Mouton de Gruyter.

Lamel, L., 1998. Spoken Language Dialog System Development and Evaluation at LIMSI. In *Proceedings International Symposium on Spoken Dialogue*. Sydney.

mySQL, 2000. http://www.mysql.com.

Sutton, S., D. G. Novick, R. Cole, P. Vermeulen, J. de Villiers, J. Schalkwyk, and M. Fanty, 1996. Building 10,000 Spoken Dialogue Systems. In *Proceedings ICSLP'96*.

Zhao, T. C. and M. Overmars, 1996. Forms Library: A Graphical User Interface Toolkit for X. http://bragg.phys.uwm.edu/xforms.