ISCA Archive
http://www.isca-speech.org/archive

6th European Conference on
Speech Communication and Technology
(EUROSPEECH'99)
Budapest, Hungary, September 5-9, 1999

# A PERVASIVE CONVERSATIONAL INTERFACE FOR INFORMATION INTERACTION

*Ganesh Ramaswamy* *Jan Kleindienst* *Daniel Coffman* *Ponani Gopalakrishnan* *Chalapathy Neti*

IBM Thomas J. Watson Research Center
Yorktown Heights, New York

10.21437/Eurospeech.1999-588

## ABSTRACT

In this paper, we describe a new pervasive conversational system that provides access to multiple desktop applications, from multiple client devices, using multiple input modalities. Client devices currently supported include desktop and telephone, and the applications incorporated include email, calendar and address book. When the access is from a desktop, both conversational natural language and graphical inputs are supported.

The paper describes the overall architecture to support such a pervasive conversational system, along with innovations in continuous speech recognition, statistical natural language understanding, and dialog management that were developed to build the system. The paper also describes the partially unsupervised Wizard-of-Oz style setup to collect real-use data, and the performance of the statistical models constructed using this data for speech recognition and natural language understanding.

## 1. INTRODUCTION

As more and more information is available in electronic form, with the information interaction becoming increasingly complex, it is desirable to provide access to information using the most natural and efficient interfaces. In particular, with several devices (such as desktops, telephones and PDAs) that can potentially be used to access information, designing interfaces that are very similar and intuitive across the wide range of access methods and input-output modalities is a design challenge and an imperative. We believe that a conversational speech interface with mixed-initiative between the user and the machine can form the basis for such a seamless and pervasive interface to information interaction.

A typical conversational speech interface consists of speech recognition, natural language understanding (NLU) and dialog management components. The speech recognition component converts the user's utterance to a string of text, and the NLU component extracts the meaning conveyed by the recognized text. The dialog management component interfaces with backend content and decides on the appropriate action to be taken and the response to be generated.

State-of-the-art conversational systems [2], [7] have focussed on a single access method, a single backend application and a single modality for inputs and outputs. In this paper, we describe a conversational system that allows access to multiple applications (email, calendar and address book) from multiple client devices (desktop or telephone), supporting multiple input modalities (conversational and graphical). To provide the most efficient interface from multiple client devices, the "system personality" and conversational context are preserved across the access methods, but the presentation of information is customized for the client device.

The remainder of the paper is divided into three sections. In Section 2, we describe the architecture of the system, highlighting the main features of the interface. In Section 3, we describe the various statistical models that were built for the speech recognition and NLU components, including the performance of the models, and Section 4 concludes the paper.

## 2. SYSTEM ARCHITECTURE

The architecture of the multi-client, multi-application, multimodal system is shown in Figure 1. The main components of the system are the Device Handling and Abstraction, the Conversational System, and the Application Abstraction.

### 2.1. Device Handling and Abstraction

The Device Handling and Abstraction component handles the connection to the client device, and is responsible for performing speech recognition on the spoken input, and supplying the recognized text along with the identity of the input device to the Conversational System. This component also presents the response returned by the Conversational System to the appropriate output device, using text-to-speech synthesis when appropriate. The specifics of the client device are hidden from the Conversational System, which sees each client device only as a Speech Interface component. For both speech recognition and synthesis, the desktop client uses the IBM ViaVoice engine, and the telephone client uses the IBM ViaVoice Telephony Toolkit.

### 2.2. Conversational System

The Conversational System consists of an NLU engine and a Dialog Manager. The NLU engine translates recognized natural language speech input into formal language. In the current version, there are more than 700 supported formal language statements relevant to the email and calendar applications. The NLU engine and the models constructed for the engine will be described in more detail in Section 3.3.

The formal language statements are presented to the Dialog Manager. While there are several approaches for dialog management, such as the forms-based approach described in [4], our Dialog Manager is based on *decision networks*. A decision network is a recipe for accomplishing a specific transaction.

In addition to the set of decision networks, the Dialog Manager consists of a Mediator and a Multimodal History. The Mediator chooses the appropriate decision network to spawn, and the appropriate device to present the response. The Multimodal History captures all system events (both conversational and graphical) from all access methods and keeps track of the system state.

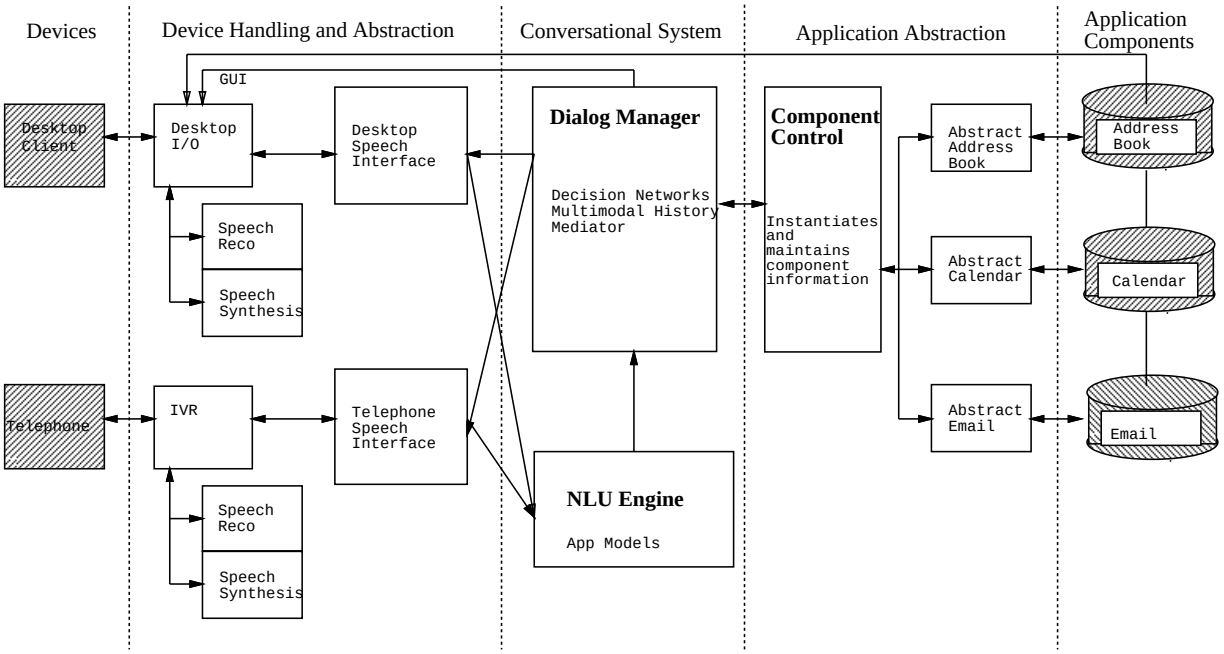


Figure 1: **Architecture of the multi-client, multi-application, multi-modal system**

The appropriate decision network to spawn is selected based on the user's command as specified by the formal language statement, and the input event source. After the decision network is determined, the Dialog Manager instantiates the network and causes it to execute within a separate thread, thereby permitting multiple transactions to be open at any given time.

The Mediator chooses the appropriate device to present the response, based on the input event source and the user's preferences. In most cases, the response will be presented to the same device that generated the input, but the user can specify a different device for the response by modifying the preferences. For example, a user may request on the telephone to view his current mail messages and have the list of messages displayed on a PDA.

The Multimodal History works as a repository, storing events generated by the user and the system within a given session. Both graphical and conversational events are saved in this repository, allowing free mixing of the input modalities. The Dialog Manager uses the Multimodal History for disambiguation and reference resolution, examples of which could be found in Section 2.4.

### 2.3. Application Abstraction

The Application Abstraction consists of two pieces: a Component Control and individual abstraction components for each application. The Component Control serves to create instances of and maintain references to the abstraction components. In addition, it functions as a switchyard for forwarding commands to the appropriate application and accumulating the responses. The abstraction components shield the Dialog Manager from the details of the applications allowing a very high level of communication, and the Dialog Manager needs to have no information on how a specific command is to be accomplished. The applications may therefore be interchanged with no changes necessary to the Dialog Manager; only the abstraction component needs to be changed.

Applications are *dynamically* created and may be addressed even when not in graphical focus. The applications to be used during a particular session are specified in a user preference file. Further, because of the Application and Device Abstraction components, a particular user's configuration may be modified with no changes to the remainder of the system.

### 2.4. System Features

In this subsection, we highlight some of the features of the system using the discourse example shown in Table 1. The first command issued by the user is a very good example to illustrate how effective a conversational interface can be in accomplishing a complex task using a single spoken command.

One of the immediately obvious system features is the ability to have multiple transactions open at any given time. Other features illustrated include several different search and query functions, disambiguation and reference resolution, seamless interaction with multiple applications, and system initiative. Features not demonstrated explicitly in the example are the support for multiple access methods and multiple input modalities. For example, the user could have issued some of the commands from a telephone and the remaining from a desktop, and the system response would have been almost identical. Similarly, the user could have mixed graphical and conversational inputs to accomplish a specific task.

## 3. STATISTICAL MODELS

This section contains a description of the various statistical models that were built for speech recognition and natural language

Table 1: **An Example of Mixed-Initiative Discourse**. The features illustrated in the example below include multiple transactions being open at any given time, system initiated transactions, ambiguity and reference resolution, multiple application interaction and support for more advanced functions such as searching and querying.

| | |
|---|---|
| 1. | U: Put all the messages from Jeff about proposal into a folder called proposal. <user tries to archive some messages> |
| 2. | S: I found no messages from Jeff. Do you want to select someone else? <system found no messages from Jeff in the inbox> |
| 3. | U: I meant Jeffrey. |
| 4. | S: Do you mean Jeffrey Smith or Jeffrey Houston? <system tries to disambiguate the name based on the address book entries> |
| 5. | S: Excuse me, you have one new message. <system takes initiative to announce arrival of new message> |
| 6. | U: Who is it from? |
| 7. | S: The author of the message is Susan McKay. |
| 8. | U: What's the subject? |
| 9. | S: The subject is let's move the meeting on September 12th to afternoon around 3 P.M. |
| 10. | U: What day is September 12th? |
| 11. | S: That is a Friday. |
| 12. | U: Find the meeting with Susan on that day. <system opens calendar, finds and highlights the referenced meeting> |
| 13. | U: Move the meeting to 3 o'clock <system moves the calendar entry to 3 P.M.> |
| 14. | U: Close this. <system closes the calendar entry> |
| 15. | U: Close this. <system minimizes the calendar> |
| 16. | U: I mean Jeffrey Houston. <user tries to finish the previous discourse> |
| 17. | S: The folder called proposal does not exist. Would you like me to create it? |
| 18. | U: Yes I would. |
| 19. | S: This may take a few seconds. <relevant messages are being selected and moved to the personal folder> |
| 20. | S: Transfer is done. |

understanding. The data collection process and setup that was used to collect domain-specific real-use data is also described.

One of the main design considerations that we followed in building the models was to ensure that these models can be shared across multiple applications and multiple access methods. Although such a constraint poses serious challenges to the model building process, it also allows the user to interact with the system seamlessly, without the user having to *explicitly* switch applications or access methods.

### 3.1. Data Collection

Using a limited amount of domain-specific data and some out-of-domain data, we built the initial models for the data collection system. The data collection system consists of all the components of the full system, and in addition also permits a human wizard (data-collector) to trap the decoded formal language statements. The human wizard examines the user's actual spoken input, the text string produced by the speech recognition engine, and the formal language statement selected by the NLU engine. The wizard makes any necessary corrections to the recognized text and the formal language statement and submits the command for automatic execution, or manually executes the command if it happens to be an unsupported command. Using this "Wizard-of-Oz" style data collection, we have collected approximately 12000 sentences relevant to the email and calendar domains.

### 3.2. Speech Recognition

The system vocabulary was limited to approximately 5000 words, including all the words found in the training data. However, *dynamic vocabulary* support was incorporated in to the system, and the details are described in Section 3.4. The acoustic models use the prototypes from IBM ViaVoice desktop and telephony products.

We built the initial language model using some limited domain-specific and some out-of-domain data, using the techniques described in [6]. This initial language model was used for the data collection process, and as more domain-specific data was collected, the dependency on out-of-domain data reduced. All iterations of the language model were built using classes, such as for names, months, days, numbers, years, and directions. The performance of these models will be described in Section 3.5.

### 3.3. Natural Language Understanding

The NLU engine consists of two sub-components: a *Tagger* and a *Translator*. The Tagger assigns words and phrases in the recognized text to appropriate classes. The Translator takes the tagged sentence and translates it to a formal language statement. For example, a sentence of the form "show me the first email from John Smith" may be tagged as

```
OPEN the COUNT MESSAGE from NAME
```

and translated into a formal language statement of the form

```
open_message(message=COUNT, sender=NAME)
```

where `COUNT` maps to 1, and `NAME` maps to John Smith. Separate statistical models were built for the Tagger and the Translator. The Tagger is based on parsing techniques similar to those described in [7], and the Translator is based on the *maximum entropy* approach, [1], [3]. A subset of the training data was first tagged, translated, and used to build the initial NLU models. These models were then used to process the remaining training

Table 2: **Examples of Sentences from Data Collection**

| | |
|---|---|
| 1. | Okay I think there is something clipped on to this and I want to open it |
| 2. | Can you give me the names of those people who attended the staff meeting last Tuesday |
| 3. | I was born December thirtieth 1965 and what'll be the day when I am fifty-nine and a half |
| 4. | Go to bank at two, deposit checks, pick up dog food and cat food at grocery store |
| 5. | Yes this is a lovely birth announcement sent to me by Kristen, I think I'm going to save it in my message log uh for later perusal |

data, followed by manual correction. Each new batch of training data usually requires introducing additional formal language statements to capture the new concepts discovered. As mentioned in Section 2, more than 700 formal language statements are currently supported, which is larger than most state-of-the-art conversational systems.

### 3.4. Other Features

In addition to the size and complexity of the statistical models, managing the vocabulary of the system is also a challenge because of the dynamic nature of the vocabulary involved in personal information management. For example, each user will have a different set of frequently used addressee names, subject line entries or folder names to begin with, and all of these can change dynamically simply by the arrival of a new message, and therefore it would be impossible to precompile all of the relevant vocabulary. Hence, we incorporated *dynamic vocabulary* support for both speech recognition and natural language understanding. For example, if a new mail arrives from a sender whose name is not currently in the system vocabulary, the baseforms for that name are automatically created and the word is automatically added to the vocabulary for speech recognition. Similarly, another preprocessing stage flags the word as potential name, thus allowing the NLU engine to tag the word correctly. As a result, the user will have instant conversational access to the new mail.

Another feature that we developed and incorporated into the system is the ability to automatically identify the end of a command. Most conversational systems usually require the user to indicate the command boundary through some form of manual input, such as pausing between commands or clicking a microphone control button on the display. Since such a requirement makes the system quite cumbersome to use, we developed a new solution for automatically identifying the command boundary, the details of which could be found in [5]. This solution makes the conversational interface much more user friendly, and allows the user to speak naturally and continuously in a hands-free manner.

### 3.5. Performance of the Models

The performance of the speech recognition and NLU models was evaluated using an independent test set. The perplexity of the language model computed using the test set was found to be 86, which is relatively high, reflecting the complexity of the task and suggesting the need for more training data. The speech recognition word error rate was 14.0%, the NLU error rate was 10.8%, and the combined speech recognition and NLU error rate was 22.5%. However, in actual use, many of the errors are corrected by the Dialog Manager, with or without additional user interaction, and therefore the effective transaction error rate would be lower. With more training data and additional tuning of the models, we expect the error rates to drop further.

As noted earlier, the requirement to share the same set of models across multiple applications and multiple access methods makes the modeling quite difficult. In addition, we discovered that the domain itself introduces additional difficulties, as illustrated by the breadth of the example sentences from our data collection, given in Table 2.

## 4. CONCLUSIONS

In this paper, we presented a new conversational system that provides access to multiple desktop applications, from multiple client devices, using multiple input modalities. We described the architecture of the system and highlighted the main system features. We also described the statistical models that were built for speech recognition and natural language understanding, and the performance of these models on an independent test set. As noted earlier, additional data collection and tuning of models is in progress, and the improvements to the transaction accuracy will be reported in the future.

## REFERENCES


[1] Della Pietra, S., Della Pietra, V., and Lafferty, J., "Inducing Features of Random Fields," Technical Report CMU-CS95-144, School of Computer Science, Carnegie-Mellon University, May 1995.

[2] Lamel, L., Rosset, S., Gauvain, J. L., and Bennacef, S., "The LIMSI ARISE System for Train Travel Information," *International Conference on Acoustics, Speech and Signal Processing*, Phoenix, Arizona, March 1999.

[3] Papineni, K., Roukos, S., and Ward, T., "Feature-Based Language Understanding," *Eurospeech*, Rhodes, Greece, September 1997.

[4] Papineni, K., Roukos, S., Ward T. "Free-Flow Dialog Management Using Forms" *Eurospeech*, Budapest, Hungary, September 1999.

[5] Ramaswamy, G. N., and Kleindienst, J., "Automatic Identification of Command Boundaries in a Conversational Natural Language User Interface," *International Conference on Spoken Language Processing*, Sydney, Australia, December 1998.

[6] Ramaswamy, G. N., Prinz, H., and Gopalakrishnan, P. S., "A Bootstrap Technique for Building Domain-Dependent Language Models," *International Conference on Spoken Language Processing*, Sydney, Australia, December 1998.

[7] Ward, T., Roukos, S., Neti, C., Gros, J., Epstein, M., and Dharanipragada, S., "Towards Speech Understanding Across Multiple Languages," *International Conference on Spoken Language Processing*, Sydney, Australia, December 1998.