Models for evaluating interaction protocols in speech recognition

Alexander I. Rudnicky and Alexander G. Hauptmann

School of Computer Science, Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

Recognition errors complicate the assessment of speech systems. This paper presents a new approach to modeling spoken language interaction protocols, based on finite Markov chains. An interaction protocol, prescribed by the interface design, defines a set of primitive transaction steps and the order of their execution. The efficiency of an interface depends on the interaction protocol as well as the cost of each different transaction step. Markov chains provide a simple and computationally efficient method for modeling errorful systems. They allow for detailed comparisons between different interaction protocols and between different modalities. The method is illustrated by application to example protocols.

KEYWORDS: Speech recognition, modeling errors, interface evaluation, user models.

INTRODUCTION

A distinguishing characteristic of spoken language interaction is the presence of noise in the transmission channel: recognition systems make errors, users are forced to compensate for these errors. User efficiency suffers from decreased task throughput, due to time spent on error correction. Any accurate characterization of a spoken language system must take this factor into account. This is particularly necessary in those cases where claims are to be made about the relative advantage of speech over conventional input modalities, such as the keyboard. For example, in an analysis of a spreadsheet task, Rudnicky et al [9] found that component actions such as movement and formula entry could be executed more rapidly by voice than by keyboard. Nevertheless, users took longer to complete the tasks by voice than by keyboard. Analysis of the sessions found that two factors, system response time and recognition errors accounted for the discrepancy. On the basis of this analysis, it was possible to specify, though just for that system, the level of recognizer performance that would equate voice and keyboard throughput.

The purpose of this paper is to describe a technique that allows such factors as recognition error and response delay to be accurately modeled and to provide a basis for quantitative comparison between different input modalities and between different interaction protocols. The technique is general and can be applied to other domains, such as OCR [3], where errors and delays constitute a significant factor in performance.

Fundamentally, we are interested in assessing the efficiency of computer interfaces. In modeling a computer interface, there are two components to consider, the *cost* of a particular transaction and the *protocol* used to enter information into the computer. Since throughput is our measure of efficiency, cost is expressed in terms of time. The interaction protocol defines the types of transactions that need to be carried out and their structural relationships.

In our discussion, we will use *single keystroke* time as the basic unit of keyboard input; for voice we will use the *single word* time. The specific values for these parameters were taken from experiments on entering digit strings into the computer [4]. Although detailed techniques have been proposed for modeling the time component in human–computer interaction [2], we will restrict ourselves to very simple and direct measures of the time cost component of the individual transaction steps. Note, however, that the techniques described below are easily extended to more elaborate models of both time cost and interaction protocols in human–computer interaction.

A BASIC PROTOCOL AND COST FUNCTION

Figure 1 shows an idealized protocol for completing a simple transaction, entering one item into the computer. **I** is the input event and **f** is the terminal state, indicating successful completion of the transaction. In a simple data entry task, where the user enters a sequence of items, transactions are more or less homogeneous; this may not be the case for more complex tasks. For now, we will only consider such simple transactions, though the approach we describe is easily extended to more complex cases. We can calculate cost for a simple digit string entry transaction by using the following expression:

$$e + dl \qquad (1)$$

where e is an overhead, d is the input time for a single unit and l is the length of an input string. Thus for Figure 1 the cost of transaction **I** is given by expression (1).

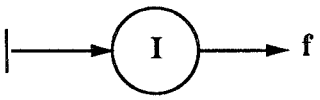

Figure 1: *Basic interaction protocol.* **I** *represents a transaction,* **f** *its successful completion.*

All times used in this paper are in seconds. For keyboard, we will use an e of 0.195, which is the time needed to depress the "enter" key on the keyboard. For voice, e is 0.450, which corresponds to the time needed by the endpoint detection algorithm in our system to operate. The item time, d is 0.310 for keyboard and 0.330 for voice. Note as well that the above values have been derived from a specific experiment [4] and should be considered approximations. Other paradigms and physical devices may produce different values for these parameters.

CONFIRMATION PROTOCOLS

The idealized protocol described in the previous section does not provide a complete description of interaction, since in reality occasional input errors occur, forcing the user to repeat or edit an input until it is satisfactory. A more realistic representation would be the one shown in Figure 2.

This figure shows a protocol in which the system accurately receives an item with only u probability. If an input error occurs, the input is repeated until it is correct, at which point the user proceeds to the next transaction. This protocol is still incomplete as it does not take into account that the system needs

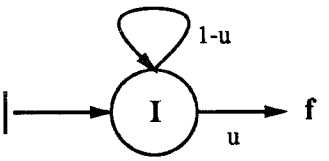

Figure 2: *Repeat-until-correct protocol for a system with a noisy input channel.* u *is the probability of successfully completing transaction* **I**.

to be told whether the current input is a repetition of the current entry or a new entry. A more practical and complete protocol would include a confirmation stage, itself susceptible to error, as modeled in Figure 3. The confirmation step includes checking the input as well as explicitly indicating to the system that the input is correct. For some types of systems, there will be probability c that the system will not accept a confirmation, forcing the user to restart the input process. For keyboard input, the values of u and c will be quite low and will be largely related to the user's conscientiousness about typographical errors. In the digit string entry experiment referenced above, the keyboard values of u and c are $> 98\%$. In the case of voice input however, recognition accuracy can have a significant impact. The lower the accuracy of the speech recognition system, the greater the amount of time spent in the repetition cycle. For the system described in [4], mean recognition accuracy for words was 92%.

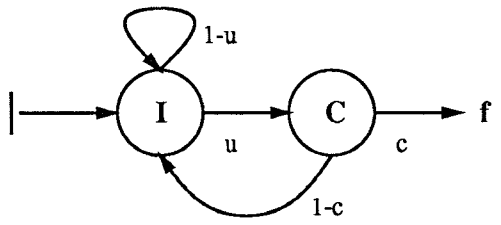

Figure 3: *Repeat-until-correct protocol, with added confirmation step (*C*), completed with probability* c *of success.*

Recognition errors and the resulting repeated attempts to complete a transaction complicate the goal of calculating the expected throughput of such a system. Calculating the expected time consumed by a transaction for a given level of recognition accuracy is simplified by the observation that the protocols shown in Figures 2 and 3 are Markov chains with a single absorbing state (**f**). Since straightforward analytic solutions exist for calculating expected state residence for such systems, we can calculate the ex-

pected transaction cost for a system for any given operating characteristics. Using standard notation, we can express a system such as Figure 3 in terms of a transition matrix:

$$\begin{array}{c|ccc} & f & I & C \\ \hline f & 1 & 0 & 0 \\ I & 0 & 1-u & u \\ C & c & 1-c & 0 \end{array} \quad (2)$$

The rows correspond (from top to bottom) to f, the final (absorbing) state, I, the input state; and C, the confirmation state. Rows are origin states and columns are destination states. The individual entries in the matrix correspond to the arcs shown in Figure 3. To estimate the total time needed for a transaction, we need to know the amount of time that the system can be expected to reside in states I and C, before being absorbed into state f. This can be determined by considering the sub-matrix consisting of just those states,

$$Q = \begin{pmatrix} 1-u & u \\ 1-c & 0 \end{pmatrix} \quad (3)$$

and performing the following transformation [5]:

$$N = (I - Q)^{-1} \quad (4)$$

For matrix (2), this operation gives us the following result:

$$N = \begin{pmatrix} \frac{1}{cu} & \frac{1}{c} \\ \frac{1-c}{cu} & \frac{1}{c} \end{pmatrix} \quad (5)$$

The cell values in this matrix correspond to the expected number of times that state will be visited, given that the system is entered at the state corresponding to a given row. Since we can meaningfully enter the system only through I, the system will reside $(1/cu)$ times in state I and $(1/c)$ times in state C. To calculate total transaction time, we need only multiply this matrix by a vector, t, containing the expected residence time in each state. For example, using the cost formula (1), we define the following cost vector for keyboard input, assuming that confirmation consists of a single keystroke (e.g., "y" or "n") followed by a carriage return:

$$t = \begin{pmatrix} e + dl \\ e + d \end{pmatrix} \quad (6)$$

If we calculate Nt, the element in the row corresponding to I gives us the formula for total cost. We are now in a position to systematically investigate the effects of different system parameters.

Figure 4 shows an application of this procedure to the calculation of a cost-per-digit surface for a family of speech recognition systems with different word

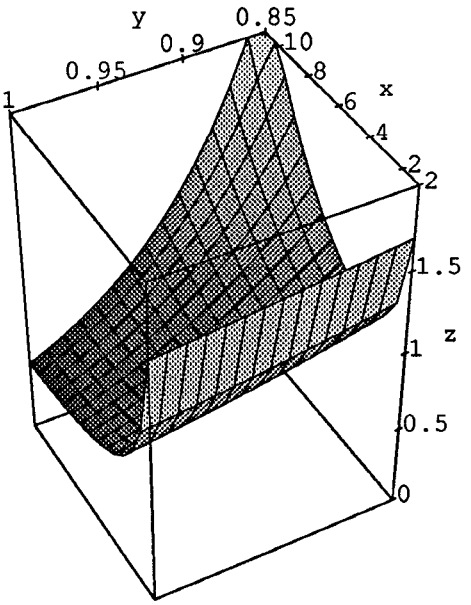

Figure 4: *Surface showing time for entering a single digit. Time is on the z-axis and ranges from 0 to 2 sec. Recognition accuracy, on the y-axis, ranges from 0.85 to 1.0. String length, on the x-axis, ranges from 1 to 11.*

accuracies. The surface is plotted for inputs of different lengths, thus the descriptors of the system are as follows:

$$Q = \begin{pmatrix} 1-w^l & w^l \\ 1-c & 0 \end{pmatrix}, t = \begin{pmatrix} 0.450 + 0.330l \\ 0.450 + 0.500 \end{pmatrix} \quad (7)$$

where w is word accuracy and w^l is therefore utterance accuracy. The cost vector assumes that it takes half a second to utter a confirmation word. For simplicity, we will also assume that $c = 1.0$, that the confirmation word is always correctly understood. In the graph, word accuracy ranges from 0.85 to 1.00 and string length ranges from 1 to 11, as in the experiments described in [4]. The vertical axis gives the expected time-per-digit for entry. The surface shown is calculated using the value of Nt, divided by the length of a string.

Not surprisingly, the total transaction time is an accelerated function of recognition error, the lower the system accuracy, the longer, proportionately, it takes to successfully enter a digit into the computer.

Modeling interaction protocols as Markov chains has a number of advantages, including ease of calculation and the relatively straightforward mapping of an interaction protocol into a concise closed-form expression for transaction time. In this sense, the current

technique is preferable to analytic solutions, e.g., as used by Ainsworth [1].

The surface in Figure 4 provides us with the cost of data entry for a noisy channel. By itself it is only of limited interest, although it can suggest "optimal" operating ranges for a recognizer, as suggested by [1]. Of greater interest is the opportunity it affords for comparing the operating characteristics of different systems. In the following two sections we will show how this modeling can be used to compare different input modalities and to compare different interaction protocols.

COMPARING VOICE AND KEYBOARD

To demonstrate the applicability of this technique to intermodal comparisons, we will consider some data from a study of digit string entry[4]. Since the protocols are the same, this comparison will focus on the consequences of different cost functions.

The particular data of interest to us are the times taken to enter digit strings of different lengths into a computer, using either voice or keyboard. The actual data are shown in Figure 5. The entry time for voice can be modeled accurately by a single cost vector, such as t in equation (7). The model for keyboard entry is more complex as the curve is nonlinear. Hauptmann and Rudnicky [4] suggest that the acceleration observed for the keyboard function is due to the need for gaze shifting between presentation medium and keyboard as well as in-line error correction. We will model the keyboard cost function in terms of a quadratic equation, which gives a satisfactory fit to the data, as shown in Figure 6. Note that our use of an arbitrary function in this case illustrates the flexibility of the procedure, allowing us to combine system descriptors from both generated and empirical sources. Using the protocol shown in Figure 3 and matrix (5), we can generate a surface for keyboard input similar to the one shown in Figure 4.

To compare voice and keyboard input, we can subtract the keyboard surface from the voice surface. Values on the resulting surface that lie below 0 indicate an advantage for voice, values above 0 indicate an advantage for keyboard. This difference surface is shown in Figure 7. For clarity, the region of the surface below 0 is rendered in white.

The difference surface has a practical use. Inspection of the surface suggests that for the class of tasks under consideration, that is, for digit entry for strings of lengths 1 to 11 and using the confirmation protocol shown in Figure 3, voice should not be considered at all if string lengths will be 6 digits or shorter, nor in any case should we consider recognition systems with word accuracies of less than about 93%.

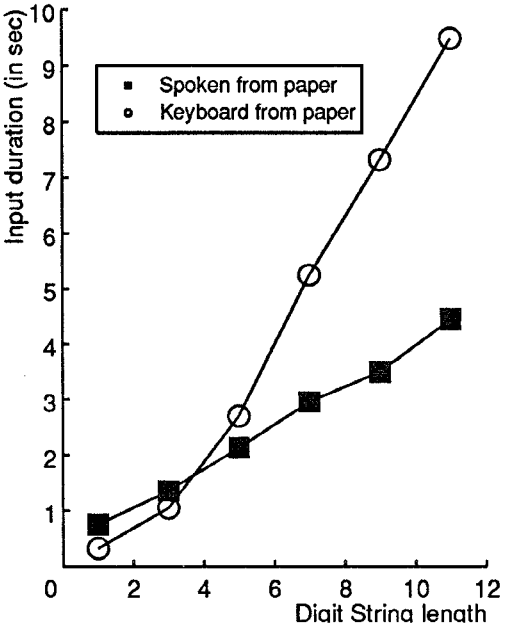

Figure 5: *Time needed to enter a digit string, by voice and by keyboard, from [4].*

Remember that we are considering a simple digit entry task, which actually puts speech at a disadvantage, since each word in this task corresponds to a single keystroke. A much larger region of usability would be found for cases in which input units require multiple keystrokes. For example, words in text require, on average, 6 keystrokes to enter.

Despite the simplifications of the example we chose, we believe that this technique provides a clear, quantitative tool for comparisons of different input modalities for specific tasks.

COMPARING DIFFERENT INTERACTION PROTOCOLS

The comparison between voice and keyboard focused on differences in cost functions. We can also use the same approach to compare different protocol structures [7] in order to chose the ones best suited to specific applications.

Consider the protocol in Figure 3. If we are so fortunate as to have a recognition system that functions

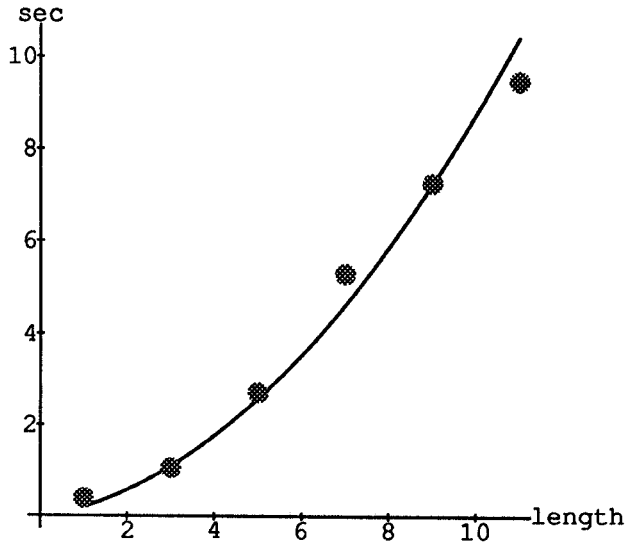

Figure 6: *Quadratic fit to the keyboard input data in Figure 5. The abscissa is string length, the ordinate is time, in sec.*

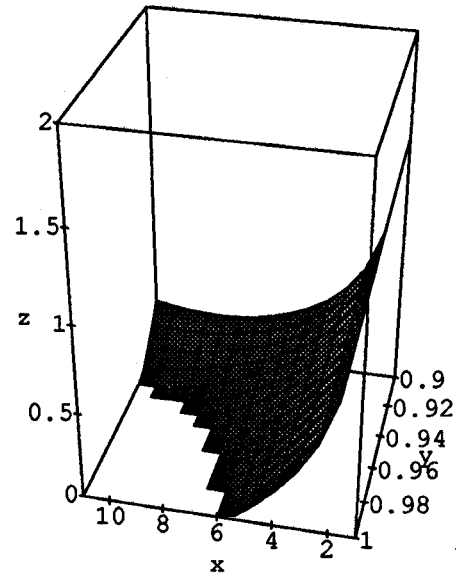

Figure 7: *Difference surface for keyboard and voice input. Word accuracy (the y axis) ranges from 0.90 to 1.0; the x axis is the digit string length ranging from 1 to 11 and the z axis is the time differential between the two modalities for a complete string entry and ranges from 0 to 2 seconds. The surface has been rotated for easier viewing.*

perfectly, then this protocol is more complex than it needs to be; the confirmation step is not necessary since every input is correct (if, of course, we ignore that users might make errors of their own). A much better protocol for high-accuracy systems might be the one shown in Figure 8, where instead of confirming the correctness of every entry, the user indicates only those utterances that have been misrecognized. The protocol in Figure 8 is clearly more efficient for low-error systems, while the protocol in Figure 3 is clearly more efficient for high-error systems. The comparison technique outlined in the previous section allows us to determine under which exact circumstances one protocol is preferable to another.

Using the protocols in Figures 3 and 8, and the cost function from (7) we can generate a difference surface, shown in Figure 9. To obtain this figure, we subtract the negation surface from the confirmation surface. This means that region below 0 sec, rendered in white, represents an advantage for confirmation. As our intuition suggests, a confirmation protocol is a poor choice, particularly for short utterances. In terms of a concrete example, the system used by [4], with word accuracy of 92%, we can assert that a confirmation protocol (Figure 3) is more efficient for tasks involving string lengths greater than 7 digits. The explicit negation protocol (Figure 8) is more efficient for strings shorter than 7.

Although the above example compares two different speech protocols, there is no reason why the comparison could not have also been made between different protocols in different modalities. Any combination is possible, inasfar as the system descriptors correspond to the same user task.

In our discussion we have assumed that system parameters remain constant over the course of interaction, though this may not be true in practice. For example, it may be the case that the repetition of a misrecognized utterance will have characteristics dif-

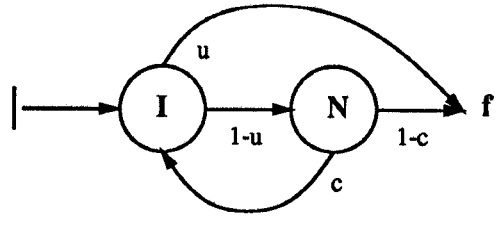

Figure 8: *Explicit negation protocol.* **N** *is the negation transaction, the other symbols are as in previous figures.*

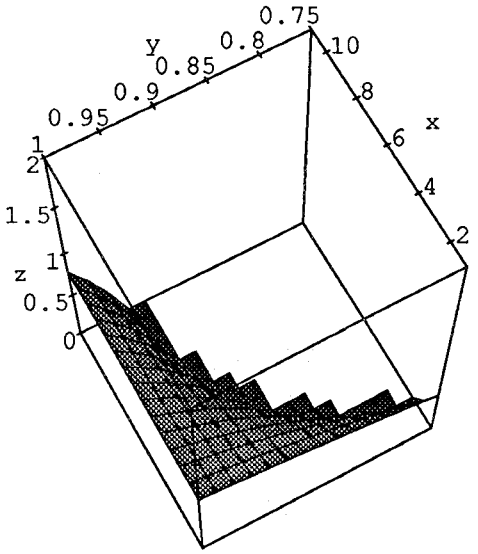

Figure 9: *Difference surface from protocol comparison. The y axis shows word recognition accuracy ranging from 0.75 to 1.0 (note that the the range is greater that the one show in Figure 7). The x axis shows the digit string length and the z axis shows the time differential (in sec) between the two protocols. The surface has been rotated for easier viewing.*

ferent from the original: it may be longer in duration or have an inherently different probability of being recognized correctly. There is empirical evidence that this is the case [8] and a more precise model of interaction would explicitly model such phenomena. Within the current approach this can be done by modeling protocols in terms of higher-order Markov chains (to capture shifts in recognition accuracy) and by introducing additional states (to capture changes in the cost vector). Figure 10 shows a modification of the protocol in Figure 3 that allows the characteristics of a repetition to be modeled separately from those of the original utterance.

CONCLUSION

We have explored the problem of modeling systems, such as speech recognizers, that are characterized by input channel errors. We have shown that suitable modeling allows us to capture the behavior of such systems and gives us a basis for comparing different systems. For example, the choice of a particular interaction protocol can now be informed by some understanding of how the resulting system will perform, taking into account differences in accuracy.

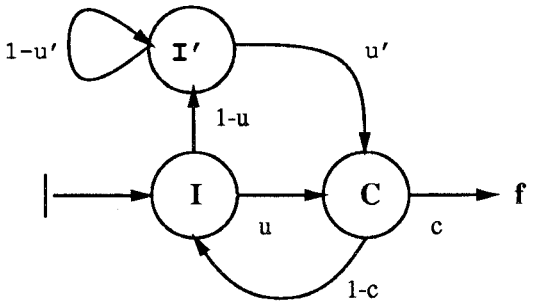

Figure 10: *A protocol that models repetitions separately from the initial input. Adding a repetition state* I′ *allows its cost (as well as any stationary change in recognition accuracy,* u′*) to be modeled more precisely.*

The modeling approach described also provides a solid basis for comparisons of different input modalities, identifying those operating regions where we can expect a given modality to provide higher throughput.

The distinctive property of speech systems, errorful transduction, led us to model their behavior in terms of clearly separable components, a *protocol* expressed in terms of a Markov chain, and a *cost* vector that specifies the (time) cost for individual operations. While we developed our approach in response to the specific problems encountered in speech recognition, the technique we have described can be applied to other domains of interaction modeling and can provide more accurate accounts of time taken to perform computer tasks.

We have described only the basic technique and should note that it has a number of shortcomings. In particular, we have only explored the modeling of externally observable events and have not attempted to model the cognitive components of interaction. Such an extension would, for example, allow protocols of differing cognitive complexity to be compared. Kieras and Polson [6] have described a procedure for specifying computer tasks in graph form which is compatible with the current approach.

The use of Markov chains has the potential to improve existing models of cognitive activity. For example, Card et al [2, pages 146–147] acknowledge the presence of user errors in their data but consider the consequent increase in variance as acceptable. More exact fits to their experimental data could be obtained, however, by adding model parameters that

reflect error probabilities for individual actions. The estimation of these parameters would be useful in itself, as it could be used to identify the location of problematic steps in task procedures.

ACKNOWLEDGMENTS

We would like to thank Jennifer Quirin for her help in collecting experimental data and Dana Scott and Marko Petkovsek for leading us to consider Markov chains. Model development and the generation of surface graphics was greatly simplified by the use of Mathematica. We also thank the reviewers for their helpful comments.

The research described in this report was sponsored by the Defense Advanced Research Projects Agency (DOD), Arpa Order No. 5167, monitored by SPAWAR under contract N00039-85-C-0163. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

References

[1] AINSWORTH, W. Optimization of string length for spoken digit input with error correction. *International Journal of Man-Machine Studies 28* (1988), 573–581.

[2] CARD, S. K., MORAN, T. P., AND NEWELL, A. *The psychology of human-computer interaction.* Erlbaum, Hillsdale, N.J., 1983.

[3] CUSHMAN, W. H., OJHA, P. S., AND DANIELS, C. M. Usable OCR: What are the minimum performance requirements? In *Proceedings of CHI* (Seattle, April 1–5 1990), ACM, New York, 1990, pp. 145–151.

[4] HAUPTMANN, A. H., AND RUDNICKY, A. I. A comparison of speech versus typed input. In *Proceedings of the Third Darpa Speech and Natural Language Workshop* (June 1990), Morgan Kaufmann, San Mateo, CA, 1990, p. in press.

[5] ISAACSON, D. L., AND MADSEN, R. W. *Markov chains, theory and applications.* Wiley, New York, 1976.

[6] KIERAS, D., AND POLSON, P. G. An approach to the formal analysis of user complexity. *International journal of Man-Machine Studies 22* (1985), 365–394.

[7] RUDNICKY, A., AND HAUPTMANN, A. Conversational interaction with speech systems. Tech. Rep. CMU-CS-89-203, Carnegie Mellon University School of Computer Science, December 1989.

[8] RUDNICKY, A. I., AND HAUPTMANN, A. G. Errors, repetition, and contrastive stress emphasis in speech recognition. Working notes from the AAAI Spoken Language Symposium, March 1989.

[9] RUDNICKY, A. I., SAKAMOTO, M. H., AND POLIFRONI, J. H. Evaluation of spoken language interaction. In *Proceedings of the Second Darpa Speech and Natural Language Workshop* (October 1989), Morgan Kaufmann, San Mateo, CA, 1989, pp. 150–159.