An Evaluation of Strategies for Selective Utterance Verification for Spoken Natural Language Dialog

Ronnie W. Smith
Department of Mathematics
Computer Science Subarea
East Carolina University
Greenville, NC 27858, USA
rws@cs.ecu.edu

Abstract

As with human-human interaction, spoken human-computer dialog will contain situations where there is miscommunication. In experimental trials consisting of eight different users, 141 problem-solving dialogs, and 2840 user utterances, the Circuit Fix-It Shop natural language dialog system misinterpreted 18.5% of user utterances. These miscommunications created various problems for the dialog interaction, ranging from repetitive dialog to experimenter intervention to occasional failure of the dialog. One natural strategy for reducing the impact of miscommunication is selective verification of the user's utterances. This paper reports on both context-independent and context-dependent strategies for utterance verification that show that the use of dialog context is crucial for intelligent selection of which utterances to verify.

1 Building Robust Spoken Natural Language Interfaces

Recent advances in speech recognition technology have raised expectations about the development of practical spoken natural language (NL) interfaces. Such interfaces can provide user flexibility as well as allow users to devote their hands and eyes to the task at hand. In particular, the ability to obtain computerized telephone assistance via a robust NL interface could provide ready access to information that currently requires direct human interaction. However, if such interfaces are to be effective with the general populous, they must be capable of dealing with miscommunication. Miscommunication can arise at several different levels, ranging from discourse structure and speech act misunderstanding (McRoy and Hirst, 1995) to misinterpretation due to misrecognition of a speaker's words. We report on a study that focuses on this latter type of miscommunication. While speech recognizer performance in controlled environments has improved dramatically in the past decade, recognition errors still occur. Furthermore, current speech recognizers cannot perform optimally in uncontrolled environments such as telephone interactions.

We examine the strategy of *verification subdialogs* for resolving miscommunications due to misrecognition. We first review how verification subdialogs can increase the rate of correct interpretation from 81.5% to 97.4% but at a cost of unnecessary verifications approximately once every five user utterances. However, by adopting a context-dependent strategy for deciding when to use a verification subdialog, we can maintain an understanding rate of 95.3% while reducing the number of unnecessary verifications by over one half.

After describing the technique of selective utterance verification, this paper gives an overview of the dialog system environment that provides the data used in testing various strategies for selective utterance verification, the Circuit Fix-It Shop. The paper concludes with a description of both context-independent and context-dependent strategies for selective utterance verification and reports on the comparative results of dialog simulations using these strategies. The results show the importance of exploiting dialog context for intelligent selection of which utterances to verify.

2 Selective Verification of Questionable User Inputs

Every system that uses natural language understanding will sometimes misunderstand its input. Misunderstandings can arise from speech recognition errors or inadequacies in the language grammar, or they may result from an input that is ungrammati-

```
Spoken:     i want to fix this circuit
Recognized: power a six a circuit

Spoken:     the one is flashing for a longer period of time
Recognized: one is flashing forth longer in a time

Spoken:     there is no wire on connector one zero four
Recognized: stays know wire i connector one zero for
```

Figure 1: Sample Utterances with Word Misrecognition

cal or ambiguous. Here we focus on misunderstandings caused by speech recognition errors. Examples of misrecognized inputs from interacting with the Circuit Fix-It Shop are given in figure 1. One method for reducing the number of misunderstandings is to require the user to verify each utterance by either speaking every utterance twice, or confirming a word-by-word read back of every utterance (e.g., (Baber and Hone, 1993)). Such verification is effective at reducing errors that result from word misrecognitions, but does nothing to reduce misunderstandings that result from other causes. Furthermore, verification of all utterances can be needlessly wearisome to the user, especially if the system is working well.

A better approach is to have the system verify its interpretation of an input only under circumstances where the accuracy of its interpretation is seriously in doubt, or correct understanding is essential to the success of the dialog. The verification is accomplished through the use of a *verification subdialog*—a short sequence of utterances intended to confirm or reject the hypothesized interpretation. The following example of a verification subdialog illustrates the idea.

```
Computer: What is the switch position when
            the LED is off?
User:     Up.
Computer: Did you mean to say that the
            switch is up?
User:     Yes.
```

Notable features of such verification subdialogs include the following.

- Verification is *selective*. A verification subdialog is initiated only if it is believed that the overall performance and accuracy of the dialog system will be improved. In this way, the dialog system responds much as a person would.
- Verification is *tunable*. The propensity of the system to verify can be adjusted so as to provide any required level of speech understanding accuracy.
- Verification *operates at the semantic level*. The system verifies an utterance's meaning, not its syntax. This helps overcome misunderstandings that result from inadequacies in the language model, or ungrammatical or ambiguous inputs.

Two important definitions concerning selective verification are the following. An *under-verification* is defined as the event where the system generates a meaning that is incorrect but not verified. An *over-verification* occurs when a correct meaning is verified. The example just given is an example of an over-verification. The goal of any algorithm for selective utterance verification is to minimize the rate of under-verifications while also holding the rate of over-verifications to as low a value as possible. That is, the goal is to only verify utterances that need verifying, and to verify as many of these as possible. In section 4 we report on the results of tests of various strategies for deciding when to engage in verification subdialogs within a specific dialog environment, the Circuit Fix-It Shop. In order to understand the strategies used, an overview of this environment must first be presented.

3 Dialog Environment: The Circuit Fix-It Shop

3.1 General Characteristics

The data used in this study were collected in experimental trials conducted with "The Circuit Fix-It Shop," a spoken NL dialog system constructed in order to test the effectiveness of an integrated dialog processing model that permits variable initiative behavior as described in (Smith and Hipp, 1994) and (Smith, Hipp, and Biermann, 1995). The implemented dialog system assists users in repairing a Radio Shack 160 in One Electronic Project Kit. The particular circuit being used causes the Light-Emitting Diode (LED) to alternately display a one and seven. The system can detect errors caused by missing wires as well as a dead battery. Speech recognition is performed by a Verbex 6000 running on an IBM PC. To improve speech recognition performance we restrict the vocabulary to 125 words. A DECtalk DTCO1 text-to-speech converter is used to provide spoken output by the computer.

After testing system prototypes with a few volunteers, eight subjects used the system during the formal experimental phase. After a warmup session where the subject trained on the speech recognizer and practiced using the system, each subject participated in two sessions where up to ten problems were attempted. Subjects attempted a total of 141 dialogs of which 118 or 84% were completed successfully.[1] The average speech rate by subjects was 2.9 sentences per minute; the average task completion times for successful dialogs were 6.5 minutes.

An excerpt from an actual interaction with the system is given in figure 2.[2] The words in parentheses represent the actual sequence of words that the speech recognizer sent to the dialog system for analysis. As can be seen from the example, the system usually understood the user utterance (but not always). We next describe two features of the system that were useful in the interpretation process: (1) error-correcting parsing; and (2) dialog expectation. In section 4 we will see how these features assist in deciding when to engage the user in a verification subdialog.

3.2 Overcoming Misrecognition by Error-Correcting Parsing

The system was able to find the correct meaning for 81.5% of the more than 2800 input utterances even though only 50% of these inputs were correctly recognized word for word by use of an error-correcting parser that uses a dynamic programming approach similar to (Ney, 1991) to compute the best n parses for the input. What constitutes "best" is determined by a cost matrix for the possible words in the vocabulary and the given grammar. The cost matrix defines the cost for inserting or deleting words as well as the cost for a word substitution when such substitutions are allowed. The intent is to permit substitutions for words which sound very similar, such as "do" and "two/to/too," words that are likely to be confused by the speech recognizer. The parser performs insertions, deletions, and substitutions in order to transform the input into a grammatical utterance. With each "grammatical" utterance is associated a *parse cost* (PC), which is the sum of the costs of each insertion, deletion, and substitution required for the transformation. For each of the best n parses, an *expectation cost* (EC) is also produced according to how likely the input is to occur according to the expectations. The *total cost* of a parse is a weighted sum of PC and EC. The values for these weights and costs have been assigned heuristically. Figure 3 shows some of the utterances successfully handled by the implemented system during the experiment. For instance, example 1 is transformed into a grammatical utterance by substituting the phonetically similar word "six" for "fix," and "and" for "can." This example would have a relatively low parse cost indicating the system has high confidence in its interpretation. On the other hand, example 3 requires a large number of costly insertions and deletions, indicating a lack of confidence in the quality of the interpretation. As will be seen in section 4, this difference in scores is an important component in deciding when to engage in verification subdialogs.

3.3 Expectation in Dialog

An important phenomenon that is exploited in computational models of language is that of *expectation* (Allen, 1995). Based on the current context, there is an expectation of what is to come next. These expectations can be for phonemes, morphemes, words, or meanings. For example, if we hear the word "the" we expect the next word to be either an adjective or noun. When the computer asks, "Is the knob position at 10?", we have greatest expectation for a response of either "yes" or "no", lesser expectation for a sentence answer such as "The knob position is five.", and still less expectation for a clarification question or comment such as "Which knob?", "Where is the knob?", or "I do not see it."

Consider the following dialog sample.

```
1. C: Is there a wire between connector
        84 and connector 99?
2. U: It is not there.
3. C: Add a wire between connector 84
        and connector 99.
4. U: I need help.
5. C: Obtain a wire.
6. U: Okay.
```

The system must use the established dialog context in order to properly interpret every user utterance as follows.

[1] Of the 23 dialogs which were not completed, misunderstandings due to misrecognition were the cause in 13 of these failures. There were a variety of causes for the failure in the other 10 dialogs, ranging from inadequate grammar coverage to subject error in connecting wires.

[2] `C` denotes utterances spoken by the computer while `U` denotes utterances spoken by the user.

```
 1.  C: This is the Circuit Fix-It Shop.  How may I help you?
 2.  U: I want to fix this circuit. (power a six a circuit)
 3.  C: I do not understand.  Please repeat what you said.
 4.  U: I want to fix this circuit (i want a fix a circuit)
 5.  C: What is the ID of the circuit?
 6.  U: RS111. (rs111)
 7.  C: I am familiar with that circuit.  The LED is supposed to be displaying
          alternately flashing one and seven.
 8.  U: The LED is displaying nothing.  (be led a displaying nothing)
 9.  C: What is the switch at when the LED is off?
10.  U: The switch is down. (the switch yes down)
11.  C: Put the switch up.
12.  U: Done. (done)
13.  C: Put the knob to one zero.
14.  U: The knob is at one zero. (the knob and add one zero)
15.  C: What is the LED displaying?
16.  U: The LED is displaying a one and seven at the same time
        (be led a displaying one an seven at the same time)
17.  C: Put the knob to zero.
18.  U: Done. (done)
19.  C: What is the LED displaying?
20.  U: The same thing.  (the same same)
21.  C: Did you mean to say the LED is displaying the same thing?
22.  U: Yes. (yes)
```

Figure 2: Sample Dialog Interaction with the Circuit Fix-It Shop

- Utterance 2: to determine what "it" refers to (i.e., the wire from 84 to 99).
- Utterance 4: to determine what the user needs help with (i.e., adding the wire).
- Utterance 6: to determine whether "okay" denotes confirmation or comprehension (i.e., confirmation that the wire has been obtained).

Effective use of expectation is necessary for constraining the search for interpretations and achieving efficient processing of NL inputs. This is particularly crucial in spoken NL dialog, where speakers expect fast response times (Oviatt and Cohen, 1989).

The system model of expectations is similar to that of (Young et al., 1989) in that we predict the meanings of possible user responses based on the current dialog goal. The details of the system model can be found in (Smith and Hipp, 1994). Here we review the key aspects that are exploited in a context-dependent strategy for verification. We define expectations based on an abstract representation of the current task goal. For example,

$$goal(user, ach(prop(Obj, PropName, PropValue)))^3$$

denotes the goal that the user achieve the value ($PropValue$) for a particular property ($PropName$), of an object (Obj). The specific values for Obj, $PropName$, and $PropValue$ are filled in according to the current goal. For example, the goal of setting the switch position to up may be represented as

$$goal(user, ach(prop(switch, position, up)))$$

while the goal of observing the knob's color would be

$$goal(user, obs(prop(knob, color, PropValue)))$$

where $PropValue$ is an uninstantiated variable whose value should be specified in the user input. General expectations for the meaning of user responses to a goal of the form $goal(user, ach(prop(\ldots)))$ include the following:

- A question about the location of Obj.
- A question about how to do the action.
- An assertion that Obj now has the value $PropValue$ for property $PropName$.

[3] This notation is an abbreviated form of the actual representation used in the system as described in (Smith and Hipp, 1994).

```
Example 1

Computer:   There is supposed to be a wire between connector 68 and connector 87.
User:       Wire connecting six eight and eight seven.
Recognized: Wire connecting fix eight can eight seven.

Example 2

Computer:   Putting the knob to one zero is desirable.
User:       The knob is at one zero.
Recognized: Seven knob use that one zero.

Example 3

Computer:   Is anything else on the LED on?
User:       LED is displaying a not flashing seven.
Recognized: Be down it be yes displaying be knob flashing seven then.
```

Figure 3: Sample Misrecognitions Correctly Parsed

- An acknowledgment that the action has been completed.

Even when using error-correcting parsing and dialog expectations, the Circuit Fix-It Shop misunderstood 18.5% of user utterances during the experimental testing. We now turn our attention to an empirical study of strategies for selective utterance verification that attempt to select for verification as many of the misunderstood utterances as possible while minimizing the selection of utterances that were understood correctly. These strategies make use of information obtainable from dialog expectation and the error-correcting parsing process.

4 Evaluating Verification Strategies

4.1 Strategy 1: Using Parse Cost Only

An enhancement to the Circuit Fix-It Shop described in (Smith and Hipp, 1994) allows for a verification subdialog only when the hypothesized meaning is in doubt or when accuracy is critical for the success of the dialog. The decision of whether or not a particular input should be verified is made by computing for each meaning a *parser confidence score* (a measure of how plausible the parser's output is—this measure is proportional to the inverse of the total cost (section 3.2) normalized for utterance length) and a *verification threshold* (a measure of how important the meaning is toward the success of the dialog—greater importance is denoted by a higher threshold). The decision rule for deciding when to initiate a verification subdialog is specified as follows:

```
IF the Parser Confidence Score > the
   Verification Threshold THEN DO NOT
   engage in a verification subdialog
ELSE
   engage in a verification subdialog
```

This basic capability for verification subdialogs was not available during the 141 dialog experiment. However, simulations run on the collected data raised the percentage of utterances that are correctly understood from 81.5% to 97.4%.[4] Unfortunately, besides improving understanding through verification of utterances initially misinterpreted, the system also verified 19.2% of the utterances initially interpreted correctly. An example would be asking, "Did you mean to say the switch is up?", when that is what the user originally said. These over-verifications result in extraneous dialog, and if excessive, will limit usability.

4.2 Strategy 2: Using Context Only

The previous decision rule for utterance verification focused exclusively on the local information about parsing cost and ignores dialog context. In that situation the over-verification rate was 19.2% while the

[4] Consequently, the under-verification rate is 2.6%. We say that an utterance is *correctly understood* if it is either correctly interpreted initially, or is an utterance for which the system will engage the user in a verification subdialog. It is of course possible that the verification subdialog may not succeed, but we have not yet assessed the likelihood of that and thus do not consider this possibility during the evaluation of the various strategies.

- $obs(prop(Obj, PropName, PropValue))$ ($PropValue$ unspecified)—observing a property.
  Example: a wh-question (e.g., "What is the switch position?")
  Main Expectation: direct answer (e.g., "The switch is up.").

- $obs(prop(Obj, PropName, PropValue))$ ($PropValue$ specified)
  Example: a yes-no question (e.g., "Is the switch up?")
  Main Expectation: (1) yes/no response and (2) a direct answer as in the above case.

- $obs(meas(Des, Val))$—observing a measurement described by Des where Val is the value.
  Example: a wh-question (e.g., "What is the voltage between connectors 121 and 34?")
  Main Expectation: direct answer (e.g., "Seven" or "The voltage is seven").

- $obs(behav(Obs, Cond))$—observing a behavior where the result of the observation (Obs), depends on the underlying physical conditions present ($Cond$) when the observation was made.
  Example: a wh-question (e.g., "What is the LED displaying when the switch is up?")
  Main Expectation: a direct answer (e.g., "The LED is displaying only a not flashing seven.").

- $ach(prop(Obj, PropName, PropValue))$—achieving a property.
  Example: a command (e.g., "Put the switch up.")
  Main Expectation: (1) completion acknowledgement (e.g.,"Okay" or "Done") and (2) assertion that the desired property now exists (e.g., "The switch is up.").

- $learn(Fact)$—learning a fact. The fact could concern a piece of state information (e.g., that the switch is located in the lower left portion of the circuit), that an action needs completing (e.g., "Putting the switch up is desirable,"), or that a certain property should or should not be true (e.g., there should be a wire between connectors 34 and 80). In all cases, one main expectation is an acknowledgment that the $Fact$ is understood. In the case of an action completion or a property status, there is also a main expectation for either that the user completed the action (e.g., "Done" or "The switch is up"), or that the property status is verified (e.g., "Wire connecting 34 and 80").

Figure 4: Summary of Main Expectations for Major Goals

under-verification rate was 2.6%. What about using a rule solely based on context? For each abstract task goal, we define a subset of the expectations as the *main expectation*. This subset consists of the expected meanings that denote a normal continuation of the task. Figure 4 lists these expectations for the major task goals of the model. For cooperative task-assistance dialog, making the assumption that the meaning of the user's utterance will belong to a very small subset of the expectations for each abstract goal allows us to define the following context-dependent decision rule for utterance verification.

```
IF utterance in the Main Expectation THEN
     DO NOT engage in a verification
     subdialog
ELSE
     engage in a verification subdialog
```

Using this decision rule, the over-verification rate rises to 31.8% while the under-verification rate falls to 1.4%. Although it significantly reduces the under-verification rate, this strategy clearly leads to an excessive number of over-verifications. We next consider combination strategies that look at both parse cost and context.

4.3 Strategy 3: Parse Cost/Context Combination

The Strategy 1 decision rule for utterance verification says to engage in a verification subdialog if the parser confidence value falls below the verification threshold. With context-dependent verification we additionally require that the utterance meaning cannot be part of the main expectation. Thus, the decision rule for verification may be revised as follows:

```
IF the Parser Confidence Score > the
   Verification Threshold THEN DO NOT
   engage in a verification subdialog
ELSE IF utterance meaning in the Main
   Expectation THEN DO NOT engage in a
   verification subdialog
ELSE
   engage in a verification subdialog
```

Using this decision rule and comparing it to Strategy 1, the over-verification rate drops from 19.2% to 7.6% while the under-verification rate rises from 2.6% to 4.7% (i.e., the percentage of utterances correctly understood falls from 97.4% to 95.3%). This corresponds to a reduction in over-verifications from once every 5.2 user utterances to once every 13.2 user utterances while under-verifications (i.e., undetected misunderstandings) rises from once every 38.5 user utterances to once every 21.3 user utterances. It should be noted that on average, users spoke 20 utterances per dialog. We now examine a context-dependent strategy that takes into account specific domain information.

4.4 Strategy 4: Domain-Dependent Exceptions

As previously noted, correctly interpreting certain utterances is crucial for efficient continuation of the dialog. In the Circuit Fix-It Shop, the crucial condition was correct determination of the LED display. Several utterances in each dialog concerned a discussion of the LED display. Consequently, assertions about the LED display were often part of the main expectation.

However, due to the myriad of possible LED displays and the frequent misrecognition of key words and phrases in these descriptions, an effective dialog system would want to be careful to ascertain correctness in interpreting these descriptions. Consequently, we modify the verification decision rule as follows:

```
IF the Parser Confidence Score > the
   Verification Threshold THEN DO NOT
   engage in verification subdialog
ELSE IF the utterance meaning is an
   assertion about the LED display THEN
   engage in a verification subdialog
ELSE IF the utterance meaning is in the
   Main Expectation THEN DO NOT engage in
   a verification subdialog
ELSE
   engage in a verification subdialog
```

As a result, the decision rule for verifying utterances concerning the LED focuses solely on the local information about parsing cost and does not consider dialog context information about expectation.[5] Such a modification might also be appropriate in other domains for information deemed essential to continuing progress.

For this final decision rule the the over-verification rate is 9.8% while the under-verification rate is 3.7%.

4.5 Strategy Comparsion

Table 1 summarizes the results of the four strategies for a fixed Verification Threshold. We conclude that the combination of considering both the local information of the parsing cost and the dialog context information about expectation provides the best strategy. We also note that inclusion of domain-dependent information does not show any notable improvement in the over-verification/under-verification tradeoff as compared with the context-dependent but domain-independent Strategy 3.[6] We believe the results show that for task-oriented domains where there are fairly strong expectations for utterances that relate directly to task goals such as those described in figure 4, a context-dependent verification strategy is effective at reducing the over-verification rate to a reasonable amount while keeping the number of under-verifications to a near minimum. Further study is needed to determine the practical usefulness of this strategy in an actual experimental situation and it is an open question as to whether or not such strategies are feasible for less task-specific domains such as advisory dialogs and database query environments.

4.6 Improving Accuracy

Obtaining a higher accuracy requires reducing the under-verification rate. For Strategy 1 we explored the impact of raising and lowering the threshold on the over- and under-verification rates. Not surprisingly, there was a tradeoff. As the threshold was raised, more utterances are verified, resulting in fewer under-verifications but more over-verifications. Lowering the threshold had the opposite impact. In fact, using just the strategy of lowering the threshold to reduce the over-verification rate to 9.3% causes the under-verification rate to rise to 8.0%. In contrast, the new context-dependent strategy, Strategy 3, achieves an over-verification rate of 7.6%, but the under-verification rate is only 4.7%. Clearly, the use of dialog context in the verification subdialog decision rule improves system performance. Nevertheless, a small set of under-verifications remains. Are there any possibilities for further reductions in the under-verifications without a substantial increase in the over-verification rate?

[5] In actuality, a small component of the total parsing cost is the expectation cost based on dialog context, but that weighting is negligible compared to the weighting of the parse cost, the predominant factor in computing total cost.

[6] This of course, does not preclude the possibility that domain-dependent interaction may be more useful in other domains.

Strategy	Under-verification Rate	Over-verification Rate
1. Parse Cost Only	2.6%	19.2%
2. Context Only	1.4%	31.8%
3. Parse Cost/Context Combination	**4.7%**	**7.6%**
4. Domain-Dependent Exceptions	3.7%	9.8%

Table 1: Comparative Performance of Verification Subdialog Decision Strategies

An analysis of the 133 under-verifications that occur with the new strategy indicates that while some of the under-verifications are due to deficiencies in the grammar, there is a a core group of under-verifications where misrecognition of the speaker's words is impossible to overcome. Incorrect recognition of digits, lost content words, and misrecognized content words can cause the system to have high confidence in an incorrect interpretation. One approach that may prove helpful with this problem is the use of speech recognition systems that provide alternate hypotheses for the speech signal along with scoring information. Another possibility is word by word verification of the speaker input (see (Baber and Hone, 1993)), but such a strategy is too time-consuming and tedious for general spoken natural language dialog, especially when the user does not have access to a visual display of what the system hypothesizes was spoken. In general, experimental trials to observe subject reaction to verification subdialogs are needed.

In conclusion, while useful, there appear to be limits to the effectiveness of verification subdialogs. Consequently, strategies for delayed detection and resolution of miscommunication (e.g. (McRoy and Hirst, 1995), (Brennan and Hulteen, 1995), and (Lambert and Carberry, 1992)) become necessary and remain an area of continued investigation. These include both computer-initiated as well as user-initiated strategies.

5 Acknowledgments

The author expresses his appreciation to D. Richard Hipp for his work on the error-correcting parser and for his initial work on context-independent verification. The author also wishes to express his thanks to Steven A. Gordon and Robert D. Hoggard for their suggestions concerning this work and an earlier draft of this paper. Other researchers who contributed to the development of the experimental system include Alan W. Biermann, Robert D. Rodman, Ruth S. Day, Dania Egedi, and Robin Gambill. This research has been supported by National Science Foundation Grant IRI-9501571.

References

Allen, J.F. 1995. *Natural Language Understanding.* The Benjamin/Cummings Publishing Company, Inc., Menlo Park, California, 2nd edition.

Baber, C. and K.S. Hone. 1993. Modelling error recovery and repair in automatic speech recognition. *Intl. J. Man-Machine Studies*, 39:495–515.

Brennan, S.E. and E.A. Hulteen. 1995. Interaction and feedback in a spoken language system: a theoretical framework. *Knowledge-Based Systems*, 8:143–151.

Lambert, L. and S. Carberry. 1992. Modeling negotiation subdialogues. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 193–200.

McRoy, S. and G. Hirst. 1995. The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, pages 435–478.

Ney, H. 1991. Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, 39(2):336–340.

Oviatt, S.L. and P.R. Cohen. 1989. The effects of interaction on spoken discourse. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, pages 126–134.

Smith, R.W. and D.R. Hipp. 1994. *Spoken Natural Language Dialog Systems: A Practical Approach.* Oxford University Press, New York.

Smith, R.W., D.R. Hipp, and A.W. Biermann. 1995. An architecture for voice dialog systems based on Prolog-style theorem-proving. *Computational Linguistics*, pages 281–320.

Young, S.R., A.G. Hauptmann, W.H. Ward, E.T. Smith, and P. Werner. 1989. High level knowledge sources in usable speech recognition systems. *Communications of the ACM*, pages 183–194, February.